The Journal of Machine Learning Research

Volume 6
Print-Archive Edition

Pages 1–1098

Microtome Publishing
Brookline, Massachusetts
www.mtome.com

The Journal of Machine Learning Research
Volume 6
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 2005.

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

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

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

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

JMLR

Journal of Machine Learning Research Vol. 6 2005

Journal of Machine Learning Research 6 (2005) 1-35 Submitted 7/03; Revised 2/04; Published 1/05

Asymptotic Model Selection for Naive Bayesian Networks

Dmitry Rusakov RUSAKOV@CS.TECHNION.AC.IL
Dan Geiger DANG@CS.TECHNION.AC.IL
Computer Science Department
Technion - Israel Institute of Technology
Haifa, 32000, Israel

Editor: David Madigan

Abstract

We develop a closed form asymptotic formula to compute the marginal likelihood of data given a naive Bayesian network model with two hidden states and binary features. This formula deviates from the standard BIC score. Our work provides a concrete example that the BIC score is generally incorrect for statistical models that belong to stratified exponential families. This claim stands in contrast to linear and curved exponential families, where the BIC score has been proven to provide a correct asymptotic approximation for the marginal likelihood.

Keywords: Bayesian networks, asymptotic model selection, Bayesian information criterion (BIC)

1. Introduction

Statisticians are often faced with the problem of choosing the appropriate model that best fits a given set of observations. One example of such problem is the choice of structure in learning of Bayesian networks (Heckerman et al., 1995; Cooper and Herskovits, 1992). In such cases the maximum likelihood principle would tend to select the model of highest possible dimension, contrary to the intuitive notion of choosing the right model. Penalized likelihood approaches such as AIC have been proposed to remedy this deficiency (Akaike, 1974).

We focus on the Bayesian approach to model selection by which a model M is chosen according to the maximum posteriori probability given the observed data D:

$$P(M|D) \propto P(M,D) = P(M)P(D|M) = P(M)\int_\Omega P(D|M,\omega)P(\omega|M)d\omega,$$

where ω denotes the model parameters and Ω denotes the domain of the model parameters. In particular, we focus on model selection using large sample approximation for $P(M|D)$, called *BIC - Bayesian Information Criterion*.

The critical computational part in using this criterion is evaluating the marginal likelihood integral $P(D|M) = \int_\Omega P(D|M,\omega)P(\omega|M)d\omega$. Given an exponential model M we write $P(D|M)$ as a function of the averaged sufficient statistics Y_D of the data D, and the number N of data points in D:

$$\mathbb{I}[N,Y_D,M] = \int_\Omega e^{\mathcal{L}(Y_D,N|\omega,M)}\mu(\omega|M)d\omega, \tag{1}$$

where $\mu(\omega|M)$ is the prior parameter density for model M, and $\mathcal{L}$ is the log-likelihood function of model M. Recall that the sufficient statistics for multinomial samples of n binary variables

$(X_1,\ldots,X_n)$ is simply the counts $N \cdot Y_D$ for each of the possible 2^n joint states. Often the prior $P(M)$ is assumed to be equal for all models, in which case Bayesian model selection is performed by maximizing $\mathbb{I}[N,Y_D,M]$. The quantity represented by $S(N,Y_D,M) \equiv \ln \mathbb{I}[N,Y_D,M]$ is called the *BIC score* of model M.

For many types of models the asymptotic evaluation of Eq. 1, as $N \to \infty$, uses a classical Laplace procedure. This evaluation was first performed for Linear Exponential (LE) models (Schwarz, 1978) and then for Curved Exponential (CE) models under some additional technical assumptions (Haughton, 1988). It was shown that

$$S(N,Y_D,M) = N \cdot \ln P(Y_D|\omega_{ML}) - \frac{d}{2}\ln N + R, \tag{2}$$

where $\ln P(Y_D|\omega_{ML})$ is the log-likelihood of Y_D given the maximum likelihood parameters of the model and d is the model dimension, i.e., the number of parameters. The error term $R = R(N,Y_D,M)$ was shown to be bounded for a fixed Y_D (Schwarz, 1978) and uniformly bounded for all $Y_D \to Y$ in CE models (Haughton, 1988) as $N \to \infty$. For convenience, the dependence on M is suppressed from our notation in the rest of this paper.

The use of BIC score for Bayesian model selection for Graphical Models is valid for Undirected Graphical Models without hidden variables because these are LE models (Lauritzen, 1996). The justification of this score for Directed Graphical Models (called Bayesian Networks) is somewhat more complicated. On one hand discrete and Gaussian DAG models are CE models (Geiger et al., 2001; Spirtes et al., 1997). On the other hand, the theoretical justification of the BIC score for CE models has been established under the assumption that the model contains the true distribution - the one that has generated the observed data. This assumption limits the applicability of the proof of BIC score's validity for Bayesian networks in practical setups.

Haughton (1988) proves that if at least one of several models contains the true distribution, then the BIC score is the correct approximation to $\mathbb{I}[N,Y_D]$ and the correct model will be chosen by BIC score with probability 1 as $N \to \infty$. However, this claim does not guarantee correctness of the asymptotic expansion of $\mathbb{I}[N,Y_D]$ for models that do not contain the true distribution, nor does it guarantee correctness of model selection for finite N. The last problem is common to all asymptotic methods, but having a correct asymptotic approximation for $\mathbb{I}[N,Y_D]$ provides some confidence in this choice.

The evaluation of the marginal likelihood $\mathbb{I}[N,Y_D]$ for Bayesian networks with hidden variables is a wide open problem because the class of distributions represented by Bayesian networks with hidden variables is significantly richer than curved exponential models and it falls into the class of Stratified Exponential (SE) models (Geiger et al., 2001). The evaluation of the marginal likelihood for this class is complicated by two factors. First, some of the parameters of the model may be redundant, and should not be accounted in the BIC score (Geiger et al., 1996; Settimi and Smith, 1998). Second, the set of maximum likelihood points is sometimes a complex self-intersecting surface rather than a single maximum likelihood point as in the proven cases for linear and curved exponential models. Recently, major progress has been achieved in analyzing and evaluating this type of integrals (Watanabe, 2001). Herein, we apply these techniques to model selection among Bayesian networks with hidden variables.

The focus of this paper is the asymptotic evaluation of $\mathbb{I}[N,Y_D]$ for a binary naive Bayesian model with binary features. This model, described fully in Section 3, is useful in classification of binary vectors into two classes (Friedman et al., 1997). Our results are derived under similar assumptions to the ones made by Schwarz (1978) and Haughton (1988). In this sense, our paper generalizes the mentioned works, providing valid asymptotic formulas for a new type of marginal likelihood integrals. The resulting asymptotic approximations, presented in Theorem 4, deviate from the standard BIC score. Hence the standard BIC score is not justified for Bayesian model selection among Bayesian networks with hidden variables. Moreover, no uniform score formula exists for such models; our *adjusted BIC score* changes depending on the different types of singularities of the sufficient statistics, namely, the coefficient of the $\ln N$ term (Eq. 2) is no longer $-\frac{d}{2}$ but rather a function of the sufficient statistics. An additional result presented in Theorem 5 describes the asymptotic marginal likelihood given a degenerate (missing links) naive Bayesian model; it complements the main result presented by Theorem 4.

The rest of this paper is organized as follows. Section 2 introduces the concept of asymptotic expansions and presents methods of asymptotic approximation of integrals. Section 3 reviews naive Bayesian models and explicates the relevant marginal likelihood integrals for these models. Section 4 states and explains our main results and Section 5 gives a proof outline of Theorem 4 that demonstrates the mathematical techniques used herein. The full proof of our theorems is deferred to Appendices A and B. Section 6 discusses our contributions and outlines future research directions.

2. Asymptotic Approximation of Integrals

Exact analytical formulas are not available for many integrals arising in practice. In such cases approximate or asymptotic solutions are of interest. Asymptotic analysis is a branch of analysis that is concerned with obtaining approximate analytical solutions to problems where a parameter or some variable in an equation or integral becomes either very large or very small. In this section we review basic definitions and results of asymptotic analysis in relation to the integral $\mathbb{I}[N,Y_D]$ under study.

Let z represent a large parameter. We say that $f(z)$ is *asymptotically equal* to $g(z)$ for $z \to \infty$ if $\lim_{z\to\infty} f/g = 1$, and write

$$f(z) \sim g(z), \text{ as } z \to \infty.$$

Equivalently, $f(z)$ is asymptotically equal to $g(z)$ if $\lim_{z\to\infty} r/g = 0$, denoted $r = o(g)$, where $r(z) = f(z) - g(z)$ is the absolute error of approximation.

We often approximate $f(z)$ by several terms via an iterative approximation of the error terms. An asymptotic approximation by m terms has the form $f(z) = \sum_{n=1}^{m} a_n g_n(z) + o(g_m(z))$, as $z \to \infty$, where $\{g_n\}$ is an *asymptotic sequence* which means that $g_{n+1}(z) = o(g_n(z))$ as $z \to \infty$. An equivalent definition is

$$f(z) = \sum_{n=1}^{m-1} a_n g_n(z) + O(g_m(z)), \text{ as } z \to \infty,$$

where the big 'O' symbol states that the error term is bounded by a constant multiple of $g_m(z)$. The latter definition of asymptotic approximation is often more convenient and we use it herein, mostly for $m = 3$. A good introduction to asymptotic analysis can be found in (Murray, 1984).

The objective of this paper is deriving asymptotic approximation of marginal likelihood integrals as represented by Eq. 1, which for exponential families have the form

$$\mathbb{I}[N,Y_D] = \int_{\Omega} e^{-Nf(\omega,Y_D)} \mu(\omega) d\omega \tag{3}$$

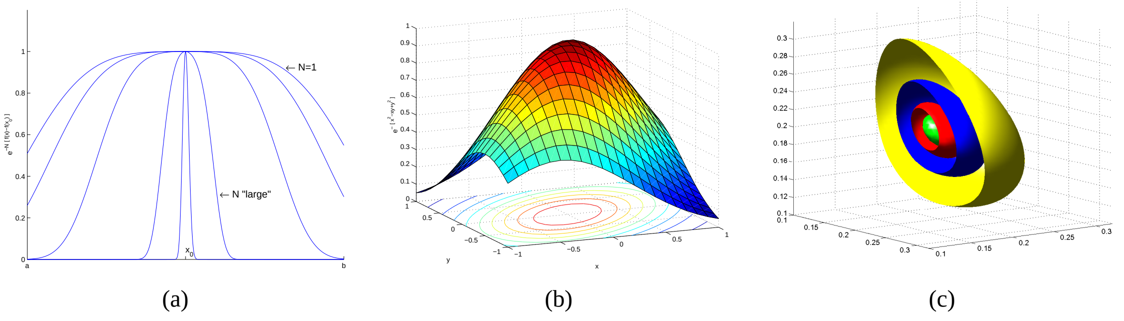


Figure 1: The classical Laplace procedure for approximation of integrals $\int e^{-Nf(x)}\mu(x)dx$, where f achieves single minimum in the range of integration. (a) The exponential integrand functions in one dimension, for different N. The large N the more mass of the function is concentrated in the small neighborhood of the extremum. (b) The two dimensional integrand function $e^{-(x^2-xy+y^2)}$, $(N=1)$. The isosurfaces are ellipses. (c) Ellipsoid-like isosurfaces of the three dimensional log-likelihood function function $f = -[0.2\ln\theta_1 + 0.2\ln\theta_2 + 0.2\ln\theta_3 + 0.4\ln(1-\theta_1-\theta_2-\theta_3)]$.

where $f(\omega, Y_D) = -\mathcal{L}(Y_D|\omega)$ is the minus log-likelihood function. We focus on exponential models, for which the log-likelihood of sampled data is equal to N times the log-likelihood of the averaged sufficient statistics. Note that the specific models discussed in this paper are indeed exponential.

Consider Eq. 3 for some fixed Y_D. For large N, the main contribution to the integral comes from the neighborhood of the minimum of f, i.e., the maximum of $-Nf(\omega, Y_D)$. See illustration on Figure 1(a,b). Thus, intuitively, the approximation of $\mathbb{I}[N, Y_D]$ is determined by the form of f near its minimum on Ω. In the simplest case $f(\omega)$ achieves a single minimum at ω_{ML} in the interior of Ω and this minimum is non-degenerate, i.e., the Hessian matrix $\mathcal{H}f(\omega_{ML})$ of f at ω_{ML} is of full rank. In this case the isosurfaces of the integrand function near the minimum f are ellipsoids (see Figure 1b,c) and the approximation of $\mathbb{I}[N, Y_D]$ for $N \longrightarrow \infty$ is the classical Laplace approximation (see, e.g., Wong, 1989, page 495) as follows.

Lemma 1 (Laplace Approximation) *Let*

$$I(N) = \int_U e^{-Nf(u)}\mu(u)du,$$

where $U \subset \mathbb{R}^d$. Suppose that f is twice differentiable and convex (i.e., $\mathcal{H}f(u)$ is positive definite), the minimum of f on U is achieved on a single internal point u_0, μ is continuous and $\mu(u_0) \neq 0$. If $I(N)$ absolutely converges, then

$$I(N) \sim Ce^{-Nf(u_0)}N^{-d/2}, \tag{4}$$

where $C = (2\pi)^{d/2}\mu(u_0)[\det \mathcal{H}f(u_0)]^{-\frac{1}{2}}$ is a constant.

Note that the logarithm of Eq. 4 yields the form of BIC score as presented by Eq. 2.

However, in many cases, and, in particular, in the case of naive Bayesian networks to be defined in the next section, the minimum of f is achieved not at a single point in Ω but rather on a variety $W_0 \subset \Omega$. Sometimes, this variety may be d'-dimensional surface (smooth manifold) in Ω in which case the computation of the integral is locally equivalent to the $d-d'$ dimensional classical case. The hardest cases to evaluate happen when the variety W_0 contains self-intersections.

Recently, an advanced mathematical method for approximating this type of integrals has been introduced to the machine learning community by Watanabe (2001). Below we briefly describe this method and state the main results. First, we introduce the main theorem that enables us to evaluate the asymptotic form of $\mathbb{I}[N, Y_D]$ as $N \to \infty$ computed in a neighborhood of a maximum likelihood point.[1]

Theorem 2 (based on Watanabe, 2001) *Let*

$$I(N) = \int_{W_\varepsilon} e^{-Nf(w)} \mu(w) dw$$

where W_ε is some closed ε-box around w_0, which is a minimum point of f in W_ε, and $f(w_0) = 0$. Assume that f and μ are analytic functions, $\mu(w_0) \neq 0$. Then,

$$\ln I(N) = \lambda_1 \ln N + (m_1 - 1) \ln \ln N + O(1)$$

where the rational number $\lambda_1 < 0$ and the natural number m_1 are the largest pole and its multiplicity of the meromorphic (analytic + poles) function that is analytically continued from

$$J(\lambda) = \int_{f(w)<\varepsilon} f(w)^\lambda \mu(w) dw \quad (Re(\lambda) > 0) \tag{5}$$

where $\varepsilon > 0$ is a sufficiently small constant.[2]

The above theorem states the main claim of the proof of Theorem 1 in (Watanabe, 2001). Consequently, the approximation of the marginal likelihood integral $\mathbb{I}[N, Y_D]$ (Eq. 3) can be determined by the poles of

$$J_{w_0}(\lambda) = \int_{W_\varepsilon} [f(w) - f(w_0)]^\lambda \mu(w) dw$$

evaluated in the neighborhoods W_ε of points w_0 on which f attains its minimum. This claim, which is further developed in Section 5, holds because the minimum of $f(w) - f(w_0)$ is zero and the main contribution to $\mathbb{I}[N, Y_D]$ comes from the neighborhoods around the minimums of f.

Often, however, it is not easy to find the largest pole and multiplicity of $J(\lambda)$ defined by Eq. 5. Here, another fundamental mathematical theory is helpful. The *resolution of singularities* in algebraic geometry transforms the integral $J(\lambda)$ into a direct product of integrals of a single variable.

Theorem 3 (Atiyah, 1970**, Resolution Theorem)** *Let $f(w)$ be a real analytic function defined in a neighborhood of $0 \in \mathbb{R}^d$. Then there exists an open set W that includes 0, a real analytic manifold U, and a proper analytic map $g : U \to W$ such that:*

1. *$g : U \setminus U_0 \to W \setminus W_0$ is an isomorphism, where $W_0 = f^{-1}(0)$ and $U_0 = g^{-1}(W_0)$.*

1. Throughout this paper we use styled '$\mathbb{I}$' symbol to denote our particular marginal likelihood integrals rather than standard 'I' symbol that denote general integrals appearing in theorems, examples and auxiliary derivations.
2. Recall that the pole of the complex function $f(z)$ is the point where it has a finite number of negative terms in its Laurent expansion, i.e., $f(z) = a_{-m}/(z-z_0)^m + \ldots + a_0 + a_1(z-z_0) + \ldots$. In this case it is said that $f(z)$ has a pole of order (or multiplicity) m at z_0. (See, e.g., Lang (1993), Section 5.3.)

2. *For each point $p \in U$ there are local analytic coordinates $(u_1, \dots, u_d)$ centered at p so that, locally near p, we have*

$$f(g(u_1, \dots, u_d)) = a(u_1, \dots, u_d) u_1^{k_1} \dots u_d^{k_d},$$

where $k_i \geq 0$ and $a(u)$ is an analytic function with analytic inverse $1/a(u)$.

This theorem is based on the fundamental results of Hironaka (1964) and the process of changing to u-coordinates is known as resolution of singularities.

Theorems 2 and 3 provide an approach for computing the leading terms in the asymptotic expansion of $\ln \mathbb{I}[N, Y_D]$:

1. Cover the integration domain Ω by a finite union of open neighborhoods W_α. This is possible under the assumption that Ω is compact.

2. Find a resolution map g_α and manifold U_α for each neighborhood W_α by resolution of singularities. Note that in the process of resolution of singularities U_α may be further divided into subregions $U_{\alpha\beta}$ by neighborhoods of different points $p \in U_\alpha$, as specified by Theorem 3. Select a finite cover of U_α by $U_{\alpha\beta}$, this is possible since closure of each U_α is also compact.

3. Compute the integral $J(\lambda)$ (Eq. 5) in each region $W_{\alpha\beta} = g_\alpha(U_{\alpha\beta})$ and find its poles and their multiplicity. This integral, denoted by $J_{\alpha\beta}$, becomes

$$\begin{aligned} J_{\alpha\beta}(\lambda) &= \int_{W_{\alpha\beta}} f(w)^\lambda \mu(w) dw \\ &= \int_{U_{\alpha\beta}} f(g_\alpha(w))^\lambda \mu(g_\alpha(u)) |g'_\alpha(u)| du \\ &= \int_{U_{\alpha\beta}} a(u)^\lambda u_1^{\lambda k_1} u_2^{\lambda k_2} \dots u_d^{\lambda k_d} \mu(g_\alpha(u)) \, |g'_\alpha(u)| \, du. \end{aligned} \tag{6}$$

   where $|g'_a(u)|$ is the Jacobian determinant. The last integration (up to a constant) is done by bounding $a(u)$ and $\mu(g_\alpha(u))$, using the Taylor expansion for $|g'_\alpha|$, and integrating each variable u_i separately. The largest pole $\lambda_{\alpha\beta}$ of $J_{\alpha\beta}$ and its multiplicity $m_{\alpha\beta}$ are now found.

4. The largest pole and multiplicity of $J(\lambda)$ are $\lambda_{(\alpha\beta)^*} = \max_{(\alpha\beta)} \lambda_{\alpha\beta}$ and the corresponding multiplicity $m_{(\alpha\beta)^*}$. If the $(\alpha\beta)^*$ values that maximize $\lambda_{\alpha\beta}$ are not unique, then the $(\alpha\beta)^*$ value that maximizes the corresponding multiplicity $m_{(\alpha\beta)^*}$ is chosen.

In order to demonstrate the above method, we conclude this section with an example, approximating the integral

$$I[N] = \int_{-\varepsilon}^{+\varepsilon} \int_{-\varepsilon}^{+\varepsilon} \int_{-\varepsilon}^{+\varepsilon} e^{-N(u_1^2 u_2^2 + u_1^2 u_3^2 + u_2^2 u_3^2)} \, du_1 du_2 du_3 \tag{7}$$

as N tends to infinity. This approximation of $I[N]$ is an important component in establishing our main results. The key properties of the integrand function in Eq. 7 are illustrated in Figure 2.

Watanabe's method calls for the analysis of the poles of the following function

$$J(\lambda) = \int_{-\varepsilon}^{+\varepsilon} \int_{-\varepsilon}^{+\varepsilon} \int_{-\varepsilon}^{+\varepsilon} (u_1^2 u_2^2 + u_1^2 u_3^2 + u_2^2 u_3^2)^\lambda \, du_1 du_2 du_3. \tag{8}$$

To find the poles of $J(\lambda)$ we transform the integrand function into a more convenient form by changing to new coordinates via the process of resolution of singularities. To obtain the needed

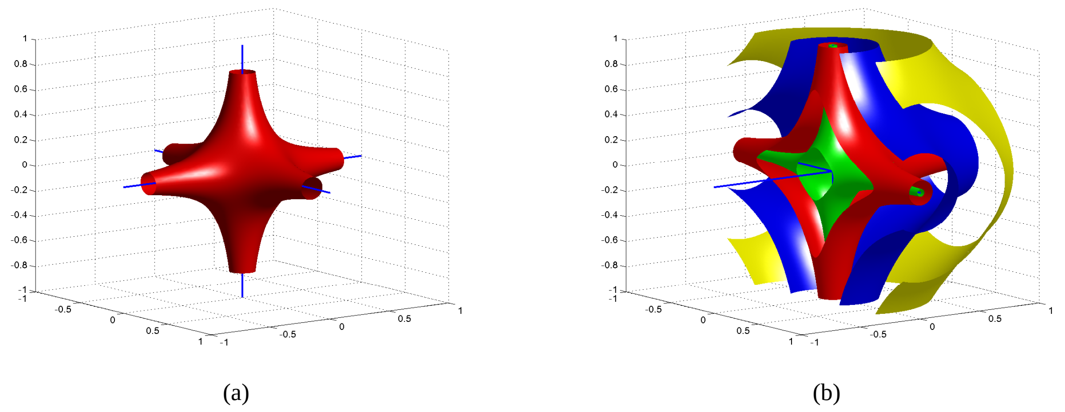


Figure 2: Part (a) depicts an isosurface of $e^{-N(u_1^2u_2^2+u_1^2u_3^2+u_2^2u_3^2)}$ (or alternatively of $u_1^2u_2^2+u_1^2u_3^2+u_2^2u_3^2$) and its set of maximum (minimum) points which coincide with the three axis. Part (b) depicts four isosurfaces of the same function for its different values. The isosurfaces are not ellipsoids as in the classical Laplace case of a single maximum (see Figure 1c).

transformations for the integral under study, we apply a technique called *blowing-up* which consists of a series of *quadratic transformations*. For an introduction to these techniques see (Abhyankar, 1990).

Rescaling the integration range to $(-1,1)$ and then taking only the positive octant yields

$$\begin{aligned} J(\lambda) &= 8\varepsilon^{4\lambda+3}\int_{(0,1)^3}(u_1^2u_2^2+u_1^2u_3^2+u_2^2u_3^2)^\lambda du \\ &= 8\varepsilon^{4\lambda+3}\left(\int_{0<u_2,u_3<u_1<1}+\int_{0<u_1,u_3<u_2<1}+\int_{0<u_1,u_2<u_3<1}\right)(u_1^2u_2^2+u_1^2u_3^2+u_2^2u_3^2)^\lambda du. \end{aligned}$$

The three integrals are symmetric, so we evaluate only the first. Using the quadratic transformation $u_2 = u_1u_2$, $u_3 = u_1u_3$, which modifies the integration range $0 < u_2, u_3 < u_1 < 1$ to be $(0,1)^3$, yields

$$J_1(\lambda) = \int_{0<u_2,u_3<u_1<1}(u_1^2u_2^2+u_1^2u_3^2+u_2^2u_3^2)^\lambda du = \int_{(0,1)^3} u_1^{4\lambda+2}(u_2^2+u_3^2+u_2^2u_3^2)^\lambda du.$$

We now divide the range $(0,1)^3$ to the regions $0 < u_3 < u_2 < 1$ and $0 < u_2 < u_3 < 1$. Again these cases are symmetric and so we continue to evaluate only the first using the transformation $u_3 = u_2u_3$,

$$J_{11}(\lambda) = \int_{0<u_3<u_2<1} u_1^{4\lambda+2}(u_2^2+u_3^2+u_2^2u_3^2)^\lambda du = \int_{(0,1)^3} u_1^{4\lambda+2}u_2^{2\lambda+1}(1+u_3^2+u_2^2u_3^2)du.$$

Since the function $(1+u_3^2+u_2^2u_3^2)$ is bounded on the region of integration, namely $1 \le 1+u_3^2+u_2^2u_3^2 \le 3$ for all $0 \le u_2, u_3 \le 1$, it follows that

$$8\varepsilon^{4\lambda+3}\int_{(0,1)^2} u_1^{4\lambda+2}u_2^{2\lambda+1}du_1du_2 \;\le\; J(\lambda) \;\le\; 24\varepsilon^{4\lambda+3}\int_{(0,1)^2} u_1^{4\lambda+2}u_2^{2\lambda+1}du_1du_2, \tag{9}$$

yielding

$$8\varepsilon^{4\lambda+3}\frac{1}{(4\lambda+3)(2\lambda+2)} \;\le\; J(\lambda) \;\le\; 24\varepsilon^{4\lambda+3}\frac{1}{(4\lambda+3)(2\lambda+2)}.$$

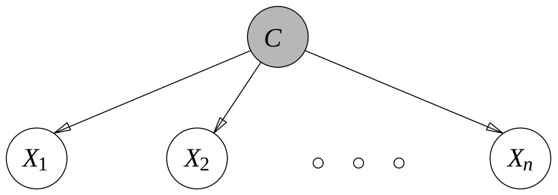


Figure 3: A naive Bayesian model. Class variable C is latent.

Thus $J(\lambda)$ has poles at $\lambda = -3/4$ and $\lambda = -1$ with multiplicity $m = 1$. The largest pole is $\lambda = -3/4$ with multiplicity $m = 1$. We conclude, using Theorem 2, that $I[N]$ defined by Eq. 7 is asymptotically equal to $cN^{-\frac{3}{4}}$.

We note that in this process of resolution of singularities we have implicitly computed the terms k_1, k_2, k_3, the function $a(u)$ and the Jacobian determinant $|g'(u)|$ (in Eq. 6). In particular, we have established that $k_1 = 4$, $k_2 = 2$, $k_3 = 0$, $a(u) = 1 + u_3^2 + u_2^2 u_3^2$ and $|g'(u)| = u_1^2 u_2$ for the appropriate range under study. The mapping g (of Theorem 3) is the composition of the two transformations we used and is defined via $u_1 = u_1$, $u_2 = u_1 u_2$ and $u_3 = u_1 u_2 u_3$. However, this explicit form is not needed for the evaluation of the target integral, as long as the values of k_i and $|g'(u)|$ are derived.

In the proof of our theorems we perform a similar process of resolution of singularities producing implicitly the mapping g which is guaranteed to exist according to Theorem 3, and which determines the values of k_i and $|g'(u)|$ needed for evaluation of poles of function $J(\lambda)$ as required by Theorem 2.

3. Naive Bayesian Models

A naive Bayesian model M for discrete variables $X = \{X_1, \ldots, X_n\}$ is a set of joint distributions for X that factor according to the tree structure depicted on Figure 3, where the class variable C is never observed. Formally, a probability distribution $P(X = x)$ belongs to a naive Bayesian model if and only if

$$P(X = x) = \sum_{j=1}^{r} P(C = c_j) \prod_{i=1}^{n} P(X_i = x_i | C = c_j),$$

where $x = (x_1, \ldots, x_n)$ is the n-dimensional binary vector of values of X, r is the number of hidden states and c_j denotes a particular unobserved state (class). Intuitively, this model describes the generation of data x that comes from r sources $c_1, \ldots, c_r$. Naive Bayesian models are a subclass of Bayesian networks (Pearl, 1988) and they are widely used in clustering (Cheeseman and Stutz, 1995).

In this work we focus on naive Bayesian networks that have two hidden states ($r = 2$) and n binary feature variables $X_1, \ldots, X_n$. We denote the parameters defining $p(x_i = 1|c_1)$ by a_i, the parameters defining $p(x_i = 1|c_2)$ by b_i, and the parameters defining $p(c_1 = 1)$ by t. These parameters are called the *model parameters*. We denote the *joint space parameters* $P(X = x)$ by θ_x. The following mapping, named T, relates these two sets of parameters:

$$\theta_x = t \prod_{i=1}^{n} a_i^{x_i} (1 - a_i)^{1 - x_i} + (1 - t) \prod_{i=1}^{n} b_i^{x_i} (1 - b_i)^{1 - x_i}, \tag{10}$$

and the marginal likelihood integral (Eq. 1) for these models becomes

$$\mathbb{I}[N,Y_D] = \int_{(0,1)^{2n+1}} e^{N\sum_x Y_x \ln\theta_x(\omega)} \mu(\omega) d\omega \tag{11}$$

where $\omega = (a_1,\ldots,a_n,b_1,\ldots,b_n,t)$ are the model parameters, N is the sample size, and the averaged sufficient statistics Y_x is the number of samples for which $X = x$ divided by the sample size N.

4. Main Results

This section presents an asymptotic approximation of the integral $\mathbb{I}[N,Y_D]$ (Eq. 11) for naive Bayesian networks consisting of binary variables $X_1,\ldots,X_n$ and two hidden states. It is based on two results. First, the classification of singular points for these types of models (Geiger et al., 2001). Second, Watanabe's approach as explained in Section 2, which provides a method to obtain the correct asymptotic formula of $\mathbb{I}[N,Y_D]$ for the singular points not covered by the classical Laplace approximation scheme.

Let $\Upsilon = \{(y_1,\ldots,y_{2^n}) | y_i \geq 0, \sum y_i = 1\}$ be the set of possible values of the averaged sufficient statistics $Y_D = (Y_1,\ldots,Y_{2^n})$ for data $D = \{(x_{i,1},\ldots,x_{i,n})\}_{i=1}^N$. In our asymptotic analysis we let the sample size N grow to infinity.

Let $\Upsilon_0 \subset \Upsilon$ be the points $(y_1,\ldots,y_{2^n})$ that correspond to the joint space parameters of the distributions that can be represented by binary naive Bayesian models with n binary variables. In other words, assuming the indices of y_i are written as vectors $(\delta_1,\ldots,\delta_n)$ of n zeros and ones, points in Υ_0 are those that can be parameterized via

$$y_{(\delta_1,\ldots,\delta_n)} = t\prod_{i=1}^n a_i^{\delta_i}(1-a_i)^{1-\delta_i} + (1-t)\prod_{i=1}^n b_i^{\delta_i}(1-b_i)^{1-\delta_i} \tag{12}$$

where t, $a = (a_1,\ldots,a_n)$ and $b = (b_1,\ldots,b_n)$ are the $2n+1$ model parameters, as defined in Section 3.

Geiger et al. (2001) classify the singular points of the algebraic variety of the parameters of binary naive Bayesian networks into two classes S and S'. This classification is used here to classify the possible statistics arising from binary naive Bayesian networks with different parameters; The set S is the set of points $(y_1,\ldots,y_{2^n})$ such that Eq. 12 holds and all $a_i = b_i$ except for at most two indices in $\{1,\ldots,n\}$. Intuitively, each such point represents a probability distribution that can be defined by a naive Bayesian model (Figure 3) with all links removed except at most two.

The set $S' \subset S$ is the set of points represented by a naive Bayesian model, just as the set S does, but with all links removed; namely, a distribution where all variables are mutually independent and independent of the class variable as well. These statistics are parameterized via $y_{(\delta_1,\ldots,\delta_n)} = \prod_{i=1}^n a_i^{\delta_i}(1-a_i)^{1-\delta_i}$.

Clearly $S' \subset S \subset \Upsilon_0 \subset \Upsilon$. We call points in $\Upsilon_0 \setminus S$ *regular points*, and points in sets $S \setminus S'$ and S' *type* 1 and *type* 2 *singularities*, respectively. We now present our main result.

Theorem 4 (Asymptotic Marginal Likelihood Formula) *Let $\mathbb{I}[N,Y_D]$ (Eqs. 10 and 11) be the marginal likelihood of data with averaged sufficient statistics Y_D given the naive Bayesian model with binary variables and two hidden states with parameters $\omega = (a,b,t)$. Namely,*

$$\begin{aligned} \mathbb{I}[N,Y_D] &= \textstyle\int_{(0,1)^{2n+1}} e^{N\sum_x Y_x \ln\theta_x(\omega)}\mu(\omega)d\omega, \\ \theta_{(x_1,\ldots,x_n)} &= t\textstyle\prod_{i=1}^n a_i^{x_i}(1-a_i)^{1-x_i} + (1-t)\prod_{i=1}^n b_i^{x_i}(1-b_i)^{1-x_i}, \end{aligned} \tag{13}$$

where $x = (x_1, \ldots, x_n)$ denotes the binary vector of length n and the vectors Y_D and θ of length 2^n are indexed by x. Let Y_D and μ satisfy the following assumptions:

A1 Bounded density. *The density $\mu(\omega)$ is bounded and bounded away from zero on $\Omega = (0,1)^{2n+1}$.*

A2 Positive statistics. *The statistics $Y_D = (Y_1, \ldots, Y_{2^n})$ are such that $Y_i > 0$ for $i = 1, \ldots, 2^n$.*

A3 Statistics stability. *There exists a sample size N_0 such that the averaged sufficient statistics Y_D is equal to a limiting statistics Y for all sample sizes $N \geq N_0$.*

Then, for $n \geq 3$ as $N \to \infty$:

(a) If $Y \in \Upsilon_0 \setminus S$ (regular point)

$$\ln \mathbb{I}[N, Y_D] = N \ln P(Y|\omega_{ML}) - \frac{2n+1}{2} \ln N + O(1), \tag{14}$$

(b) If $Y \in S \setminus S'$ (type 1 singularity)

$$\ln \mathbb{I}[N, Y_D] = N \ln P(Y|\omega_{ML}) - \frac{2n-1}{2} \ln N + O(1), \tag{15}$$

(c) If $Y \in S'$ (type 2 singularity)

$$\ln \mathbb{I}[N, Y_D] = N \ln P(Y|\omega_{ML}) - \frac{n+1}{2} \ln N + O(1), \tag{16}$$

where ω_{ML} are the maximum likelihood parameters for the averaged sufficient statistic Y. Moreover, for $n = 2$, $S = \Upsilon_0 = \Upsilon$ and

(d) If $Y \notin S'$ (namely, $Y \in S \setminus S'$),

$$\ln \mathbb{I}[N, Y_D] = N \ln P(Y|\omega_{ML}) - \frac{3}{2} \ln N + O(1), \tag{17}$$

(e) If $Y \in S'$,

$$\ln \mathbb{I}[N, Y_D] = N \ln P(Y|\omega_{ML}) - \frac{3}{2} \ln N + 2 \ln \ln N + O(1), \tag{18}$$

and for $n = 1$,

(f)
$$\ln \mathbb{I}[N, Y_D] = N \ln P(Y|\omega_{ML}) - \frac{1}{2} \ln N + O(1), \tag{19}$$

as $N \to \infty$.

The first assumption that the prior density μ is bounded has been made by all earlier works; in some applications it holds and in some it does not. The proof and results, however, can be easily modified to apply to any particular kind of singularity of μ, as long as the form of singularity is specified. The second and third assumptions are made to ease the proof; the third assumption was also made by (Schwarz, 1978). Removing these assumptions is beyond the scope of this paper.

Note that Eq. 15 corresponds to selecting $\lambda_1 = -\frac{2n-1}{2}$ and $m_1 = 1$ in Watanabe's method and Eq. 16 corresponds to selecting $\lambda_1 = -\frac{n+1}{2}$ and $m_1 = 1$. Both formulas are different from the standard BIC score, given by Eq. 14, which only applies to regular points, namely, the points in $\Upsilon_0 \setminus S$. In contrast to the standard BIC score, which is uniform for all points Y_D, the asymptotic approximation given by our *adjusted BIC score* depends on the value of $Y = Y_D$ through the coefficient of $\ln N$.

One might be tempted to think that the coefficient of the $-\ln N$ term can be guessed by various intuitive considerations. We now discuss three such erroneous attempts. First, the number of parameters of the model that generates a singular point Y_D is $n+1$ for case (c) because there are $n+1$ independent binary variables (the class variable and n feature variables). This may seem to explain the coefficient of $\ln N$ in case (c). However, using the same reasoning for case (b) yields the coefficient $(n+3)/2$ which differs from the correct coefficient. Another attempt is to claim that the coefficient of $-\ln N$ is half the number of parameters in the naive Bayesian model minus the number of redundant parameters in the model that generates Y_D. In particular, for case (b), the number of redundant parameters in the generative model is $(n+3)-(n+1) = 2$ and so the speculated coefficient should be $(2n+1-2)/2 = (2n-1)/2$ which is the correct coefficient. However, using the same reasoning for case (c) yields the coefficient $2n/2$ which is wrong. Finally, computing the maximum rank of the Jacobian of the map from the model parameters to the joint space parameters (defined by Eq. 22) at the maximum likelihood parameters w_{ML} for singular statistics Y_D yields the correct coefficient for case (b) but the wrong coefficient $(2n-1)/2$ for case (c).

The next theorem specifies the asymptotic behavior of marginal likelihood integrals for degenerate naive Bayesian models, namely, when some of the links are missing. This theorem complements Theorem 4 and its proof is explicated in Appendix B.

Theorem 5 *Let M be the degenerate naive Bayesian model with two hidden states and n binary feature variables of which m are independent of the hidden state and let*

$$\omega = (a_1, \ldots, a_{n-m}, b_1, \ldots, b_{n-m}, t, c_{n-m+1}, \ldots, c_n)$$

be the $2n-m+1$ *model parameters of M. Let* $\mathbb{I}[N, Y_D]$ *be the marginal likelihood of data D with averaged sufficient statistics* Y_D *given model M. Namely,*

$$\begin{aligned} \mathbb{I}[N,Y_D] &= \int_{(0,1)^{2n+1}} e^{N\sum_x Y_x \ln\theta_x(\omega)} \mu(\omega) d\omega, \\ \theta_x &= \left(t\prod_{i=1}^{n-m} a_i^{x_i}(1-a_i)^{1-x_i} + (1-t)\prod_{i=1}^{n-m} b_i^{x_i}(1-b_i)^{1-x_i}\right) \prod_{i=n-m+1}^{n} c_i^{x_i}(1-c_i)^{1-x_i}, \end{aligned} \tag{20}$$

where $x = (x_1, \ldots, x_n)$ *denotes the binary vector of length n and the vectors* Y_D *and* θ *of length* 2^n *are indexed by x. Let* Y_D *and* μ *satisfy the following assumptions:*

A1 Bounded density. *The density* $\mu(\omega)$ *is bounded and bounded away from zero on* $\Omega = (0,1)^{2n+1}$.

A2 Positive statistics. *The statistics* $Y_D = (Y_1, \ldots, Y_{2^n})$ *are such that* $Y_i > 0$ *for* $i = 1, \ldots, 2^n$.

A3 Statistics stability. *There exists a sample size* N_0 *such that the averaged sufficient statistics* Y_D *is equal to a limiting statistics Y for all sample sizes* $N \geq N_0$.

Assume also that $Y \in \Upsilon_0$ *and that the parameterization of Y (as is Eq. 12) corresponds to a binary naive Bayesian model M′, which shares k links with model M. Then, for* $m \leq n-3$ *as* $N \to \infty$*:*

(a) If $k \geq 3$ (regular point)

$$\ln \mathbb{I}[N, Y_D] = N \ln P(Y|\omega_{ML}) - \frac{2n+1-m}{2} \ln N + O(1),$$

(b) If $k = 2$ (type 1 singularity)

$$\ln \mathbb{I}[N, Y_D] = N \ln P(Y|\omega_{ML}) - \frac{2n-1-m}{2} \ln N + O(1),$$

(c) If $k = 0$ or $k = 1$ (type 2 singularity)

$$\ln \mathbb{I}[N, Y_D] = N \ln P(Y|\omega_{ML}) - \frac{n+1}{2} \ln N + O(1),$$

where ω_{ML} are the maximum likelihood parameters of statistics Y.
Furthermore, for $m = n - 2$

(d) If $k = 2$ (type 1 singularity)

$$\ln \mathbb{I}[N, Y_D] = N \ln P(Y|\omega_{ML}) - \frac{n+1}{2} \ln N + O(1).$$

Note that here $n + 1 = 2n - m - 1$, since $m = n - 2$.

(e) If $k = 0$ or $k = 1$ (type 2 singularity)

$$\ln \mathbb{I}[N, Y_D] = N \ln P(Y|\omega_{ML}) - \frac{n+1}{2} \ln N + 2 \ln \ln N + O(1),$$

and for $m = n - 1$ or $m = n$,

$$(f) \qquad \ln \mathbb{I}[N, Y_D] = N \ln P(Y|\omega_{ML}) - \frac{n}{2} \ln N + O(1),$$

regardless of k as $N \rightarrow \infty$.

An adversary may argue that evaluating the marginal likelihood on singular points is not needed because one could exclude from the model all singular points which only have measure zero. The remaining set would be a smooth manifold defining a curved exponential model, and so the standard BIC score would be a correct asymptotic expansion as long as the point Y_D has not been excluded. However, this proposed remedy is not perfect because in some situations the data may come from a model that yields singular statistics relative to the models being compared.

As an example of incorrect Bayesian model selection by the standard BIC score, consider the problem of selecting between two naive Bayesian models M_1 and M_2, as depicted on Figure 4. Suppose that the data is generated by the third model M_T. Both models M_1 and M_2 can not represent the target distribution (M_T) exactly, therefore, given a large enough sample, the choice of the model depends on the particular distribution represented by M_T and its parameters. Intuitively, if the dependencies of X_1 and X_2 on the hidden node C in model M_T are stronger than the dependency of X_4 on the hidden node, then one should prefer model M_1 over model M_2, and vice versa.

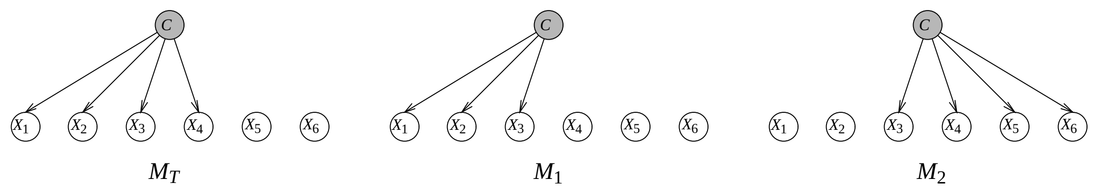


Figure 4: An example of incorrect Bayesian model selection by the standard BIC score. M_T represents the generating model, and M_1, M_2 represent models being compared. If the maximum likelihoods of data given M_1 and M_2 happen to be equal, e.g., for true model parameters $a_1 = 0.75$, $a_2 = 0.2$, $a_3 = 0.12$, $a_4 = 0.17$, $b_1 = 0.33$, $b_2 = 0.12$, $b_3 = 0.07$, $b_4 = 0.77$, $a_5 = b_5 = 0.2$, $a_6 = b_6 = 0.6$, $t = 0.42$, then the model selection procedure based on the standard BIC score will prefer model M_1, as it is less penalized compared to M_2. Using the adjusted BIC formula (Theorem 5), on the other hand, gives an advantage to M_2, reflecting its higher marginal likelihood.

Now, if the maximum likelihoods of the data given model M_1 and given model M_2 happen to be equal, which is possible when X_4 depends strongly on C in M_T (Figure 4), then the standard choice of the model is dictated by the penalty term of the BIC score (Eq. 2). The penalty term is smaller for M_1, which contains less parameters than M_2, and, consequently, the model preferred by the standard BIC score is M_1. However, the adjusted BIC approximation formula for the marginal likelihood for models with hidden variables penalizes model M_2 less than model M_1 (Theorem 5). Therefore, the marginal likelihood of the data given model M_2 is asymptotically larger than that of model M_1 and it should be chosen according to a Bayesian model selection procedure, given enough data.

Note that when comparing a naive Bayesian model versus a sub-model, where the data comes from the smaller model, then the standard BIC score may underevaluate the larger model, but this would not lead to an incorrect model selection.

5. Proof Outline of Theorem 4

The proof of Theorem 4 consists of two logical parts. The first part is the proof of claim (a) of Theorem 4 that follows from the fact that for regular statistics $Y \in \Upsilon_0 \setminus S$ there are only two (symmetric) maximum likelihood points at each of which the log-likelihood function is properly convex. Hence, the marginal likelihood integral can be approximated by the classical Laplace method (Lemma 1). The proof of Theorem 4a, which reflects standard practice, is provided in Appendix A.2. The second logical part consists of the proofs of claims (b) and (c) of Theorem 4 and requires the advanced techniques of Watanabe (Section 2). First, the integral $\mathbb{I}[N, Y_D]$ is transformed by a series of transformations into a simpler one. Second, the sets of extremum points of the exponent (maximum log-likelihood points) are found, and then the new integral is computed in the neighborhoods of extremum points. Finally, the logarithm of the largest contribution gives the desired asymptotic approximation of the original integral. We focus on one thread of our proof, which demonstrates this method, deferring the full proof to Appendix A.

5.1 Useful Transformations

Decomposing the transformation T from the model parameters (a,b,t) to the joint space parameters θ_x, as defined by Eq. 13, facilitates the evaluation of the integral $\mathbb{I}[N,Y_D]$. We decompose T into a series of three transformations T_1, T_2, T_3 such that $T = T_3 \circ T_2 \circ T_1$. We call the model parameters (a,b,t) - *the source coordinates* and the parameters θ_x - *the target coordinates*. The transformations T_1 and T_3 are diffeomorphisms, namely, one-to-one differentiable mappings with differentiable inverses, that change the source and target coordinates, respectively, and are defined in such a way that the intermediate transformation T_2, which carries all the information about the singularities, is simple to analyze. These transformations are from (Geiger et al., 2001).

Denote the domain of the model parameters by $\Omega = [0,1]^{2n+1}$ and the domain of the joint space parameters by $\Theta = \overline{\Delta}_{2^n-1}$, where $\overline{\Delta}_{2^n-1} = \{(\alpha_1,\ldots,\alpha_{2^n-1}) | \alpha_i \geq 0, \sum \alpha_i \leq 1\}$ is the closed 2^n-1 dimensional unit simplex. Let $U = T_1(\Omega)$ be the image of T_1, $\Lambda = T_3^{-1}(\Theta)$ be the preimage of T_3, and $T_2 : U \to \Lambda$ be the transformation that relates these sets. These transformations are chained as follows:

$$\Omega_{(a,b,t)} \xleftrightarrow{\mathbf{T_1}} U_{(x,u,s)} \xrightarrow{\mathbf{T_2}} \Lambda_{(z)} \xleftrightarrow{\mathbf{T_3}} \Theta_{(\theta)}$$

where the indices denote the names of the coordinates used to describe the corresponding spaces. We now present these three transformations.

Transformation T_1: We define $T_1 : \Omega \to U$ via

$$s = 2t-1, \quad u_i = \frac{a_i - b_i}{2}, \quad x_i = ta_i + (1-t)b_i, \quad i = 1,\ldots,n. \tag{21}$$

The mapping T_1 is a diffeomorphism with $|\det J_{T_1}| = 2^{-n+1}$. The inverse transformation is given by

$$t = (s+1)/2, \quad a_i = x_i + (1-s)u_i, \quad b_i = x_i - (1+s)u_i, \quad i = 1,\ldots,n. \tag{22}$$

Furthermore, it can be verified that U is the set of points $(x,u,s) \in \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}$ such that

$$0 \leq x_i \leq 1, \quad -1 \leq s \leq 1, \quad -x_i \leq (1-s)u_i \leq 1-x_i, \quad x_i - 1 \leq (1+s)u_i \leq x_i. \tag{23}$$

Transformation T_3: We define $T_3 : \Lambda \to \Theta$ as the inverse of a composition of two transformations T_{31} and T_{32}. First, consider the nonsingular transformation $T_{31} : \Theta \to \Lambda'$ defined by

$$\nu_{ij\ldots k} = \sum_{(x_1,\ldots,x_n),\ \text{s.t. } x_i = x_j = \ldots = x_k = 1} \theta_{(x_1,\ldots,x_n)}$$

where ν_i stands for the probability of the ith feature being true, ν_{ij} stands for the probability that the ith and jth features are both true, etc. We now express $\nu_{ij\ldots k}$ using the model parameters (a,b,t) via

$$\nu_{ij\ldots k} = ta_i a_j \ldots a_k + (1-t) b_i b_j \ldots b_k. \tag{24}$$

Using Eq. 22, we rewrite Eq. 24 obtaining

$$\begin{aligned} &\nu_i = x_i, \quad \nu_{ij} = x_i x_j + (1-s^2) u_i u_j, \\ &\nu_{ijk} = x_i x_j x_k + (1-s^2)(x_i u_j u_k + u_i x_j u_k + u_i u_j x_k) - 2s(1-s^2) u_i u_j u_k \\ &\nu_{12\ldots r} = x_1 x_2 \cdots x_r + \textstyle\sum_{i=2}^{r} p_i(s) \left(\sum \text{“products of } i\ u'\text{s and } r-i\ x'\text{s''}\right) \end{aligned} \tag{25}$$

where $p_i(s) = 1/2\left[(1-s)^i(1+s)+(-1-s)^i(1-s)\right]$, and, in particular, $p_2(s) = 1-s^2$ and $p_3(s) = -2s(1-s^2)$.

Now we subtract products of the first n coordinates to remove the leading terms. So, we do $z_{ij} = \nu_{ij} - \nu_i\nu_j$. Then we subtract products of the first n coordinates with one of the new coordinates to remove the second terms, namely, $z_{ijk} = \nu_{ijk} - \nu_i\nu_j\nu_k - z_{ij}\nu_k - z_{ik}\nu_j - z_{jk}\nu_i$, and so forth. We end up with the transformation $T_{32} : \Lambda' \to \Lambda$ defined by

$$z_i = \nu_i, \quad z_{ij} = \nu_{ij} - \nu_i\nu_j, \quad z_{ijk} = \nu_{ijk} - \nu_i\nu_j\nu_k - z_{ij}\nu_k - z_{ik}\nu_j - z_{jk}\nu_i, \quad \text{etc.} \tag{26}$$

where the indices of the z coordinates are non-empty subsets of $\{1,\ldots,n\}$. In particular, the z coordinate corresponding to a set $I \subseteq \{1,\ldots,n\}$ is z_I, the z coordinate corresponding to $\{i\}$ is z_i, and the z coordinate corresponding to $\{i,j,k\} \subseteq \{1,\ldots,n\}$ is z_{ijk}, etc.

The transformations T_{31} and T_{32} are diffeomorphisms with Jacobian determinant 1. The transformation T_3 is defined by $T_3 = T_{31}^{-1} \circ T_{32}^{-1} : \Lambda \to \Theta$. Hence, T_3 is a diffeomorphism with Jacobian determinant equal to 1.

Transformation T_2: We define $T_2 : U \subset \mathbb{R}^{2n+1} \to \Lambda \subset \mathbb{R}^{2^n-1}$ via

$$z_i = x_i, \quad z_{ij} = p_2(s)u_iu_j, \quad \ldots, \quad z_{12\ldots r} = p_r(s)u_1u_2\ldots u_r \tag{27}$$

obtained by combining Eqs. 25 and 26. We use the notation $z_I(x,u,s)$ when the dependence of z_I on (x,u,s) needs to be explicated. Note that this transformation is not a diffeomorphism for $n > 3$.

Transformations T_1, T_2 and T_3 are similar to transformations used by (Settimi and Smith, 2000) in the study of the geometry of parametric spaces for Bayesian networks with hidden variables. These transformations can be regarded as reparameterizations of the naive Bayesian models in terms of moments. In particular, if the hidden and observable nodes are assumed to have states -1 and 1, then $s = \mathbb{E}[C]$, $u_i = Cov(X_i,C)/Var(C)$, $p_i(s) = \mathbb{E}[(C-s)^i]$ and $z_{12\ldots r} = \mathbb{E}[\prod_{i=1}^r (X_i - \mathbb{E}[X_i])]$.

5.2 Preliminary Lemmas

Based on the transformations T_1, T_2 and T_3, we present two lemmas that facilitate the evaluation of the integral $\mathbb{I}[N,Y_D]$. The first lemma states that under Assumptions $A1$ and $A3$, the integral $\mathbb{I}[N,Y_D]$ can be asymptotically evaluated in the (x,u,s) coordinates for a limiting statistics Y, while dismissing the contribution of the density function μ. The second lemma shows that the resulting integral $\tilde{\mathbb{I}}[N,Y]$ can be evaluated using the quadratic form in the z coordinates.

Lemma 6 *Let $\mathbb{I}[N,Y_D]$ be defined by Eq. 13, namely,*

$$\mathbb{I}[N,Y_D] = \int_{(0,1)^{2n+1}} e^{N\sum_x Y_x \ln\theta_x(\omega)}\mu(\omega)d\omega$$

and assume μ is bounded ($A1$) and Y_D is stable ($A3$). Let

$$\tilde{\mathbb{I}}[N,Y] = \int_U e^{-Nf(\theta[x,u,s])}dx\,du\,ds \tag{28}$$

where

$$\begin{aligned} f(x,u,s) &= f_Y - \textstyle\sum_{i=1}^{2^n} Y_i \ln\theta_i[x,u,s], \\ \theta[x,u,s] &= (T_3 \circ T_2)[x,u,s], \quad \theta_{2^n}[x,u,s] = 1 - \textstyle\sum_{i=1}^{2^n-1}\theta_i[x,u,s], \end{aligned} \tag{29}$$

and where $f_Y = \max_{(x,u,s)\in U} \sum_{i=1}^{2^n} Y_i \ln \theta_i[x,u,s]$ *and* Y *is the limiting statistics of* Y_D *as specified by Assumption A3, namely,* $Y_D = Y$ *for* $N \geq N_0$.

Then, $f_Y = P(Y|\omega_{ML})$ *and*

$$\ln \mathbb{I}[N, Y_D] = N f_Y + \ln \tilde{\mathbb{I}}[N, Y] + O(1) \tag{30}$$

for all $N > 1$.

Proof: Since T_1 is a diffeomorphism, $f_Y = P(Y|\omega_{ML})$ and the integral $\mathbb{I}[N, Y_D]$ can be evaluated in (x,u,s) coordinates by introducing the constant factor of Jacobian determinant of transformation T_1, $J_{T_1} = 2^{-n+1}$. Moreover $\mu(\omega)$ is bounded and thus the integral evaluated with $\mu(\omega) \equiv 1$ is within a constant factor of $\mathbb{I}[N, Y_D]$ and since Y_D is equal to Y starting from N_0, fixing Y_D to Y introduces finite number of approximation errors for $N < N_0$ that can be bounded. Thus, $\tilde{\mathbb{I}}[N, Y]$ is within a constant factor of the integral $\mathbb{I}[N, Y_D]$ multiplied by $e^{N f_Y}$ with the constants independent on N and Y_D. Eq. 30 expresses this fact in a logarithmic scale. ■

Lemma 7 *Consider* $\tilde{\mathbb{I}}[N, Y]$ *and* $f(x,u,s)$ *as defined in Lemma 6 (Eqs. 28 and 29). Let the zero set* $U_0 = \arg\min_{(x,u,s)\in U} f(x,u,s)$ *be the set of minimum points of* $f(x,u,s)$ *in* U. *Let*

$$\mathbb{J}[N, Y] = \max_{p_0 \in U_0} \mathbb{J}_{p_0}[N, Y] \quad \text{and} \quad \mathbb{J}_{p_0}[N] = \int_{U_\varepsilon \cap U} e^{-N \sum_I (z_I(x,u,s) - z'_I)^2} dx\, du\, ds, \tag{31}$$

where $z_I(x,u,s)$ *is the I-th coordinate of* $z(x,u,s) = T_2[x,u,s]$, z'_I *is the I-th coordinate of* $T_2[x',u',s']$ *and* U_ε *is an* ε*-box neighborhood of* $p_0 = (x',u',s') \in U_0$. *(Note that* $\mathbb{J}_{p_0}[N]$ *does not depend on* Y, *while* $\mathbb{J}[N, Y]$ *depends on* Y *through the form of set* U_0.*)*

If Y *is positive (A2) and* $Y \in \Upsilon_0$, *then*

$$\ln \tilde{\mathbb{I}}[N, Y] = \ln \mathbb{J}[N, Y] + O(1) \quad \text{for all } N > 1. \tag{32}$$

The proof of this lemma uses the facts that T_3 is a diffeomorphism, U is compact, the contributions of non-maximum regions of $-f$ are exponentially small, and the 2^n dimensional point $Y > 0$ corresponds to a maximum likelihood parameters of naive Bayesian network with binary variables and two hidden states. The proof is explicated in Appendix A.1.

Lemmas 6 and 7 jointly state that the asymptotic forms of $\ln \mathbb{J}[N, Y]$ and $\ln \mathbb{I}[N, Y_D]$ are identical up to an additive term $N f_Y$ and a constant provided that Y is the limiting statistics of Y_D (Assumption A3).

5.3 Analysis of Type 2 Singularity

We now focus on the proof of Theorem 4c that deals with the singular points in S'. Let $Y \in S'$. Our starting point in proving Theorem 4c is integral $\mathbb{J}[N, Y]$ (Eq. 31), which by Lemmas 6 and 7 specifies the asymptotic form of $\mathbb{I}[N, Y_D]$. We evaluate the contributions $\mathbb{J}_{p_0}[N]$ to $\mathbb{J}[N, Y]$ from the neighborhoods of extremum points $p_0 = (x',u',s') \in U_0$. The largest contribution determines the asymptotic form of integral $\mathbb{I}[N, Y_D]$ as $N \to \infty$ and $Y_D = Y$.

Let $\gamma = (\gamma_1, \ldots, \gamma_n)$ be the model parameters of the n independent variables that define the 2^n dimensional point $Y \in S'$, namely

$$\gamma_j = \sum_{\delta \in \{0,1\}^n, \text{s.t. } \delta_j = 1} Y_{(\delta_1, \ldots, \delta_n)}, \qquad j = 1, \ldots, n. \tag{33}$$

Furthermore, $Y \in S'$ if and only if for all $\delta \in \{0,1\}^n$, equality $Y_{(\delta_1,\ldots,\delta_n)} = \prod_{i=1}^n \gamma_i^{\delta_i}(1-\gamma_i)^{1-\delta_i}$ holds for $\gamma = \{\gamma_1,\ldots,\gamma_n\}$ given by Eq. 33.

Let $\bar{V}$ denote the closure of a set V. The zero set U_0 can be written as the union of $n+2$ sets

$$U_0 = \bar{U}_{0-} \cup \bar{U}_{0+} \cup \bigcup_{j=1}^{n} \bar{U}_{0j}, \tag{34}$$

where

$$\begin{aligned}
U_{0-} &= \left\{(x=\gamma, u, s=-1) \mid u_i \in \left(\tfrac{-\gamma_i}{2}, \tfrac{1-\gamma_i}{2}\right),\ i=1,\ldots,n\right\}, & W_{0-} &= \{(a, b=\gamma, t=0) \mid a_i \in (0,1)\},\\
U_{0+} &= \left\{(x=\gamma, u, s=1) \mid u_i \in \left(\tfrac{\gamma_i-1}{2}, \tfrac{\gamma_i}{2}\right),\ i=1,\ldots,n\right\}, & W_{0+} &= \{(a=\gamma, b, t=1) \mid b_i \in (0,1)\},\\
U_{0j} &= \left\{(x=\gamma,u,s) \mid \begin{array}{l} u_i = 0, \forall i \neq j;\\ u_j \in \left(-\tfrac{1}{2}, \tfrac{1}{2}\right); s \in (-1,1);\\ -\gamma_j < (1-s)u_j < 1-\gamma_j,\\ \gamma_j - 1 < (1+s)u_j < \gamma_j \end{array}\right\}, & W_{0j} &= \left\{(a,b,t) \mid \begin{array}{l} a_i = b_i = \gamma_i, \forall i \neq j;\\ ta_j + (1-t)b_j = \gamma_j \end{array}\right\},
\end{aligned} \tag{35}$$

and where $W_{0-} = T_1^{-1}(U_{0-})$, $W_{0+} = T_1^{-1}(U_{0+})$, and $W_{0j} = T_1^{-1}(U_{0j})$ are the same sets expressed using the model parameters (a,b,t).

The zero set U_0, namely the minimum points of f, is divided into five disjoint sets:

C1: $(x',u',s') \in U_{0j} \setminus \bigcup_{i\neq j} U_{0i}$.

C2: $(x',u',s') \in \bigcap_j U_{0j}$.

C3: $(x',u',s') \in U_{0-} \cup U_{0+} \setminus \bigcup_j \bar{U}_{0j}$.

C4: $(x',u',s') \in \bigcup_j \left[U_{0-} \cup U_{0+} \cap \bar{U}_{0j} \setminus \bigcup_{i\neq j} \bar{U}_{0i}\right]$.

C5: $(x',u',s') \in (U_{0-} \cup U_{0+}) \bigcap_j \bar{U}_{0j}$.

These five disjoint sets and their boundaries cover U_0, because $U_{0+} \cap U_{0-} = \emptyset$ and $U_{0i} \cap U_{0j} = \bigcap_k U_{0k}$. The set U_0 is shown in Figure 5 along with a representative point from $C1$ through $C5$.

Note that U_0 is a union of two n-dimensional planes U_{0-}, U_{0+} and n two-dimensional planes U_{0j}, $j=1,\ldots,n$. Consequently, one could perhaps guess from the classical Laplace approximation analysis that because the zero subsets U_{0-}, U_{0+} have dimension n, the coefficient of the $\ln N$ term would be at least $-(2n+1-n)/2 = -(n+1)/2$. Indeed this happens, but a formal proof requires to closely examine the form of f near the different minimum points. This evaluation is complicated by the fact that the zero planes intersect (see Figure 5), and such cases ($C2$, $C4$, $C5$) are not covered by the classical Laplace approximation analysis.

The proof proceeds case by case by evaluating the integrals $\mathbb{J}_{p_0}[N]$ (Eq. 31) around points $p_0 = (x',u',s')$ from the sets $C1$ through $C5$. Then, the maximal asymptotic value of $\mathbb{J}_{p_0}[N]$ is the approximation of $\mathbb{J}[N,Y]$, as specified by Lemma 7. We now treat case $C2$ which demonstrates the main ideas, deferring the other cases to Appendix A.

According to case $C2$, $(x',u',s') = \bigcap_j U_{0j}$. Each point of case $C2$ satisfies $u_i' = 0$ and $x_i' = \gamma_i$ for $i=1,\ldots,n$ and $s' \neq \pm 1$. Furthermore, its z coordinates satisfy $z_i' = x_i'$ for all $i=1,\ldots,n$ and $z_I' = 0$ for all other indices. Let $\phi(x,u,s) = \sum_I \left[z_I(x'+x, u'+u, s'+s) - z_I'\right]^2$. Note that $\phi(x,u,s)$ is term

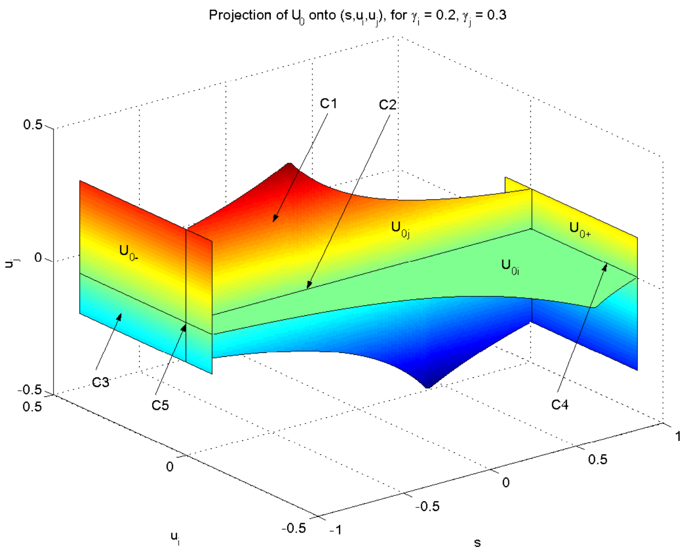


Figure 5: The set U_0 projected on (s,u_i,u_j), for $x_i=\gamma_i=0.2$, $x_j=\gamma_j=0.3$. Examples of points of types $C1$-$C5$ are marked.

in the exponent of the integrand of $\mathbb{J}[N,Y]$ centered around the minimum point (x',u',s'). Using transformation T_2 (Eq. 27), we obtain

$$\begin{aligned}\phi(x,u,s) = &\ \textstyle\sum_I \left[z_I(x'+x,u'+u,s'+s)-z'_I\right]^2 \\ = &\ \textstyle\sum_i [z_i-z'_i]^2+\sum_{ij,i\neq j}[z_{ij}-z'_{ij}]^2+\sum_{ijk,i\neq j\neq k}[z_{ijk}-z'_{ijk}]^2+\dots \\ = &\ \textstyle\sum_i \left[(x'_i+x_i)-x'_i\right]^2+\sum_{ij,i\neq j}\left[(1-(s'+s)^2)u_iu_j-0\right]^2+\text{“higher order terms”} \\ = &\ \textstyle\sum_i x_i^2+\sum_{ij,i\neq j}\left[(1-s'^2)u_iu_j-(s+2s')su_iu_j\right]^2+\text{“higher order terms”}.\end{aligned} \tag{36}$$

The higher order terms are multiplication of three, four and more u_i's and their contribution is bounded by the terms explicitly written in Eq. 36. For example, third terms are of form $(z_{ijk}-z'_{ijk})^2=4(s'+s)^2(1-(s'+s)^2)^2u_i^2u_j^2u_k^2\leq 5\varepsilon^2u_i^2u_j^2$ for all $s,u_i,u_j,u_k<\varepsilon$ for ε small enough. Similar bounds can be obtained for all high order terms in Eq. 36. Thus, the principal part of ϕ, that bounds ϕ within the multiplicative constant near zero, is given by

$$\tilde{\phi}(x,u,s)=\sum_i x_i^2+\sum_{ij,i\neq j}u_i^2u_j^2. \tag{37}$$

and $\tilde{\phi}(x,u,s)\leq\phi(x,u,s)\leq 2\tilde{\phi}(x,u,s)$ for all $s,u_i,u_j<\varepsilon$ for ε small enough.

Since the multiplicative constants in the exponent can be transferred to the multiplicative constants of integral itself by changing the integration range around zero and rescaling, we only need to evaluate the asymptotic form of integral $\int e^{-N(\sum_i x_i^2+\sum_{ij,i\neq j}u_i^2u_j^2)}dxduds$ in order to get the asymptotic form of integral $\mathbb{J}[N,Y\in S']$ (Eq. 31) within a constant multiply.

The quadratic form in x_i's contributes an $N^{-n/2}$ factor to the integral $\tilde{\mathbb{J}}[N]$. This can be shown by decomposing the integral and integrating out the x_i's. We are left with the evaluation of the integral

$$\tilde{\mathbb{J}}[N] = \int_{(-\varepsilon,+\varepsilon)^n} e^{-N\sum_{ij,i\neq j} u_i^2 u_j^2} du.$$

For $n = 3$, this is precisely the integral evaluated as example in Section 2 which was found to be asymptotically equal to $cN^{-\frac{3}{4}}$. Generalizing the approach demonstrated in the example in Section 2 to $n \geq 3$ we obtain that the largest pole of $J(\lambda)$ is $\lambda_1 = -n/4$ with multiplicity $m = 1$, so $\tilde{\mathbb{J}}[N]$ is asymptotically equal to $cN^{-\frac{n}{4}}$. Thus the contribution of the neighborhood of $(x', u', s') \in \bigcap_j U_{0j}$ to $\mathbb{J}[N, Y \in S']$ is $cN^{-\frac{3n}{4}}$.

In summary, we have analyzed case $C2$, showing that the contribution to $\mathbb{J}[N, Y \in S']$ is $cN^{-\frac{3n}{4}}$. The dominating contributions in the cases $C3$, $C4$, and $C5$, are all equal to $cN^{-\frac{n+1}{2}}$ (the proof of this claim is given in Appendix A). The dominating contribution in case $C1$ is only $cN^{-\frac{2n-1}{2}}$. Also, the various border points of U_0 do not contribute more than the corresponding internal points. Thus, $\mathbb{J}[N, Y] = cN^{-\frac{n+1}{2}}$ for $Y' \in S$. Consequently, due to Lemmas 6 and 7, $\ln \mathbb{I}[N, Y_D] = N \cdot P(Y|\omega_{ML}) - \frac{n+1}{2} \ln N + O(1)$, as claimed by Theorem 4c. ■

6. Discussion

This paper presents an asymptotic approximation of the marginal likelihood of data given a naive Bayesian model with binary variables (Theorem 4). This Theorem proves that the classical BIC score that penalizes the log-likelihood of a model by $\frac{d}{2} \ln N$ is incorrect for Bayesian networks with hidden variables and suggests an adjusted BIC score. Moreover, no uniform penalty term exists for such models in the sense that the penalty term, i.e., the coefficient of $\ln N$, depends on the averaged sufficient statistics. This result resolves an open problem regarding the validity of the classical BIC score for stratified exponential families, raised in (Geiger et al., 2001).

The major limitation of Theorem 4 arises from Assumptions $A2$ and $A3$. While Assumption $A1$ (bounded density) is often satisfied in applications, Assumption $A2$ (positive statistics) is only sometimes satisfied and Assumption $A3$ (statistics stability) is never satisfied in practice. Nevertheless, this Theorem is an essential advance towards developing asymptotic Bayesian methods for model selection among naive Bayesian models in particular, and for Bayesian networks with hidden variables in general. We now highlight the steps required for obtaining a valid, practical asymptotic model selection score for arbitrary latent Bayesian networks, namely, for Bayesian networks with hidden variables.

1. Develop a closed form asymptotic formula for marginal likelihood integrals for all types of statistics Y given an arbitrary latent Bayesian model.

2. Extend these solutions by developing *uniform* asymptotic approximations valid for converging statistics $Y_D \to Y$ as $N \to \infty$. A uniform asymptotic approximation is an approximation that has the error term bounded for all Y_D near Y and for all N.

3. Develop an algorithm that, given a Bayesian network with hidden variables and a data set with statistics Y_D, determines the possible singularity types of the limit statistics Y and applies the appropriate asymptotic formula developed in step 2.

Our work provides a first step for naive Bayesian networks and a concrete framework to pursue these tasks.

Theorem 4 shows that when comparing the classical BIC score with our adjusted BIC score (Eq. 2 versus Eqs. 15, 16), one can see that a naive Bayesian network with all links present is somewhat under-evaluated using the classical BIC score for singular statistics Y because the penalty term reduces from $(2n+1)/2$ in the classical score to $(2n-1)/2$ (or $(n+1)/2$) in the adjusted score. We conjecture that such under evaluation occurs for general Bayesian networks with hidden variables. As a result, when the data shows weak dependencies for some links, often resulting in evaluation of the marginal likelihood near singular points of the model, then those models with more links might be under evaluated using BIC, but correctly evaluated with a uniform asymptotic formula that takes the proximity to a singular points into account. An illustrative example of incorrect model choice by the standard BIC score has been presented in Figure 4.

We conclude with two remarks. First, we note that the adjusted penalty term (Eqs. 15, 16) falls within the range of penalty terms, studied by Keribin (2000), that lead to sure consistency estimators in a frequentist's interpretation.

Second, we note that, the sets of singular points S and S' are defined in (Geiger et al., 2001) as the singular points of the algebraic varieties of distributions represented by binary naive Bayesian networks in the joint space parameters space, while here the same sets are defined as sets of *statistics* points Y which give rise to singular maximum likelihood in the model parameters space. At the singular points of the joint space parameters space, regular local coordinates do not exist and the usual coordinates (i.e., the model parameters) that parameterize the rest of the model variety have a number of coordinates crushed into a single point. This results in complex surfaces of maximum likelihood points in the model parameter space and, consequently, a non-standard behavior of marginal likelihood integrals which we have started to explore in this paper. Another ramification of this observation is that a bounded prior density defined on the model parameters may accumulate massively on a single point on the model variety in the joint space parameter space, violating the boundedness assumption of the prior density and thus yielding non-standard approximations to marginal likelihood integrals in the joint space parameters.

Acknowledgments

The second author thanks David Heckerman and Chris Meek for years of collaboration on this subject. An early version of this paper, without proofs and without Theorem 5, has been presented at the 18th UAI Conference (Rusakov and Geiger, 2002).

Appendix A. Proof of Theorem 4 (The Main Theorem)

We start with the proof of Lemma 7, which requires two additional lemmas. Then we proceed with a case by case proof of Theorem 4.

A.1 Proof of Lemma 7

The proof of Lemma 7 uses Lemmas 8 and 9. In particular, Lemma 8 states that a local version of the claim made by Lemma 7 (Eq. 32) holds in the neighborhood of extremum points p_0 under two additional assumptions denoted by $B1$ and $B2$. Lemma 9 shows that $B1$ and $B2$ hold. Finally, the proof of Lemma 7 elevates the local version to the global claim.

Lemma 8 *Let*

$$\begin{aligned} & f(x,u,s) = f_Y - \textstyle\sum_{i=1}^{2^n} Y_i \ln \theta_i[x,u,s], \\ & \theta[x,u,s] = (T_3 \circ T_2)[x,u,s], \quad \theta_{2^n}[x,u,s] = 1 - \textstyle\sum_{i=1}^{2^n-1} \theta_i[x,u,s], \end{aligned} \tag{38}$$

where $f_Y = \max_{(x,u,s)\in U} \sum_{i=1}^{2^n} Y_i \ln \theta_i[x,u,s]$ *and* $Y = (Y_1, \ldots, Y_{2^n})$ *is a non-negative vector with sum of elements equal to* 1. *Let the zero set* $U_0 = \arg\min_{(x,u,s)\in U} f(x,u,s)$ *be the set of minimum points of* $f(x,u,s)$ *on* U*, let* $p_0 = (x', u', s')$ *be a point in* U_0 *and let*

$$\tilde{\mathbb{I}}_{p_0}[N,Y] = \int_{U_\varepsilon} e^{-N f(x,u,s)} dx\, du\, ds, \tag{39}$$

where U_ε *is some small neighborhood of* p_0*. Also, let*

$$\mathbb{J}_{p_0}[N] = \int_{U_\varepsilon} e^{-N \sum_I (z_I(x,u,s) - z'_I)^2} dx\, du\, ds,$$

where $z_I(x,u,s)$ *is the* I*-th coordinate of* $z(x,u,s) = T_2[x,u,s]$ *and* z'_I *is the* I*-th coordinate of* $T_2[x',u',s']$.
Further assume that (x',u',s') *satisfies*

B1. $\theta' = T_3 \circ T_2(x',u',s')$ *is a minimum of* f *as function of* θ, $f(\theta') = 0$ *and* $\nabla_\theta f(\theta') = 0$.

B2. f*, as a function of* θ*, is strictly convex at* $\theta' = \theta(x',u',s')$*, i.e., the matrix* $\mathcal{H}_\theta f(\theta')$ *is positive definite.*

Then,

$$\ln \tilde{\mathbb{I}}_{p_0}[N,Y] = \ln \mathbb{J}_{p_0}[N] + O(1) \quad \text{for all } N > 1. \tag{40}$$

(The right hand side of Eq. 40 depends on Y *through the* $O(1)$ *term.)*

Proof: Since $\nabla_\theta f(\theta') = 0$, $\mathcal{H}_\theta f(\theta')$ is positive definite and $T_3 : \Lambda_{(z)} \to \Theta_{(\theta)}$ is a diffeomorphism, it follows that $\nabla_z f(z') = 0$ and $\mathcal{H}_z f(z')$ is positive definite. Also, $f(z') = 0$. Therefore, f as a function of z can be approximated by a quadratic form near $z' = T_2(x',u',s')$ via

$$\eta_1 \sum_I (z_I - z'_I)^2 \; < \; f(z) \; < \; \eta_2 \sum_I (z_I - z'_I)^2, \quad \text{for } z \in \Lambda_\varepsilon, \tag{41}$$

where Λ_ε is some sufficiently small neighborhood of z', and $\eta_1, \eta_2 > 0$ are slightly smaller and larger, respectively, than all eigenvalues of $\mathcal{H}_z f(z')$. Consequently, since $T_2 : U \to \Lambda$ is continuous, there exists neighborhood U_ε of p_0 such that $T_2(U_\varepsilon) \subseteq \Lambda_\varepsilon$ and Inequality 41 holds for $z(x,u,s) = T_2(x,u,s)$ for all points (x,u,s) in U_ε. Using Inequality 41 for evaluating $\tilde{\mathbb{I}}_{p_0}[N,Y]$ (Eq. 39) yields

$$\int_{U_\varepsilon} e^{-\eta_2 N \sum_I (z_I(x,u,s) - z'_I)^2} dx\, du\, ds \; < \; \tilde{\mathbb{I}}_{p_0}[N,Y] \; < \; \int_{U_\varepsilon} e^{-\eta_1 N \sum_I (z_I(x,u,s) - z'_I)^2} dx\, du\, ds.$$

Due to Theorem 2, the bounding integrals are asymptotically equivalent up to a multiplicative constant, because the poles and multiplicities of the corresponding $J(\lambda)$ functions (Eq. 5) that determine their asymptotic behavior are the same for any constant multiples of $\sum (z_I(x,u,s) - z'_I)^2$, and in particular, for the multipliers η_1, η_2 and 1. ■

Lemma 9 *Let $f(u,x,s)$ be as defined by Eq. 38, namely,*

$$f(x,u,s) = f_Y - \textstyle\sum_{i=1}^{2^n} Y_i \ln\theta_i[x,u,s],$$

$$\theta[x,u,s] = (T_3 \circ T_2)[x,u,s], \quad \theta_{2^n}[x,u,s] = 1 - \textstyle\sum_{i=1}^{2^n-1} \theta_i[x,u,s],$$

where $f_Y = \max_{(x,u,s)\in U} \sum_{i=1}^{2^n} Y_i \ln\theta_i[x,u,s]$ and $Y = (Y_1,\dots,Y_{2^n})$ is a vector in Υ_0 (defined by Eq. 12) such that $Y_i > 0$ (A2). Let the zero set $U_0 = \arg\min_{(x,u,s)\in U} f(x,u,s)$ be the set of minimum points of f, and let (x',u',s') be a point in U_0. Then $f(x',u',s') = 0$, and

B1. $\theta' = T_3 \circ T_2(x',u',s')$ is a minimum point of f as function of θ on Θ, $f(\theta') = 0$ and $\nabla_\theta f(\theta') = 0$. Furthermore, $\theta' = (Y_1,\dots,Y_{2^n-1})$ and $\nabla f(x',u',s') = 0$.

B2. f as a function of θ is strictly convex at θ', i.e., $\mathcal{H}_\theta f(\theta')$ is positive definite.

B3. If $n \geq 3$ and $Y \in \Upsilon_0 \setminus S$, then $f(x,u,s)$ is strictly convex at (x',u',s'), that is, the matrix $\mathcal{H}_{(x,u,s)} f(x',u',s')$ is positive definite.

B4. Also, if $n \geq 3$ and $Y \in \Upsilon_0 \setminus S$, then U_0 consists only of two distinct points (x',u',s') and (x'',u'',s''), such that $x' = x''$, $u' = -u''$ and $s' = -s''$.

Proof: The claim $f(x',u',s') = f(\theta') = 0$ follows directly from the definitions of f, θ' and f_Y.

Consider Claim $B1$. The point $\theta_0 = (Y_1,\dots,Y_{2^n-1})$ is the unique minimum of f, as a function of θ, on Θ, because $f_Y - f(\theta) = \sum_i Y_i \ln\theta_i[x,u,s]$ is the logarithm of a multinomial distribution. Since $Y \in \Upsilon_0$, the distribution specified by θ_0 can be represented by the model parameters, namely, $\theta_0 \in (T_3 \circ T_2)[U_0]$. Consequently, $\theta_0 = (T_3 \circ T_2)[U_0]$ because θ_0 is the unique minimum of f. So, $\theta' = \theta_0 = (Y_1,\dots,Y_{2^n-1})$. Furthermore, because $Y > 0$, θ' is an internal point of Θ yielding $\nabla_\theta f(\theta') = 0$. Finally $\nabla f(x_0,u_0,s_0) = J^T_{(T_3\circ T_2)}(x_0,u_0,s_0)\nabla_\theta f(\theta') = 0$ as well.

Claim $B2$ is established by explicit calculations. The Hessian matrix $\mathcal{H}_\theta f(\theta')$ at $\theta' = (Y_1,\dots,Y_{2^n-1})$ is given by

$$\left[\mathcal{H}_\theta f(\theta')\right]_{ij} = \begin{cases} \frac{1}{Y_{2^n}} & \text{for } i \neq j \\ \frac{1}{Y_i} + \frac{1}{Y_{2^n}} & \text{for } i = j \end{cases}$$

Consequently, for any $a \in \mathbb{R}^{2^n-1}$, $a \neq 0$, it follows that

$$a^T \cdot \mathcal{H}_\theta f(\theta') \cdot a = \sum_{i=1}^{2^n-1} \frac{a_i^2}{Y_i} + \frac{1}{Y_{2^n}} \left[\sum_{i=1}^{2^n-1} a_i\right]^2 > 0.$$

Claim $B3$ follows from the proof of Theorem 12 of (Geiger et al., 2001), which shows that the Jacobian of the transformation T_2 is of maximal rank for $n \geq 3$ for points (x',u',s') that satisfy $\theta' = T_2[x',u',s'] \in \Upsilon_0 \setminus S$. The mentioned theorem and claim $B2$ imply that for all $a \in \mathbb{R}^{2n+1}$, $a \neq 0$,

$$\begin{aligned} a^T \cdot \mathcal{H}_{(x,u,s)} f(x_0,u_0,s_0) \cdot a &= a^T \cdot \left[J^T_{(T_3\circ T_2)}(x_0,u_0,s_0) \cdot \mathcal{H}_\theta f(\theta') \cdot J_{(T_3\circ T_2)}(x_0,u_0,s_0)\right] \cdot a \\ &= \left[J_{(T_3\circ T_2)}(x_0,u_0,s_0)\cdot a\right]^T \cdot \mathcal{H}_\theta f(\theta') \cdot \left[J_{(T_3\circ T_2)}(x_0,u_0,s_0)\cdot a\right] = b^T \cdot \mathcal{H}_\theta f(\theta') \cdot b > 0, \end{aligned}$$

where $b = J_{(T_3\circ T_2)}(x_0,u_0,s_0)\cdot a$. This proves Claim $B3$ because $\mathcal{H}_\theta f(\theta')$ is positive definite and $b \neq 0$ lest $J_{(T_3\circ T_2)}$ would not be of maximal rank.

Claim $B4$ follows from claim $B1$ that $\theta' = (Y_1, \ldots, Y_{2^n-1})$ and from Theorem 13 in (Geiger et al., 2001), which states that for $\theta' \in \Upsilon_0 \setminus S$, there are exactly two source points (x,u,s), precisely the ones specified by Claim $B4$, that satisfy $\theta' = T_2[x,u,s]$. ■

Proof of Lemma 7: Lemma 8 combined with Lemma 9 establish the asymptotic behavior of $\tilde{\mathbb{I}}[N,Y]$ in the ε_{p_0} neighborhood of a single minimum p_0 (Eq. 40). Now, since U is closed and bounded (Eq. 23), it is *compact*. Hence, from an arbitrary infinite set of ε-neighborhoods of points in U, there exist a finite subset of disjoint neighborhoods of points in U that cover U. The neighborhoods that do not contain minimum points can be discarded since their contribution to the integral is exponentially small, i.e., a contribution bounded by e^{-Nc_1} versus e^{-Nc_2} where $c_1 > c_2$. Let $U_0' \subseteq U_0$ denote the finite set of points from U_0, the neighborhoods of which are chosen to cover U_0. Also, let $\mathbb{J}[N,Y]$ denote the maximal contribution to $\tilde{\mathbb{I}}[N,Y]$, as in Lemma 7 (Eq. 31). We obtain

$$\mathbb{J}[N,Y] \le \tilde{\mathbb{I}}[N,Y] \le \sum_{p_0 \in U_0'} \mathbb{J}_{p_0}[N] \le k \cdot \mathbb{J}[N,Y], \tag{42}$$

where k is the number of points in U_0'. Taking the logarithm of Eq. 42 yields Eq. 32 which establishes Lemma 7. ■

Claims $B3$ and $B4$ of Lemma 9 have not been used in the proof of Lemma 7. These claims are needed in the next section.

A.2 Proof of Theorem 4a (Regular Statistics Case)

Theorem 4a rephrases standard facts regarding asymptotic expansion of integrals around a single extremum point. Recall that Theorem 4a states that if $Y_D = Y$ for $N \ge N_0$, $Y_i > 0$ for $i = 1, \ldots, 2^n$ and $Y \in \Upsilon_0 \setminus S$, then asymptotic approximation of $\ln \mathbb{I}[N,Y_D]$ (Eq. 13) equals $N \ln P(Y|w_{ML}) - \frac{2n+1}{2} \ln N + O(1)$ (Eq. 14). To prove this claim we use Lemma 6 which states that $\mathbb{I}[N,Y_D]$ and $\tilde{\mathbb{I}}[N,Y]$ have the same asymptotic approximation up to a multiplicative constant e^{Nf_Y} and compute $\tilde{\mathbb{I}}[N,Y]$ using Lemma 1 (Laplace approximation).

We start by noticing that $\mathbb{I}[N,Y_D]$ absolutely converges for any $N \ge 1$ and $Y_D \ge 0$. That is because the integrand function $e^{N \sum_x Y_x \ln \theta_x(w)} = \prod_x \theta_x(w)^{NY_x}$ satisfies $0 \le \theta_x(w)^{NY_x} \le 1$ for all N, Y_D, i and $w = (a,b,t) \in \Omega$ and because $\mu(a,b,t)$ is a probability density function on Ω, thus integral $\mathbb{I}[N,Y_D]$ is finite (and less than 1). Consequently, $\tilde{\mathbb{I}}[N,Y]$ also absolutely converges for any $N \ge 1$ and any $Y \ge 0$, as required in order to use Lemma 1.

Consider now the integral $\tilde{\mathbb{I}}[N,Y] = \int_U e^{-Nf(x,u,s)} dxduds$. Since the value of $e^{-Nf(x,u,s)}$ outside the small neighborhoods of the minimums f is exponentially small, so the asymptotic behavior of $\tilde{\mathbb{I}}[N,Y]$ on U is actually described by integration of $\tilde{\mathbb{I}}[N,Y]$ in the small neighborhoods of minimums of f (Lemma 7). Since $\tilde{\mathbb{I}}[N,Y]$ converges and Claims $B1$, $B3$ and $B4$ of Lemma 9 hold, it follows that in sufficiently small neighborhoods of the two internal minimum points of f, the integral $\tilde{\mathbb{I}}[N,Y]$ can be computed by Lemma 1 (Laplace Approximation).

Consequently, integrating $\tilde{\mathbb{I}}[N,Y]$ in the full neighborhoods of the maximum likelihood points $(x',u',s') \in U_0$, that lie on the border of U, introduces only a constant multiplicative errors to the approximation. This is shown by considering the integral $\tilde{\mathbb{I}}[N,Y]$ around minimum points of f in the equivalent (since T_1 is a diffeomorphism) coordinates (a,b,t), which have the full integration domain $\Omega = (0,1)^{2n+1}$. In these coordinates, approximating f by a quadratic form (as performed by Laplace approximation) on (a,b,t) and integrating in a full neighborhood of border point results in multiplicative error factor of 2^k where k is the number of border coordinates.

We now apply Lemma 1 to the small neighborhoods of the two minimum points of f and by combining Eq. 30 with the logarithm of sum of two approximations described by Eq. 4 we obtain the Theorem 4a. ■

Theorem 4a does not specify the $O(1)$ term. The constant term C is well known in explicit form when the minimum of f is achieved on a single point, as specified by Lemma 1. In our case, the minimum of f is achieved on two points (x', u', s') and (x'', u'', s'') and by taking the integrals $\mathbb{J}_{(x',u',s')}[N]$ and $\mathbb{J}_{(x'',u'',s'')}[N]$ in (a,b,t) coordinates and accounting for the partial integration domains for the border points we obtain

$$C = \frac{2n+1}{2}\ln(2\pi) + \ln\left[\frac{\mu(a',b',t')}{\sqrt{\det \mathcal{H} f(a',b',t')}} + \frac{\mu(a'',b'',t'')}{\sqrt{\det \mathcal{H} f(a'',b'',t'')}}\right] - k\ln 2,$$

where $(a',b',t') = T_1^{-1}(x',u',s')$, $(a'',b'',t'') = T_1^{-1}(x'',u'',s'')$ and k is the number of border coordinates of (a',b',t') (or equivalently of (a'',b'',t'')). Note that $a' = b''$, $b' = a''$ and $t' = 1-t''$.

A.3 Proof of Theorem 4b (Type 1 Singularity)

Theorem 4b states that if $Y_D = Y$ for $N \geq N_0$, $Y_i > 0$ for $i = 1,\ldots,2^n$ and $Y \in S \setminus S'$, then $\ln \mathbb{I}[N, Y_D]$ (Eq. 13) is asymptotically equal to $NP(Y|w_{ML}) - \frac{2n-1}{2}\ln N + O(1)$ (Eq. 15). To prove this claim we first employ Lemma 6, which relates $\mathbb{I}[N,Y_D]$ with $\tilde{\mathbb{I}}[N,Y]$ (Eqs. 28 and 30) and Lemma 7, which relates $\tilde{\mathbb{I}}[N,Y]$ with $\mathbb{J}[N,Y]$ (Eqs. 31 and 32). Consequently, it remains to evaluate $\mathbb{J}[N,Y] = \max_{p_0 \in U_0} \mathbb{J}_{p_0}[N]$. For this task, one needs to examine the neighborhoods of arbitrary minimum points $p_0 \in U_0$ of f. However, for $Y \in S \setminus S'$ (singularity of type 1), the function f can not be approximated by quadratic form and Lemma 1 (Laplace Approximation) no longer applies. Instead we use Watanabe's method.

Let (a,b,t) be the parameterization of $Y \in S$ as described by the definition of S (Eq. 12) with $a_i = b_i$ for all $i \neq l,k$. Also, let $z'_{lk} = T_3^{-1}(Y)_{lk} = (1-(2t-1)^2)\cdot\frac{a_l-b_l}{2}\cdot\frac{a_k-b_k}{2}$. The zero set U_0 is given by

$$U_0 = \left\{ (x,u,s) \in U \;\middle|\; \begin{array}{l} x_i = a_i, \ \forall i = 1,\ldots,n,\ i \neq l,k, \\ x_l = ta_l + (1-t)b_l, \\ x_k = ta_k + (1-t)b_k; \\ u_i = 0, \ \forall i = 1,\ldots,n,\ i \neq l,k, \\ u_l, u_k, s, \text{ such that } (1-s^2)u_l u_k = z'_{lk} \end{array} \right\}. \tag{43}$$

Note that $z'_{lk} \neq 0$ and $u'_l, u'_k \neq 0$, $s' \neq \pm 1$ for $(x',u',s') \in U_0$, because $Y \notin S'$. The set U_0 is depicted in Figure 6.

We now apply the method of Watanabe, as described in Section 2, to evaluate the integrals $\mathbb{J}_{p_0}[N,Y]$ for $p_0 \in U_0$. We examine the form of the exponent function in $\mathbb{J}_{p_0}[N,Y]$, $\phi(x,u,s)$ which is equal to $\sum_I [z_I(x,u,s) - z'_I]^2$, in a small neighborhood of $p_0 = (x',u',s') \in U_0$. The coordinates of $z' = T_2(x',u',s')$ are $z'_i = x_i$ for all i, $z'_{lk} = (1-s'^2)u'_l u'_k$ and all other z'_I's are zero. Substituting z_I as a function of (x,u,s) into ϕ and translating the (x,u,s) coordinates so that (x',u',s') becomes the

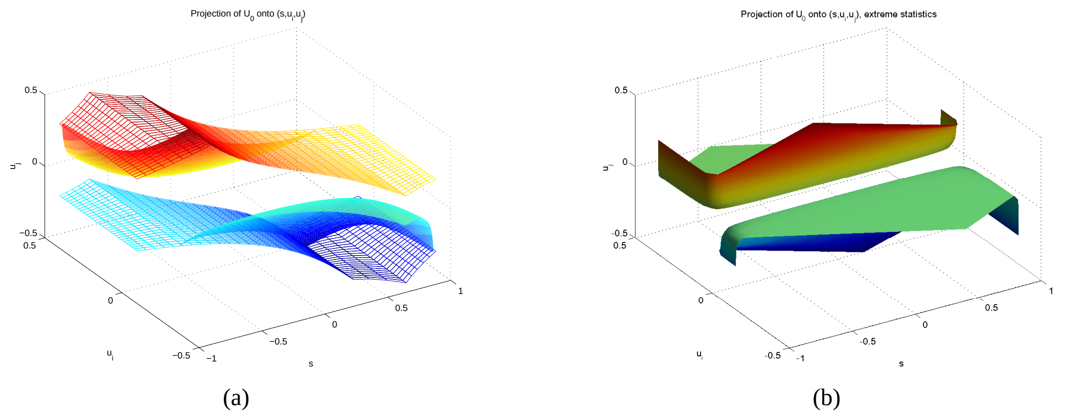


Figure 6: The projection of set U_0 onto (s,u_1,u_2) space. The zero set U_0 is defined by type 1 singularity statistics. (a) Illustration is for $x'_1 = 0.18$, $x'_2 = 0.28$, $z'_{12} = 0.0096$, that correspond to statistics Y generated by true distribution: $a_1 = 0.1$, $a_2 = 0.2$, $b_1 = 0.3$, $b_2 = 0.4$ and $t = 0.6$. Upper and lower bounds on u_1 are shown by mesh-grid. (b) Illustration of set U_0 for extreme (almost type 2) singular statistics of type 1 that is generated by $a_1 = 0.1$, $a_2 = 0.2$, $b_1 = 0.3$, $b_2 = 0.4$ and $t = 0.005$. The zero set is very close to the zero set for type 2 singularity statistics depicted in Figure 5(b).

origin, yields

$$\begin{aligned}
\phi(x,u,s) = \; & \sum_I \left[z_I(x'+x,u'+u,s'+s) - z'_I\right]^2 \\
= \; & \sum_i [z_i(x'+x,u'+u,s'+s) - z'_i]^2 \\
& + \left[z_{lk}(x'+x,u'+u,s'+s) - z'_{lk}\right]^2 \\
& + \sum_{i\neq l,k} \left[z_{il}(x'+x,u'+u,s'+s) - z'_{il}\right]^2 + \left[z_{ik}(x'+x,u'+u,s'+s) - z'_{ik}\right]^2 \\
& + \sum_{i,j\neq l,k} \left[z_{ij}(x'+x,u'+u,s'+s) - z'_{ij}\right]^2 + \ldots \\
= \; & \sum_{i=1}^n [(x'_i + x_i) - x'_i]^2 \\
& + \left[(1-(s'+s)^2)(u'_l+u_l)(u'_k+u_k) - (1-s'^2)u'_l u'_k\right]^2 \\
& + \sum_{i\neq l,k} \left[(1-(s'+s)^2)(u'_l+u_l)u_i - 0\right]^2 + \left[(1-(s'+s)^2)(u'_k+u_k)u_i - 0\right]^2 \\
& + \sum_{i,j\neq l,k} \left[(1-(s'+s)^2)u_i u_j - 0\right]^2 + \ldots \\
= \; & \sum_{i=1}^n x_i^2 \\
& + \left[-2s'u'_l u'_k s + (1-s'^2)u'_k u_l + (1-s'^2)u'_l u_k + \text{“smaller terms”}\right]^2 \\
& + \sum_{i\neq l,k} \left[(1-s'^2)u'_l u_i + \text{“smaller terms”}\right]^2 + \left[(1-s'^2)u'_k u_i + \ldots\right]^2 \\
& + \sum_{i,j\neq l,k} \left[(1-s'^2)u_i u_j + \text{“smaller terms”}\right]^2 + \ldots .
\end{aligned} \tag{44}$$

The phrase “smaller terms” and dots denotes higher order terms that include variables that are present in the explicit terms of the sum and can be discarded for sufficiently small (x,u,s). In particular, the term $z_{lk}(x,u,s)-z'_{lk}$ is rewritten via

$$\begin{aligned} z_{lk}(x'+x,u'+u,s'+s)-z'_{lk} = \quad & (1-(s+s')^2)(u_l+u'_l)(u_k+u'_k)-(1-s'^2)u'_lu'_k \\ = \quad & -(2s'+s)u'_lu'_ks+((1-s'^2)-2s's-s^2)(u'_k+u_k)u_l \\ & +((1-s'^2)-2s's-s^2)u'_lu_k. \end{aligned}$$

Consequently for $s'\neq 0$, sufficiently small ε and $s,u_l,u_k\in(-\varepsilon,\varepsilon)$ it follows that $C_1^- < -(2s'+s)u'_lu'_k < C_1^+$, $C_2^- < [(1-s'^2)-2s's-s^2][u'_k+u_k] < C_2^+$ and $C_3^- < [(1-s'^2)-2s's-s^2]u'_l < C_3^+$ for C_1^-, C_1^+, C_2^-, C_2^+, C_3^-, C_3^+ slightly smaller and larger than $C_1=-2s'u'_lu'_k$, $C_2=(1-s'^2)u'_k$, $C_3=(1-s'^2)u'_l$.

Consequently, in order to approximate the integral $\mathbb{J}_{p_0}[N]$ (Eq. 31) for $p_0=(x',u',s')$ with $s'\neq 0$, it remains to approximate the integral

$$\begin{aligned} &\tilde{\mathbb{J}}_1[N]=\int e^{-N\tilde{\phi}_1(x,u,s)}dxduds, \\ &\text{where}\ \ \tilde{\phi}_1(x,u,s)=\sum_i x_i^2+\left[\tilde{C}_1s+\tilde{C}_2u_l+\tilde{C}_3u_k\right]^2+\sum_{i\neq l,k}\tilde{c}_iu_i^2 \end{aligned} \tag{45}$$

and where $\tilde{C}_1$, $\tilde{C}_2$, $\tilde{C}_3$ and $\tilde{c}_i$ are non-zero constants.

Similar analysis of the principal part of $\phi(x,u,s)$ (Eq. 44) function can be applied for the neighborhoods of $p_0=(x',u',s')$ with $s'=0$. It reveals that in order to approximate $\mathbb{J}_{p_0}[N]$ for $p_0=(x',u',s')$ with $s'=0$ we should approximate the integral

$$\begin{aligned} &\tilde{\mathbb{J}}_2[N]=\int e^{-N\tilde{\phi}_2(x,u,s)}dxduds, \\ &\text{where}\ \ \tilde{\phi}_2(x,u,s)=\sum_i x_i^2+\left[\hat{C}_1s^2+\hat{C}_2u_l+\hat{C}_3u_k\right]^2+\sum_{i\neq l,k}\hat{c}_iu_i^2, \end{aligned} \tag{46}$$

and where $\hat{C}_1$, $\hat{C}_2$, $\hat{C}_3$ and $\hat{c}_i$ are non-zero constants that are slightly larger or smaller than $u'_lu'_k$, u'_k, u'_l and $u'^2_l+u'^2_k$.

From Eq. 45, by changing the coordinates to $v=\tilde{C}_1s+\tilde{C}_2u_l+\tilde{C}_3u_k$, we obtain that in the neighborhoods of the points in U_0 with $s'\neq 0$, that f can be described by quadratic form in $2n-1$ variables, so their contribution to $\mathbb{J}[N,Y]$ is $cN^{\frac{2n-1}{2}}$.

The analysis of neighborhoods of points in U_0 with $s'=0$ is harder. Integrating out x_i and u_i variables yields $N^{-\frac{2n-2}{2}}$ multiplicative factor to the asymptotic approximation of $\tilde{\mathbb{J}}_2[N]$, leaving us to compute of the contribution of $\int e^{-N[\hat{C}_1s^2+\hat{C}_2u_i+\hat{C}_3u_j]^2}ds\,du_i\,du_j$. The changes of variables $t=(\tilde{C}_2u_i+\tilde{C}_3u_j)/\tilde{C}_1$ transforms the remaining part of $\tilde{\mathbb{J}}_2[N]$ to

$$\tilde{\mathbb{J}}_3[N]=\int_{-\varepsilon_1}^{+\varepsilon_1}\int_{-\varepsilon_2}^{+\varepsilon_2}e^{-N(s^2+t)^2}dsdt.$$

The zero set of the exponent function is a one-dimensional curve $t=-s^2$, so we expect $\tilde{\mathbb{J}}_3[N]$ be at least $cN^{-\frac{1}{2}}$, as verified below.

Watanabe's method for $\tilde{\mathbb{J}}_3[N]$ calls for the analysis of the poles of the function

$$J(\lambda)=\int_{(-1,1)^2}(s^2+t)^{2\lambda}dsdt.$$

Here, we transform the original integration range into $(-1,1)$ by rescaling, introducing only constant multipliers to the integral. The analysis of the poles of $J(\lambda)$ is in the spirit of example shown in Section 2. We present this analysis completely to demonstrate a number of important subtle points in the evaluation of integrals by resolution of singularities. E.g., we can not use the binomial formula for expanding $(s^2+t)^{2\lambda}$, since λ is not necessarily an integer.

The integral $J(\lambda)$ is symmetric relative to s, so we consider only $s>0$ for the evaluation of its poles. Changing the coordinates via $t=\pm t^2$ we obtain

$$\frac{1}{2}J(\lambda)=\int_{-1}^{1}\int_{0}^{1}(s^2+t)^{2\lambda}dsdt=\int_{0}^{1}\int_{0}^{1}2t(s^2+t^2)^{2\lambda}dsdt+\int_{0}^{1}\int_{0}^{1}2t(s^2-t^2)^{2\lambda}dsdt.$$

The first integral is easy to evaluate by standard substitutions $s=ts$ for $0<s<t<1$ and $t=st$ for $0<t<s<1$. Thus, the first integral contributes a pole at $\lambda=-\frac{3}{4}$ with multiplicity 1. The second integral, however, can not be evaluated in this way, since, the substitution $s=ts$ for $0<s<t<1$ gives the integral $\int_0^1\int_0^1 2t^{4\lambda+2}(s^2-1)^{2\lambda}dsdt$, where the term (s^2-1) is not bounded away from zero on $(0,1)$ and thus can not be ignored when identifying the poles.

To overcome this difficulty let $v=s+t$ and $u=s-t$, yielding

$$\int_{0}^{1}\int_{0}^{1}2t(s^2-t^2)^{2\lambda}dsdt=\frac{1}{2}\int_{0}^{2}\int_{\max(-v,v-2)}^{\min(v,2-v)}(v-u)u^{2\lambda}v^{2\lambda}dudv$$

and

$$\frac{1}{2}\int_{0}^{1}\int_{-v}^{v}(v-u)u^{2\lambda}v^{2\lambda}dudv<\int_{0}^{1}\int_{0}^{1}2t(s^2-t^2)^{2\lambda}dsdt<\frac{1}{2}\int_{0}^{2}\int_{-v}^{v}(v-u)u^{2\lambda}v^{2\lambda}dudv. \tag{47}$$

Computing the lower bound in Eq. 47, we obtain

$$\begin{aligned}\tfrac{1}{2}\textstyle\int_0^1\int_{-v}^{v}(v-u)u^{2\lambda}v^{2\lambda}dudv &= \tfrac{1}{2}\textstyle\int_0^1\left[v^{2\lambda+1}\frac{1}{2\lambda+1}u^{2\lambda+1}-v^{2\lambda}\frac{1}{2\lambda+2}u^{2\lambda+2}\Big|_{-v}^{v}\right]dv\\ &= \tfrac{1}{2}\textstyle\int_0^1\frac{2}{2\lambda+1}v^{4\lambda+2}dv=\frac{1}{(2\lambda+1)(4\lambda+3)}.\end{aligned}$$

The upper limit is correspondingly $\frac{2^{4\lambda+3}}{(2\lambda+1)(4\lambda+3)}$. Hence, the largest pole of $J(\lambda)$ is $\lambda=-\frac{1}{2}$, with multiplicity $m=1$ and the overall contribution of the neighborhoods of points p_0 with $s'=0$ to $\mathbb{J}[N,Y]$ is again $cN^{-\frac{2n-1}{2}}$, and it is the same as for points p_0 for which $s'\neq 0$. The point (x',u',s') need not be an internal point of U. Such border points have a smaller domain of integration than an internal point, therefore they do not contribute more to $\mathbb{J}[N,Y]$ than internal points. ■

It is interesting to compare Figure 6b and Figure 5, to see that as a point $Y\in S\setminus S$ approaches $Y'\in S'$, the zero set for Y depicted by Figure 6 approaches the zero set for Y' depicted in Figure 5.

A.4 Proof of Theorem 4c (Type 2 Singularity)

The outline of the proof of Theorem 4c is presented in Section 5.3 including the specification of the zero set U_0 and five principal cases $C1$-$C5$ that correspond to different locations of extremum points $(x',u',s')\in U_0$. Recall that we are interested in the evaluation of the contribution of the neighborhood of each of the points of types $C1$-$C5$ to the integral $\mathbb{J}[N,Y]$ (Eq. 31). The maximal contribution determine, according to Lemmas 6 and 7, the asymptotic behavior of the integral $\mathbb{I}[N,Y_D]$ (Eq. 11) of interest. We now treat these cases one by one.

Case C1: $(x',u',s')\in U_{0j}\setminus\cup_{i\neq j}U_{0i}$ for some j. Each such point (x',u',s') satisfies $u'_i=0$, for all $i=1,\ldots,n$, $i\neq j$; $u'_j\neq 0$; $s'\neq\pm 1$; $z'_i=x'_i$; and $z'_{ij..k}=0$. Using the approach of Watanabe we analyze the form of the exponent function ϕ of integrand of $\mathbb{J}_{p_0}[N]$ near the minimum point $p_0 = (x', u', s')$. Centering (x, u, s) around (x', u', s') we obtain

$$\begin{aligned}
\phi(x,u,s) = \quad & \textstyle\sum_I [z_I(x'+x, x'+u, s'+s) - z'_I]^2 \\
= \quad & \textstyle\sum_i [z_i(x'+x, u'+u, s'+s) - z'_i]^2 + \sum_{i\neq j}[z_{ij}(x'+x, u'+u, s'+s) - z'_{ij}]^2 \\
& \textstyle + \sum_{i,k\neq j}[z_{ik}(x'+x, u'+u, s'+s) - z'_{ik}]^2 + \text{“higher order terms''} \\
= \quad & \textstyle\sum_i [(x'_i + x_i) - x'_i]^2 + \sum_{i\neq j}\left[(1-(s'+s)^2)(u'_j + u_j)u_i - 0\right]^2 \\
& \textstyle + \sum_{i,k\neq j}\left[(1-(s'+s)^2)u_i u_k - 0\right]^2 + \text{“higher order terms''} \\
= \quad & \textstyle\sum_i x_i^2 + \sum_{i\neq j}\left[(1-s'^2)u'_j u_i + \text{“smaller terms''}\right]^2 \\
& \textstyle + \sum_{i,k\neq j}\left[(1-s'^2)u_i u_k - (s+2s')s u_i u_k\right]^2 + \text{“higher order terms''}.
\end{aligned}$$

Since, $u'_j \neq 0$ and $s \neq \pm 1$, the principal part of ϕ, that bounds ϕ within a multiplicative constant, is

$$\tilde{\phi}(x,u,s) = \sum_{i=1,\ldots,n} x_i^2 \; + \sum_{i=1,\ldots,n;\, i\neq j} u_i^2.$$

Hence, $\mathbb{J}_{p_0}[N]$ is $cN^{-\frac{2n-1}{2}}$. One should have expected this result because the zero set $U_{0,j}$ is a 2-dimensional surface, yielding a dimensionality drop of 2 due to two locally redundant parameters.

Case C2: $(x', u', s') = \bigcap_j U_{0j}$. This case is analyzed in Section 5.3.

Case C3: $(x', u', s') \in U_{0-} \cup U_{0+} \setminus \cup_j \bar{U}_{0j}$. Each such point (x', u', s') satisfies $u'_j \neq 0$ for all $j = 1, \ldots, n$ and $s' = \pm 1$. We have

$$\begin{aligned}
\phi(x,u,s) = \quad & \textstyle\sum_I [z_I(x'+x, u'+u, s'+s) - z'_I]^2 \\
= \quad & \textstyle\sum_i [z_i(x'+x, u'+u, s'+s) - z'_i]^2 + \sum_{i,j}[z_{ij}(x'+x, u'+u, s'+s) - z'_{ij}]^2 \\
& \textstyle + \sum_{i,j,k}[z_{ijk}(x'+x, u'+u, s'+s) - z'_{ijk}]^2 + \ldots \\
= \quad & \textstyle\sum_i [(x'_i + x_i) - x'_i]^2 + \sum_{i,j}\left[(1-(s'+s)^2)(u'_i + u_i)(u'_j + u_j) - 0\right]^2 \\
& \textstyle + \sum_{i,j,k}\left[-2(s'+s)(1-(s'+s)^2)(u'_i + u_i)(u'_j + u_j)(u'_k + u_k) - 0\right]^2 + \ldots \\
= \quad & \textstyle\sum_i x_i^2 + \sum_{i,j}\left[-2s' u'_i u'_j s + \text{“smaller terms''}\right]^2 \\
& \textstyle + \sum_{i,j,k}\left[4u'_i u'_j u'_k s + \text{“smaller terms''}\right]^2 + \text{“higher order terms''}.
\end{aligned}$$

So, the principal part of ϕ is of the form $\sum_i x_i^2 + s^2$. The fact that integration range for s is one sided, i.e. $s > 0$ (or $s < 0$) changes the integral $\mathbb{J}_{p_0}[N]$ only by a constant multiply $(1/2)$ relatively to the “full” neighborhood. Thus the contribution of this region to $\mathbb{J}[N, Y]$ is $cN^{-\frac{n+1}{2}}$.

Case C4: $(x',u',s') \in \bigcup_j \left[U_{0-} \cup U_{0+} \cap \bar{U}_{0j} \setminus \cap_{i\neq j} \bar{U}_{0i}\right]$, for some j. Each such point (x',u',s') satisfies $u'_j \neq 0$ for some j; $u'_i = 0$ for all $i \neq j$; and $s' = \pm 1$. We have

$$\begin{aligned}
\phi(x,u,s) = \;& \textstyle\sum_I [z_I(x'+x,u'+u,s'+s) - z'_I]^2 \\
= \;& \textstyle\sum_i [z_i(x'+x,u'+u,s'+s) - z'_i]^2 + \sum_{i\neq j}[z_{ij}(x'+x,u'+u,s'+s) - z'_{ij}]^2 \\
& + \textstyle\sum_{i,k\neq j}[z_{ik}(x'+x,u'+u,s'+s) - z'_{ik}]^2 + \text{“higher order terms''} \\
= \;& \textstyle\sum_i [(x'_i + x_i) - x'_i]^2 + \sum_{i\neq j}\left[(1-(s'+s)^2)(u'_j+u_j)u_i - 0\right]^2 \\
& + \textstyle\sum_{i,k\neq j}\left[(1-(s'+s)^2)u_iu_k - 0\right]^2 + \text{“higher order terms''} \\
= \;& \textstyle\sum_i x_i^2 + \sum_{i\neq j}\left[\mp 2su'_ju_i \mp 2su_ju_i - s^2u'_ju_i - s^2u_ju_i\right]^2 \\
& + \textstyle\sum_{i,k\neq j}\left[\mp 2su_iu_k - s^2u_iu_k\right]^2 + \text{“higher order terms''} \\
\approx \;& \textstyle\sum_i x_i^2 + s^2 \sum_{i\neq j} u_i^2.
\end{aligned} \tag{48}$$

Integrating out the $\sum_i x_i^2$ terms from $\mathbb{J}_{p_0}[N]$, we see that they contribute factor of $N^{-\frac{n}{2}}$ to $\mathbb{J}_{p_0}[N]$. So, we are left with analysis of the poles of

$$J(\lambda) = \int_{W_\varepsilon} s^{2\lambda}\left(\sum_{i=1}^{n-1} u_i^2\right)^{\lambda} dsdu.$$

The standard change of variables to $u_i = u_1u_i$ for $i = 2,\ldots,n-1$ gives

$$J(\lambda) = c\int_{(0,1)^n} s^{2\lambda}u_1^{2\lambda+n-2}(1+\sum_{i=2}^{n-1}u_i^2)^{\lambda}dsdu.$$

Thus the largest pole of $J(\lambda)$ (for $n > 2$) is $\lambda = -\frac{1}{2}$ with multiplicity $m = 1$ and the contribution of the neighborhood of this (x',u',s') is $cN^{-\frac{n+1}{2}}$.

Case C5: $(x',u',s') \in (U_{0-} \cup U_{0+}) \bigcap_j \bar{U}_{0j}$. Each such point (x',u',s') satisfies $u'_i = 0$ for all $i = 1,\ldots,n$ and $s' = \pm 1$. This is the deepest singularity, the crossing of all (except one) zero planes of U_0. We have

$$\begin{aligned}
\phi(x,u,s) = \;& \textstyle\sum_I [z_I(x'+x,u'+u,s'+s) - z'_I]^2 \\
= \;& \textstyle\sum_i [z_i(x'+x,u'+u,s'+s) - z'_i]^2 + \sum_{i,j}[z_{ij}(x'+x,u'+u,s'+s) - z'_{ij}]^2 \\
& + \textstyle\sum_{i,j,k}[z_{ijk}(x'+x,u'+u,s'+s) - z'_{ijk}]^2 + \ldots \\
= \;& \textstyle\sum_i [(x'_i + x_i) - x'_i]^2 + \sum_{i,j}\left[(1-(s'+s)^2)u_iu_j - 0\right]^2 \\
& + \textstyle\sum_{i,j,k}\left[-2(s'+s)(1-(s'+s)^2)u_iu_ju_k - 0\right]^2 + \text{“higher order terms''} \\
= \;& \textstyle\sum_i x_i^2 + \sum_{i,j}\left[\mp 2su_iu_j - s^2u_iu_j\right]^2 \\
& + \textstyle\sum_{i,j,k}[4su_iu_ju_k + \text{“smaller terms''}]^2 + \text{“higher order terms''} \\
\approx \;& \textstyle\sum_i x_i^2 + s^2\sum_{i,j}u_i^2u_j^2.
\end{aligned} \tag{49}$$

The higher order terms are bounded by some $s^2u_i^2u_j^2$ term, because of the special form of $p_i(s)$ term in $z_{12\ldots i}$ (Eq. 27). I.e., the function $p_i(s'+s) = 1/2(1-(s'+s)^2)[(1-(s'+s))^{i-1} - (-1)^{i-1}(1+(s'+$

$s))^{i-1}]$ can be rewritten around $s = \pm 1$ as $p_i(s'+s) = s \cdot 1/2(2s'+s)[(1-(s'+s))^{i-1} - (-1)^{i-1}(1+(s'+s))^{i-1}]$. Thus, any high-order term $z^2_{ij\ldots r}(x'+x,u,\pm 1+s)$ is of form

$$z^2_{ij\ldots k}(x'+x,u,\pm 1+s) = s^2 u_i^2 u_j^2 \ldots u_k^2 \cdot \tilde{p}(s),$$

where $\tilde{p}(s) = 1/4(2s'+s)^2[(1-(s'+s))^{r-1} - (-1)^{r-1}(1+(s'+s))^{r-1}]^2$ and where r is the size of index set $\{ij\ldots k\}$. Consequently, this term is bounded by $s^2 u_i^2 u_j^2$ for s and u small enough.

The $\sum_i x_i^2$ terms contribute $N^{-\frac{n}{2}}$ multiplicative factor to $\mathbb{J}_{p_0}[N]$, so we should only analyze the poles of

$$J(\lambda) = \int_{(0,1)^{n+1}} s^{2\lambda} \left(\sum_{l,k} u_l^2 u_k^2 \right)^{\lambda} ds du.$$

The analysis is similar to the one presented in Section 2, but with additional variable s. Thus the largest pole of $J(\lambda)$ this time is $\lambda = -\frac{1}{2}$ and not $\lambda = -n/4$. The multiplicity of the pole $\lambda = -\frac{1}{2}$ is one and so the contribution of the neighborhoods of $(x',0,\pm 1)$ is $cN^{-\frac{n+1}{2}}$. This analysis is incorrect for $n = 2$ because then the sum $\sum_{l,k} u_l^2 u_k^2$ contains only one term and this results in increasing the multiplicity of the pole $\lambda = -1/2$.

The interesting fact about the last two cases is that in the neighborhood of U_{0-} and U_{0+} the growth of the function ϕ is dominated by s^2 and thus the multiplicity of the maximal pole of $J(\lambda)$ is always one and the $\ln\ln N$ terms do not appear in the approximation of $\ln \mathbb{J}_{p_0}[N]$. This changes in the case $n = 2$, where the dimensionalities of U_{0-} and U_{0+} are the same as of U_{0j}'s, as explicated in the next section.

Summary of Proof of Theorem 4 for type 2 *singularity,* $Y \in S'$: Among the possible cases $C1$-$C5$ the largest contribution to the $\mathbb{J}[N,Y]$ comes from points with $s' = \pm 1$. Note that various border points of U_0 that we do not consider in the above analysis do not contribute more than the corresponding internal points because their domain of integration is smaller. Thus, $\ln \mathbb{J}[N,Y] = -\frac{n+1}{2} \ln N + O(1)$ and due to Lemmas 6 and 7, $\ln \mathbb{I}[N,Y_D] = NP(Y|w_{ML}) - \frac{n+1}{2} \ln N + O(1)$ as claimed. ■

A.5 Proof of Claims (d,e) of Theorem 4 (Case $n = 2$)

Claims (d,e) of Theorem 4 state that if $n = 2$, $Y_D = Y$ for $N \geq N_0$ and $Y_i > 0$ for $i = 1,\ldots,2^n$, $\ln \mathbb{I}[N,Y_D]$ (Eq. 13) is asymptotically equal to $NP(Y|w_{ML}) - \frac{3}{2} \ln N + O(1)$ (Eq. 17) for $Y \notin S'$ and asymptotically equal to $NP(Y|w_{ML}) - \frac{3}{2} \ln N + 2 \ln\ln N + O(1)$ (Eq. 18) for $Y \in S'$. Similar to the proofs of Claims (b,c), we first employ Lemma 6, which relates $\mathbb{I}[N,Y_D]$ with $\tilde{\mathbb{I}}[N,Y]$ (Eqs. 28 and 30) and Lemma 7, which relates $\tilde{\mathbb{I}}[N,Y]$ with $\mathbb{J}[N,Y]$ (Eqs. 31 and 32). Consequently, it remains to evaluate $\mathbb{J}[N,Y] = \max_{p_0 \in U_0} \mathbb{J}_{p_0}[N]$. For this task, one needs examine the neighborhoods of arbitrary minimum points $p_0 \in U_0$ of the function f. From the definition of Υ, Υ_0 and S (Section 4) it follows that $S = \Upsilon_0 = \Upsilon$ for $n = 2$. Note that there is no regular points in this case. We now modify the proofs of type 1 and type 2 singularities to fit to the case $n = 2$.

Type 1 *singularity:* The zero set U_0 is the same set as described by Eq. 43 with $l = 1$ and $k = 2$. The analysis of the form of the exponent function ϕ of the integrand of $\mathbb{J}_{p_0}[N]$ gives Eqs. 45 and 46 without the $\sum_{l \neq i,j} c_l u_l^2$ terms. Thus, by the same analysis, the contribution of these regions to the integral $\mathbb{J}[N,Y]$ is $cN^{-\frac{3}{2}}$ and application of Lemmas 6 and 7 concludes the proof.

Type 2 *singularity:* The zero set $U_0 = \overline{U}_{0-} \cup \overline{U}_{0+} \cup \overline{U}_{01} \cup \overline{U}_{02}$ is the same set as described by Eqs. 34 and 35. Now, however, $\overline{U}_{0-}$, $\overline{U}_{0+}$, $\overline{U}_{01}$ and $\overline{U}_{02}$ are of the same dimension, namely, two. This fact changes the asymptotic approximation.

Consider the cases $C1$-$C5$ one by one. There is no change in cases $C1$ and $C3$ where the point (x', u', s') lies on the proper two dimensional surfaces U_{01}, U_{02} or U_{0-}, U_{0+}. Here, the function ϕ can be approximated by 3 variables, resulting in the contribution $cN^{-3/2}$ of these regions to $\mathbb{J}[N,Y]$.

The more complex situation is in $C2$, $C4$ and $C5$ cases, where zero planes of the same dimension meet. Generally, the intersection points of zero surfaces of the same dimension are expected to give rise to a $\ln\ln N$ term. While this is not always a case, e.g., see example in Section 2, the $\ln\ln N$ term does appear now. We have:

C2: The principal part of ϕ is $x_1^2 + x_2^2 + u_1^2 u_2^2$, as specified by Eq. 37. Integrating out the x_i^2 terms we obtain through the analysis of the poles of $J(\lambda) = \int u_1^{2\lambda} u_2^{2\lambda} du_1 du_2$ that the largest pole of $J(\lambda)$ is $\lambda = -1/2$ with multiplicity $m = 2$. Thus the contribution of this region to $\mathbb{J}[N,Y]$ is $cN^{-3/2} \ln N$.

C4: The principal part of ϕ is $x_1^2 + x_2^2 + s^2 u_2^2$ or $x_1^2 + x_2^2 + s^2 u_1^2$ (see Eq. 48). Similarly to the case $C2$, the contribution of this region to $\mathbb{J}[N,Y]$ is $cN^{-3/2} \ln N$.

C5: Here, the principal part of ϕ is $x_1^2 + x_2^2 + s^2 u_1^2 u_2^2$ (see Eq. 49). Once again, we integrate out the x_i variables and analyze the poles of $J(\lambda) = \int s^{2\lambda} u_1^{2\lambda} u_2^{2\lambda} ds du_1 du_2$. The largest pole is $\lambda = -1/2$ with multiplicity $m = 3$, and thus the contribution of this region to $\mathbb{J}[N,Y]$, including the factors from integrating out the x_i's, is $cN^{-3/2} \ln^2 N$.

Summarizing the contributions of the neighborhoods of various critical points for $Y \in S'$, we see that $\mathbb{J}[N,Y] \sim cN^{-3/2} \ln^2 N$ and, consequently, $\ln \mathbb{I}[N,Y] = N f_Y - \frac{3}{2} \ln N + 2 \ln\ln N + O(1)$. ■

A.6 Proof of Theorem 4f (Case $n = 1$)

Theorem 4f states that if $n = 1$, $Y_D = Y$ for $N \geq N_0$ and $Y_1, Y_2 > 0$, then $\ln \mathbb{I}[N, Y_D]$ (Eq. 13) is asymptotically equal to $NP(Y|w_{ML}) - \frac{1}{2} \ln N + O(1)$ (Eq. 19). Once again, we first employ Lemma 6, which relates $\mathbb{I}[N, Y_D]$ with $\tilde{\mathbb{I}}[N,Y]$ (Eqs. 28 and 30) and Lemma 7, which relates $\tilde{\mathbb{I}}[N,Y]$ with $\mathbb{J}[N,Y]$ (Eqs. 31 and 32). Consequently, it remains to evaluate $\mathbb{J}[N,Y] = \max_{p_0 \in U_0} \mathbb{J}_{p_0}[N]$. For this task, one needs examine $\mathbb{J}_{p_0}[N]$ in the neighborhoods of arbitrary minimum points $p_0 \in U_0$ of the function f.

From the definitions of Υ, Υ_0 and S', for $n = 1$, there is no distinction between different type of statistics and $\Upsilon = \Upsilon_0 = S'$. Moreover, according to Theorem 2 the asymptotic form of the integral $J_{p_0}[N] = \int_{U_\varepsilon} e^{-N(z_1(x,u,s) - z_1')^2} dx\, du\, ds$ is determined by the poles of $J(\lambda) = \int_{U_\varepsilon} (z_1(x,u,s) - z_1')^{2\lambda} dx\, du\, ds$, where, in this case, $z_1(x,u,s) - z_1' = x_1^2$. Once again, contributions of points $p_0 \in U_0$ lying on the boundary of U can be ignored, since their domains of integration are smaller than domains of integration of the corresponding internal points. Thus, the largest pole of $J(\lambda)$ is $\lambda = -1/2$ with multiplicity $m = 1$ and $\ln I[N, Y_D]$ is asymptotically equal to $N f_Y - \frac{1}{2} \ln N + O(1)$. ■

We can also compute the integral $\mathbb{I}[N, Y_D]$ (Eq. 11) directly for $n = 1$ and $Y_D = Y$. It is

$$\mathbb{I}[N,Y] = \int_{(0,1)^3} e^{N(Y_0 \ln[at + b(1-t)] + Y_1 \ln[(1-a)t + (1-b)(1-t)])} \mu(a,b,t)\, da\, db\, dt$$

where $Y_1 = 1 - Y_0$. Ignoring the density $\mu(a,b,t)$ by using the assumption of bounded density ($A1$) and changing the variables to $x = at + b(1-t)$, we rewrite $\mathbb{I}[N,Y]$ is asymptotically equivalent form

$$\tilde{\mathbb{I}}[N,Y] = \int_0^1 \int_0^1 \frac{1}{b-a} \int_a^b e^{N(Y_0 \ln x + Y_1 \ln[1-x])} dx\, da\, db.$$

Consider now

$$\mathbb{I}_1[N,Y] = \int_a^b e^{N(Y_0 \ln x + Y_1 \ln(1-x))} dx$$

for some $0 \le a < b \le 1$ (the case $b > a$ is symmetric). This is the integral of the beta distribution with $\alpha = NY_0 + 1$ and $\beta = NY_1 + 1$ (DeGroot, 1970, page 40). Let $f(x) = Y_0 \ln x + Y_1 \ln(1-x)$. The maximum of the integrand function $f(x)$ on $[0,1]$ is achieved at $x_0 = Y_0$ and it is $e^{Nf(Y_0)}$. There are three cases to consider according to the location of x_0 relative to (a,b).

1. *Internal point,* $x_0 = Y_0 \in (a,b)$. In this case

$$\begin{aligned} f(Y_0+x) = {} & f(Y_0) + Y_0 \ln\left(1 + \tfrac{x}{Y_0}\right) + (1-Y_0)\ln\left(1 - \tfrac{x}{1-Y_0}\right) \\ = {} & f(Y_0) + Y_0\left(\tfrac{x}{Y_0} - \tfrac{x^2}{2Y_0^2} + O(x^3)\right) + (1-Y_0)\left(\tfrac{-x}{1-Y_0} - \tfrac{x^2}{2(1-Y_0)^2} + O(x^3)\right) \\ = {} & f(Y_0) - \tfrac{1}{2Y_0(1-Y_0)} x^2 + O(x^3). \end{aligned}$$

   Thus, in the small neighborhood of x_0, f can be approximated by quadratic form and the classic Laplace approximation (Lemma 1) can be applied yielding $\mathbb{I}_1[N,Y] \sim c_1 e^{Nf_Y} N^{-1/2}$. Moreover, since $\mathbb{I}_1[N,Y]$ and $e^{Nf(Y_0)}$ are continuous functions of N and $x_0 = Y_0$, uniform asymptotic bounds on $\mathbb{I}_1[N,Y]$ exists for all x_0 in a proper closed subset of (a,b) as $N \to \infty$. I.e., the integral $\mathbb{I}_1[N,Y]$ is bounded within a constant multiplies of $e^{Nf_Y} N^{-\frac{1}{2}}$ and these constants are independent of x_0 and N for all $x_0 \in [a+\varepsilon, b-\varepsilon]$ and $N \ge 1$. Note that the above approximation of f is only valid for $Y_0 \ne 0,1$ (Assumption $A2$). Otherwise, the approximation of f is non-quadratic.

2. *Border point,* $x_0 = Y_0 \in \{a,b\}$. The expansion for $f(Y_0+x)$ is the same, but the integration is performed only on the half of the interval, which results in half the constant factor to the final approximation compared with the previous case.

3. *Maximum of f is outside of* $[a,b]$. Let m denote the maximum of $e^{f(x)}$ on $[a,b]$, i.e., $m = \max_{x\in[a,b]} e^{f(x)}$. We have $\mathbb{I}_1[N,Y] \le (b-a)m^N < c_3 e^{Nf_Y} N^{-1/2}$, for some appropriate constant c_3.

The above analysis shows that $\mathbb{I}_1[N,Y] < c_{upp} e^{Nf_Y} N^{-1/2}$ for some constant c_{upp} for all a and b. Furthermore, $\mathbb{I}[N,Y] > c_{low} e^{Nf_Y} N^{-1/2}$ for some $c_{low} > 0$ for $(a,b) \in \{(a,b) | a < Y_0, b > Y_0, b-a > 2\varepsilon > 0\}$. Since the later region has a non-zero Lebesgue measure, it follows that $\mathbb{I}[N,Y] \sim c e^{Nf_Y} N^{-1/2}$ and $\ln \mathbb{I}[N,Y] = Nf_Y - \frac{1}{2}\ln N + O(1)$.

Appendix B. Proof of Theorem 5

Theorem 5 states the asymptotic approximation for the marginal likelihood given a degenerate binary naive Bayesian model M that has m missing links. In order to prove this theorem we examine the log-likelihood function of the degenerate model and decompose it into a degenerate part and a naive Bayesian part. These parts define two probability functions that are independent and the marginal likelihood of data is computed relevant to each one of them. Combining the results gives Theorem 5.

Let ψ be the log-likelihood function of the marginal likelihood integral (Eq. 20) for the degenerate binary naive Bayesian network described in Theorem 5. We have

$$\begin{aligned}
\tfrac{1}{N}\psi(a,b,t,c) = {} & \textstyle\sum_x Y_x \ln\theta_x(\omega) \\
= {} & \textstyle\sum_x Y_x \left[\ln\theta_{(x_1,\ldots,x_{n-m})}(a,b,t) + \sum_{i=n-m+1}^{n}(x_i \ln c_i + (1-x_i)\ln(1-c_i))\right] \\
= {} & \textstyle\sum_{(x_1,\ldots,x_{n-m})} \left[\ln\theta_{(x_1,\ldots,x_{n-m})}(a,b,t)\cdot\sum_{(x_{n-m+1},\ldots,x_n)} Y_x\right] \\
& \textstyle + \sum_{i=n-m+1}^{n}(\sum_x Y_x x_i \ln c_i + \sum_x Y_x(1-x_i)\ln(1-c_i)) \\
= {} & \textstyle\sum_{(x_1,\ldots,x_{n-m})} Y_{(x_1,\ldots,x_{n-m})}\ln\theta_{(x_1,\ldots,x_{n-m})}(a,b,t) \\
& \textstyle + \sum_{i=n-m+1}^{n}(Y_i \ln c_i + (1-Y_i)\ln(1-c_i))
\end{aligned}$$

where $(x_1,\ldots,x_k)$ are binary vectors of length k, $Y_{(x_1,\ldots,x_{n-m})} = \sum_{(x_{n-m+1},\ldots,x_n)} Y_{(x_1,\ldots,x_n)}$ and $Y_i = \sum_{(x_1,\ldots,x_{i-1},1,x_{i+1},\ldots,x_n)} Y_x$. The new statistics $Y_{(x_1,\ldots,x_{n-m})}$ and Y_i's are positive, because Y is positive ($A2$). Using the assumptions of bounded density ($A1$) and stable statistics ($A3$), the marginal likelihood integral $\mathbb{I}[N,Y]$ (Eq. 20) can be rewritten as

$$\mathbb{I}[N,Y_D] \sim \hat{\mathbb{I}}[N,Y] = \left[\prod_{i=n-m+1}^{n}\int_0^1 c_i^{NY_i}(1-c_i)^{N(1-Y_i)}dc_i\right]\int_{(0,1)^{2n-2m+1}} e^{N\sum_{\tilde{x}} Y_{\tilde{x}}\ln\theta_{\tilde{x}}(\omega)}d\omega. \tag{50}$$

where $\tilde{x} = (x_1,\ldots,x_{n-m})$. The first m integrals are integrals over the beta distribution (DeGroot, 1970, page 40) and

$$\int_0^1 c_i^{NY_i}(1-c_i)^{N(1-Y_i)}dc_i = \frac{\Gamma(NY_i+1)\Gamma(N(1-Y_i)+1)}{\Gamma(N+2)}$$

The asymptotic behavior of Gamma function is well understood and it is described by Stirling formula, $\Gamma(z) = e^{-z}z^{z-\frac{1}{2}}\sqrt{2\pi}\left[1+O(z^{-1})\right]$ (Murray, 1984, page 38), and thus $\ln\Gamma(z) = -z+(z-\frac{1}{2})\ln z + O(1)$. Using the equality $\ln(YN+1) = \ln YN + O(1)$, we obtain

$$\begin{aligned}
&\ln\tfrac{\Gamma(NY_i+1)\Gamma(N(1-Y_i)+1)}{\Gamma(N+2)} \\
&\quad = (NY_i+\tfrac{1}{2})\ln(NY_i+1) + (N(1-Y_i)+\tfrac{1}{2})\ln(N(1-Y_i)+1) - (N+\tfrac{3}{2})\ln(N+2) + O(1) \\
&\quad = (NY_i+\tfrac{1}{2})\ln NY_i + (N(1-Y_i)+\tfrac{1}{2})\ln N(1-Y_i) - (N+\tfrac{3}{2})\ln N + O(1) \\
&\quad = -\tfrac{1}{2}\ln N + N\,(Y_i\ln Y_i + (1-Y_i)\ln(1-Y_i)) + O(1).
\end{aligned}$$

Hence, the contribution of the first m integrals to $\ln\hat{\mathbb{I}}[N,Y]$ is $N\ln p(Y_{n-m+1},\ldots,Y_n|c_{ML}) - \frac{m}{2}\ln N$. The second integral in Eq. 50 is exactly of the type analyzed in Theorem 4, and the theorem follows by summing up the contributions of these two parts. ■

References

Shreeram S. Abhyankar. *Algebraic Geometry for Scientists and Engineers*. Number 35 in Mathematical Surveys and Monographs. American Mathematical Society, 1990.

Hirotugu Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, December 1974.

M.F. Atiyah. Resolution of singularities and division of distributions. *Communications on Pure and Applied Mathematics*, 13:145–150, 1970.

Peter Cheeseman and John Stutz. Bayesian classification (AutoClass): Theory and results. In U. Fayyad, G. Piatesky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 153–180. AAAI Press, 1995.

Gregory F. Cooper and Edward Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347, October 1992.

Morris H. DeGroot. *Optimal Statistical Decisions*. McGraw-Hill Book Company, 1970.

Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29(2-3):131–163, 1997.

Dan Geiger, David Heckerman, Henry King, and Christopher Meek. Stratified exponential families: Graphical models and model selection. *Annals of Statistics*, 29(2):505–529, 2001.

Dan Geiger, David Heckerman, and Christopher Meek. Asymptotic model selection for directed networks with hidden variables. In Eric Horvitz and Finn Jensen, editors, *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, pages 283–290. Morgan Kaufmann Publishers, Inc., 1996.

Dominique Haughton. On the choice of a model to fit data from an exponential family. *Annals of Statistics*, 16(1):342–355, 1988.

David Heckerman, Dan Geiger, and David M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, 1995.

Heisuke Hironaka. Resolution of singularities of an algebraic variety over a field of characteristic zero. *Annals of Mathematics*, 7(1,2):109–326, 1964.

Christine Keribin. Consistent estimation of the order of mixture models. *Sankhya, Series A*, 62(1), February 2000.

Serge Lang. *Complex Analysis*. Springer-Verlag, 3rd edition, 1993.

Steffen L. Lauritzen. *Graphical Models*. Number 17 in Oxford Statistical Science Series. Clarendon Press, 1996.

James D. Murray. *Asymptotic Analysis*. Number 48 in Applied Mathematical Sciences. Springer-Verlag, 1984.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

Dmitry Rusakov and Dan Geiger. Asymptotic model selection for naive Bayesian networks. In Adnan Darwiche and Nir Friedman, editors, *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI-02)*, 2002.

Gideon Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464, 1978.

Raffaella Settimi and Jim Q. Smith. On the geometry of Bayesian graphical models with hidden variables. In Gregory F. Cooper and Serafin Moral, editors, *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 472–479. Morgan Kaufmann Publishers, Inc., 1998.

Raffaella Settimi and Jim Q. Smith. Geometry, moments and conditional independence trees with hidden variables. *Annals of Statistics*, 28:1179–1205, 2000.

Peter Spirtes, T Richardson, and Christopher Meek. The dimensionality of mixed ancestral graphs. Technical Report CMU-PHIL-83, Philosophy Department, Carnegie Mellon University, 1997.

Sumio Watanabe. Algebraic analysis for nonidentifiable learning machines. *Neural Computation*, 13(4):899–933, 2001.

Roderick Wong. *Asymptotic Approximations of Integrals*. Computer Science and Scientific Computing. Academic Press, 1989.

Journal of Machine Learning Research 6 (2005) 37–53 Submitted 3/03; Revised 9/03; Published 1/05

Dimension Reduction in Text Classification with Support Vector Machines

Hyunsoo Kim HSKIM@CS.UMN.EDU
Peg Howland HOWLAND@CS.UMN.EDU
Haesun Park HPARK@CS.UMN.EDU
Department of Computer Science and Engineering
University of Minnesota
200 Union Street S.E., 4-192 EE/CS Building
Minneapolis MN 55455, USA

Editor: Nello Christianini

Abstract

Support vector machines (SVMs) have been recognized as one of the most successful classification methods for many applications including text classification. Even though the learning ability and computational complexity of training in support vector machines may be independent of the dimension of the feature space, reducing computational complexity is an essential issue to efficiently handle a large number of terms in practical applications of text classification. In this paper, we adopt novel dimension reduction methods to reduce the dimension of the document vectors dramatically. We also introduce decision functions for the centroid-based classification algorithm and support vector classifiers to handle the classification problem where a document may belong to multiple classes. Our substantial experimental results show that with several dimension reduction methods that are designed particularly for clustered data, higher efficiency for both training and testing can be achieved without sacrificing prediction accuracy of text classification even when the dimension of the input space is significantly reduced.

Keywords: dimension reduction, support vector machines, text classification, linear discriminant analysis, centroids

1. Introduction

Text classification is a supervised learning task for assigning text documents to pre-defined classes of documents. It is used to find valuable information from a huge collection of text documents available in digital libraries, knowledge databases, the world wide web (WWW), and company-wide intranets, to name a few. Several characteristics have been observed in vector space based methods for text classification (20; 21), including the high dimensionality of the input space, sparsity of document vectors, linear separability in most text classification problems, and the belief that few features are irrelevant. It has been conjectured that an aggressive dimension reduction may result in a significant loss of information, and therefore, result in poor classification results (13).

Assume that training data $(\mathbf{x}_i, y_i)$ with $y_i \in \{-1, +1\}$ for $1 \le i \le n$ are given. The dual formulation of soft margin support vector machines (SVMs) with a kernel function K and control parameter

C is

$$\begin{aligned} \max_{\alpha_i} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j), \\ s.t. \quad & \sum_{i=1}^{n} \alpha_i y_i = 0, \;\; 0 \le \alpha_i \le C, \;\; i = 1, \ldots, n. \end{aligned} \tag{1}$$

The kernel function

$$K(\mathbf{x}_i, \mathbf{x}_j) =< \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) >,$$

where $<,>$ denotes an inner product between two vectors, is introduced to handle nonlinearly separable cases without any explicit knowledge of the feature mapping ϕ. The formulation (1) shows that the computational complexity of SVM training depends on the number of training data samples which is denoted as n. The dimension of the feature space does not influence the computational complexity of training or testing due to the use of the kernel function.

However, an often neglected fact is that the computational complexity of training depends on the *dimension of the input space*. This is clear when we consider some typical kernel functions such as the linear kernel

$$K(\mathbf{x}, \mathbf{x}_i) =< \mathbf{x}, \mathbf{x}_i >,$$

the polynomial kernel

$$K(\mathbf{x}, \mathbf{x}_i) = [< \mathbf{x}, \mathbf{x}_i > + \beta]^d,$$

where d is the degree of the polynomial, and the Gaussian RBF (radial basis function) kernel

$$K(\mathbf{x}, \mathbf{x}_i) = \exp(-\gamma \|\mathbf{x} - \mathbf{x}_i\|^2),$$

where γ is a parameter to control. The evaluation of the kernel function *depends on the dimension of the input data*, since the kernel functions contain the inner product of two input vectors for the linear or polynomial kernels or the distance of two vectors for the Gaussian RBF kernel. Let α_i^* denote the optimal solution for (1). The optimal separating hyperplane $f(\mathbf{x}, \alpha^*, b)$ also requires evaluation of the kernel function since

$$f(\mathbf{x}, \alpha^*, b) = \sum_{\mathbf{x}_i \in SV} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$$

where SV denotes the set of support vectors, b is a bias given by

$$b = -\frac{min_{y_i=1} < w^*, \phi(\mathbf{x}_i) > + \max_{y_i=-1} < w^*, \phi(\mathbf{x}_i) >}{2}$$

and

$$w^* = \sum_{i=1}^{l} y_i {\alpha_i}^* \phi(\mathbf{x}_i).$$

Therefore, more efficient testing as well as training is expected from dimension reduction.

Throughout the paper, we will assume that the document set is represented in an $m \times n$ term-document matrix $A = (a_{ij})$, in which each column represents a document, and each entry a_{ij} represents the weighted frequency of term i in document j (1; 2). The clustering of data is assumed to be performed previously.

In the next section, we review Latent Semantic Indexing (LSI) (2; 1), which uses the truncated singular value decomposition (SVD) as a low-rank approximation of A. Although the truncated SVD provides the closest approximation to A in Frobenius or L_2 norm, LSI ignores the cluster structure while reducing the dimension of the data. In contrast, in Section 3, we review several dimension reduction methods that are especially effective for classification of clustered data: two methods based on centroids (16; 12), and one method which is a generalization of linear discriminant analysis (LDA) using the generalized singular value decomposition (GSVD) (10). With dimension reduction, computational complexity can be dramatically reduced for all classifiers including support vector machines and k-nearest neighbor classification. For k-nearest neighbor classification (kNN), the distances of vector pairs need to be computed when finding k nearest neighbors. Therefore, one can significantly reduce computational complexity by dimension reduction.

In many document data sets, documents can be assigned to more than one cluster upon classification. To handle this problem more effectively, we introduce a threshold based extension of several classification algorithms in Section 4. Our numerical experiments illustrate that the cluster-preserving dimension reduction algorithms we employ reduce the data dimension without any significant loss of information. In fact, in many cases, they seem to have the effect of noise reduction, since prediction accuracy becomes better after dimension reduction when compared to that in the original high dimensional input space.

2. Low-Rank Approximation Using Latent Semantic Indexing

LSI is based on the assumption that there is some underlying latent semantic structure in the term-document matrix that is corrupted by the wide variety of words used in documents and queries. This is referred to as the problem of polysemy and synonymy (6). The basic idea is that if two document vectors represent the same topic, they will share many associating words with a keyword, and they will have very close semantic structures after dimension reduction via SVD. Thus LSI/SVD breaks the original relationship of the data into linearly independent components (6), where the original term vectors are represented by left singular vectors and document vectors by right singular vectors. That is, if $l \leq \text{rank}(A)$, then

$$A \approx U_l \Sigma_l V_l^T$$

, where the columns of U_l are the leading l left singular vectors, Σ_l is an $l \times l$ diagonal matrix with the l largest singular values in nonincreasing order along its diagonal, and the columns of V_l are the leading l right singular vectors. Then $\Sigma_l V_l^T$ is the reduced dimensional representation of A, or equivalently, a new document $\mathbf{q} \in \mathbb{R}^{m \times 1}$ can be represented in the l-dimensional space as $\hat{\mathbf{q}} = U_l^T \mathbf{q}$.

This low-rank approximation has been widely applied in information retrieval (2). Since the complete orthogonal decomposition such as ULV or URV has computational advantages over the SVD including easier updating (22; 23; 24) and downdating (17), dimension reduction by these faster low-rank orthogonal decompositions has also been exploited (3). However, LSI ignores the cluster structure while reducing the dimension. In addition, since there is no theoretical optimum value for the reduced dimension, potentially expensive experimentation may be required to determine a reduced dimension l. As we report in Section 5, classification results after LSI vary depending upon the reduced dimension, classification method, and similarity measure employed. The experimental results confirm that when the data set is already clustered, the dimension reduction methods we present in the next section are more effective for classification of new data.

Algorithm 1 : Centroid algorithm for Dimension Reduction

Given a data set $A \in \mathbb{R}^{m \times n}$ with p clusters and a vector $\mathbf{q} \in \mathbb{R}^{m \times 1}$, this algorithm computes a p dimensional representation $\hat{\mathbf{q}} \in \mathbb{R}^{p \times 1}$ of $\mathbf{q}$.

```
1. Compute the centroid $\mathbf{c}_i$ of the $i$th cluster, $1 \le i \le p$
2. Set $C = \begin{bmatrix} \mathbf{c}_1 & \mathbf{c}_2 & \cdots & \mathbf{c}_p \end{bmatrix}$
3. Solve $\min_{\hat{\mathbf{q}}} \|C\hat{\mathbf{q}} - \mathbf{q}\|_2$
```

Algorithm 2 : Orthogonal Centroid algorithm for Dimension Reduction

Given a data set $A \in \mathbb{R}^{m \times n}$ with p clusters and a vector $\mathbf{q} \in \mathbb{R}^{m \times 1}$, this algorithm computes a p dimensional representation $\hat{\mathbf{q}}$ of $\mathbf{q}$.

```
1. Compute the centroid $\mathbf{c}_i$ of the $i$th cluster, $1 \le i \le p$
2. Set $C = \begin{bmatrix} \mathbf{c}_1 & \mathbf{c}_2 & \cdots & \mathbf{c}_p \end{bmatrix}$
3. Compute the reduced QR decomposition of $C$, which is $C = Q_p R$
4. $\hat{\mathbf{q}} = Q_p^T \mathbf{q}$
```

3. Dimension Reduction Algorithms for Clustered Data

To achieve greater efficiency in manipulating data represented in a high dimensional space, it is often necessary to reduce the dimension *dramatically*. In this section, several dimension reduction methods that preserve the cluster structure are reviewed. Each method attempts to choose a projection to a reduced dimensional space that will capture the cluster structure of the data collection as much as possible.

3.1 Centroid-based Algorithms for Dimension Reduction of Clustered Data

Suppose we are given a data matrix A whose columns are grouped into p clusters. Instead of treating each column of the matrix A equally regardless of its membership in a specific cluster as in LSI/SVD, we want to find a lower dimensional representation Y of A so that the p clusters are preserved in Y. Given a term-document matrix, the problem is to find a transformation that maps each document vector in the m dimensional space to a vector in the l dimensional space for some $l < m$. For this, either the dimension reducing transformation $G^T \in \mathbb{R}^{l \times m}$ is computed explicitly or the problem is formulated as a rank reducing approximation where the given matrix A is to be decomposed into two matrices B and Y. That is,

$$A \approx BY \tag{2}$$

where $B \in \mathbb{R}^{m \times l}$ with rank(B) = l and $Y \in \mathbb{R}^{l \times n}$ with rank(Y) = l. The matrix B accounts for the dimension reducing transformation. However, it is not necessary to compute the dimension reducing transformation G from B explicitly, as long as we can find the reduced dimensional representation of a given data item. If the matrix B is already determined, the matrix Y can be computed by solving the least squares problem (8; 12; 16)

$$\min_{B,Y} \|BY - A\|_F. \tag{3}$$

Any given document $\mathbf{q} \in \mathbb{R}^{m\times 1}$ can be transformed to the lower dimensional space by solving the minimization problem

$$\min_{\hat{\mathbf{q}}\in\mathbb{R}^{l\times 1}} \|B\hat{\mathbf{q}} - \mathbf{q}\|_2. \tag{4}$$

Latent Semantic Indexing that utilizes the SVD (LSI/SVD) can be viewed as a variation of the model (2) with $B = U_l$ (16), where $U_l \Sigma_l V_l^T$ is the rank l truncated SVD of A. Then $\hat{\mathbf{q}} = U_l^T \mathbf{q}$ is obtained by solving the least squares problem

$$\min_{\hat{\mathbf{q}}\in R^{l\times 1}} \|B\hat{\mathbf{q}} - \mathbf{q}\|_2 = \min_{\hat{\mathbf{q}}\in R^{l\times 1}} \|U_l\hat{\mathbf{q}} - \mathbf{q}\|_2. \tag{5}$$

In the Centroid dimension reduction algorithm (see Algorithm 1), the ith column of B is the centroid vector of the ith cluster, which is the average of the data items in the ith cluster, for $1 \le i \le p$. This matrix B is called the centroid matrix. Then, any vector $\mathbf{q} \in \mathbb{R}^{m\times 1}$ can be represented in the p dimensional space as $\hat{\mathbf{q}}$, the solution of the least squares problem (4), where B is the centroid matrix. In the Orthogonal Centroid algorithm (see Algorithm 2), the p dimensional representation of a data vector $\mathbf{q} \in \mathbb{R}^{m\times 1}$ is given as $\hat{\mathbf{q}} = Q_p^T \mathbf{q}$ where Q_p is an orthonormal basis for the centroid matrix obtained from its QR decomposition.

The centroid-based dimension reduction algorithms are computationally less costly than LSI/SVD. They are also more effective when the data are already clustered. Although the centroid-based schemes can be applied only when the data are linearly separable, they are suitable for text classification problems, since text data is usually linearly separable in the original dimensional space (13). For a nonlinear extension of the Orthogonal Centroid method that utilizes kernel functions, see (18).

3.2 Generalized Discriminant Analysis based on the Generalized Singular Value Decomposition

Recently, a new algorithm has been developed for cluster-preserving dimension reduction based on the generalized singular value decomposition (GSVD) (10). This algorithm generalizes classical discriminant analysis, by extending its application to very high-dimensional data such as that encountered in text classification.

Classical discriminant analysis (7; 25) preserves cluster structure by maximizing the scatter between clusters while minimizing the scatter within clusters. For this purpose, the within-cluster scatter matrix S_w and the between-cluster scatter matrix S_b are defined. If we denote by N_i the set of column indices that belong to the cluster i, n_i the number of columns in cluster i, and $\mathbf{c}$ the global centroid, then

$$S_w = \sum_{i=1}^{p} \sum_{j\in N_i} (\mathbf{a}_j - \mathbf{c}_i)(\mathbf{a}_j - \mathbf{c}_i)^T,$$

and

$$\begin{aligned} S_b &= \sum_{i=1}^{p} \sum_{j\in N_i} (\mathbf{c}_i - \mathbf{c})(\mathbf{c}_i - \mathbf{c})^T \\ &= \sum_{i=1}^{p} n_i (\mathbf{c}_i - \mathbf{c})(\mathbf{c}_i - \mathbf{c})^T. \end{aligned}$$

Algorithm 3 LDA/GSVD

```
Given a data matrix $A \in \mathbb{R}^{m\times n}$ with $p$ clusters, this algorithm computes the columns of the matrix
$G \in \mathbb{R}^{m\times(p-1)}$, which preserves the cluster structure in the reduced dimensional space, and it also
computes the $p-1$ dimensional representation $Y$ of $A$.

1. Compute $H_b \in \mathbb{R}^{m\times p}$ and $H_w \in \mathbb{R}^{m\times n}$ from $A$ according to Eqns. (7) and (6), respectively.
2. Compute the complete orthogonal decomposition of $H = (H_b, H_w)^T \in \mathbb{R}^{(p+n)\times m}$, which is
       $P^T H Q = \begin{pmatrix} R & 0 \\ 0 & 0 \end{pmatrix}$.
3. Let $t = \text{rank}(H)$.
4. Compute W from the SVD of $P(1:p, 1:t)$, which is $U^T P(1:p,1:t)W = \Sigma_A$.
5. Compute the first $p-1$ columns of
       $X = Q\begin{pmatrix} R^{-1}W & 0 \\ 0 & I \end{pmatrix}$,
   and assign them to $G$.
6. $Y = G^T A$
```

Since

$$trace(S_w) = \sum_{i=1}^{p}\sum_{j\in N_i} \|\mathbf{a}_j - \mathbf{c}_i\|_2^2$$

measures the closeness within the clusters, and

$$trace(S_b) = \sum_{i=1}^{p}\sum_{j\in N_i} \|\mathbf{c}_i - \mathbf{c}\|_2^2$$

measures the remoteness between the clusters, the goal is to minimize the former while maximizing the latter in the reduced dimensional space. Once again letting $G^T \in \mathbb{R}^{l\times m}$ denote the transformation that maps a column of A in the m dimensional space to a vector in the l dimensional space, the goal can be expressed as the simultaneous minimization of $\text{trace}(G^T S_w G)$ and maximization of $\text{trace}(G^T S_b G)$.

When S_w is nonsingular, this simultaneous optimization is commonly approximated by maximizing

$$J_1(G) = \text{trace}((G^T S_w G)^{-1}(G^T S_b G)).$$

It is well known that the global maximum is achieved when the columns of G are the eigenvectors of $S_w^{-1}S_b$ that correspond to the l largest eigenvalues (7; 25). In fact, when the reduced dimension $l \geq p-1$, $\text{trace}(S_w^{-1}S_b)$ is exactly preserved upon dimension reduction, and equals $\lambda_1 + \cdots + \lambda_{p-1}$, where each $\lambda_i \geq 0$. Without loss of generality, we assume that the term-document matrix A is partitioned as

$$A = [A_1, \quad \cdots, \quad A_p]$$

where the columns of each block $A_i \in \mathbb{R}^{m\times n_i}$ belong to the cluster i. Defining the matrices

$$H_w = [\mathbf{a}_1 - \mathbf{c}_1, \mathbf{a}_2 - \mathbf{c}_1, \ldots, \mathbf{a}_n - \mathbf{c}_p] \in \mathbb{R}^{m\times n} \tag{6}$$

and

$$H_b = [\sqrt{n_1}(\mathbf{c}_1 - \mathbf{c}), \ldots, \sqrt{n_p}(\mathbf{c}_p - \mathbf{c})] \in \mathbb{R}^{m\times p}, \tag{7}$$

then

$$S_w = H_w H_w^T \quad \text{and} \quad S_b = H_b H_b^T.$$

As the product of an $m \times n$ matrix with an $n \times m$ matrix, S_w will be singular when the number of terms m exceeds the number of documents n. In that case, classical discriminant analysis fails. However, if we rewrite the eigenvalue problem $S_w^{-1} S_b \mathbf{x}_i = \lambda_i \mathbf{x}_i$ as

$$\beta_i^2 H_b H_b^T \mathbf{x}_i = \alpha_i^2 H_w H_w^T \mathbf{x}_i,$$

it can be solved by the GSVD.

The resulting algorithm, called LDA/GSVD, is summarized in Algorithm 3. It follows the construction of the Paige and Saunders (15) proof, but only computes the necessary part of the GSVD. The most expensive step of LDA/GSVD is the complete orthogonal decomposition of the composite H matrix in Step 2. When $\max(p,n) \ll m$, the SVD of $H = [H_b^T, H_w^T] \in \mathbb{R}^{(p+n)\times m}$ can be computed by first computing the reduced QR decomposition $H^T = Q_H R_H$, and then computing the SVD of $R_H \in \mathbb{R}^{(p+n)\times(p+n)}$ as

$$R_H = Z \begin{pmatrix} \Sigma_H & 0 \\ 0 & 0 \end{pmatrix} P^T.$$

This gives

$$H = R_H^T Q_H^T = P \begin{pmatrix} \Sigma_H & 0 \\ 0 & 0 \end{pmatrix} Z^T Q_H^T,$$

where the columns of $Q_H Z \in \mathbb{R}^{m\times(p+n)}$ are orthonormal. There exists othogonal $Q \in \mathbb{R}^{m\times m}$ whose first $p+n$ columns are $Q_H Z$. Hence

$$H = P \begin{pmatrix} \Sigma_H & 0 \\ 0 & 0 \end{pmatrix} Q^T,$$

where there are now $m-t$ zero columns to the right of Σ_H. Since $R_H \in \mathbb{R}^{(p+n)\times(p+n)}$ is a much smaller matrix than $H \in \mathbb{R}^{(p+n)\times m}$, the required memory is substantially reduced. In addition, the computational complexity of the algorithm is reduced to $O(mn^2) + O(n^3)$ (8), since this step is the dominating part.

4. Classification Methods

To test the effect of dimension reduction in text classification, three different classification methods were used: centroid-based classification, k-nearest neighbor (kNN), and support vector machines (SVMs). Each classification method is modified by introducing some threshold values to perform classification correctly when a document has membership in multiple classes. In this section, we briefly review the three classification methods and discuss their modifications.

Algorithm 4 : Centroid-based Classification

Given a data matrix A with p clusters and p corresponding centroids, $\mathbf{c}_i$, $1 \le i \le p$, and a vector $\mathbf{q} \in \mathbb{R}^{m \times 1}$, this method finds the index j of the cluster in which the vector $\mathbf{q}$ belongs.

- find the index j such that $sim(\mathbf{q}, \mathbf{c}_i)$, $1 \le i \le p$, is minimum (or maximum), where $sim(\mathbf{q}, \mathbf{c}_i)$ is the similarity measure between $\mathbf{q}$ and $\mathbf{c}_i$. (For example, $sim(\mathbf{q}, \mathbf{c}_i) = \|\mathbf{q} - \mathbf{c}_i\|_2$ using the L_2 norm, and we take the index with the minimum value. Using the cosine measure,

$$sim(\mathbf{q}, \mathbf{c}_i) = cos(\mathbf{q}, \mathbf{c}_i) = \frac{\mathbf{q}^T \mathbf{c}_i}{\|\mathbf{q}\|_2 \|\mathbf{c}_i\|_2},$$

  and we take the index with the maximum value.)

4.1 Centroid-based Classification

Centroid-based classification, summarized in Algorithm 4, is one of the simplest classification methods. A test document is assigned to a class that has the most similar centroid. Using the cosine similarity measure, we can classify a test document $\mathbf{q}$ by computing

$$arg \max_{1 \le i \le p} \frac{\mathbf{q}^T \mathbf{c}_i}{\|\mathbf{q}\|_2 \|\mathbf{c}_i\|_2} \tag{8}$$

where $\mathbf{c}_i$ is the centroid of the ith cluster of the training data. When dimension reduction is performed by the Centroid algorithm, the centroids of the full space become the columns $\mathbf{e}_i \in \mathbb{R}^{p \times 1}$ of the identity matrix. Then the decision rule becomes

$$arg \max_{1 \le i \le p} \frac{\hat{\mathbf{q}}^T \mathbf{e}_i}{\|\hat{\mathbf{q}}\|_2 \|\mathbf{e}_i\|_2}, \tag{9}$$

where $\hat{\mathbf{q}}$ is the reduced dimensional representation of the document $\mathbf{q}$. This shows that classification can be performed by simply finding the index i of the vector $\hat{\mathbf{q}}$ with the largest component. Centroid-based classification has the advantage that the computation involved is extremely simple. We can also classify using the L_2 norm similarity measure by finding the centroid that is closest to $\mathbf{q}$ in L_2 norm.

The original form of centroid-based classification finds the nearest centroid and assigns the corresponding class as the predicted class. To allow an assignment of any document to multiple classes, we introduce the decision rule for centroid-based classification as

$$y(\mathbf{x}, j) = \text{sign}\{sim(\mathbf{x}, \mathbf{c}_j) - \theta_j^c\}, \tag{10}$$

where $y(\mathbf{x}, j) \in \{+1, -1\}$ is the classification for document $\mathbf{x}$ with respect to class j (if $y > 0$ then the class is j, else the class is not j), $sim(\mathbf{x}, \mathbf{c}_j)$ is the similarity between the test document $\mathbf{x}$ and the centroid vector $\mathbf{c}_j$ for the class j, and θ_j^c is the class specific threshold for the binary decision for $y(\mathbf{x}, j)$ in centroid-based classification. In this way, document $\mathbf{x}$ will be a member of class j if its similarity to the centroid vector $\mathbf{c}_j$ for the class is above the threshold.

Algorithm 5 : k Nearest Neighbor (kNN) Classification

Given a data matrix $A = [\mathbf{a}_1, \ldots, \mathbf{a}_n]$ with p clusters and a vector $\mathbf{q} \in \mathbb{R}^{m \times 1}$, this method finds the cluster in which the vector $\mathbf{q}$ belongs.

```
1. Using the similarity measure $sim(\mathbf{q}, \mathbf{a}_j)$ for $1 \le j \le n$, find the $k$ nearest neighbors of $\mathbf{q}$.
2. Among these $k$ vectors, count the number belonging to each cluster.
3. Assign $\mathbf{q}$ to the cluster with the greatest count in the previous step.
```

4.2 k-Nearest Neighbor Classification

The kNN algorithm, summarized in Algorithm 5, is one of the most commonly used classification methods. To correctly predict the membership of a document which belongs to multiple classes, we used the following modified decision rule for kNN (29):

$$y(\mathbf{x}, j) = \text{sign}\{\sum_{\mathbf{d}_i \in kNN} sim(\mathbf{x}, \mathbf{d}_i) y(\mathbf{d}_i, j) - \theta_j^{kNN}\} \tag{11}$$

where kNN is the set of k nearest neighbors for document $\mathbf{x}$, $y(\mathbf{d}_i, j) \in \{+1, -1\}$ is the classification for document $\mathbf{d}_i$ with respect to class j (if $y > 0$ then the class is j, else the class is not j), $sim(\mathbf{x}, \mathbf{d}_i)$ is the similarity between the test document $\mathbf{x}$ and the training document $\mathbf{d}_i$, and θ_j^{kNN} is the class specific threshold for kNN classification.

4.3 Support Vector Machines

The optimal separating hyperplane of the one-vs-rest binary classifier can be obtained by conventional SVMs. We introduce the following decision rule for support vector machines as

$$y(\mathbf{x}, j) = \text{sign}\{\sum_{\mathbf{x}_i \in SV} \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) + b - \theta_j^{SVM}\}, \tag{12}$$

where $y(\mathbf{x}, j) \in \{+1, -1\}$ is the classification for document $\mathbf{x}$ with respect to class j, SV is the set of support vectors, and θ_j^{SVM} is the class specific threshold for the binary decision. This threshold is set so that a new document $\mathbf{x}$ must not be classified to belong to class j when it is located very close to the optimal separating hyperplane, i.e. when the decision is made with a low reliability. We use the linear kernel $K = < \mathbf{x}, \mathbf{x}_i >$, the polynomial kernel $K = [< \mathbf{x}, \mathbf{x}_i > +1]^d$, where d is the degree of the polynomial, and the Gaussian RBF (radial basis function) kernel $K = \exp(-\gamma \|\mathbf{x} - \mathbf{x}_i\|^2)$, where γ is a parameter that controls the width of the Gaussian function.

5. Experimental Results

Prediction results are compared for the test documents in the full space without any dimension reduction as well as those in the reduced space obtained by LSI/SVD, Centroid, Orthogonal Centroid, and LDA/GSVD dimension reduction methods. For SVMs, we optimized the regularization parameter C, polynomial degree d for the polynomial kernel, and γ for the Gaussian RBF (radial basis function) kernel for each full and reduced dimension data set.

classification	The rank-l approximation of LSI/SVD								
methods	l=5	l=100	l=200	l=300	l=500	l=1000	l=1246	l=1247	Full
centroid (L_2)	71.6	82.2	83.4	83.9	84.8	84.9	85.2	85.2	85.2
centroid (Cosine)	78.5	86.9	87.1	87.6	88.0	88.2	88.3	88.3	88.3
5NN (L_2)	77.8	68.8	55.4	49.2	63.8	76.9	79.0	79.0	79.0
15NN (L_2)	77.5	69.7	52.7	50.3	76.3	74.7	83.4	83.4	83.4
30NN (L_2)	77.5	64.3	47.8	58.0	80.8	73.2	83.8	83.8	83.8
5NN (Cosine)	77.8	82.2	79.1	79.6	79.4	78.7	77.8	77.8	77.8
15NN (Cosine)	80.2	83.1	82.5	83.6	82.9	82.5	82.5	82.5	82.5
30NN (Cosine)	79.8	83.4	83.8	84.1	84.2	84.1	83.8	83.8	83.8
SVM	79.1	87.6	88.4	88.5	88.6	89.2	89.7	89.7	88.9

Table 1: Text classification accuracy (%) using centroid-based classification, k-nearest neighbor classification, and SVMs, with LSI/SVD dimension reduction on the MEDLINE data set. The Euclidean norm (L_2) and the cosine similarity measure (Cosine) were used for the centroid-based and kNN classification.

The first data set that we used was a subset of the MEDLINE database with 5 classes. Each class has 500 documents. The set was divided into 1250 training documents and 1250 test documents. After stemming and stoplist removal, the training set contains 22095 distinct terms. For this data, each document belongs to only one class, and we used the original form of the three classification algorithms without introducing the threshold.

The second data set was the “ModApte” split of the Reuter-21578 text collection. We only used 90 classes for which there is at least one training and one test example in each class. It contains 7769 training documents and 3019 test documents. The training set contains 11941 distinct terms after preprocessing with stoplist removal and stemming. The Reuter data set contains documents that belong to multiple classes, so the classification methods utilize thresholds.

We used a standard weight factor for each word stem:

$$\phi_i(\mathbf{x}) = \frac{tf_i \log(idf_i)}{\kappa}, \tag{13}$$

where tf_i is the number of occurrences of term i in document $\mathbf{x}$, $idf_i = n/d$ is the ratio between the total number of documents n and the number of documents d containing the term, and κ is the normalization constant that makes $\|\phi\|_2 = 1$.

Table 1 reports text classification accuracy for the MEDLINE data set using LSI/SVD with a range of values for the reduced dimension. The smallest reduced dimension, $l = 5$, is included in order to compare with centroid-based and LDA/GSVD methods, which reduce the dimension to 5 and 4, respectively. Since the training set has the nearly-full rank of 1246, we include the reduced dimensions 1246 and 1247 at the high end of the range. For a training set of size 1250, the reduced dimension $l = 300$ is generous. However, we observe that kNN classification with L_2 norm similarity produces poor classification results for l values from 100 to 500. This is consistent with the common belief that cosine similarity performs better with unnormalized text data. Also, classification accuracy using 5NN lags that for higher values of k, suggesting that k=5 is too small for classes

kernel		Dimension reduction methods			
	Full	Centroid	Orthogonal Centroid	LDA/ GSVD4	LDA/ GSVD5
	22095×1250	5×1250	5×1250	4×1250	5×1250
linear (C=1.0)	88.1	88.9	85.9	86.5	86.6
linear (C=10.0)	88.9	88.5	88.3	86.7	86.7
linear (C=50.0)	88.9	87.7	88.8	87.1	87.1
linear^{opt}	88.9	88.9	89.0	87.4	87.4
polynomial(d=2)	88.6	88.9	88.9	87.3	87.3
polynomial(d=3)	88.0	89.0	88.8	87.4	87.4
polynomial(d=4)	87.5	88.9	88.8	87.2	87.2
polynomial(d=5)	86.5	88.6	88.8	87.1	87.1
polynomial^{opt}	88.6	89.0	88.9	87.4	87.4
RBF ($\gamma = 0.5$)	88.5	89.0	89.0	87.1	87.2
RBF ($\gamma = 1.0$)	87.6	89.2	89.0	87.3	87.2
RBF ($\gamma = 1.5$)	86.3	89.1	88.8	87.4	87.3
RBF^{opt}	88.7	89.2	89.0	87.4	87.3

Table 2: Text classification accuracy (%) with different kernels in SVMs with and without dimension reduction on the MEDLINE data set. The regularization parameter C for each case was optimized by numerical experiments. Dimension of each training term-document matrix is shown. LDA/GSVD4 and LDA/GSVD5 represent the results from LDA/GSVD where the reduced dimensions are 4 and 5, respectively.

of size 250. It is noteworthy that even with LSI, which makes no attempt to preserve the cluster structure upon dimension reduction, SVM classification achieves very consistent classification results for reduced dimensions of 100 or greater, and the SVM accuracy exceeds that of the other classification methods.

Table 2 shows text classification accuracy (%) with different kernels in SVMs, with and without dimension reduction on the MEDLINE data set. Note that the linear^{opt} values are optimal over all the values of the regularization parameter C that we tried, and the RBF^{opt} values are optimal over all the γ values we tried. This table shows that the prediction results in the reduced dimension are similar to those in the original full dimensional space, while achieving a significant reduction in time and space complexity. In the reduced space obtained by the Orthogonal Centroid dimension reduction algorithm, the classification accuracy is insensitive to the choice of the kernel. Thus, we can choose the linear kernel in this case instead of the computationally more expensive polynomial or RBF kernel.

Table 3 shows classification accuracy obtained by all three classification methods – centroid-based, kNN with three different values of k, and the optimal result from SVM – for each dimension reduced data set and the full space. For the LDA/GSVD dimension reduction method, the classification accuracy with cosine similarity measure is lower with centroid-based classification as well as with kNN, while the results with L_2 norm are better. This is due to the formulation of trace optimization criteria in terms of the L_2 norm. With LDA/GSVD, documents from the same class in

classification methods	Full	Dimension reduction methods			
		Centroid	Orthogonal Centroid	LDA/ GSVD4	LDA/ GSVD5
	22095×1250	5×1250	5×1250	4×1250	5×1250
centroid (L_2)	85.2	88.0	85.2	88.7	88.7
centroid (Cosine)	88.3	88.0	88.3	83.9	83.9
5NN (L_2)	79.0	88.4	88.6	81.5	86.6
15NN (L_2)	83.4	88.3	87.8	88.7	88.6
30NN (L_2)	83.8	88.8	88.5	88.7	88.5
5NN (Cosine)	77.8	88.6	88.2	83.8	84.1
15NN (Cosine)	82.5	88.2	88.5	83.8	84.1
30NN (Cosine)	83.8	88.3	88.6	83.8	84.1
SVM	88.9	89.2	89.0	87.4	87.4

Table 3: Text classification accuracy (%) using centroid-based classification, k-nearest neighbor classification, and SVMs, with and without dimension reduction on the MEDLINE data set. The Euclidean norm (L_2) and the cosine similarity measure (Cosine) were used for centroid-based and kNN classification.

class	Full	Dimension reduction			
		Centroid	Orthogonal Centroid	LDA/ GSVD4	LDA/ GSVD5
	22095×1250	5×1250	5×1250	4×1250	5×1250
heart attack	92.4	94.4	94.4	92.4	92.4
colon cancer	84.8	84.8	86.0	83.2	83.2
glycemic	95.6	97.6	98.0	95.2	95.2
oral cancer	82.0	75.2	73.6	78.8	78.8
tooth decay	89.6	94.0	92.8	87.2	87.2
microavg	88.9	89.2	89.0	87.4	87.4

Table 4: Text classification accuracy (%) of the 5 classes and the microaveraged performance over all 5 classes on the MEDLINE data set. All results are from SVMs using optimal kernels.

the full dimensional space tend to be transformed to a very tight cluster or even to a single point in the reduced space, since the LDA/GSVD algorithm tends to minimize the trace of the within cluster scatter. This seems to make it difficult for SVMs to find a binary classifier with low generalization error.

Table 4 shows text classification accuracy for the 5 classes using SVMs with and without dimension reduction methods on the MEDLINE data set. The colon cancer and oral cancer documents were relatively hard to classify correctly.

The REUTERS data set has many documents that are classified to more than 2 classes, whereas no document is classified to belong to more than one class in the MEDLINE data set. While we

classification methods	Full	Dimension reduction	
		Centroid	Orthogonal Centroid
	11941×9579	90×9579	90×9579
centroid(L_2)	78.89	73.32	78.00
centroid(Cosine)	80.45	74.79	80.46
15NN	78.65	81.70	85.51
30NN	80.21	81.94	86.19
45NN	80.29	81.01	84.79
SVM	87.11	84.54	87.03

Table 5: Comparison of micro-averaged F_1 scores for 3 different classification methods with and without dimension reduction on the REUTERS data set. The Euclidean norm (L_2) and the cosine similarity measure (Cosine) were used for the centroid-based classification. The cosine similarity measure was used for the kNN classification. The dimension of the full training term-document matrix is 11941×9579 and that of the reduced matrix is 90×9579.

could handle relatively large matrices using a sparse matrix representation and sparse QR decomposition in the Centroid and Orthogonal Centroid dimension reduction methods, results for the LDA/GSVD dimension reduction method are not reported, since we ran out of memory while computing the GSVD. For this data set, we built a series of threshold-based classifiers, optimizing the thresholds to capture the multiple class membership. All class specific thresholds (θ_j^{kNN}, θ_j^c, θ_j^{SVM}) are determined by numerical experiments. Though we obtained precision/recall break even points by optimizing the thresholds, we report values of the F_1 measure (26) which is defined as

$$F_1 = \frac{2rp}{r+p}, \tag{14}$$

where r is recall and p is precision for a binary classification. Table 5 shows that the effectiveness of classification was preserved for the Orthogonal Centroid dimension reduction algorithm, while it became worse for the Centroid dimension reduction algorithm. This is due to a property of the Centroid algorithm that the centroids of the full space are projected to the columns of the identity matrix in the reduced space. This orthogonality between the centroids may make it difficult to represent the multiclass membership of a document by separating closely related classes after dimension reduction. The pattern of prediction measure F_1 for each class is also preserved by Orthogonal Centroid in Table 6. The macro-averaged F_1 and micro-averaged F_1 for the 10 most frequent classes are also presented.

6. Conclusion and Discussion

In this paper, we applied three methods, Centroid, Orthogonal Centroid, and LDA/GSVD, which are designed for reducing the dimension of clustered data. For comparison, we also applied LSI/SVD, which does not attempt to preserve cluster structure upon dimension reduction. We tested the effectiveness in classification with dimension reduction using three different classification methods:

class		Dimension reduction	
	Full	Centroid	Orthogonal Centroid
	11941×9579	90×9579	90×9579
earn	98.25	97.49	96.60
acq	95.57	95.45	94.94
money-fx	75.78	77.97	79.44
grain	92.88	86.62	92.26
crude	88.11	86.49	87.70
trade	75.32	75.11	77.25
interest	77.99	78.13	83.21
ship	84.09	85.71	88.00
wheat	84.14	81.94	84.06
corn	87.27	74.78	89.47
microavg (top 10)	92.21	91.32	92.21
avg (top 10)	85.94	83.96	87.32
microavg(all)	87.11	84.54	87.03

Table 6: F_1 scores of the 10 most frequent classes and micro-averaged performance over all 90 classes on the REUTERS data set. All results are from SVMs using optimal kernels. The dimension of the full training term-document matrix is 11941×9579 and that of the reduced matrix is 90×9579.

SVMs, kNN, and centroid-based classification. For the three cluster-preserving methods, the results show surprisingly high prediction accuracy, which is essentially the same as in the original full space, even with very dramatic dimension reduction. They justify dimension reduction as a worthwhile preprocessing stage for achieving high efficiency and effectiveness. Especially for kNN classification, the savings in computational complexity in classification after dimension reduction are significant. In the case of SVM the savings are also clear, since the distance between two pairs of input data points need to be computed repeatedly with and without the use of the kernel function, and the vectors become significantly shorter with dimension reduction.

We have also introduced threshold based classifiers for centroid-based classification and SVMs in order to capture the overlap structure between closely related classes. Prediction results with the Centroid dimension reduction method became better compared to those from the full space for the completely disjoint MEDLINE data set, but became worse for the REUTERS data set. Since the Centroid dimension reduction method maps the centroids to unit vectors $\mathbf{e}_i$ which are orthogonal to each other, it is helpful for the disjoint data set, but not for a data set which contains documents belonging multiple classes. We observed that prediction accuracy with the Orthogonal Centroid dimension reduction algorithm was preserved for SVMs as well as with centroid-based classification. The Orthogonal Centroid dimension reduction method maximizes the between cluster relationship using the relatively inexpensive reduced QR decomposition, compared to LDA/GSVD which also considers the within cluster relationship but requires a more expensive rank revealing decomposition such as the singular value decomposition (10; 11).

The better prediction accuracy using SVMs is due to low generalization error by maximizing the margin, and the capability to handle non-linearity by kernel choice. Although most classes of the Reuters-21578 data set are linearly separable (13), there seems to be some level of non-linearity. For non-linearly separable data, SVMs with appropriate nonlinear kernel functions would work as a better classifier. Another way to handle non-linearly separable data is to apply nonlinear extensions of the dimension reduction methods, including those presented in (18; 19). All of the dimension reduction methods presented here can also be applied to visualize the higher dimensional structure by reducing the dimension to 2- or 3-dimensional space.

We conclude that dramatic dimension reduction of text documents can be achieved, without sacrificing classification accuracy. For the document sets we tested, the Orthogonal Centroid method did particularly well at preserving the cluster structure from the full dimensional representation. That is, the prediction accuracies for Orthogonal Centroid rival those of the full space, even though the dimension is reduced to the number of clusters. The savings in computational complexity are significant using either kNN classification or SVM.

Acknowledgments

This material is based upon work supported by the National Science Foundation Grant No. CCR-0204109. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation (NSF). The authors would also like to thank University of Minnesota Supercomputing Institute (MSI) for providing the computing facilities.

References

[1] M. W. Berry, Z. Drmac, and E. R. Jessup. Matrices, vector spaces, and information retrieval. *SIAM Review*, 41:335–362, 1999.

[2] M. W. Berry, S. T. Dumais, and G. W. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37:573–595, 1995.

[3] M. W. Berry and R. D. Fierro. Low-rank orthogonal decompositions for information retrieval applications. *Numerical Linear Algebra with Applications*, 3(4):301–327, 1996.

[4] Å. Björck. *Numerical Methods for Least Square Problems*. SIAM, Philadelphia, PA, 1996.

[5] N. Cristianini and J. Shawe-Taylor. *Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[6] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the Society for Information Science*, 41:391-407, 1990.

[7] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Second ed., Academic Press, 1990.

[8] G. H. Golub and C. F. Van Loan. *Matrix Computations, third edition*. Johns Hopkins University Press, Baltimore, 1996.

[9] M. Heiler. *Optimization Criteria and Learning Algorithms for Large Margin Classifiers*. Diploma Thesis, University of Mannheim., 2002.

[10] P. Howland, M. Jeon, and H. Park. Structure Preserving Dimension Reduction for Clustered Text Data based on the Generalized Singular Value Decomposition. *SIAM Journal of Matrix Analysis and Applications*, 25(1):165–179, 2003.

[11] P. Howland and H. Park. Generalizing discriminant analysis using the generalized singular value decomposition, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8): 995-1006, 2004.

[12] M. Jeon, H. Park, and J. B. Rosen. Dimensional reduction based on centroids and least squares for efficient processing of text data. In *Proceedings for the First SIAM International Workshop on Text Mining*. Chicago, IL, 2001.

[13] 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.

[14] H. Lodhi, N. Cristianini, J. Shawe-Taylor, and C. Watkins. Text classification using string kernels. *Advances in Neural Information Processing Systems*, 13:563–569, 2000.

[15] C. C. Paige and M. A. Saunders, Towards a generalized singular value decomposition, *SIAM Journal of Numerical Analysis*, 18, pp. 398–405, 1981.

[16] H. Park, M. Jeon, and J. B. Rosen. Lower dimensional representation of text data based on centroids and least squares, *BIT Numerical Mathematics*, 42(2):1–22, 2003.

[17] H. Park and L. Eldén. Downdating the rank-revealing URV decomposition. *SIAM Journal of Matrix Analysis and Applications*, 16, pp. 138–155, 1995.

[18] C. Park and H. Park. Nonlinear feature extraction based on centroids and kernel functions. *Pattern Recognition*, to appear.

[19] C. Park and H. Park. Kernel discriminant analysis based on the generalized singular value decomposition. Technical report 03-017, Department of Computer Science and Engineering, University of Minnesota, 2003.

[20] G. Salton, *The SMART Retrieval System*, Prentice Hall, 1971.

[21] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, 1983.

[22] G. W. Stewart. An updating algorithm for subspace tracking. *IEEE Transactions on Signal Processing*, 40:1535–1541, 1992.

[23] G. W. Stewart. Updating URV decompositions in parallel. *Parallel Computing*, 20(2):151–172, 1994.

[24] M. Stewart and P. Van Dooren. Updating a generalized URV decomposition. *SIAM Journal of Matrix Analysis and Applications*, 22(2):479–500, 2000.

[25] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, Academic Press, 1999.

[26] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 1979.

[27] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.

[28] V. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, New York, 1998.

[29] Y. Yang and X. Liu. A re-examination of text categorization methods. In *22nd Annual International SIGIR*, pages 42–49, Berkeley, August 1999.

Journal of Machine Learning Research 6 (2005) 55–79 Submitted 2/04; Revised 8/04; Published 1/05

Stability of Randomized Learning Algorithms

Andre Elisseeff AEL@ZURICH.IBM.COM
IBM Zurich Research Lab
8803 Rueschlikon, Switzerland

Theodoros Evgeniou THEODOROS.EVGENIOU@INSEAD.EDU
Technology Management
INSEAD
77300 Fontainebleau, France

Massimiliano Pontil M.PONTIL@CS.UCL.AC.UK
Department of Computer Science
University College London
Gower Street, London WC1E, UK

Editor: Leslie Pack Kaelbling

Abstract

We extend existing theory on stability, namely how much changes in the training data influence the estimated models, and generalization performance of deterministic learning algorithms to the case of randomized algorithms. We give formal definitions of stability for randomized algorithms and prove non-asymptotic bounds on the difference between the empirical and expected error as well as the leave-one-out and expected error of such algorithms that depend on their random stability. The setup we develop for this purpose can be also used for generally studying randomized learning algorithms. We then use these general results to study the effects of bagging on the stability of a learning method and to prove non-asymptotic bounds on the predictive performance of bagging which have not been possible to prove with the existing theory of stability for deterministic learning algorithms.[1]

Keywords: stability, randomized learning algorithms, sensitivity analysis, bagging, bootstrap methods, generalization error, leave-one-out error.

1. Introduction

The stability of a learning algorithm, namely how changes to the training data influence the result of the algorithm, has been used by many researchers to study the generalization performance of several learning algorithms (Devroye and Wagner, 1979; Breiman, 1996b; Kearns and Ron, 1999; Bousquet and Elisseeff, 2002; Kutin and Niyogi, 2002; Poggio et al., 2004). Despite certain difficulties with theories about stability, such as the lack so far of tight bounds as well as lower bounds (Bousquet and Elisseeff, 2002), the study of learning methods using notions of stability is promising although it is still at its infancy. For example, recently Poggio et al. (2004) have shown conditions for the generalization of learning methods in terms of a stability notion that have possible implications for new insights on diverse learning problems.

1. This work was done while A.E. was at the Max Planck Institute for Biological Cybernetics in Tuebingen, Germany.

The existing theory, however, is developed only for deterministic learning algorithms (Bousquet and Elisseeff, 2002), therefore it cannot be used to study a large number of algorithms which are randomized, such as bagging (Breiman, 1996a), neural networks, or certain Bayesian learning methods. The *goal of this paper* is to improve upon this analysis. To this end, we present a natural generalization of the existing theory to the case of randomized algorithms, thereby extending the results of (Bousquet and Elisseeff, 2002), and formally prove bounds on the performance of randomized learning algorithms using notions of randomized stability that we define. To prove our results we have also extended the results of (Bousquet and Elisseeff 2002) that hold only for symmetric learning algorithms to the case of asymmetric ones. We then prove, as an application of our results, new non-asymptotic bounds for bagging (Breiman, 1996a), a randomized learning method. Finally, we note that our work also provides an approach that can be used for extending other studies, for example other results on stability, done for deterministic algorithms to the case of randomized learning algorithms.

The paper is organized as follows. For completeness and comparison we first replicate in Section 2 the key notions of stability and the generalization bounds we extend derived for deterministic methods in the literature. We then extend these notions — Definitions 7, 10, and 13 — and generalization bounds — Theorems 9, 12 and 15 — to the case of randomized methods in Section 3. Finally, in Section 4 we present an analysis of bagging within the stability theory framework.

2. Stability and Generalization for Deterministic Algorithms

In this section we briefly review the results in (Devroye and Wagner 1979; Kearns and Ron, 1999; Bousquet and Elisseeff, 2002) that show that stability is linked to generalization for deterministic learning methods. We assume here that all algorithms are symmetric, that is, their outcome does not change when the elements in the training set are permuted. In the next section, we will extend stability concepts to the case of randomized learning methods and remove this symmetry assumption.

2.1 Basic Notation

In the following, calligraphic font is used for sets and capital letters refer to numbers unless explicitly defined. Let $\mathcal{X}$ be a set, $\mathcal{Y}$ a subset of a Hilbert space and define $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. $\mathcal{X}$ is identified as the input space and $\mathcal{Y}$ as the output space. Given a learning algorithm A we define $f_{\mathcal{D}}$ to be the solution of the algorithm when the training set $\mathcal{D} = \{z_i = (x_i, y_i),\ i = 1, \ldots, m\} \in \mathcal{Z}^m$ drawn i.i.d. from a distribution $\mathbb{P}$ is used. Algorithm A is thus interpreted as a function from $\mathcal{Z}^m$ to $(\mathcal{Y})^{\mathcal{X}}$, the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$, and we use the notation $A(\mathcal{D}) = f_{\mathcal{D}}$. We denote by $\mathcal{D}^{\setminus i}$ the training set $\mathcal{D} \setminus \{z_i\}$ obtained by removing point (x_i, y_i). More formally, point i is replaced by the empty set which we assume the learning method treats as having this point simply removed – we will need this for our analysis below. We denote by $\mathcal{D}^i$ the training set obtained by changing point (x_i, y_i) from $\mathcal{D}$ into $z' = (x', y')$, that is the set $(\mathcal{D} \setminus \{z_i\}) \cup z'$.

For any point $z = (x, y)$ and function f (real valued or binary) we denote by $\ell(f, z)$ the loss (error) when $f(x)$ is predicted instead of y (ℓ is the loss function). We define the expected error of f also known as *generalization error* by the equation

$$R_{gen}[f] = \mathbf{E}_z[\ell(f, z)].$$

We also define the *empirical error* as

$$R_{emp}[f] = \frac{1}{m}\sum_{i=1}^{m} \ell(f, z_i)$$

and the *leave–one–out error* as

$$R_{loo}[f] = \frac{1}{m}\sum_{i=1}^{m} \ell(f_{\mathcal{D}^{\setminus i}}, z_i).$$

Note that the last two errors are functions of $\mathcal{D}$. For the case of classification we use $\theta(-yf(x))$ as the loss function ℓ, where $\theta(\cdot)$ is the Heavyside function. The analysis we will do concerns classification as well as regression. For the latter we will mainly focus on the case that ℓ is a Lipschitzian loss function, that is, we assume that there exists a positive constant B such that, for every $f_1, f_2 \in (\mathcal{Y})^{\mathcal{X}}$ and $z = (x,y) \in \mathcal{Z}$, there holds the inequality $|\ell(f_1,z) - \ell(f_2,z)| \leq B|y_1 - y_2|$. Note that the absolute value satisfies this condition with $B = 1$, whereas the square loss satisfies the condition provided the set $\mathcal{Y}$ is compact.

2.2 Hypothesis Stability

The first notion of stability we consider has been stated in (Bousquet and Elisseeff, 2002) and is inspired by the work of Devroye and Wagner (1979). It is very close to what Kearns and Ron (1999) defined as hypothesis stability:

Definition 1 (Hypothesis Stability) *An algorithm A has* hypothesis stability β_m *w.r.t. the loss function ℓ if the following holds:*

$$\forall i \in \{1, \ldots, m\},\ \mathbf{E}_{\mathcal{D},z}[|\ell(f_{\mathcal{D}}, z) - \ell(f_{\mathcal{D}^{\setminus i}}, z)|] \leq \beta_m.$$

It can be shown (Bousquet and Elisseeff, 2002) that when an algorithm has hypothesis stability β_m and for *all* training sets $\mathcal{D}$ we have, for every $z \in \mathcal{Z}$, that $0 \leq \ell(f_{\mathcal{D}}, z) \leq M$, M being a positive constant, then the following relation between the leave-one-out error and the expected error holds:

Theorem 2 (Hypothesis stability leave-one-out error bound) *Let $f_{\mathcal{D}}$ be the outcome of a learning algorithm with hypothesis stability β_m (w.r.t. a loss function ℓ such that $0 \leq \ell(f,z) \leq M$). Then with probability $1-\delta$ over the random draw of the training set $\mathcal{D}$,*

$$R_{gen}[f_{\mathcal{D}}] \leq R_{\ell oo}[f_{\mathcal{D}}] + \sqrt{\delta^{-1}\frac{M^2 + 6Mm\beta_m}{2m}}. \tag{1}$$

The proof consists of first bounding the second order moment of $(R_{gen}[f_{\mathcal{D}}] - R_{\ell oo}[f_{\mathcal{D}}])$ and then applying Chebychev's inequality. A similar bound on $(R_{gen}[f_{\mathcal{D}}] - R_{\ell oo}[f_{\mathcal{D}}])^2$ holds. Theorem 2 holds for any loss functions as long as stability can be proved w.r.t. this loss function.
In the following, we will say that an algorithm is stable when its stability scales like $1/m$, in which case the difference between the generalization and leave-one-out error is of the order $O(1/\sqrt{m})$. Many algorithms are stable according to this definition, see (Devroye et al., 1996; Bousquet and Elisseeff, 2002) for a discussion. For example, with respect to the classification loss, k-Nearest Neighbor ($k-$NN) is k/m stable. This is discussed in the next example.

Example 1 (Hypothesis Stability of k-Nearest Neighbor (k-NN)) *With respect to the classification loss, k-NN is at least $\frac{k}{m}$ stable. This can be seen via symmetrization arguments. For the sake of simplicity we give here the proof for the* 1*-NN only. Let v_i be the neighborhood of z_i such that the closest point in the training set to any point of v_i is z_i. The $1-NN$ algorithm computes its output via the following equation (we assume here that the probability that x_i appears twice in the training set is negligible):*

$$f_{\mathcal{D}}(x) = \sum_{i=1}^{m} y_i \mathbf{1}_{x\in v_i}(x)$$

where $\mathbf{1}_S$ is the indicator function of set S. The difference between the losses $\ell(f_{\mathcal{D}}, z)$ and $\ell(f_{\mathcal{D}^{\setminus i}}, z)$ is then defined by the set v_i. Here we assume that ℓ is the classification loss. We then have that

$$\mathbf{E}_z[|\ell(f_{\mathcal{D}_m}, z) - \ell(f_{\mathcal{D}^{\setminus i}}, z)|] \leq \mathbb{P}(v_i).$$

Note that v_i depends on $\mathcal{D}$. Now averaging over $\mathcal{D}$ we need to compute $\mathbf{E}_{\mathcal{D}}[\mathbb{P}(v_i)]$ which is the same for all i because the z_i are drawn i.i.d. from the same distribution. But, we have,

$$1 = \mathbf{E}_{\mathcal{D},z}[|f_{\mathcal{D}}(x)|] = \mathbf{E}_{\mathcal{D},z}\left[\left|\sum_{i=1}^{m} y_i \mathbf{1}_{x\in v_i}(x)\right|\right] = \mathbf{E}_{\mathcal{D},z}\left[\sum_{i=1}^{m} \mathbf{1}_{x\in v_i}(x)\right].$$

The last equality comes from the fact that for fixed $\mathcal{D}$ and z, only one $\mathbf{1}_{x\in v_i}(x)$ is non-zero. We also have that

$$1 = \mathbf{E}_{\mathcal{D},z}\left[\sum_{i=1}^{m} \mathbf{1}_{x\in v_i}(x)\right] = m\mathbf{E}_{\mathcal{D}}[\mathbb{P}(v_i)].$$

Consequently, $\mathbf{E}_{\mathcal{D}}[\mathbb{P}(v_i)] = \frac{1}{m}$ and the 1*-NN has hypothesis stability bounded above by $1/m$.*

A bound similar to Equation (1) can be derived for the empirical error when a slightly different notion of stability is used (Bousquet and Elisseeff, 2002).[2]

Definition 3 (Pointwise hypothesis stability) *An algorithm A has* pointwise hypothesis stability *β_m w.r.t. the loss function ℓ if the following holds :*

$$\forall i \in \{1,\ldots,m\},\ \mathbf{E}_{\mathcal{D},z}\left[\left|\ell(f_{\mathcal{D}}, z_i) - \ell(f_{\mathcal{D}^{\setminus i}\cup z}, z_i)\right|\right] \leq \beta_m.$$

Note that we adopted the same notation β_m for all notions of stability since it should always be clear from the context which is the referred notion. As for the case of hypothesis stability and leave-one-out error above, it can also be shown (Bousquet and Elisseeff, 2002) that when an algorithm has pointwise hypothesis stability β_m and if for all training sets $\mathcal{D}$, $0 \leq \ell(f, z) \leq M$, then the following relation between the empirical error and the expected error holds:

Theorem 4 (Pointwise hypothesis stability empirical error bound) *Let $f_{\mathcal{D}}$ be the outcome of a learning algorithm with pointwise hypothesis stability β_m (w.r.t. a loss function ℓ such that $0 \leq \ell(f_{\mathcal{D}}, z) \leq M$). Then with probability $1-\delta$ over the random draw of the training set $\mathcal{D}$,*

$$R_{gen}[f_{\mathcal{D}}] \leq R_{emp}[f_{\mathcal{D}}] + \sqrt{\delta^{-1}\frac{M^2 + 12Mm\beta_m}{2m}}. \tag{2}$$

2. We slightly changed the definition to correct one mistake that has been pointed out by Poggio et al., (2004): the difference of losses is taken here between two outcomes trained on datasets of equal sizes.

2.3 Uniform Stability

The application of bound (1) to different algorithms $f_1, \dots, f_Q$ with stabilities β_m^q, $q = 1, \dots, Q$, is usually done by using the union bound (Vapnik, 1998). Applying Theorem 2 Q times, we get with probability $1 - \delta$,

$$\forall q \in \{1, \dots, Q\}, \quad R_{gen}[f_q] \leq R_{\ell oo}[f_q] + \sqrt{\delta^{-1} Q \frac{M^2 + 6Mm\beta_m^q}{2m}}. \tag{3}$$

In such situations, we would like to have a dependence in $\log(Q)$ so that we can have large values of Q without increasing the bound too much. To this end, we need a stronger notion of stability called uniform stability (Bousquet and Elisseeff, 2002).

Definition 5 (Uniform Stability) *An algorithm A has* uniform stability β_m *w.r.t. the loss function ℓ if the following holds*

$$\forall \mathcal{D} \in \mathcal{Z}^m, \ \forall i \in \{1, \dots, m\}, \ \|\ell(f_{\mathcal{D}}, .) - \ell(f_{\mathcal{D}^{\setminus i}}, .)\|_\infty \leq \beta_m. \tag{4}$$

It is easily seen that the uniform stability is an upper bound on hypothesis and pointwise hypothesis stability (Bousquet and Elisseeff, 2002). Uniform stability can be used in the context of regression to get bounds as follows (Bousquet and Elisseeff, 2002):

Theorem 6 *Let $f_{\mathcal{D}}$ be the outcome of an algorithm with uniform stability β_m w.r.t. a loss function ℓ such that $0 \leq \ell(f_{\mathcal{D}}, z) \leq M$, for all $z \in \mathcal{Z}$ and all sets $\mathcal{D}$. Then, for any $m \geq 1$, and any $\delta \in (0,1)$, each of the following bounds holds with probability $1 - \delta$ over the random draw of the training set $\mathcal{D}$,*

$$R_{gen}[f_{\mathcal{D}}] \leq R_{emp}[f_{\mathcal{D}}] + 2\beta_m + (4m\beta_m + M)\sqrt{\frac{\log(1/\delta)}{2m}}, \tag{5}$$

and

$$R_{gen}[f_{\mathcal{D}}] \leq R_{\ell oo}[f_{\mathcal{D}}] + \beta_m + (4m\beta_m + M)\sqrt{\frac{\log(1/\delta)}{2m}}. \tag{6}$$

The dependence on δ is $\sqrt{\log(1/\delta)}$ which is better than the bounds given in terms of hypothesis and pointwise hypothesis stability.

It is important to note that these bounds hold only for regression. Uniform stability can also be used for classification with margin classifiers to get similar bounds, but we do not pursue this here for simplicity. In the next section, for simplicity we also consider random uniform stability only for regression. Classification can be treated with appropriate changes like in (Bousquet and Elisseeff, 2002).

Example 2 (Uniform Stability of regularization methods) *Regularization-based learning algorithms such as Regularization Networks (RN's) (Poggio and Girosi, 1990) and Support Vector Machines (SVM's), see, for example, (Vapnik, 1998), are obtained by minimizing the functional*

$$\sum_{i=1}^{m} \ell(f, z_i) + \lambda \|f\|_K^2$$

where $\lambda > 0$ is a regularization parameter and $\|f\|_K$ is the norm of f in a reproducing kernel Hilbert space associated to a symmetric and positive definite kernel $K : X \times X \to \mathbb{R}$. A typical example is the Gaussian, $K(x,t) = \exp(-\|x-t\|^2/2\sigma^2)$, where σ is a parameter controlling the width of the kernel. Depending on the loss function used, we obtain different learning methods. RN's use the square loss while SVM's regression uses the loss $\ell(f,z) = |f(x)-y|_\varepsilon$, where $|\xi|_\varepsilon = |\xi| - \varepsilon$ if $|\xi| > \varepsilon$, and zero otherwise.[3]

It can be shown (Bousquet and Elisseeff, 2002) that for Lipschitz loss functions, the uniform stability of these regularization methods scales as $1/\lambda$. This results is in agreement with the fact that for small λ, the solution tends to fit perfectly the data and Theorem 6 does not give an interesting bound. On the contrary, when λ is large the solution is more stable and Theorem 6 gives a tight bound. Hence, there is a trade-off between stability and deviation between generalization and empirical error that is illustrated here by the role of the regularization parameter λ.

Finally, we note that the notion of uniform stability may appear a little restrictive since the inequality in Equation (4) has to hold over all training sets $\mathcal{D}$. A weaker notion of stability has been introduced by Kutin and Niyogi (2002) with related exponential bounds. We do not discuss this issue here for simplicity, and we conjecture that the analysis we do below can be generally adapted for other notions of stability.

3. Stability and Generalization for Randomized Algorithms

The results summarized in the previous section concern only deterministic learning algorithms. For example they cannot be applied to certain neural networks as well as bagging methods. In this section we generalize the theory to include randomized learning algorithms.

3.1 Informal Reasoning

Let A be a randomized learning algorithm, that is a function from $\mathcal{Z}^m \times \mathcal{R}$ onto $(\mathcal{Y})^{\mathcal{X}}$ where $\mathcal{R}$ is a space containing elements $\mathbf{r}$ that model the randomization of the algorithm and is endowed with a probability measure $\mathbb{P}_\mathbf{r}$. For notational convenience, we will use the shorthand $f_{\mathcal{D},\mathbf{r}}$ to denote the outcome of the algorithm A applied on a training set $\mathcal{D}$ with a random parameter $\mathbf{r}$. We should distinguish between two types of randomness that are exemplified by the following examples.

Example 3 (Bootstrapping once) *Let $\mathcal{R} = \{1,\dots,m\}^p$, $p \le m$, and define $\mathbb{P}_\mathbf{r}$, for $\mathbf{r} \in \mathcal{R}$, to be a multinomial distribution with m parameters $(1/m,\dots,1/m)$. This random process models the sub-sampling with replacement of p elements from a set of m distinct elements. An algorithm A that takes as input a training set $\mathcal{D}$, performs a sub-sampling with replacement and runs a method such as a decision tree on the sub-sampled training set is typically modeled as a randomized algorithm taking as inputs a training set and an element $\mathbf{r} \in \mathcal{R}$ just described.*

In this first example we see that the randomness depends on m, which is different from what the second example describes.

3. Note that in the statistical learning theory literature (Vapnik, 1998), SVM are usually presented in term of mathematical programming problems and the parameter λ is replaced by $C = 1/(2\lambda)$ which now appears in front of the empirical error.

Example 4 (Initialization weights) *Let $\mathcal{R} = [0,1]^k$ and define $\mathbb{P}_{\mathbf{r}}$ to be the uniform distribution over $\mathcal{R}$. Such a random process appear in the initialization procedure of Neural Networks when the initial weights are chosen randomly. In the latter case, a multi-layer perceptron with k weights can be understood as an algorithm A taking a training set and a random vector $\mathbf{r} \in \mathcal{R}$ as inputs.*

We consider the following issues for the definitions of stability for randomized algorithms below.

- We give stability definitions that reduce to deterministic stability concepts when there is no randomness, that is, $\mathcal{R}$ is reduced to one element with probability 1.

- We assume that the randomness of an algorithm (randomness of $\mathbf{r}$) is independent of the training set $\mathcal{D}$, although $\mathbf{r}$ may depend on the size of this set, m. There are two main reasons for this: first, it simplifies the calculations; second, the randomness of $\mathbf{r}$ has generally nothing to do with the randomness of the training set $\mathcal{D}$. Most of the time our knowledge about the distribution over $\mathbf{r}$ is known perfectly, like in the examples above, and we would like to take advantage of that. Adding some dependencies between $\mathbf{r}$ and $\mathcal{D}$ reduces this knowledge since nothing is assumed about the distribution over $\mathcal{Z}$ from which $\mathcal{D}$ is drawn.

- We also consider the general case that the randomization parameter $\mathbf{r} \in \mathcal{R}^T$ is decomposed as a vector of independent random parameters $\mathbf{r} = (\mathbf{r}_1, \ldots, \mathbf{r}_T)$ where each $\mathbf{r}_t$ is drawn from the distribution $\mathbb{P}^t_{\mathbf{r}_t}$. For example, this model can be used to model the randomization of bagging (Breiman, 1996a), where each $\mathbf{r}_t$ corresponds to one random subsampling from the data, and the T subsamples are all drawn independently. To summarize, we will make use of the following assumption:

  Assumption 1: *We assume that $\mathbf{r} = (\mathbf{r}_1, \ldots, \mathbf{r}_T)$ where $\mathbf{r}_t$, $t = 1, \ldots, T$ are random elements drawn independently from the same distribution and write $\mathbf{r} \in \mathcal{R}^T$ to indicate the product nature of* $\mathbf{r}$.

- Finally we assume that we can re-use a draw of $\mathbf{r}$ for different training set sizes, for example for m and $m-1$. We need this assumption for the definitions of stability below to be well defined as well as for the leave-one-out error definition we use for randomized methods.

To develop the last issue further, let us consider how to compute a leave-one-out error estimate when the algorithm depends on a random vector $\mathbf{r}$ that changes with the number of training examples. One way is to sample a new random vector $\mathbf{r}$ (which in this case will concern only $m-1$ training points) for each fold/iteration. This is done, for example, by Kearns and Ron (1999) when they introduce the notion of the random error stability. However, this introduces more instabilities to the algorithms whose behavior can be different not only because of changes in the training set but also because of changes in the random part $\mathbf{r}$. A more stable leave-one-out procedure for a randomized algorithm would be to fix $\mathbf{r}$ and to apply the leave-one-out method only on the sampling of the training set – a deterministic leave-one-out error (Evgeniou et al., 2004). Therefore for each leave-one-out iteration, when we leave one point out — which is replaced, as we discussed in Section 2.1, with an empty set which we assume the learning method does not use — we use the same $\mathbf{r}$ for the remaining $m-1$ points. For instance, in Example 3.1 we would use the same bootstrap samples that we used when having all m points, with the point left out replaced by the empty set that is not used for training, for each leave-one-out iteration. In that case, we don't need to re-sample $\mathbf{r}$ and the leave-one-out estimate concerns an algorithm that is closer to what we consider on m points.

Therefore, in what follows, keeping in mind Example 3, we assume the following:

Assumption 2: *The same* $\mathbf{r}$ *can be applied to* $f_{\mathcal{D}}$ *and* $f_{\mathcal{D}^{\setminus i}}$ *where* $\mathcal{D}^{\setminus i}$ *is the set* $\mathcal{D}$ *where point* i *is replaced by the empty set. We also consider the deterministic leave-one-out error computed as described above.*

Note that this assumption is not restrictive about the kind of learning methods we can consider. For example both in Example 3.1 and 3.2 the same $\mathbf{r}$ (i.e. subsamples or initialization of neural network weights) can be used for m and $m-1$ training points.

3.2 Random Hypothesis Stability

The first definition we consider is inspired by the hypothesis stability for deterministic algorithms.

Definition 7 (Random Hypothesis Stability) *A randomized algorithm A has* random hypothesis stability β_m *w.r.t. the loss function* ℓ *if the following holds:*

$$\forall i \in \{1,\ldots,m\}, \mathbf{E}_{\mathcal{D},z,\mathbf{r}}\left[\left|\ell(f_{\mathcal{D},\mathbf{r}},z) - \ell(f_{\mathcal{D}^{\setminus i},\mathbf{r}},z)\right|\right] \leq \beta_m. \tag{7}$$

Note that the value in the left hand side (l.h.s.) of Equation (7) can vary for different indexes i. If $\mathbf{r}$ is fixed then the random hypothesis stability is exactly the same as the hypothesis stability except that the resulting algorithm need not be symmetric anymore: if we sample the training data using a fixed $\mathbf{r}$, permuting two data points might lead to different samplings and hence to a different outcome. This means that we cannot apply the results for the case of deterministic algorithms and we have to consider other bounds on the variance of the difference between the generalization and empirical (or leave-one-out) errors. We prove in the appendix the following lemma.

Lemma 8 *For any (non-symmetric) learning algorithm A and loss function* ℓ *such that* $0 \leq \ell(f,z) \leq M$ *we have for the leave-one-out error:*

$$\mathbf{E}_{\mathcal{D}}\left[(R_{gen} - R_{\ell oo})^2\right] \leq \frac{2M^2}{m} + \frac{12M}{m}\sum_{i=1}^{m}\mathbf{E}_{\mathcal{D},z}\left[\left|\ell(f_{\mathcal{D}},z) - \ell(f_{\mathcal{D}^{\setminus i}},z)\right|\right]. \tag{8}$$

Using Chebychev's inequality, this lemma leads to the inequality

$$\mathbb{P}_{\mathcal{D}}\left(R_{gen}[f_{\mathcal{D},\mathbf{r}}] - R_{\ell oo}[f_{\mathcal{D},\mathbf{r}}] \geq \varepsilon \mid \mathbf{r}\right) \leq \frac{2M^2}{m\varepsilon^2} + \frac{12M\sum_{i=1}^{m}\mathbf{E}_{\mathcal{D},z}\left[\left|\ell(f_{\mathcal{D},\mathbf{r}},z) - \ell(f_{\mathcal{D}^{\setminus i},\mathbf{r}},z)\right|, \mathbf{r}\right]}{m\varepsilon^2}, \tag{9}$$

where we use the notation $\mathbb{E}[X,Y]$ for the expectation of X conditioned on Y, and $\mathbb{P}[.|\mathbf{r}]$ for the conditional probability. By integrating Equation (9) with respect to $\mathbf{r}$ and using the property $\mathbb{E}_Y\left[\mathbb{E}_X[g(X,Y)|Y]\right] = \mathbb{E}_{X,Y}[g(X,Y)]$ we derive the following theorem about the generalization and leave-one-out errors of randomized learning methods:

Theorem 9 *Let $f_{\mathcal{D},\mathbf{r}}$ be the outcome of a randomized algorithm with random hypothesis stability β_m w.r.t. a loss function ℓ such that $0 \leq \ell(f,z) \leq M$, for all $y \in \mathcal{Y}$, $\mathbf{r} \in \mathcal{R}$ and all sets $\mathcal{D}$. Then with probability $1-\delta$ with respect to the random draw of the $\mathcal{D}$ and $\mathbf{r}$,*

$$R_{gen}(f_{\mathcal{D},\mathbf{r}}) \leq R_{\ell oo}[f_{\mathcal{D},\mathbf{r}}] + \sqrt{\delta^{-1}\frac{2M^2 + 12Mm\beta_m}{m}}. \tag{10}$$

Notice that in the case that we make Assumption 1 nothing changes since the integration of (9) w.r.t. $\mathbf{r}$ does not depend on the decomposition nature of $\mathbf{r}$ made in Assumption 1.

As in the deterministic case, it is possible to define a different notion of stability to derive bounds on the deviation between the empirical error and the generalization error of randomized algorithms:

Definition 10 (Random Pointwise Hypothesis Stability) *A randomized algorithm A has* random pointwise hypothesis stability *β_m w.r.t. the loss function ℓ if the following holds:*

$$\forall i \in \{1,\dots,m\}, E_{\mathcal{D}_m,\mathbf{r},z}\left|\ell(f_{\mathcal{D},\mathbf{r}},z_i) - \ell(f_{\mathcal{D}^{\setminus i}\cup z,\mathbf{r}},z_i)\right| \leq \beta_m. \tag{11}$$

Using the following lemma proved in the appendix,

Lemma 11 *For any (non-symmetric) learning algorithm A and loss function ℓ such that $0 \leq \ell(f,z) \leq M$ we have for the empirical error,*

$$\mathbf{E}_{\mathcal{D}}\left[(R_{gen} - R_{emp})^2\right] \leq \frac{2M^2}{m} + \frac{12M}{m}\sum_{i=1}^{m}\mathbf{E}_{\mathcal{D},z}\left[|\ell(f_{\mathcal{D}},z_i) - \ell(f_{\mathcal{D}^{\setminus i}\cup z},z_i)|\right]. \tag{12}$$

we can derive as before the theorem:

Theorem 12 *Let $f_{\mathcal{D},\mathbf{r}}$ be the outcome of a random algorithm with random pointwise hypothesis stability β_m w.r.t. a loss function ℓ such that $0 \leq \ell(f,z) \leq M$, for all $y \in \mathcal{Y}$, $\mathbf{r} \in \mathcal{R}$ and all sets $\mathcal{D}$. Then with probability $1-\delta$ with respect to the random draw of the $\mathcal{D}$ and $\mathbf{r}$,*

$$R_{gen}(f_{\mathcal{D},\mathbf{r}}) \leq R_{emp}[f_{\mathcal{D},\mathbf{r}}] + \sqrt{\delta^{-1}\frac{2M^2 + 12Mm\beta_m}{m}}. \tag{13}$$

We note that both for Theorems 9 and 12 (Lemmas 8 and 11) one can further improve the constants of the bounds – as is typically the case with bounds in the literature.

The parallel with the deterministic case is striking. However when we consider a random space $\mathcal{R}$ reduced to only one element, then the bounds we obtain here are worse since we assume non-symmetric learning algorithms.

3.3 Random Uniform Stability

The uniform stability definition (Definition 5) for deterministic algorithms can be extended as follows:

Definition 13 (Uniform Stability of Randomized Algorithms) *We say that a randomized learning algorithm has uniform stability β_m w.r.t. the loss function ℓ if, for every $i = 1, \ldots, m$*

$$\sup_{\mathcal{D},z} \left| \mathbf{E}_{\mathbf{r}} \left[\ell(f_{\mathcal{D},\mathbf{r}}, z) \right] - \mathbf{E}_{\mathbf{r}} \left[\ell(f_{\mathcal{D}^{\setminus i},\mathbf{r}}, z) \right] \right| \leq \beta_m. \tag{14}$$

Note that this definition is consistent with Definition 5 which holds for deterministic symmetric learning algorithms.

To link uniform stability to generalization, the following result by McDiarmid (1989), see also (Devroye et al., 1996), is central.

Theorem 14 (Bounded Difference Inequality) *Let $\mathbf{r} = (\mathbf{r}_1, \ldots, \mathbf{r}_T) \in \mathcal{R}$ be T independent random variables ($\mathbf{r}_t$ can be vectors, as in Assumption 1, or scalars) drawn from the same probability distribution $\mathbb{P}_{\mathbf{r}}$. Assume that the function $G : \mathcal{R}^T \rightarrow \mathbb{R}$ satisfies*

$$\sup_{\mathbf{r}_1,\ldots,\mathbf{r}_T,\mathbf{r}'_t} \left| G(\mathbf{r}_1, \ldots, \mathbf{r}_T) - G(\mathbf{r}_1, \ldots, \mathbf{r}_{t-1}, \mathbf{r}'_t, \mathbf{r}_{t+1}, \ldots, \mathbf{r}_T) \right| \leq c_t,\ t = 1, \ldots, T. \tag{15}$$

where c_t is a nonnegative function of t. Then, for every $\varepsilon > 0$

$$\mathbb{P}\left[G(\mathbf{r}_1, \ldots, \mathbf{r}_T) - \mathbf{E}_{\mathbf{r}}\left[G(\mathbf{r}_1, \ldots, \mathbf{r}_T)\right] \geq \varepsilon\right] \leq \exp\left\{ -2\varepsilon^2 / \sum_{t=1}^{T} c_t^2 \right\}. \tag{16}$$

For the next theorem we replace the G of Theorem 14 with $\ell(f_{\mathcal{D},\mathbf{r}}, z)$ and require that, for every $\mathcal{D} \in \mathcal{Z}^m$ and $z \in \mathcal{Z}$, $\ell(f_{\mathcal{D},\mathbf{r}}, z)$ satisfies the inequality in Equation (15). This is a mild assumption but the bounds below will be interesting only if, for $T \rightarrow \infty$, c_t goes to zero at least as $1/\sqrt{T}$. We use ρ as the supremum of the c_ts of Theorem 14.

Theorem 15 *Let $f_{\mathcal{D},\mathbf{r}}$ be the outcome of a randomized learning algorithm satisfying Assumptions 1 and 2 with uniform stability β_m w.r.t. the loss function ℓ. Let ρ be such that for all t*

$$\sup_{\mathbf{r}_1,\ldots,\mathbf{r}_T,\mathbf{r}'_t} \sup_{z} \left| \ell(f_{\mathcal{D},(\mathbf{r}_1,\ldots,\mathbf{r}_T)}, z) - \ell(f_{\mathcal{D},(\mathbf{r}_1,\ldots,\mathbf{r}_{t-1},\mathbf{r}'_t,\mathbf{r}_{t+1},\ldots,\mathbf{r}_T)}, z) \right| \leq \rho,$$

as in Equation (15) for G being $\ell(f_{\mathcal{D},\mathbf{r}}, z)$ and $\mathbf{r} = (\mathbf{r}_1, \ldots, \mathbf{r}_T)$. The following bound holds with probability at least $1 - \delta$ with respect to the random draw of the $\mathcal{D}$ and $\mathbf{r}$,

$$R_{gen}(f_{\mathcal{D},\mathbf{r}}) \leq R_{emp}(f_{\mathcal{D},\mathbf{r}}) + 2\beta_m + \left(\frac{M + 4m\beta_m}{\sqrt{2m}} + \sqrt{2T}\rho \right) (\sqrt{\log 2/\delta}), \tag{17}$$

and,

$$R_{gen}(f_{\mathcal{D},\mathbf{r}}) \leq R_{\ell oo}(f_{\mathcal{D},\mathbf{r}}) + \beta_m + \left(\frac{M + 2m\beta_{m-1} + 2m\beta_m}{\sqrt{2m}} + \sqrt{2T}\rho \right) (\sqrt{\log(2/\delta)}). \tag{18}$$

Furthermore, assuming that β_{m-1}, the random uniform stability for training sets of size $m-1$, is greater than β_m, we can simplify Equation (18) to:

$$R_{gen}(f_{\mathcal{D},\mathbf{r}}) \leq R_{\ell oo}(f_{\mathcal{D},\mathbf{r}}) + \beta_m + \left(\frac{M + 4m\beta_{m-1}}{\sqrt{2m}} + \sqrt{2T}\rho \right) (\sqrt{\log(2/\delta)}). \tag{19}$$

Notice that the assumption for the simplification we make in the theorem that β_{m-1} is greater than β_m is natural: when points are added to the training set, the outcome of a learning algorithm is usually more stable. Moreover, bounds on β_m can be used here so that the condition $\beta_{m-1} \geq \beta_m$ can be replaced by a condition on these bounds: we would require that the bounds on β_m are non-increasing in m.

We note that ρ may depend both on the number of random variables T and the number of training data m. In the bagging example below we estimate a bound on ρ that depends only on T, the number of subsamples we do for the bagging process – it may or may not be possible to show that ρ depends on m, too, but this is an open question. We do not know of an example where ρ also depends on m or, alternatively, of a case where it can be shown that it is not possible to have ρ depend on m. The latter case would imply that for fixed T the empirical (leave-one-out) error does not converge to the expected error as m increases. This is, however, an open question and potentially a weakness for the framework we develop here.

Finally note that, as in the deterministic case discussed in Section 2, results similar to those in Theorem 15 can be given for classification following the same line as in (Bousquet and Elisseeff, 2002).

4. Stability of Bagging and Subbagging

In this section we discuss an application of the results derived above to bagging (Breiman, 1996a) and subbagging, see, for example, (Andonova et al., 2002), two randomized algorithms which work by averaging the solutions of a learning algorithm trained a number of times on random subsets of the training set. We will analyze these methods within the stability framework presented above. To this end, we need to study how bagging and subbagging "affect" the stability of the base (underlying) learning algorithm. First we present more formally what we mean by bagging.

4.1 Bagging

Bagging consists of training the same learning algorithm on a number T of different bootstrap sets of a training set $\mathcal{D}$ and by averaging the obtained solutions. We denote these bootstrap sets by $\mathcal{D}(\mathbf{r}_t)$ for $t = 1, \ldots, T$, where the $\mathbf{r}_t \in \mathcal{R} = \{1, \ldots, m\}^m$ are instances of a random variable corresponding to sampling *with* replacement of m elements from the training set $\mathcal{D}$ (recall the notation in Example 3). Such random variables have a multinomial distribution with parameters $(\frac{1}{m}, \ldots, \frac{1}{m})$. The overall bagging model can thus be written as:

$$F_{\mathcal{D},\mathbf{r}} = \frac{1}{T} \sum_{t=1}^{T} f_{\mathcal{D}(\mathbf{r}_t)}. \tag{20}$$

Here we assume that the base learning method ($f_{\mathcal{D}}$) treats multiple copies of a training point (for example when many copies of the same point are sampled) as one point.[4] Extending the results below to the case where multiple copies of a point are treated as such is an open question.

The reader should also keep in mind that the base learning algorithm may be itself randomized with random parameter $\mathbf{s}$. When trained on the $t-$th bootstrap set, $\mathcal{D}(\mathbf{r}_t)$, this algorithm will output the solution $f_{\mathcal{D}(\mathbf{r}_t),\mathbf{s}_t}$. However, to simplify the notation, we suppress the symbol $\mathbf{s}_t$ in our discussion below.

In what follows, we compute an upper bound on the random hypothesis stability for bagging. For regression, we have then the following proposition:

Proposition 4.1 (Random hypothesis stability of bagging for regression) *Assume that the loss ℓ is $B-$lipschitzian w.r.t. its first variable. Let $F_{\mathcal{D},\mathbf{r}}$, $\mathbf{r} \in \mathcal{R}^T$, be the outcome of a bagging algorithm whose base machine ($f_{\mathcal{D}}$) has (pointwise) hypothesis stability γ_m w.r.t. the ℓ_1 loss function. Then the random (pointwise) hypothesis stability β_m of $F_{\mathcal{D},\mathbf{r}}$ with respect to ℓ is bounded by*

$$\beta_m \leq B \sum_{k=1}^{m} \frac{k\gamma_k}{m} \mathbb{P}_{\mathbf{r}}\left[d(\mathbf{r}) = k\right],$$

where $d(\mathbf{r})$, $\mathbf{r} \in \mathcal{R}$, is the number of distinct sampled points in one bootstrap iteration.

Proof

We first focus on hypothesis stability. Let us assume first that $\mathcal{D}$ is fixed and z too. We would like to bound:

$$I(\mathcal{D},z) = \mathbf{E}_{\mathbf{r}_1,\ldots,\mathbf{r}_T}\left[\left|\ell\left(\frac{1}{T}\sum_{t=1}^{T} f_{\mathcal{D}(\mathbf{r}_t)}, z\right) - \ell\left(\frac{1}{T}\sum_{t=1}^{T} f_{\mathcal{D}^{\setminus i}(\mathbf{r}_t)}, z\right)\right|\right]$$

where $\mathbf{r}_1, \ldots, \mathbf{r}_T$ are i.i.d. random variables modeling the random sampling of bagging and having the same distribution as $\mathbf{r}$. Since ℓ is $B-$lipschitzian, and the $\mathbf{r}_t$ are i.i.d., $I(\mathcal{D},z)$ can be bounded as:

$$\begin{aligned} I(\mathcal{D},z) &\leq \frac{B}{T}\,\mathbf{E}_{\mathbf{r}_1,\ldots,\mathbf{r}_T}\left[\left|\sum_{t=1}^{T}\left(f_{\mathcal{D}(\mathbf{r}_t)}(x) - f_{\mathcal{D}^{\setminus i}(\mathbf{r}_t)}(x)\right)\right|\right] \\ &\leq \frac{B}{T}\sum_{t=1}^{T}\mathbf{E}_{\mathbf{r}_t}\left[\left|f_{\mathcal{D}(\mathbf{r}_t)}(x) - f_{\mathcal{D}^{\setminus i}(\mathbf{r}_t)}(x)\right|\right] = B\,\mathbf{E}_{\mathbf{r}}\left[\left|f_{\mathcal{D}(\mathbf{r})}(x) - f_{\mathcal{D}^{\setminus i}(\mathbf{r})}(x)\right|\right]. \end{aligned}$$

To simplify the notation we denote by $\Delta(\mathcal{D}(\mathbf{r}),x)$ the difference between $f_{\mathcal{D}^{\setminus i}(\mathbf{r})}(x)$ and $f_{\mathcal{D}(\mathbf{r})}(x)$. We have that

$$\begin{aligned} \mathbf{E}_{\mathbf{r}}\left[|\Delta(\mathcal{D}(\mathbf{r}),x)|\right] &= \mathbf{E}_{\mathbf{r}}\left[|\Delta(\mathcal{D}(\mathbf{r}),x)|\,(\mathbf{1}_{i\in\mathbf{r}} + \mathbf{1}_{i\notin\mathbf{r}})\right] \\ &= \mathbf{E}_{\mathbf{r}}\left[|\Delta(\mathcal{D}(\mathbf{r}),x)|\,\mathbf{1}_{i\in\mathbf{r}}\right] + \mathbf{E}_{\mathbf{r}}\left[|\Delta(\mathcal{D}(\mathbf{r}),x)|\,\mathbf{1}_{i\notin\mathbf{r}}\right]. \end{aligned}$$

Note that the second part of the last line is equal to zero because when i is not in $\mathbf{r}$, point z_i does not belong to $\mathcal{D}(\mathbf{r})$ and, thus, $\mathcal{D}(\mathbf{r}) = \mathcal{D}^{\setminus i}(\mathbf{r})$. We conclude that

$$I(\mathcal{D},z) \leq B\mathbf{E}_{\mathbf{r}}\left[|\Delta(\mathcal{D}(\mathbf{r}),x)|\,\mathbf{1}_{i\in\mathbf{r}}\right].$$

4. This means that if for example the underlying learning algorithm is a neural network, this algorithm is modified by a preprocessing step so that the training set consists only of distinct data points.

We now take the average w.r.t. $\mathcal{D}$ and z:

$$\mathbf{E}_{\mathcal{D},z}\left[I(\mathcal{D},z)\right] \leq B\mathbf{E}_{\mathbf{r},\mathcal{D},x}\left[|\Delta(\mathcal{D}(\mathbf{r}),x)|\mathbf{1}_{i\in\mathbf{r}}\right] =$$

$$= B\mathbf{E}_{\mathbf{r}}\left[\mathbf{E}_{\mathcal{D},x}\left[|\Delta(\mathcal{D}(\mathbf{r}),x)|\right]\mathbf{1}_{i\in\mathbf{r}}\right] = B\mathbf{E}_{\mathbf{r}}\left[\gamma_{d(\mathbf{r})}\mathbf{1}_{i\in\mathbf{r}}\right], \quad (21)$$

where the last equality follows by noting that $\mathbf{E}_{\mathcal{D},x}\left[|\Delta(\mathcal{D}(\mathbf{r}),x)|\right]$ is bounded by the hypothesis stability $\gamma_{d(\mathbf{r})}$ of a training set of size $d(\mathbf{r})$. We now note that when averaging w.r.t. $\mathbf{r}$, the important variable about $\mathbf{r}$ is the size $d(\mathbf{r})$:

$$\mathbf{E}_{\mathbf{r}}\left[\gamma_{d(\mathbf{r})}\mathbf{1}_{i\in\mathbf{r}}\right] = \sum_{k=1}^{m}\mathbb{P}_{\mathbf{r}}\left[d(\mathbf{r})=k\right]\gamma_k\mathbf{E}_{\mathbf{r}}\left[\mathbf{1}_{i\in\mathbf{r}};d(\mathbf{r})=k\right].$$

Now note that, by symmetry, $\mathbf{E}_{\mathbf{r}}\left[\mathbf{1}_{i\in\mathbf{r}};d(\mathbf{r})=k\right]=k/m$. This concludes the proof for hypothesis stability. The proof for pointwise stability is exactly the same except that in Equation (21) there is no expectation w.r.t. z and z is replaced by z_i. ■

The bounds we just proved depend on the quantities $\mathbb{P}_{\mathbf{r}}[d(\mathbf{r})=k]$, where, we recall that $d(\mathbf{r})$, $\mathbf{r}\in\mathcal{R}$, is the number of distinct sampled points in one bootstrap iteration. It can be shown, for example by applying Theorem 14, that the random variable $d(\mathbf{r})$ is sharply concentrated around its mode which is for $k=(1-\frac{1}{e})m\approx 0.632m$. For that reason, in what follows we will assume that the previous bounds can be approximately rewritten as:

$$\beta_m \leq .632B\gamma_{.632m}.$$

For example if $B=1$ and γ_m scales appropriately with m the bounds on the random (pointwise) hypothesis stability of the bagging predictor are better than those on the (pointwise) hypothesis stability of a single predictor trained on the whole training set. Notice also that .632 is the probability that the bootstrapped set will contain a specific (any) point, also used to justify the .632 bootstrap error estimates (Efron and Tibshirani, 1997).

Similar results can be shown for the random (pointwise) hypothesis stability for classification. In particular:

Proposition 4.2 (Random hypothesis stability of bagging for classification) *Let $F_{\mathcal{D},\mathbf{r}}$ be the outcome of a bagging algorithm whose base machine has (pointwise) hypothesis stability γ_m w.r.t. the classification loss function. Then, the (pointwise) random hypothesis stability β_m of $F_{\mathcal{D},\mathbf{r}}$ w.r.t. the ℓ_1 loss function is bounded by*

$$\beta_m \leq 2\sum_{k=1}^{m}\frac{k\gamma_k}{m}\mathbb{P}_{\mathbf{r}}\left[d(\mathbf{r})=k\right].$$

Proof The proof is the same as in the above proposition except that the loss appearing therein is the ℓ_1 loss and, so, $B=1$. The functions $f^{(t)}$ being $\{+1,-1\}$ valued, the term:

$$\mathbf{E}_{\mathcal{D},z}\left[|f_{\mathcal{D}}(x)-f_{\mathcal{D}^{\setminus i}}(x)|\right]$$

is equal to the term

$$2\mathbf{E}_{\mathcal{D},z}\left[\theta(-yf_{\mathcal{D}}(x))-\theta(-yf_{\mathcal{D}^{\setminus i}}(x))\right].$$

So that stability w.r.t. the ℓ_1 loss function can be replaced by stability w.r.t. the classification loss, and the proof can be transposed directly. ■

Example 5 (k-NN) *As previously seen, k-NN has hypothesis stability equal to $\frac{k}{m}$ such that bagging k-NN has stability with respect to classification loss bounded by*

$$2\sum_{j=1}^{m}\frac{j\beta_j}{m}\mathbb{P}_{\mathbf{r}}\left[d(\mathbf{r})=j\right]=2\sum_{j=1}^{m}\frac{j\frac{k}{j}}{m}\mathbb{P}_{\mathbf{r}}\left[d(\mathbf{r})=j\right]=2\frac{k}{m}\sum_{j=1}^{m}\mathbb{P}_{\mathbf{r}}\left[d(\mathbf{r})=j\right]=2\frac{k}{m}$$

So bagging does not improve stability, which is also experimentally verified by Breiman (1996a).

The next proposition establishes the link between the uniform stability of bagging and that of the base learning algorithm for regression. As before, classification can be treated similarly, see (Bousquet and Elisseeff, 2002).

Proposition 4.3 (Random uniform stability of bagging for regression) *Assume that the loss ℓ is B-lipschitzian with respect to its first variable. Let $F_{\mathcal{D},\mathbf{r}}$ be the outcome of a bagging algorithm whose base machine has uniform stability γ_m w.r.t. the ℓ_1 loss function. Then the random uniform stability β_m of $F_{\mathcal{D},\mathbf{r}}$ with respect to ℓ is bounded by*

$$\beta_m \leq B\sum_{k=1}^{m}\frac{k\gamma_k}{m}\mathbb{P}_{\mathbf{r}}\left[d(\mathbf{r})=k\right]. \tag{22}$$

Proof The random uniform stability of bagging is given by

$$\beta_m=\sup_{\mathcal{D},z}\left|\mathbf{E}_{\mathbf{r}_1,\ldots,\mathbf{r}_t}\left[\ell\left(\frac{1}{T}\sum_{t=1}^{T}f_{\mathcal{D}(\mathbf{r}_t)},z\right)-\ell\left(\frac{1}{T}\sum_{t=1}^{T}f_{\mathcal{D}^{\setminus i}(\mathbf{r}_t)},z\right)\right]\right|.$$

This can be bound by taking the absolute valued inside the expectation. Then, following the same lines as in the proof of Proposition 4.1 we have:

$$\beta_m \leq B\sup_{\mathcal{D},x}\left\{\mathbf{E}_{\mathbf{r}}\left[\Delta(\mathcal{D}(\mathbf{r}),x)\mathbf{1}_{i\in\mathbf{r}}\right]\right\}$$

where, we recall, $\Delta(\mathcal{D}(\mathbf{r}),x)=|f_{\mathcal{D}(\mathbf{r})}-f_{\mathcal{D}^{\setminus i}(\mathbf{r})}|$ and function $\mathbf{1}_{i\in\mathbf{r}}$ is equal to one if point i is sampled during bootstrapping and zero otherwise. We then have

$$\beta_m \leq B\,\mathbf{E}_{\mathbf{r}}\left[\sup_{\mathcal{D},x}\left\{\Delta(\mathcal{D}(\mathbf{r}),x)\right\}\mathbf{1}_{i\in\mathbf{r}}\right].$$

Now we observe that

$$\sup_{\mathcal{D},x}\left\{\Delta(\mathcal{D}(\mathbf{r}),x)\right\}=\sup_{\mathcal{D}(\mathbf{r}),x}\left\{\Delta(\mathcal{D}(\mathbf{r}),x)\right\}=\gamma_{d(\mathbf{r})}.$$

Placing this bound in the previous one gives

$$\beta_m \leq \mathbf{E}_{\mathbf{r}}\left[\gamma_{d(\mathbf{r})}\mathbf{1}_{i\in\mathbf{r}}\right].$$

The proof is now exactly the same as in the final part of Proposition 4.1. ■

Example 6 (SVM regression) *We have seen in Example 2 that the uniform stability of a SVM w.r.t. the ℓ_1 loss is bounded by $1/\lambda$. The uniform stability of bagging SVM is then roughly bounded by $0.632/\lambda$ if the SVM is trained on all bootstrap sets with the same λ. So that the bound on the random uniform stability of a bagged SVM is better than the bound on the uniform stability for a single SVM trained on the whole training set with the same λ.*

4.2 Subbagging

Subbagging is a variation of bagging where the sets $\mathcal{D}(\mathbf{r}_t)$, $t = 1, \ldots, T$ are obtained by sampling $p \leq m$ points from $\mathcal{D}$ *without* replacement. Like in bagging, a base learning algorithm is trained on each set $\mathcal{D}(\mathbf{r}_t)$ and the obtained solutions $f_{\mathcal{D}(\mathbf{r}_t)}$ are combined by average.

The proofs above can then be used here directly which gives the following upper bounds on stability for subbagging:

Proposition 4.4 (Stability of subbagging for regression) *Assume that the loss ℓ is B-lipschitzian w.r.t. its first variable. Let $F_{\mathcal{D},\mathbf{r}}$ be the outcome of a subbagging algorithm whose base machine is symmetric and has uniform (resp. hypothesis or pointwise hypothesis) stability γ_m w.r.t. the ℓ_1 loss function, and subbagging is done by sampling p points without replacement. Then the random uniform (resp. hypothesis or pointwise hypothesis) stability β_m of $F_{\mathcal{D},\mathbf{r}}$ w.r.t. ℓ is bounded by*

$$\beta_m \leq B\gamma_p \frac{p}{m}.$$

For classification, we have also the following proposition, again only for hypothesis or pointwise hypothesis stability as in Section 2:

Proposition 4.5 ((P.) Hypothesis stability of subbagging for classification) *Let $F_{\mathcal{D},\mathbf{r}}$ be the outcome of a subbagging algorithm whose base machine is symmetric and has hypothesis (resp. pointwise hypothesis) stability γ_m with respect to classification loss, and subbagging is done by sampling p points without replacement. Then the random hypothesis (resp. pointwise hypothesis) stability β_m of $F_{\mathcal{D},\mathbf{r}}$ with respect to the ℓ_1 loss function is bounded by*

$$\beta_m \leq 2\gamma_p \frac{p}{m}.$$

4.3 Bounds on the Performance of Subbagging

We can now prove bounds on the performance of bagging and subbagging. We present the following theorems for subbagging but the same statements hold true for bagging where, in the bounds below, $\frac{p\gamma_p}{m}$ is replaced by $\sum_{k=1}^{m} \frac{k\gamma_k}{m} \mathbb{P}_{\mathbf{r}}[d(\mathbf{r}) = k]$ which is roughly equal to $0.632\gamma_{0.632m}$ when m is sufficiently large.

Theorem 16 *Assume that the loss ℓ is B-lipschitzian w.r.t. its first variable. Let $F_{\mathcal{D},\mathbf{r}}$ be the outcome of a subbagging algorithm. Assume subbagging is done with T sets of size p subsampled without replacement from $\mathcal{D}$ and the base learning algorithm has hypothesis stability γ_m and pointwise* *hypothesis stability γ'_m, both stabilities being w.r.t. the ℓ loss. The following bounds hold separately with probability at least* $1-\delta$

$$R_{gen}(F_{\mathcal{D},\mathbf{r}}) \leq R_{\ell oo}(F_{\mathcal{D},\mathbf{r}}) + \sqrt{\delta^{-1}\frac{2M^2+12MBp\gamma_p}{m}} \tag{23}$$

$$R_{gen}(F_{\mathcal{D},\mathbf{r}}) \leq R_{emp}(F_{\mathcal{D},\mathbf{r}}) + \sqrt{\delta^{-1}\frac{2M^2+12MBp\gamma'_p}{m}}. \tag{24}$$

Proof The inequalities follow directly from plugging the result of Proposition 4.4 in Theorems 9 and 12 respectively. ■

Note that, as in Proposition 4.2, the same result holds for classification if we set $B=2$ and $M=1$.

The following theorem holds for regression. The extension to the case of classification can be done again as in (Bousquet and Elisseeff, 2002).

Theorem 17 *Assume that the loss ℓ is B-lipschitzian w.r.t. its first variable. Let $F_{\mathcal{D},\mathbf{r}}$ be the outcome of a subbagging algorithm. Assume subbagging is done with T sets of size p subsampled without replacement from $\mathcal{D}$ and the base learning algorithm has uniform stability γ_m w.r.t. the ℓ loss. The following bounds hold separately with probability at least $1-\delta$ in the case of regression*

$$R_{gen}(F_{\mathcal{D},\mathbf{r}}) \leq R_{\ell oo}(F_{\mathcal{D},\mathbf{r}}) + \frac{Bp\gamma_p}{m} + \left(\frac{M+4B(m/m-1)p\gamma_p}{\sqrt{2m}} + \frac{\sqrt{2}BM}{\sqrt{T}}\right)\sqrt{\log(2/\delta)}, \tag{25}$$

and

$$R_{gen}(F_{\mathcal{D},\mathbf{r}}) \leq R_{emp}(F_{\mathcal{D},\mathbf{r}}) + 2\frac{Bp\gamma_p}{m} + \left(\frac{M+4Bp\gamma_p}{\sqrt{2m}} + \frac{\sqrt{2}BM}{\sqrt{T}}\right)\sqrt{\log 2/\delta}. \tag{26}$$

Proof We recall that $\mathbf{r}=(\mathbf{r}_1,\ldots,\mathbf{r}_T)$ and introducethe notation

$$\mathbf{r}^t = (\mathbf{r}_1,\ldots,\mathbf{r}_{t-1},\mathbf{r}',\mathbf{r}_{t+1},\ldots,\mathbf{r}_T).$$

Note that

$$\left|\ell(F_{\mathcal{D},\mathbf{r}},z)-\ell(F_{\mathcal{D},\mathbf{r}^t},z)\right| = \left|\ell\left(\sum_{s=1}^{T} f_{\mathcal{D}(\mathbf{r}_s)},z\right) - \ell\left(\sum_{s=1,s\neq t}^{T} f_{\mathcal{D}(\mathbf{r}_s)} + f_{\mathcal{D}(\mathbf{r}')},z\right)\right| \leq$$
$$\leq \frac{B}{T}\left|f_{\mathcal{D}(\mathbf{r}')}\right| \leq \frac{B}{T}M$$

Thus, the constant ρ in Theorem 15 is bounded as

$$\rho = \sup_{\mathbf{r},\mathbf{r}'_t}\left|\ell(F_{\mathcal{D},\mathbf{r}},z)-\ell(F_{\mathcal{D},\mathbf{r}^t},z)\right| \leq \frac{B}{T}M.$$

The result then follows by using this theorem and Proposition 4.4. ■

We comment on some characteristics of the above bounds for subbagging:

- In Theorem 16 if, as $m \to \infty$, $\frac{p\gamma_p}{m} \to 0$ then the empirical or leave-one-out error converge to the expected error. In particular, if $p = O(1)$ as $m \to \infty$ the empirical or leave-one-out error converge to the expected one as $O(1/\sqrt{m})$. This convergence is in probability as opposed to the convergence provided by Theorem 17 which is almost surely.

- Although we can derive bounds for bagging using our theory in section 3 that were not possible to derive with the existing theory summarized in Section 2, our results for bagging do not show that bagging actually improves performance. Indeed, for example comparing Theorems 17 and 6, it is not clear which bound is tighter as that depends on the constants (e.g. M, B, and other constants) and the behavior of γ_p as p increases. Developing tighter bounds or lower bounds within our analysis for bagging is needed for this purpose. This is an open problem.

- Theorem 17 indicates that the effects of the number of subsamples T is of the form $\frac{1}{\sqrt{T}}$, so there is no need for a large T, as also observed in practice (Breiman, 1996a). For example, it is sufficient that T scales as $\sqrt{m}$. This result improves upon the analysis of (Evgeniou et al., 2004) where in order to have convergence of the empirical or leave-one-our error to the expected error it was required that T is infinite.

- The bounds provided by Theorem 17 imply that the empirical or leave-one-out error converge to the expected error provided, as $m \to \infty$, that $\frac{p\gamma_p}{\sqrt{m}} \to 0$ *and* $T \to \infty$. The latter condition is not a problem in practice, for example one could choose $T = O(\sqrt{m})$ to get convergence, but it indicates a weak point of the uniform stability analysis as opposed to the hypothesis stability analysis above. As we discussed above, it may be possible to show that parameter ρ appearing in Theorem 15 depends on m for the case of bagging, or to show that this is not possible in which case it will be a limitation of our approach. This is an open problem.

5. Conclusions

We presented a theory of random stability for randomized learning methods that we also applied to study the effects of bagging on the stability of a learning method. This is an extension of the existing theory about the stability and generalization performance of deterministic (symmetric) learning methods (Bousquet and Elisseeff 2002). We note that the setup that we developed for this analysis, such as the issues and assumptions that we considered in Section 3, may be also used for other studies of randomized learning algorithms – such as extensions of other theories about stability from deterministic to randomized learning methods. The bounds we proved show formally the relation of the generalization error to the stability of the (random) algorithm. There is currently no lower bound hence we cannot practically use the bounds when the number of data m is small (e.g., several hundreds or thousands, which is the case in many current applications). This issue concerns both the deterministic (Bousquet and Elisseeff, 2002) as well as the random case. Developing tighter bounds as well as lower bounds in order to be able to use the theory developed here in practice is an open question.

Acknowledgments

We wish to thank Sayan Mukherjee, Tomaso Poggio, Ryan Rifkin, and the anonymous reviewers for their useful comments.

Appendix A. Proofs of Lemmas 3.1 and 3.2

The proofs of Lemmas 3.1 and 3.2 follow directly the proof that has been given in (Bousquet and Elisseeff, 2002). We reproduce the proof here with the changes that are required to handle non symmetric algorithms. Before entering the core of the calculations, let us introduce some convenient notation. We will denote by

$$\ell_{ij}(z,z',z'') = \ell(f_{\mathcal{D}_{ij}(z,z')},z'') \tag{27}$$

the loss of an algorithm A trained on

$$\mathcal{D}_{i,j}(z,z') = \left(z_1,\ldots,z_{i-1},z,z_{i+1},\ldots,z_{j-1},z',z_{j+1},\ldots,z_m\right)$$

which represents the training set $\mathcal{D}$ where z_i and z_j have been replaced by z and z'. When $i=j$, it is required that $z=z'$. Note that the position of z_i and z_j matters here since the algorithm is not symmetric. Since we have $\mathcal{D}_{i,j}(z_i,z_j) = \mathcal{D}_{k,l}(z_k,z_l)$ for any i,j and k,l in $\{1,\ldots,m\}$, we use the notation $\ell(z)$ to denote $\ell_{ij}(z_i,z_j,z)$ for all i and j in $\{1,\ldots,m\}$. According to these notations we have

$$\ell_{ij}(\emptyset,z_j,z_i) = \ell(f_{\mathcal{D}^{\setminus i}},z_i),$$

that is, we replace z_i by the empty set when it is removed from the training set. Since $\ell_{ij}(\emptyset,z_j,z_i)$ does not depend on j, we will denote it by ℓ_i.

Different tricks such as decomposing sums, renaming and permuting variables will be used in the following calculations. Since the proofs are very technical and mostly formal, we explain here more precisely what these steps are. Decomposing sums is the main step of the calculations. The idea is to transform a difference $a-b$ into a sum $a-b=\sum_{i=1}^{k} a_i - a_{i+1}$ ($a_1 = a$ and $a_{k+1} = b$) so that the quantities $a_i - a_{i+1}$ in the sum can be bounded by terms of the form $\mathbf{E}_{\mathcal{D},z}\left[\left|\ell_{ij}(z,z_j,z_i)-\ell(z_i)\right|\right]$, the latter being directly related to the notion of stability we defined. Renaming variables corresponds to simply changing the name of one variable into another one. Most of time, this change will be done between z, z_i and z_j using the fact that z and the z_i's are independently and identically distributed so that averaging w.r.t. z is the same as w.r.t. z_i. The last technique we use is symmetrization. The following simple lemma will allow us to perform some symmetrization without changing significantly the outcome of a (stable) learning algorithm.

Lemma 18 *Let A be a (non-symmetric) algorithm and let ℓ be as defined in Equation (27), we have* $\forall (i,j) \in \{1,\ldots,m\}^2$

$$\mathbf{E}_{\mathcal{D},z}\left[\left|\ell(z)-\ell_{ij}(z_j,z_i,z)\right|\right] \le \frac{3}{2}\left(\mathbf{E}_{\mathcal{D},z,z'}\left[\left|\ell_{ij}(z',z_j,z)-\ell(z)\right|\right] + \mathbf{E}_{\mathcal{D},z,z'}\left[\left|\ell_{ij}(z_i,z',z)-\ell(z)\right|\right]\right). \tag{28}$$

Proof We have

$$\begin{aligned}\mathbf{E}_{\mathcal{D},z}\left[\left|\ell(z)-\ell_{ij}(z_j,z_i,z)\right|\right] &\le \mathbf{E}_{\mathcal{D},z,z'}\left[\left|\ell(z)-\ell_{ij}(z',z_j,z)\right|\right] \\ &+ \mathbf{E}_{\mathcal{D},z,z'}\left[\left|\ell_{ij}(z',z_j,z)-\ell_{ij}(z',z_i,z)\right|\right] + \mathbf{E}_{\mathcal{D},z,z'}\left[\left|\ell_{ij}(z',z_i,z)-\ell_{ij}(z_j,z_i,z)\right|\right]\end{aligned} \tag{29}$$

Since the distribution over $\mathcal{D}$ is i.i.d., integrating with respect to z_i is the same as integrating w.r.t. z_j or z', and we can swap the role of z' and z_i in the second term of the r.h.s. , and of z_i and z_j in the last term.

$$\begin{aligned}\mathbf{E}_{\mathcal{D},z,z'}\left[\left|\ell_{ij}(z',z_j,z)-\ell_{ij}(z',z_i,z)\right|\right] &= \mathbf{E}_{\mathcal{D},z,z'}\left[\left|\ell(z)-\ell_{ij}(z_i,z',z)\right|\right] \\ \mathbf{E}_{\mathcal{D},z,z'}\left[\left|\ell_{ij}(z',z_i,z)-\ell_{ij}(z_j,z_i,z)\right|\right] &= \mathbf{E}_{\mathcal{D},z,z'}\left[\left|\ell_{ij}(z',z_j,z)-\ell(z)\right|\right],\end{aligned}$$

which gives the following result:

$$\mathbf{E}_{\mathcal{D},z}\left[\left|\ell(z)-\ell_{ij}(z_j,z_i,z)\right|\right] \quad\leq\quad 2\mathbf{E}_{\mathcal{D},z}\left[\left|\ell_{ij}(z',z_j,z)-\ell(z)\right|\right] \;+\; \mathbf{E}_{\mathcal{D},z}\left[\left|\ell_{ij}(z_i,z',z)-\ell(z)\right|\right] \quad (30)$$

If instead of (29) we used the following decomposition,

$$\begin{aligned}\mathbf{E}_{\mathcal{D},z}\left[\left|\ell(z)-\ell_{ij}(z_j,z_i,z)\right|\right] \leq \mathbf{E}_{\mathcal{D},z,z'}\left[\left|\ell(z)-\ell_{ij}(z_i,z',z)\right|\right] \\ +\mathbf{E}_{\mathcal{D},z,z'}\left[\left|\ell_{ij}(z_i,z',z)-\ell(z_j,z',z)\right|\right]+\mathbf{E}_{\mathcal{D},z,z'}\left[\left|\ell(z_j,z',z)-\ell_{ij}(z_j,z_i,z)\right|\right],\end{aligned}$$

it would have led to

$$\mathbf{E}_{\mathcal{D},z}\left[\left|\ell(z)-\ell_{ij}(z_j,z_i,z)\right|\right] \leq \mathbf{E}_{\mathcal{D},z}\left[\left|\ell_{ij}(z',z_j,z)-\ell(z)\right|\right]+2\mathbf{E}_{\mathcal{D},z}\left[\left|\ell_{ij}(z_i,z',z)-\ell(z)\right|\right].$$

Averaging this inequality with (30), we get the final result. ■

Note that the quantity appearing in the r.h.s. of Equation (28) can be bounded by different quantities related to pointwise hypothesis stability or to hypothesis stability. We have indeed

$$\mathbf{E}_{\mathcal{D},z}\left[\left|\ell(z)-\ell_{ij}(z_j,z_i,z)\right|\right] \leq 3\left(\mathbf{E}_{\mathcal{D},z}\left[\left|\ell_{ij}(z,z_j,z_i)-\ell(z_i)\right|\right] +\mathbf{E}_{\mathcal{D},z}\left[\left|\ell_{ij}(z_i,z,z_j)-\ell(z_j)\right|\right]\right), \quad (31)$$

which is related to the definition of pointwise hypothesis stability and will be used when the focus is on empirical error. We have also

$$\mathbf{E}_{\mathcal{D},z}\left[\left|\ell(z)-\ell_{ij}(z_j,z_i,z)\right|\right] \leq 3\left(\mathbf{E}_{\mathcal{D},z}\left[\left|\ell_{ij}(\emptyset,z_j,z)-\ell(z)\right|\right] +\mathbf{E}_{\mathcal{D},z}\left[\left|\ell_{ij}(z_i,\emptyset,z)-\ell(z)\right|\right]\right),$$

which is related to bounds on the leave-one-out error. Both bounds have the same structure and it will turn out that the following calculations are almost identical for leave-one-out error and empirical error. We can now start the main part of the proofs. The notations are difficult to digest but the ideas are simple and use only the few formal steps we have described before. We first state the following lemma as in (Bousquet and Elisseeff, 2002):

Lemma 19 *For any (non-symmetric) learning algorithm A, we have*

$$\begin{aligned}\mathbf{E}_{\mathcal{D}}\left[(R_{gen}-R_{emp})^2\right] \leq \frac{1}{m^2}\sum_{i\neq j}\mathbf{E}_{\mathcal{D},z,z'}\left[\ell(z)\ell(z')\right]-\frac{2}{m^2}\sum_{i\neq j}^{m}\mathbf{E}_{\mathcal{D},z}\left[\ell(z)\ell(z_i)\right] \\ +\frac{1}{m^2}\sum_{i\neq j}\mathbf{E}_{\mathcal{D}}\left[\ell(z_i)\ell(z_j)\right]+\frac{1}{m^2}\sum_{i=1}^{m}\left(\mathbf{E}_{\mathcal{D},z,z'}\left[\ell(z)\ell(z')\right]-2\mathbf{E}_{\mathcal{D},z}\left[\ell(z)\ell(z_i)\right]+\mathbf{E}_{\mathcal{D}}\left[\ell(z_i)^2\right]\right)\end{aligned}$$

and

$$\begin{aligned}\mathbf{E}_{\mathcal{D}}\left[(R_{gen}-R_{\ell oo})^2\right] \leq \frac{1}{m^2}\sum_{i\neq j}\mathbf{E}_{\mathcal{D},z,z'}\left[\ell(z)\ell(z')\right]-\frac{2}{m^2}\sum_{i\neq j}\mathbf{E}_{\mathcal{D},z}\left[\ell(z)\ell_i\right] \\ +\frac{1}{m^2}\sum_{i\neq j}\mathbf{E}_{\mathcal{D}}\left[\ell_i\ell_{ij}(z_i,\emptyset,z_j)\right] \\ +\frac{1}{m^2}\sum_{i=1}^{m}\left(\mathbf{E}_{\mathcal{D},z,z'}\left[\ell(z)\ell(z')\right]-2\mathbf{E}_{\mathcal{D},z}\left[\ell(z)\ell_i\right]+\mathbf{E}_{\mathcal{D}}\left[\ell_i^2\right]\right).\end{aligned}$$

Proof We have

$$\begin{aligned}
\mathbf{E}_{\mathcal{D}}\left[R_{gen}^2\right] &= \mathbf{E}_{\mathcal{D}}\left[\mathbf{E}_z \ell(z)^2\right] \\
&= \mathbf{E}_{\mathcal{D},z,z'}\left[\ell(z)\ell(z')\right] \\
&= \frac{1}{m^2}\sum_{i\neq j}\mathbf{E}_{\mathcal{D},z,z'}\left[\ell(z)\ell(z')\right] + \frac{1}{m^2}\sum_{i=1}^m \mathbf{E}_{\mathcal{D},z,z'}\left[\ell(z)\ell(z')\right],
\end{aligned}$$

and also

$$\begin{aligned}
\mathbf{E}_{\mathcal{D}}\left[R_{gen}R_{emp}\right] &= \mathbf{E}_{\mathcal{D}}\left[R_{gen}\frac{1}{m}\sum_{i=1}^m \ell(z_i)\right] \\
&= \frac{1}{m}\sum_{i=1}^m \mathbf{E}_{\mathcal{D}}\left[R_{gen}\ell(z_i)\right] \\
&= \frac{1}{m}\sum_{i=1}^m \mathbf{E}_{\mathcal{D},z}\left[\ell(z)\ell(z_i)\right] \\
&= \frac{1}{m^2}\sum_{i\neq j}\mathbf{E}_{\mathcal{D},z}\left[\ell(z)\ell(z_i)\right] + \frac{1}{m^2}\sum_{i=1}^m \mathbf{E}_{\mathcal{D},z}\left[\ell(z)\ell(z_i)\right],
\end{aligned}$$

and also

$$\begin{aligned}
\mathbf{E}_{\mathcal{D}}\left[R_{gen}R_{\ell oo}\right] &= \mathbf{E}_{\mathcal{D}}\left[R_{gen}\frac{1}{m}\sum_{i=1}^m \ell_i\right] \\
&= \frac{1}{m}\sum_{i=1}^m \mathbf{E}_{\mathcal{D}}\left[R_{gen}\ell_i\right] \\
&= \frac{1}{m}\sum_{i=1}^m \mathbf{E}_{\mathcal{D},z}\left[\ell(z)\ell_i\right] \\
&= \frac{1}{m^2}\sum_{i\neq j}^m \mathbf{E}_{\mathcal{D},z}\left[\ell(z)\ell_i\right] + \frac{1}{m}\sum_{i=1}^m \mathbf{E}_{\mathcal{D},z}\left[\ell(z)\ell_i\right].
\end{aligned}$$

Also we have

$$\mathbf{E}_{\mathcal{D}}\left[R_{emp}^2\right] = \frac{1}{m^2}\sum_{i=1}^m \mathbf{E}_{\mathcal{D}}\left[\ell(z_i)^2\right] + \frac{1}{m^2}\sum_{i\neq j}\mathbf{E}_{\mathcal{D}}\left[\ell(z_i)\ell(z_j)\right]$$

and

$$\mathbf{E}_{\mathcal{D}}\left[R_{\ell oo}^2\right] = \frac{1}{m^2}\sum_{i=1}^m \mathbf{E}_{\mathcal{D}}\left[\ell_i^2\right] + \frac{1}{m^2}\sum_{i\neq j}\mathbf{E}_{\mathcal{D}}\left[\ell_i \ell_{ij}(z_i,\emptyset,z_j)\right],$$

which concludes the proof. ■

Continuing the proof of Lemma 3.2, we now formulate the first inequality of Lemma 19 as

$$\mathbf{E}_{\mathcal{D}}\left[(R_{gen}-R_{emp})^2\right] \leq \frac{1}{m^2}\sum_{i\neq j}\underbrace{\mathbf{E}_{\mathcal{D},z,z'}\left[\ell(z)\ell(z')\right]-\mathbf{E}_{\mathcal{D},z}\left[\ell(z)\ell(z_i)\right]}_{I}$$
$$+\frac{1}{m^2}\sum_{i\neq j}^{m}\underbrace{\mathbf{E}_{\mathcal{D}}\left[\ell(z_i)\ell(z_j)\right]-\mathbf{E}_{\mathcal{D},z}\left[\ell(z)\ell(z_i)\right]}_{J}$$
$$+\frac{1}{m^2}\sum_{i=1}^{m}\underbrace{\mathbf{E}_{\mathcal{D},z,z'}\left[\ell(z)\ell(z')\right]-2\mathbf{E}_{\mathcal{D},z}\left[\ell(z)\ell(z_i)\right]+\mathbf{E}_{\mathcal{D}}\left[\ell(z_i)^2\right]}_{K}.$$

Using the fact that the loss is bounded by M, we have

$$\begin{aligned}K &= \mathbf{E}_{\mathcal{D},z,z'}\left[\ell(z)\left(\ell(z')-\ell(z_i)\right)\right]+\mathbf{E}_{\mathcal{D},z}\left[\ell(z_i)\left(\ell(z_i)-\ell(z)\right)\right]\\ &\leq 2M^2.\end{aligned}$$

Now we rewrite I as

$$\mathbf{E}_{\mathcal{D},z,z'}\left[\ell(z)\ell(z')\right]-\mathbf{E}_{\mathcal{D},z}\left[\ell(z)\ell(z_i)\right]=$$
$$=\mathbf{E}_{\mathcal{D},z,z'}\left[\ell(z)\ell(z')-\ell_{ij}(z',z_j,z)\ell_{ij}(z',z_j,z')\right],$$

where we renamed z_i as z' in the second term. We have then

$$I=\mathbf{E}_{\mathcal{D},z,z'}\left[(\ell(z)-\ell_{ij}(z,z_j,z))\ell(z')\right]$$
$$+\mathbf{E}_{\mathcal{D},z,z'}\left[(\ell_{ij}(z,z_j,z)-\ell_{ij}(z',z_j,z))\ell(z')\right]$$
$$+\mathbf{E}_{\mathcal{D},z,z'}\left[(\ell(z')-\ell_{ij}(z',z_j,z'))\ell_{ij}(z',z_j,z)\right].$$

Thus,

$$|I|\leq 3M\mathbf{E}_{\mathcal{D},z,z'}\left[\left|\ell_{ij}(z,z_j,z)-\ell(z)\right|\right]. \qquad (32)$$

Next we rewrite J as

$$\mathbf{E}_{\mathcal{D}}\left[\ell(z_i)\ell(z_j)\right]-\mathbf{E}_{\mathcal{D},z}\left[\ell(z)\ell(z_i)\right]=\mathbf{E}_{\mathcal{D},z,z'}\left[\ell_{ij}(z,z',z)\ell_{ij}(z,z',z')-\ell(z)\ell(z_i)\right]$$

where we renamed z_j as z' and z_i as z in the first term. We have also

$$J=\mathbf{E}_{\mathcal{D},z,z'}\left[\ell_{ij}(z,z',z)\ell_{ij}(z,z',z')-\ell_{ij}(z',z_i,z)\ell_{ij}(z',z_i,z')\right]$$

where we renamed z_i as z' and z_j as z_i in the second term. Using Equation 31, we have

$$J\leq\underbrace{\mathbf{E}_{\mathcal{D},z,z'}\left[\ell_{ij}(z,z',z)\ell_{ij}(z,z',z')-\ell_{ij}(z_i,z',z)\ell_{ij}(z',z_i,z')\right]}_{J_1}$$
$$+3M\left(\mathbf{E}_{\mathcal{D},z}\left[\left|\ell_{ij}(z,z_j,z_i)-\ell(z_i)\right|\right]+\mathbf{E}_{\mathcal{D},z}\left[\left|\ell_{ij}(z_i,z,z_j)-\ell(z_j)\right|\right]\right). \qquad (33)$$

Let us focus on J_1, we have

$$J_1=\mathbf{E}_{\mathcal{D},z,z'}\left[(\ell_{ij}(z,z',z')-\ell_{ij}(z,z_i,z')\ell_{ij}(z,z',z)\right]$$
$$+\mathbf{E}_{\mathcal{D},z,z'}\left[(\ell_{ij}(z,z',z)-\ell_{ij}(z_i,z',z))\ell_{ij}(z,z_i,z')\right]$$
$$+\mathbf{E}_{\mathcal{D},z,z'}\left[(\ell_{ij}(z,z_i,z')-\ell_{ij}(z',z_i,z'))\ell_{ij}(z_i,z',z)\right]$$

and

$$\begin{aligned}J_1 = \mathbf{E}_{\mathcal{D},z,z'}\left[(\ell_{ij}(z_i,z_j,z_j) - \ell_{ij}(z_i,z,z_j))\ell_{ij}(z_i,z_j,z_i)\right] \\ + \mathbf{E}_{\mathcal{D},z,z'}\left[(\ell_{ij}(z_i,z_j,z_i) - \ell_{ij}(z,z_j,z_i))\ell_{ij}(z_i,z,z_j)\right] \\ + \mathbf{E}_{\mathcal{D},z,z'}\left[(\ell_{ij}(z,z_j,z_i) - \ell_{ij}(z_i,z_j,z_i))\ell_{ij}(z_j,z_i,z)\right]\end{aligned}$$

where we replaced z by z_i, z_i by z and z' by z_j in the first term, and z by z_i and z' by z_j and z_i by z in the second term and, in the last term, we renamed z' by z_i and z_i by z_j. Thus,

$$|J_1| \leq 2M\mathbf{E}_{\mathcal{D},z}\left[\left|\ell_{ij}(z,z_j,z_i) - \ell(z_i)\right|\right] + M\mathbf{E}_{\mathcal{D},z,z'}\left[\left|\ell_{ij}(z_i,z,z_j) - \ell(z_j)\right|\right]. \tag{34}$$

Summing Equation (32) with the inequality on J derived from Equations (34) and (33), we obtain

$$I + J \quad \leq \quad 8M\mathbf{E}_{\mathcal{D},z}\left[\left|\ell_{ij}(z,z_j,z_i) - \ell(z_i)\right|\right] + 4M\mathbf{E}_{\mathcal{D},z}\left[\left|\ell_{ij}(z_i,z,z_j) - \ell(z_j)\right|\right].$$

To bound $I+J$, we can swap the role of i and j (note that i and j are under a sum and that we can permute the role of i and j in this sum without changing anything). In that case, we obtain

$$I + J \leq 4M\mathbf{E}_{\mathcal{D},z}\left[\left|\ell_{ij}(z,z_j,z_i) - \ell(z_i)\right|\right] + 8M\mathbf{E}_{\mathcal{D},z}\left[\left|\ell_{ij}(z_i,z,z_j) - \ell(z_j)\right|\right].$$

Averaging over this bound and the previous one, we finally obtain

$$I + J \leq 6M\left(\mathbf{E}_{\mathcal{D},z}\left[\left|\ell_{ij}(z,z_j,z_i) - \ell(z_i)\right|\right] + \mathbf{E}_{\mathcal{D},z}\left[\left|\ell_{ij}(z_i,z,z_j) - \ell(z_j)\right|\right]\right).$$

The above concludes the proof of the bound for the empirical error (Lemma 3.2).

The bound for the leave-one-out error (Lemma 3.1) can be obtained in a similar way. Indeed, we notice that if we rewrite the derivation for the empirical error, we simply have to remove from the training set the point at which the loss is computed. That is, we simply have to replace all the quantities of the form $\ell_{ij}(z,z',z)$ by $\ell_{ij}(\emptyset,z',z)$. It is easy to see that the above results are modified in a way that gives the correct bound for the leave-one-out error.

Appendix B. Proof of Theorem 3.4

Proof We first prove Equation (17) and then show how to derive Equation (19). Both proofs are very similar except for some calculations.

Let $K(\mathcal{D},\mathbf{r}) = R_{gen}(f_{\mathcal{D},\mathbf{r}}) - R_{emp}(f_{\mathcal{D},\mathbf{r}})$ the random variable which we would like to bound. For this purpose, we first show that K is close to its expectation w.r.t. $\mathbf{r}$ and then show how this average algorithm is controlled by its stability.

For every $\mathbf{r},\mathbf{s} \in \mathcal{R}^T$, and $T \in \mathbb{N}$, we have

$$|K(\mathcal{D},\mathbf{r}) - K(\mathcal{D},\mathbf{s})| =$$

$$= \left|\mathbf{E}_z\left[\ell(f_{\mathcal{D},\mathbf{r}},z) - \ell(f_{\mathcal{D},\mathbf{s}},z)\right] - \frac{1}{m}\sum_{i=1}^{m}\left(\ell(f_{\mathcal{D},\mathbf{r}},z_i) - \ell(f_{\mathcal{D},\mathbf{s}},z_i)\right)\right|$$

$$\leq \mathbf{E}_z\left[\left|\ell(f_{\mathcal{D},\mathbf{r}},z) - \ell(f_{\mathcal{D},\mathbf{s}},z)\right|\right] + \frac{1}{m}\sum_{i=1}^{m}\left|\ell(f_{\mathcal{D},\mathbf{r}},z_i) - \ell(f_{\mathcal{D},\mathbf{s}},z_i)\right|.$$

Thus, using the definition of ρ, this equation implies (when $\mathbf{r}$ and $\mathbf{s}$ differ only in one of the T coordinates) that

$$\sup_{\mathbf{r}_1,\ldots,\mathbf{r}_T,\mathbf{r}'_t} \left| K(\mathcal{D},\mathbf{r}_1,\ldots,\mathbf{r}_T) - K(\mathcal{D},\mathbf{r}_1,\ldots,\mathbf{r}_{t-1},\mathbf{r}'_t,\mathbf{r}_{t+1},\ldots,\mathbf{r}_T) \right| \leq 2\rho$$

and applying Theorem 14 we obtain (note that $\mathcal{D}$ is independent of $\mathbf{r}$)

$$\mathbb{P}_{\mathbf{r}}\left[K(\mathcal{D},\mathbf{r}) - \mathbf{E}_{\mathbf{r}}\left[K(\mathcal{D},\mathbf{r})\right] \geq \varepsilon \mid \mathcal{D}\right] \leq \exp\left\{-\varepsilon^2/2T\rho^2\right\}.$$

We also have

$$\begin{aligned} &\mathbf{E}_{\mathcal{D}}\left[\mathbb{P}_{\mathbf{r}}\left[K(\mathcal{D},\mathbf{r}) - \mathbf{E}_{\mathbf{r}}K(\mathcal{D},\mathbf{r}) \geq \varepsilon\right]\right] = \\ &= \mathbf{E}_{\mathcal{D}}\left[\mathbb{P}_{\mathbf{r}}\left[K(\mathcal{D},\mathbf{r}) - \mathbf{E}_{\mathbf{r}}K(\mathcal{D},\mathbf{r}) \geq \varepsilon \mid \mathcal{D}\right]\right] \leq \exp\left\{-\varepsilon^2/2T\rho^2\right\}. \end{aligned}$$

Setting the r.h.s. equal to δ and writing ε as a function of δ we have that with probability at least $1-\delta$ w.r.t. the random sampling of $\mathcal{D}$ and $\mathbf{r}$:

$$K(\mathcal{D},\mathbf{r}) - \mathbf{E}_{\mathbf{r}}K(\mathcal{D},\mathbf{r}) \leq \sqrt{2T}\rho\sqrt{\log(1/\delta)}. \tag{35}$$

We first bound the expectation of $K(\mathcal{D},\mathbf{r})$. We define $G(\mathcal{D},z) := \mathbf{E}_{\mathbf{r}}[\ell(f_{\mathcal{D},\mathbf{r}},z)]$. We have

$$\begin{aligned} \mathbf{E}_{\mathcal{D},\mathbf{r}}\left[K(\mathcal{D},\mathbf{r})\right] &= \mathbf{E}_{\mathcal{D}}\left[\mathbf{E}_z\left[G(\mathcal{D},z) - \frac{1}{m}\sum_{i=1}^m G(\mathcal{D},z_i)\right]\right] \\ &= \mathbf{E}_{\mathcal{D},z}\left[G(\mathcal{D},z)\right] - \frac{1}{m}\sum_{i=1}^m \mathbf{E}_{\mathcal{D}}\left[G(\mathcal{D},z_i)\right] \\ &\overset{(a)}{\leq} 2\beta_m + \mathbf{E}_{\mathcal{D}^{\setminus i},z}\left[G(\mathcal{D}^{\setminus i},z)\right] - \frac{1}{m}\sum_{i=1}^m \mathbf{E}_{\mathcal{D}}\left[G(\mathcal{D}^{\setminus i},z_i)\right] \\ &\overset{(b)}{=} 2\beta_m \end{aligned} \tag{36}$$

where (a) is derived from the fact that the algorithm has random uniform stability β_m, that is,

$$\sup_{\mathcal{D},z}\left|G(\mathcal{D},z) - G(\mathcal{D}^{\setminus i},z)\right| \leq \beta_m,$$

and (b) comes from $\mathbf{E}_{\mathcal{D}}\left[G(\mathcal{D}^{\setminus i},z_i)\right] = \mathbf{E}_{\mathcal{D}^{\setminus i},z}\left[G(\mathcal{D}^{\setminus i},z)\right]$ (it amounts to changing z_i into z). We would like now to apply Theorem 14 to $\mathbf{E}_{\mathbf{r}}\left[K(\mathcal{D},\mathbf{r})\right]$. To this aim, we bound (recall that $\mathcal{D}^i = \mathcal{D}^{\setminus i} \cup z'$):

$$\begin{aligned} &\left|\mathbf{E}_{\mathbf{r}}\left[\mathbf{E}_{\mathbf{r}}\left[K(\mathcal{D},\mathbf{r}) - K(\mathcal{D}^i,\mathbf{r})\right]\right]\right| = \\ &\Bigg| \underbrace{\frac{1}{m}\left(\mathbf{E}_{\mathbf{r}}\left[\ell(f_{\mathcal{D}^i,\mathbf{r}},z')\right] - \mathbf{E}_{\mathbf{r}}\left[\ell(f_{\mathcal{D},\mathbf{r}},z_i)\right]\right)}_{(a)} + \frac{1}{m}\sum_{i\neq j}\underbrace{\mathbf{E}_{\mathbf{r}}[\ell(f_{\mathcal{D}^{\setminus i},\mathbf{r}},z_j)] - \mathbf{E}_{\mathbf{r}}\left[\ell(f_{\mathcal{D},\mathbf{r}},z_j)\right]}_{(b)} \\ &+ \frac{1}{m}\sum_{i\neq j}\underbrace{\mathbf{E}_{\mathbf{r}}\left[\ell(f_{\mathcal{D}^i,\mathbf{r}},z_j)\right] - \mathbf{E}_{\mathbf{r}}[\ell(f_{\mathcal{D}^{\setminus i},\mathbf{r}},z_j)]}_{(c)} + \underbrace{\mathbf{E}_{\mathbf{r}}\left[\mathbf{E}_z\left[\ell(f_{\mathcal{D},\mathbf{r}},z) - \ell(f_{\mathcal{D}^i,\mathbf{r}},z)\right]\right]}_{(d)} \Bigg| \end{aligned} \tag{37}$$

where (a) is bounded by $\frac{M}{m}$, (b), (c) are bounded by β_m and (d) is similarly bounded by $2\beta_m$. So that $\sup_{\mathcal{D},z',z} \left|\mathbf{E_r}\left[K(\mathcal{D},\mathbf{r})\right] - \mathbf{E_r}\left[K(\mathcal{D}^i,\mathbf{r})\right]\right| \le \frac{M}{m} + 4\beta_m$ and we derive that

$$\mathbb{P}_{\mathcal{D}}\left[\mathbf{E_r}\left[K(\mathcal{D},\mathbf{r})\right] \ge \varepsilon + 2\beta_m\right] \le \exp\left\{-\frac{2m\varepsilon^2}{(M+4m\beta_m)^2}\right\},$$

which implies that with probability at least $1-\delta$ w.r.t. the random sampling of $\mathcal{D}$ and r

$$\mathbf{E_r}\left[K(\mathcal{D},\mathbf{r})\right] \le 2\beta_m + \frac{M+4m\beta_m}{\sqrt{2m}}\sqrt{\log(1/\delta)}. \tag{38}$$

Observe that the inequalities in Equations (35) and (38) hold simultaneously with probability at least $1-2\delta$. The result follows by combining those inequalities and setting $\delta = \delta/2$.

The proof of Equation (19) follows the same reasoning except that the chain of Equations (36) and (37) are different. We have

$$\begin{aligned}
\mathbf{E}_{\mathcal{D},\mathbf{r}}\left[K(\mathcal{D},\mathbf{r})\right] &= \mathbf{E}_{\mathcal{D}}\left[\mathbf{E}_z\left[G(\mathcal{D},z)\right] - \frac{1}{m}\sum_{i=1}^m G(\mathcal{D}^{\setminus i},z_i)\right] \\
&= \mathbf{E}_{\mathcal{D},z}\left[G(\mathcal{D},z)\right] - \frac{1}{m}\sum_{i=1}^m \mathbf{E}_{\mathcal{D},z}\left[G(\mathcal{D}^{\setminus i},z)\right] \\
&\le \beta_m,
\end{aligned}$$

and denoting $\mathcal{D}^{\setminus i,j}$ the set $\mathcal{D}$ where z_i and z_j have been removed, and $\mathcal{D}^{i\setminus j}$ the set $\mathcal{D}^i$ where z_j has been removed (for $j \ne i$),

$$\begin{aligned}
&\left|\mathbf{E_r}\left[K(\mathcal{D},\mathbf{r})\right] - \mathbf{E_r}\left[K(\mathcal{D}^i,\mathbf{r})\right]\right| = \\
&\Bigg|\underbrace{\frac{1}{m}\left(\mathbf{E_r}\left[\ell(f_{\mathcal{D}^{\setminus i},\mathbf{r}},z_i)\right] - \mathbf{E_r}\left[\ell(f_{\mathcal{D}^{\setminus i},\mathbf{r}},z')\right]\right)}_{(a)} + \frac{1}{m}\sum_{i\ne j}\underbrace{\mathbf{E_r}\left[\ell(f_{\mathcal{D}^{\setminus j},\mathbf{r}},z_j)\right] - \mathbf{E_r}\left[\ell(f_{\mathcal{D}^{\setminus i,j},\mathbf{r}},z_j)\right]}_{(b)} \\
&\quad + \frac{1}{m}\sum_{j\ne j}^{m}\underbrace{\mathbf{E_r}\left[\ell(f_{\mathcal{D}^{\setminus i,j},\mathbf{r}},z_j)\right] - \mathbf{E_r}\left[\ell(f_{\mathcal{D}^{i\setminus j},\mathbf{r}},z_j)\right]}_{(c)} + \underbrace{\mathbf{E_r}\left[\mathbf{E}_z\left[\ell(f_{\mathcal{D},\mathbf{r}},z) - \ell(f_{\mathcal{D}^i,\mathbf{r}},z)\right]\right]}_{(d)}\Bigg|.
\end{aligned}$$

Finally, note that (a) is bounded by $\frac{M}{m}$, (b) and (c) are bounded by β_{m-1} and (d) by $2\beta_m$. ■

References

[1] Andonova, S., Elisseeff, A., Evgeniou, T., and Pontil, M. (2002), "A simple algorithm to learn stable machines", Proceedings of the 15th European Conference on Artificial Intelligence (ECAI) 2002.

[2] Bousquet, O., and Elisseeff, A. (2002), "Stability and generalization", *Journal of Machine Learning Research*, 2:499–526.

[3] Breiman, L. (1996a), “Bagging predictors”, *Machine Learning*, 26(2):123–140.

[4] Breiman, L. (1996b), “Heuristics of instability and stabilization in model selection”, *Annals of Statistics*, 24(6):2350–2383.

[5] Devroye, L., Györfi, L., and Lugosi, G. (1996), *A Probabilistic Theory of Pattern Recognition*, Number 31 in Applications of Mathematics, Springer, New York.

[6] Devroye, L., and Wagner, T. (1979), “Distribution-free performance bounds for potential function rules”, *IEEE Transactions on Information Theory*, 25(5):601–604.

[7] Evgeniou, T., Pontil, M., and Elisseeff, A. (2004), “Leave-one-out error, stability, and generalization of voting combinations of classifiers”, *Machine Learning*, 55:1, 2004 .

[8] Kearns, M., and Ron, D. (1999), “Algorithmic stability and sanity check bounds for leave-one-out cross validation bounds”, *Neural Computation*, 11(6):1427–1453.

[9] Kutin, S., and Niyogi, P. (2002), “Almost-everywhere algorithmic stability and generalization error”, *Uncertainty in Artificial Intelligence (UAI)*, August, 2002, Edmonton, Canada.

[10] McDiarmid, C. (1989), “On the method of bounded differences”, In *Survey in Combinatorics*, p. 148–188. Cambridge University Press, Cambridge.

[11] Poggio, T., and Girosi, F. (1990), “Networks for approximation and learning”, *Proceedings of the IEEE*, 78 (9).

[12] Poggio, T., Rifkin, R., Mukherjee, S. and Niyogi, P. (2004), “Learning Theory: general conditions for predictivity”, *Nature*, Vol. 428, 419-422.

[13] Vapnik, V. (1998), *Statistical Learning Theory*. Wiley, New York, 1998.

Journal of Machine Learning Research 6 (2005) 81–127 Submitted 9/04; Revised 12/04; Published 1/05

Learning Hidden Variable Networks: The Information Bottleneck Approach

Gal Elidan GALEL@CS.HUJI.AC.IL
Department of Engineering and Computer Science
The Hebrew University
Jerusalem, 91904, Israel

Nir Friedman NIR@CS.HUJI.AC.IL
Department of Engineering and Computer Science
The Hebrew University
Jerusalem, 91904, Israel

Editor: David Maxwell Chickering

Abstract

A central challenge in learning probabilistic graphical models is dealing with domains that involve hidden variables. The common approach for learning model parameters in such domains is the *expectation maximization* (EM) algorithm. This algorithm, however, can easily get trapped in sub-optimal local maxima. Learning the model *structure* is even more challenging. The *structural EM* algorithm can adapt the structure in the presence of hidden variables, but usually performs poorly without prior knowledge about the cardinality and location of the hidden variables. In this work, we present a general approach for learning Bayesian networks with hidden variables that overcomes these problems. The approach builds on the *information bottleneck* framework of Tishby et al. (1999). We start by proving formal correspondence between the information bottleneck objective and the standard parametric EM functional. We then use this correspondence to construct a learning algorithm that combines an information-theoretic smoothing term with a continuation procedure. Intuitively, the algorithm bypasses local maxima and achieves superior solutions by following a continuous path from a solution of, an easy and smooth, target function, to a solution of the desired likelihood function. As we show, our algorithmic framework allows learning of the parameters as well as the structure of a network. In addition, it also allows us to introduce new hidden variables during model selection and learn their cardinality. We demonstrate the performance of our procedure on several challenging real-life data sets.

Keywords: Bayesian networks, hidden variables, information bottleneck, continuation, variational methods

1. Introduction

Probabilistic graphical models have been widely used to model real world domains and are particularly appealing due to their natural interpretation. Despite extensive research in learning these models from data (Pearl, 1988; Heckerman, 1998), learning with *hidden* (or *latent*) variables has

A preliminary version of this paper appeared in the Proceedings of the Nineteenth Conference on Uncertainty in Artificial, 2003 (UAI '03).

remained a central challenge in learning graphical models in general, and Bayesian networks in particular. Hidden entities play a central role in many real-life problems: an unknown regulating mechanism can be the key to complex biological systems; correlating symptoms might hint at a hidden fundamental problem in a diagnostic system; an intentionally masked economic power might be the cause of related financial phenomena. Indeed, hidden variables typically serve as a summarizing mechanism that "captures" information from some of the observed variables and "passes" this information to some other part the network. As such, hidden variables can simplify the network structure and consequently lead to better generalization.

When learning the parameters of a Bayesian network with missing values or hidden variables, the most common approach is to use some variant of the *expectation maximization* (EM) algorithm (Dempster et al., 1977; Lauritzen, 1995). This algorithm performs a greedy search of the likelihood surface and converges to a local stationary point (usually a local maximum). Unfortunately, in challenging real-life learning problems, there are many local maxima that can trap EM in a poor solution. Attempts to address this problem use a variety of strategies (*e.g.*, Glover and Laguna (1993); Kirkpatrick et al. (1983); Rose (1998); Elidan et al. (2002)). When learning structure, the *structural EM* (SEM) algorithm (Friedman, 1997; Meila and Jordan, 1998; Thiesson et al., 1998) can adapt the network topology. In this approach, as in the classical parametric EM algorithm, we use the distribution induced by our current model, to probabilistically *complete* the data. Unlike parametric EM, we then use the completed data to evaluate different candidate structures. This allows us to perform structure improvement steps in the *M-Step* of a structural EM iteration. As in the case of EM, while convergence is guaranteed, the algorithm typically converges to a local maximum.

An even more challenging problem is that of *model selection* with hidden variables. This involves choosing the number of hidden variables, their cardinalities and the dependencies between them and the observed entities of the domain. These decisions are crucial to achieve good generalization. In particular, in hard real-life learning problems, structural EM will perform poorly unless some prior knowledge of the interaction between the hidden and observed variables exists or if the cardinality of the hidden variables is not (at least approximately) known. These challenging problems have received surprisingly little attention.

In this paper, we introduce a new approach to learning Bayesian networks with hidden variables. We pose the learning problem as an the optimization of a target function that includes a tradeoff between two information theoretic objectives. The first objective is to compress information about the training data. Intuitively, this is required when we want to generalize from the training data to new unseen instances. The second objective is to make the hidden variables informative about the observed attributes to ensure they preserve the *relevant* information. This objective is directly related to maximizing the likelihood of the training data. By exploring different relative weightings of these two objectives, we are able to bypass local maxima and learn better models.

Our approach builds on the *information bottleneck* framework of Tishby et al. (1999) and its multivariate extension (Friedman et al., 2001). This framework provides methods for constructing a set of new variables $\mathbf{T}$ that are stochastic functions of one set of variables $\mathbf{Y}$ and at the same time provide information on another set of variables $\mathbf{X}$. The intuition is that the new variables $\mathbf{T}$ capture the relevant aspects of $\mathbf{Y}$ that are informative about $\mathbf{X}$. We show how to pose the learning problem within the multivariate information bottleneck framework and derive a target Lagrangian for the hidden variables. We then show that this Lagrangian is an extension of the Lagrangian formulation of EM of Neal and Hinton (1998), with an additional regularization term. By controlling the strength of this information theoretic regularization term using a *scale parameter*, we can explore a range of target functions. On the one end of the spectrum there is a trivial target where compression of the data is total and all relevant information is lost. On the other extreme is the target function of EM.

This continuum of target functions allow us to learn using a procedure motivated by the *deterministic annealing* approach (Rose, 1998). We start with the optimum of the trivial target function and slowly change the scale parameter while tracking the local optimum solution at each step on the way. To do so, we present an alternative view of the optimization problem in the joint space of the model parameters and the scale parameter. This provides an appealing method for scanning the range of solutions as in *homotopy continuation* (Watson, 2000).

We generalize our *information bottleneck expectation maximization* (IB-EM) framework for multiple hidden variables and any Bayesian network structure. To make learning feasible for large, real-life problems we show how to introduce variational approximation assumptions into the framework. We further show that, similarly to the case of standard parametric EM, there is a formal relation between the information bottleneck objective in this case and the *variational EM* functional (Jordan et al., 1998).

We then extend the approach to deal with structure learning. As we show, we can easily incorporate our method into the structural EM framework to deal with *model selection* with hidden variables. In doing so, we perform continuation interleaved with model selection steps that change the structure and the scope of the model. On top of standard structure modification steps of adding and removing edges, we introduce two model enrichment operators that take advantage of emergent information cues during the continuation process. The first operator can adapt the cardinality of a hidden variable. Specifically, the cardinality of a hidden variable can increase during the continuation process, increasing the likelihood as long as it is beneficial to do so. The second operator introduces new hidden variables into the network structure. Intuitively, a hidden variable is introduced as a parent of a subset of nodes whose interactions are poorly explained by the current model.

We demonstrate the effectiveness of our information bottleneck EM algorithm in several learning scenarios. First, we learn parameters in general Bayesian networks for several challenging real-life data sets and show significant improvement in generalization performance on held-out test data. Second, we demonstrate the importance of cardinality adaptation for good generalization. We then show how our operator for enriching the network structure with new hidden variables leads to significantly superior models, for several complex real-life problems. Finally, we show that combining both structure enrichment and cardinality adaptation results in further improvement of test performance.

The paper is organized as follows. In Section 2, we give a short background on learning Bayesian networks and on the *Multivariate information bottleneck* of Friedman et al. (2001). In Section 3, we present the basic framework of our IB-EM algorithm. In Section 4, we show how to combine this algorithm with continuation to bypass local maxima. In Section 5 we extend the framework to multiple hidden variables. In Section 6, we demonstrate the method for parameter learning in real-life scenarios. In Section 7, we show how our method can be combined with the structural EM algorithm to learn the structure of a network with hidden variables. In Section 8, we take advantage of emergent structure during the continuation process, and present a method for learning the cardinality of the hidden variables. We apply this method to real-life data in Section 9. In Section 10, we address the model selection challenge of learning new hidden variables. We present experimental evaluation for several real-life problems in Section 11. In Section 12, we give a brief overview of relevant works, and in Section Section 13 we end with a discussion and future directions.

2. Background

In this section we briefly present the basics of learning Bayesian networks from data followed by the essentials of the *multivariate information bottleneck* framework that forms the basis of our approach.

2.1 Bayesian Networks

Consider a finite set $\mathcal{X} = \{X_1, \ldots, X_n\}$ of random variables, where each variable X_i may take on values from a finite set, denoted by $Val(X_i)$. We use capital letter such as X, Y, Z for variable names and lower case letters such as x, y, z to denote specific values taken by those variables. We use bold letters such as $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ when referring to sets of variables. A *Bayesian network* (Pearl, 1988) is an annotated directed acyclic graph that encodes a joint probability distribution over $\mathcal{X}$. Formally, a Bayesian network over $\mathcal{X}$ is a pair $B = \langle \mathcal{G}, \Theta \rangle$. The first component, $\mathcal{G}$, is a directed acyclic graph whose vertices correspond to the random variables in $\mathcal{X}$. The edges in the graph represent direct dependencies between the variables. The graph $\mathcal{G}$ represents independence properties that are assumed to hold in the underlying distribution: Each X_i is independent of its non-descendants given its parents $\mathbf{Pa}_i$ denoted by $(X_i \perp NonDescendants_i \mid \mathbf{Pa}_i)$. The second component, Θ, represent the set of parameters that quantify the network. Each node is annotated with a *conditional probability distribution* $P(X_i \mid \mathbf{Pa}_i)$, representing the conditional probability of the node X_i given its parents in $\mathcal{G}$, defined by the parameters $\theta_{x_i|\mathbf{pa}_i}$ for each value of X_i and $\mathbf{Pa}_i$. A Bayesian network defines a unique joint probability distribution over $\mathcal{X}$ given by

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid \mathbf{Pa}_i).$$

In this distribution, a variable X_i is independent of the rest of the variables given its *Markov blanket* variables. These include the variable's parents, direct children and the parents of those children (spouses).

Given a network structure $\mathcal{G}$, the problem of learning a Bayesian network can be stated as follows: Given a training set $\mathcal{D} = \{\mathbf{x}[1], \ldots, \mathbf{x}[M]\}$ of instances of $\mathbf{X} \subset \mathcal{X}$, we want to learn parameters for the network. In the *Maximum Likelihood* setting we want to find the parameter values θ that maximize the log-likelihood function

$$\log P(\mathcal{D} \mid \mathcal{G}, \theta) = \sum_m \log P(\mathbf{x}[m] \mid \mathcal{G}, \theta).$$

This function can be equivalently (up to a multiplicative constant) written as $\boldsymbol{E}_{\hat{P}}[\log P(\mathbf{X} \mid \mathcal{G}, \theta)]$ where $\hat{P}$ is the empirical distribution in $\mathcal{D}$. When all instances in $\mathcal{D}$ are complete, estimating the *maximum likelihood* parameters can be done efficiently using a closed form solution. This involves empirical sufficient statistics in the form of joint counts

$$N(x_i, \mathbf{pa}_i) = \sum_m 1\{X_i[m] = x_i, \mathbf{Pa}_i[m] = \mathbf{pa}_i\}, \tag{1}$$

where $1\{\}$ is the indicator function. When learning multinomial conditional parameterization, using Dirichlet priors (DeGroot, 1970) amounts to augmenting the empirical counts with *pseudo-counts*

$\alpha(x_i, \mathbf{pa}_i)$.[1] These can thought of as adding imaginary instances that are distributed according to a certain distribution (*e.g.*, uniform) to the training data (Heckerman, 1998). Consequently, from this point on we view priors as modifying the empirical distribution $\hat{P}$ with additional instances, and then apply the maximum likelihood principle.

When learning with hidden variables, the problem is more complex. Since we observe only partial instances, learning also involves "guessing" the values of the hidden variables. In the *expectation maximization* (EM) algorithm (Dempster et al., 1977; Lauritzen, 1995) and its variants (Neal and Hinton, 1998), this issue is addressed by using an auxiliary distribution Q that provides a proxy for the empirical distribution. In the M-step of EM we estimate parameters as though this was the true empirical distribution. In the E-step, we use the data and the current model to optimize the auxiliary distribution over the hidden values resulting in a *completed* empirical distribution. Each of these steps is simpler than the original problem and is guaranteed not to decrease the likelihood. Unfortunately, EM iterations are prone to getting trapped at local maxima, since each step is biased by the choices made by the previous ones. Attempts to address this problem use a variety of strategies (*e.g.*, Glover and Laguna (1993); Kirkpatrick et al. (1983); Rose (1998); Elidan et al. (2002)).

Learning the structure of a network poses additional challenges as the number of possible structures is super-exponential. In practice, structure learning is typically done using a local search procedure, which examines local structure changes that are easily evaluated (add, delete or reverse an edge). This search is usually guided by a scoring function such as the MDL principle based score (Lam and Bacchus, 1994) or the *Bayesian score* (BDe) (Heckerman et al., 1995). Both scores penalize the likelihood of the data to limit the model complexity. An important characteristic of these scoring functions is that when the data instances are complete (that is, each training instance assigns values to all of the variables) the score is *decomposable*. More precisely, the score can be rewritten as the sum

$$\mathcal{S}core(\mathcal{G} : \mathcal{D}) = \sum_i \text{FamScore}_{X_i}(\mathbf{Pa}_i \ : \ \mathcal{D}),$$

where FamScore_{X_i} is the *local* contribution of X_i to the total network score. This term depends only on values of X_i and $\mathbf{Pa}_{X_i}$ in the training instances. In particular, the BDe score is defined as

$$\text{Score}_{\text{BDe}}(\mathcal{G} : \mathcal{D}) = \sum_i \sum_{\mathbf{pa}_i} \left(\log \frac{\Gamma(\alpha(\mathbf{pa}_i))}{\Gamma(N(\mathbf{pa}_i) + \alpha(\mathbf{pa}_i))} + \sum_{x_i} \log \frac{\Gamma(N(x_i, \mathbf{pa}_i) + \alpha(x_i, \mathbf{pa}_i))}{\Gamma(\alpha(x_i, \mathbf{pa}_i))} \right), \qquad (2)$$

where Γ is the Gamma function that generalizes the factorial function for real numbers, the terms $\alpha()$ are hyper-parameters of the prior distributions over the parameterizations and the terms $N()$ are the corresponding empirical *sufficient statistics*.

In the presence of incomplete data or hidden variables, the structural EM framework (Friedman, 1997; Meila and Jordan, 1998; Thiesson et al., 1998) can adapt the network structure. In this approach, as in classical *parametric* EM, we use the distribution induced by our current model to probabilistically complete the data. Unlike parametric EM, we then use the completed data to evaluate different candidate structures, and perform structure improvement steps in the *M-step* of the structural EM iteration. As in the case of EM, convergence is guaranteed, albeit to a local maximum. Scoring candidate structures in this scenario is more complex, and computation of the score is typically intractable. Thus, we need to resort to approximations such as the *Cheeseman-Stutz* (CS)

[1] The use of pseudo-counts is slightly different depending on whether we do MAP or Bayesian estimation and depends on the representation used (see (Thiesson, 1997) for more details).

score (Cheeseman et al., 1988; Chickering and Heckerman, 1997), which combines the likelihoods of the parameters found by EM, with an estimate of the penalty term associated with structure.

2.2 Multivariate Information Bottleneck

The *information bottleneck* method (Tishby et al., 1999) is a general non-parametric information-theoretic clustering framework. Given a joint distribution $Q(Y,X)$ of two variables, it attempts to extract the relevant information that Y contains about X. We can think of such information extraction as partitioning the possible values of Y into coarser distinctions that are still informative about X. (The actual details are more complex, as we shall see shortly). For example, we might want to partition the words (Y) appearing in several documents in a way that is most relevant to the topics (X) of these documents.

To achieve this goal, we first need a relevance measure between two random variables X and Y with respect to some probability distribution $Q(X,Y)$. The symmetric *mutual information* measure (Cover and Thomas, 1991)

$$\boldsymbol{I}_Q(X;Y) = \sum_{x,y} Q(x,y) \log \frac{Q(x,y)}{Q(x)Q(y)}$$

is a natural choice as it measures the average number of bits needed to convey the information X contains about Y and vice versa. It is bounded from below by zero when the variables are independent, and attains its maximum when one variable is a deterministic function of the other.

The next step is to introduce a new variable T. This variable provides the *bottleneck* relation between X and Y. In our words and documents example, we want T to maintain the distinctions between words (Y) that provide information for determining the topic of a document (X). For example, the words 'music' and 'lyrics' typically occur together and are typical of the same topic, and thus the distinction between them does not contribute to the prediction of the topic. At the same time, we want T to distinguish between 'music' and 'politics' as they correlate with markedly different topics. Formally, we define T using a stochastic function $Q(T \mid Y)$. On the one hand we want T to compress Y, while on the other hand we want it to preserve information that is relevant to X. Using the mutual information defined above, a balance between these two competing goals is achieved by minimization of the Lagrangian

$$\mathcal{L}[Q] = \boldsymbol{I}_Q(Y;T) - \beta \boldsymbol{I}_Q(T;X), \tag{3}$$

where the parameter β controls the tradeoff. Tishby et al. (1999) show that the optimal partition for a given value of β satisfies

$$Q(t \mid y) = \frac{Q(t)}{Z(y,\beta)} \exp\left\{-\beta \boldsymbol{D}(Q(X \mid y) \| Q(X \mid t))\right\},$$

where

$$\boldsymbol{D}(P(\mathbf{X}) \| Q(\mathbf{X})) = \sum_{\mathbf{x}} P(\mathbf{x}) \log \frac{P(\mathbf{x})}{Q(\mathbf{x})}$$

is the Kulback Leibler divergence between the distributions P and Q over the set of random variables $\mathbf{X}$ (Cover and Thomas, 1991). Repeated iterations of these equations for all t and y converge to a (local) maximum where all equations are satisfied. Practical application of this approach for various clustering problems was demonstrated in several works (e.g., (Slonim and Tishby, 2000, 2001)).

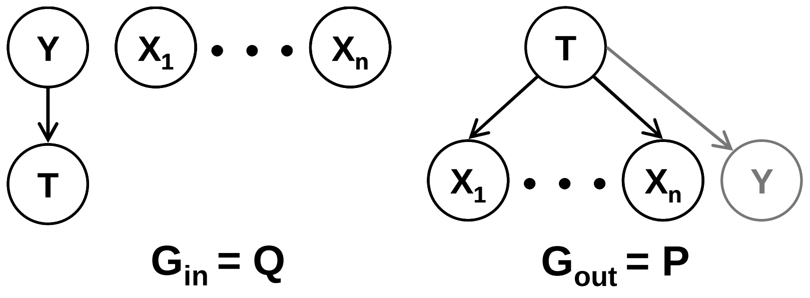


Figure 1: Definition of $\mathcal{G}_{in}$ and $\mathcal{G}_{out}$ for the multivariate information bottleneck framework. $\mathcal{G}_{in}$ encodes the distribution Q that compresses Y. $\mathcal{G}_{out}$ encodes the distribution P that we want to approximate using Q.

The multivariate extension of this framework (Friedman et al., 2001) allows us to consider the interactions of multiple observed variables using several bottleneck variables. For example, we might want to compress words (Y) in a way that preserves information both on the topic of the document (X_1) and on the author of that document (X_2). In addition, there probably is a strong correlation between the author and the topics he writes about. Evidently, the number of possible interactions may be large, and so the framework allows us to specify the interactions we desire. These interactions are represented via two Bayesian networks. The first, called $\mathcal{G}_{in}$, represents the required compression, and the second, called $\mathcal{G}_{out}$, represents the independencies that we are striving for between the bottleneck variables and the target variables. In Figure 1, $\mathcal{G}_{in}$ specifies that T is a stochastic function of its parent in the graph Y. $\mathcal{G}_{out}$ specifies that we want T to make Y and the variables X_i's independent of each other.

Formally, the framework of Friedman et al. (2001), attempts to minimize the Lagrangian

$$\mathcal{L}^{(1)}[\mathcal{G}_{in}, Gout] = I^{\mathcal{G}_{in}} - \beta I^{\mathcal{G}_{out}},$$

where

$$I^{\mathcal{G}} = \sum_i \boldsymbol{I}(X_i; \mathbf{Pa}_i^{\mathcal{G}})$$

and the information is computed with respect to the probability distribution represented by the network $\mathcal{G}$. This objective is a direct generalization of Eq. (3), and as before, tractable self-consistent equations characterize the optimal partitioning. Note that, as in the basic information bottleneck formulation, the two objective of the above Lagrangian are competing. On the one hand we want to compress the information between all bottleneck variables $\mathbf{T}$ and their parents in $\mathcal{G}_{in}$. On the other hand we want to preserve, or maximize, the information between the variables and their parents in $\mathcal{G}_{out}$.

Friedman et al. (2001) also present an analogous variational principal that will be useful in our framework. Briefly, the problem is reformulated as a tradeoff between compression of mutual information in $\mathcal{G}_{in}$ so that the bottleneck variable(s) $\mathbf{T}$ help us describe a joint distribution that follows that form of a target Bayesian network $\mathcal{G}_{out}$. Formally, they attempt to minimize the following objective function

$$\mathcal{L}^{(2)}[Q,P] = \boldsymbol{I}_Q(Y;T) + \gamma \boldsymbol{D}(Q(Y,T,\mathbf{X}) \| P(Y,T,\mathbf{X})), \tag{4}$$

where Q and P are joint probabilities that can be represented by the networks of $\mathcal{G}_{in}$ and $\mathcal{G}_{out}$, respectively. The two principals are analogous under the transformation $\beta = \frac{\gamma}{1+\gamma}$ and assuming $I^{Gin} = \boldsymbol{I}_Q(Y;T)$. See Friedman et al. (2001) for more details of the relation between the two principals.

The minimization of the above Lagrangian is over possible parameterizations of $Q(T \mid Y)$ (the marginal $Q(Y,\mathbf{X})$ is given and fixed) and over possible parameterizations of $P(Y,T,\mathbf{X})$ that can be represented by $\mathcal{G}_{out}$. In other words, we want to compress Y in such a way that the distribution defined by $\mathcal{G}_{in}$ is as close as possible to desired distribution of $\mathcal{G}_{out}$. The analogous principal gives us a new view on why these two objectives are conflicting: Consider a distribution that is consistent with $\mathcal{G}_{in}$ so that T is independent of X given Y. On the other hand, a distribution consistent with a specific choice of $\mathcal{G}_{out}$ may require that X is independent of Y given T. Constructing a distribution where both of these requirements actually hold is not useful, may results in T that is equal to either X or Y, making this bottleneck variable redundant.

The scale parameter γ balances the above two factors. When γ is zero we are only interested in compressing the variable Y and we resort to the trivial solution of a single cluster (or an equivalent parameterization). When γ is high we concentrate on choosing $Q(T \mid Y)$ that is close to a distribution satisfying the independencies encoded by $\mathcal{G}_{out}$. Returning to our word-document example. We might be willing to forgo the distinction between 'football' and 'baseball' in which case we would set γ to a relatively low value. On the other hand, we might even want to make a minute distinction between 'Pentium' and 'Celeron' in which case we would set γ to a high value. Obviously, there is no single correct value of γ but rather a range of possible tradeoffs. Accordingly, several approaches were devised to explore the spectrum of solutions as γ varies. These include Deterministic annealing like approaches that start with small value of γ and progressively increase it (Friedman et al., 2001), as well as agglomerative approaches that start with a highly refined solution and gradually compress it (Slonim and Tishby, 2000, 2001; Slonim et al., 2002).

3. Information Bottleneck Expectation Maximization

The main focus of the multivariate information bottleneck (see is on distribution $Q(T \mid Y)$ that is a local maxima solution of the Lagrangian This distribution can be thought of as a soft clustering of the original data. Our emphasis in this work is somewhat different. Given a data set $\mathcal{D} = \{\mathbf{x}[1], \ldots, \mathbf{x}[M]\}$ over the observed variables $\mathbf{X}$, we are interested in learning a better generative model describing the distribution of the observed attributes $\mathbf{X}$. That is, we want to give high probability to new data instances from the same source. In the learned network, the hidden variables will serve to summarize some part of the data while retaining the relevant information on (some) of the observed variables $\mathbf{X}$.

We start by extending the multivariate information bottleneck framework for the task of generalization where, in addition to the task of clustering, we are also interested in learning the generative model P. We emphasize that this is a conceptually different task. In particular, the common view of the information bottleneck framework is as a non-parametric information-theoretic method for clustering (the obvious exception is the work of Slonim and Weiss (2002) mentioned below). In generative learning, on the other hand, we are interested in modeling the distribution. That is, we are ultimately interested in *parameterizing* a specific model so that our generalization prediction on unseen future instances is improved. We start by considering this task for the case of a single hidden variable T and then, in Section 5, extend the framework to several hidden variables.

3.1 The Information Bottleneck EM Lagrangian

If we were only interested in the *training* data and the cardinality of the hidden variable allows it, each state of the hidden variable would have been assigned to a different instance. Consider, for example, a variable T with $|T|$ states that defines a soft clustering on the specific identity of words (Y) appearing in documents while preserving the information relevant to the topic (X) of these documents. Now suppose we are given a set of instances $\mathcal{D} = \{word[i], topic[i]\}$ where i goes from 1 to M, the number of instances. If $|T| = M$ then we could simply deterministically set $Q(T = i \mid word[i]) = 1$ and then predict $topic[i]$ perfectly. While this model achieves perfect training performance, it will clearly have no generalization abilities. Since we are also interested in unknown future samples, we intuitively require that the learned model "forget" the specifics of the training examples. However, in doing so we will also deteriorate the (previously deterministic) prediction of the observed variables. Thus, there is a tradeoff between the compression of the identity of specific instances and the preservation of the information relevant to the observed variables.

We now formalize this idea for the task of learning a generative model over the variables $\mathbf{X}$ and the hidden variable T. We define an additional variable Y to be the instance identity in the training data $\mathcal{D}$. That is, Y takes values in $\{1,\ldots,M\}$ and $Y[m] = m$. We define $Q(Y,\mathbf{X})$ to be the empirical distribution of the variables $\mathbf{X}$ in the data, augmented with the distribution of the new variable Y. For each instance y, $\mathbf{x}[y]$ are the values $\mathbf{X}$ take in the specific instance. We now apply the information bottleneck framework with the graph $\mathcal{G}_{in}$ of Figure 1. The choice of the graph $\mathcal{G}_{out}$ depends on the network model that we want to learn. We take it to be the target Bayesian network, augmented by the additional variable Y, where we set T as Y's parent. For simplicity, we consider as a running example the simple clustering model of $\mathcal{G}_{out}$ where T is the parent of $X_1,\ldots,X_n$. In practice, and as we show in Section 6 any choice of $\mathcal{G}_{out}$ can be used. We now want to optimize the Bottleneck objective as defined by these two networks. This will attempt to define a conditional probability $Q(T \mid Y)$ so that $Q(T,Y,\mathbf{X}) = Q(T \mid Y)Q(Y,\mathbf{X})$ can be approximated by a distribution that factorizes according to $\mathcal{G}_{out}$. This construction will aim to find T that captures the relevant information the instance identity has about the observed attributes. The following proposition concretely defines the objective function for the particular choice of $\mathcal{G}_{in}$ and $\mathcal{G}_{out}$ we are dealing with.

Proposition 1
Let

1. *Y be the instance identity as defined above;*
2. *$\mathcal{G}_{in}$ be a Bayesian network structure such that such that T is independent of $\mathbf{X}$ given Y; and*
3. *$\mathcal{G}_{out}$ be a Bayesian network structure such that Y is a leaf with T as its only parent.*

Then, minimizing the information bottleneck objective function in Eq. (4) is equivalent to minimizing the Lagrangian

$$\mathcal{L}_{EM} = \boldsymbol{I}_Q(T;Y) - \gamma(\boldsymbol{E}_Q[\log P(\mathbf{X},T)] - \boldsymbol{E}_Q[\log Q(T)]),$$

as a function of $Q(T \mid Y)$ and $P(\mathbf{X},T)$.

Note that once the above conditions are satisfied, we can still arbitrarily choose the structure of $\mathcal{G}_{out}$, which encodes independencies of the distribution P we ultimately wish to learn.

Proof: Using the chain rule and the fact that Y and X are independent given T in $\mathcal{G}_{out}$), we can write $P(Y,\mathbf{X},T) = P(Y \mid T)P(\mathbf{X},T)$. Similarly, using the chain rule and the fact that X and T are independent given Y in $\mathcal{G}_{in}$, we can write $Q(Y,\mathbf{X},T) = Q(Y \mid T)Q(T)Q(\mathbf{X} \mid Y)$. Thus,

$$\begin{aligned}
\boldsymbol{D}(Q(Y,\mathbf{X},T)\|P(Y,\mathbf{X},T)) &= \boldsymbol{E}_Q\left[\log\frac{Q(Y \mid T)Q(T)Q(\mathbf{X} \mid Y)}{P(Y \mid T)P(\mathbf{X},T)}\right] \\
&= \boldsymbol{D}(Q(Y \mid T)\|P(Y \mid T)) \\
&\quad + \boldsymbol{E}_Q[\log Q(\mathbf{X} \mid Y)] \\
&\quad + \boldsymbol{E}_Q[\log Q(T)] \\
&\quad - \boldsymbol{E}_Q[\log P(\mathbf{X},T)].
\end{aligned}$$

By setting $P(Y \mid T) = Q(Y \mid T)$, the first term reaches zero, its minimal value. The second term is a constant since we cannot change the input distribution $Q(\mathbf{X} \mid Y)$. Thus, we need to minimize the last two terms and the result follows immediately. ▮

An immediate question is how this target function relates to standard maximum likelihood learning. To explore the connection, we use a formulation of EM introduced by Neal and Hinton (1998). Although EM is usually thought of in terms of changing the parameters of the target function P, Neal and Hinton show how to view it as a dual optimization of P and an auxiliary distribution Q. This auxiliary distribution replaces the given empirical distribution $Q(\mathbf{X})$ with a completed empirical distribution $Q(\mathbf{X},T)$. Using our notation in the above discussion, we can write the functional defined by Neal and Hinton as

$$\mathcal{F}[Q,P] = \boldsymbol{E}_Q[\log P(\mathbf{X},T)] + \boldsymbol{H}_Q(T \mid Y), \tag{5}$$

where $\boldsymbol{H}_Q(T \mid Y) = \boldsymbol{E}_Q[-\log Q(T \mid Y)]$, and $Q(\mathbf{X},Y)$ is fixed to be the observed empirical distribution.

Theorem 2 (Neal and Hinton, 1998) *If (Q^*,P^*) is a stationary point of $\mathcal{F}$, then P^* is a stationary point of the log-likelihood function $\boldsymbol{E}_Q[\log P(\mathbf{X})]$.*

Moreover, Neal and Hinton show that an EM iteration corresponds to maximizing $\mathcal{F}[Q,P]$ with respect to $Q(T \mid Y)$ while holding P fixed, and then maximizing $\mathcal{F}[Q,P]$ with respect to P while holding $Q(T \mid Y)$ fixed. The form of $\mathcal{F}[Q,P]$ is quite similar to the IB-EM Lagrangian, and indeed we can relate the two.

Theorem 3 $\mathcal{L}_{EM} = (1-\gamma)\boldsymbol{I}_Q(T;Y) - \gamma\mathcal{F}[Q,P]$.

Proof: Plugging the identity $\boldsymbol{H}_Q(T \mid Y) = -\boldsymbol{E}_Q[\log Q(T)] - \boldsymbol{I}_Q(T;Y)$ into the EM functional we can write

$$\mathcal{F}[Q,P] = \boldsymbol{E}_Q[\log P(\mathbf{X},T)] - \boldsymbol{E}_Q[\log Q(T)] - \boldsymbol{I}_Q(T;Y).$$

If we now multiply this by γ, and re-arrange terms, we get the form of Proposition 1. ▮

As a consequence, *minimizing* the IB-EM Lagrangian is equivalent to *maximizing* the EM functional combined with an information theoretic regularization term. When $\gamma = 1$, the solutions of the Lagrangian and the EM functional coincide and finding a local minimum of $\mathcal{L}_{EM}$ is equivalent to finding a local maximum of the likelihood function. Slonim and Weiss (2002) provide a similar result for the specific case where the generative model is a mixture model of a univariate X. Their formulation is different than ours in several subtle details that do not allow a direct relation between the two methods. Nonetheless, both Slonim and Weiss (2002) and Theorem 3 show that for a particular value of γ, the information bottleneck Lagrangian coincides with the likelihood objective of EM. The main difference between the two results is the choice of generative models, in our case general multi-variate Bayesian networks, and in the case of Slonim and Weiss (2002), univariate mixture models.

3.2 The Information Bottleneck EM Algorithm

Using the above results, we can now describe the *Information Bottleneck EM* algorithm given a specific value of γ. The algorithm can be described similarly to the EM iterations of Neal and Hinton (1998).

- **E-step**: Maximize $-\mathcal{L}_{EM}$ by varying $Q(T \mid Y)$ while holding P fixed.
- **M-step**: Maximize $-\mathcal{L}_{EM}$ by varying P while holding Q fixed.

Note that the algorithm is formulated in terms of maximizing $-\mathcal{L}_{EM}$ rather than minimizing $\mathcal{L}_{EM}$ to enhance the relation between the Lagrangian and the EM objective.

The M-Step is essentially the standard maximum likelihood optimization of Bayesian networks. To see that, note that the only term that involves P is $\boldsymbol{E}_Q[\log P(\mathbf{X},T)]$. This term has the form of a log-likelihood function, where Q plays the role of the empirical distribution. Since the distribution is over all the variables, we can use sufficient statistics of P for efficient estimates, just as in the case of complete data. Thus, the M step consists of computing expected sufficient statistics given Q, and then using a closed form formula for choosing the parameters of P.

The E-step is a bit more involved. We need to maximize with respect to $Q(T \mid Y)$. To do this we use the following two results that are variants of Theorem 7.1 and Theorem 8.1 of Friedman et al. (2001) and proved using similar techniques (see Appendix A for the full proof).

Proposition 4 *Let $\mathcal{L}_{EM}$ be defined via $\mathcal{G}_{in}$ and $\mathcal{G}_{out}$ as in Proposition 1. $Q(T \mid Y)$ is a stationary point of $\mathcal{L}_{EM}$ with respect to a fixed choice of P if and only if for all values t and y of T and Y, respectively,*

$$Q(t \mid y) = \frac{1}{Z(y,\gamma)} Q(t)^{1-\gamma} P(\mathbf{x}[y],t])^{\gamma}, \tag{6}$$

where $Z(y,\gamma)$ is a normalizing constant:

$$Z(y,\gamma) = \sum_{t'} Q(t')^{1-\gamma} P(\mathbf{x}[y],t'])^{\gamma}.$$

Note that, as can be expected from Theorem 3, when $\gamma = 1$ the update equation reduces to $Q(t \mid y) \propto P(\mathbf{x}[y],t)$ which is equivalent to the standard EM update equation.

Proposition 5 *A stationary point of $\mathcal{L}_{EM}$ is achieved by iteratively applying the self-consistent equations of Proposition 4.*

Combining this result with the result of Neal and Hinton that show that optimization of P increases $F(P,Q)$, we conclude that both the E-step and the M-step increase $-\mathcal{L}_{EM}$ until we reach a stationary point. As in standard EM, in most cases the stationary convergence point reached by applying these self-consistent equations will be a local maximum of $-\mathcal{L}_{EM}$, or a local minimum of $\mathcal{L}_{EM}$.

4. Bypassing Local Maxima using Continuation

As discussed in the previous section, the parameter γ balances between compression of the data and the fit of parameters to $\mathcal{G}_{out}$. When γ is close to 0, our only objective is compressing the data and the effective dimensionality of T will be 1, leading to a trivial solution (or an equivalent parameterization). At larger values of γ we pay more and more attention to the distribution of $\mathcal{G}_{out}$, and we can expect additional states of T to be utilized. Ultimately, we can expect each sample to be assigned to a different cluster (if the dimensionality of T allows it), in which case there is no compression of Y and the information about the Xs is fully preserved. Theorem 3 also tells us that at the limit of $\gamma = 1$ our solution will actually converge to one of the standard EM solutions. In this section we show how to utilize the inherent tradeoff determined by γ to bypass local maxima towards a better solution at $\gamma = 1$.

Naively, we could allow a large cardinality for the hidden variable, set γ to a high value and find the solution of the bottleneck problem. There are several drawbacks to this approach. First, we will typically converge to a sub-optimal solution for the given cardinality and γ, all the more so for $\gamma = 1$ where there are many such maxima. Second, we often do not know the cardinality that should be assigned to the hidden variable. If we use a cardinality for T that is too large, learning will be less robust and might become intractable. If T has too low a dimensionality, we will not fully utilize the potential of the hidden variable. We would like to somehow identify the beneficial number of clusters without having to simply try many options.

To cope with this task, we adopt the *deterministic annealing* strategy (Rose, 1998). In this strategy, we start with $\gamma = 0$ where a single cluster solution is optimal and compression is total. We then progress toward higher values of γ. This gradually introduces additional structure into the learned model. Intuitively, the algorithm starts at a place where a single, easy to compute solution exists, and tracks it through various stages of progressively complex solutions hopefully bypassing local maxima by staying close to the optimal solution at each value of γ. There are several ways of executing this general strategy. The common approach is simply to increase γ in fixed steps, and after each increment apply the iterative algorithm to re-attain a (local) maxima with the new value of γ. On the problems we examine in Section 6, this naive approach did not prove successful.

Instead, we use a more refined approach that utilizes *continuation methods* for executing the annealing strategy. This approach automatically tunes the magnitude of changes in the value of γ, and also tracks the solution from one iteration to the next. To perform continuation, we view the optimization problem in the joint space of the parameters and γ. In this space we want to follow a smooth path from the trivial solution at $\gamma = 0$ to a solution at $\gamma = 1$. Furthermore, we would like this path to follow a local maximum of $\mathcal{L}_{EM}$. As was shown above, this is equivalent to requiring that the fixed point equations hold at all points along the path. Continuation theory (Watson, 2000) guarantees that, excluding degenerate cases, such a path, free of discontinuities, indeed exists. Figure 2 shows a synthetic illustration of the setup. (a) shows the likelihood function of the two extremes of the easy solution at $\gamma = 0$ and the EM function at $\gamma = 1$ in the joint (γ, Q)-space. (b) shows the range of solutions between these extremes and marks the desired path we would like to follow.

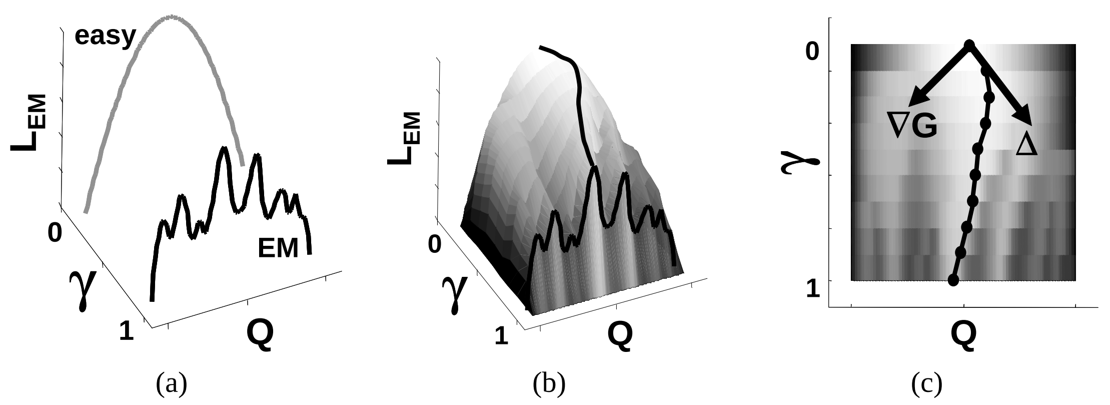


Figure 2: Synthetic illustration of the continuation process. (a) shows the easy likelihood function at $\gamma = 0$ and the complex EM function at $\gamma = 1$. (b) spans the full range of functions and marks the desired path for following the maximum. (c) demonstrates a single step in the continuation process. The gradient $\nabla_{Q,\gamma}G$ is computed and then the orthogonal direction is taken.

We start by characterizing such paths. Note that once we fix the parameters $Q(T \mid Y)$, the M-step maximization of the parameters in P is fully determined as a function of Q. Thus, we take $Q(T \mid Y)$ and γ as the only free parameters in our problem. As we have shown in Proposition 4, when the gradient of the Lagrangian is zero, Eq. (6) holds for each value of t and y. Thus, we want to consider paths where all of these equations hold. Rearranging terms and taking a log of Eq. (6) we define

$$G_{t,y}(Q,\gamma) = -\log Q(t \mid y) + (1-\gamma)\log Q(t) + \gamma \log P(\mathbf{x}[y], y) - \log Z(y,\gamma). \tag{7}$$

Clearly, $G_{t,y}(Q,\gamma) = 0$ exactly when Eq. (6) holds for all t and y. Our goal is then to follow an equi-potential path where all $G_{t,y}(Q,\gamma)$ functions are zero starting from some small value of γ up to the desired EM solution at $\gamma = 1$.

Suppose we are at a point (Q_0, γ_0), where $G_{t,y}(Q_0,\gamma_0) = 0$ for all t and y. We want to move in a direction $\Delta = (dQ, d\gamma)$ so that $(Q_0 + dQ, \gamma_0 + d\gamma)$ also satisfies the fixed point equations. To do so, we want to find a direction Δ, so that

$$\forall t, y, \;\; \nabla_{Q,\gamma} G_{t,y}(Q_0,\gamma_0) \cdot \Delta = 0, \tag{8}$$

where $\nabla_{Q,\gamma} G_{t,y}(Q_0,\gamma_0)$ is the gradient of $G_{t,y}(Q_0,\gamma_0)$ with respect to the parameters Q and γ. Computing these derivatives with respect to each of the parameters results in a derivative matrix

$$H_{t,y}(Q,\gamma) = \left(\frac{\partial G_{t,y}(Q,\gamma))}{\partial Q(t|y)} \;\middle|\; \frac{\partial G_{t,y}(Q,\gamma)}{\partial \gamma} \right). \tag{9}$$

Rows of the matrix correspond to each of the $L = |T| \times |Y|$ functions of Eq. (7), corresponding to joint combinations of the $|T|$ states of the bottleneck variable T and the $|Y| = M$ number of possible values of the instance identity variable Y. The columns correspond to the L parameters of Q as well as γ. The entries correspond to the partial derivative of the function associated with the row with respect to the parameter associated with the column.

To find a direction Δ that satisfies Eq. (8) we need to satisfy the matrix equation

$$H_{t,y}(Q_0,\gamma_0)\Delta = 0. \tag{10}$$

In other words, we are trying to find a vector in the null-space of $H_{t,y}(Q_0,\gamma_0)(Q_0,\gamma_0)$. The matrix H is an $L \times (L+1)$ matrix and its null-space is defined by the intersection of L tangent planes, and is of dimension $L+1-\text{Rank}(H_{t,y}(Q,\gamma))$. Numerically, excluding measure zero cases (Watson, 2000), we expect $\text{Rank}(H_{t,y}(Q_0,\gamma_0))$ to be full, *i.e.*, L. Thus, a unique line that (up to scaling) defines the null space, and we can choose any vector along it. To follow the path to our target objective at $\gamma = 1$ we choose the direction that always increases γ (we discuss the choice of the length of this vector below). Returning to Figure 2, (c) illustrates this process. Shown is joint (γ, Q)-space with the grey-level denoting the value of the likelihood function. At each point in the learning process the gradient of G is evaluated and the orthogonal direction is taken to follow the desired path.

Finding this direction, however, can be costly. Notice that $H_{t,y}(Q,\gamma)$ is of size $L(L+1)$. This number is quadratic in the training set size, and full computation of the matrix is impractical even for small data sets. Instead, we resort to approximating $H_{t,y}(Q,\gamma)$ by a matrix that contains only the diagonal entries $\frac{\partial G_{t,y}(Q,\gamma)}{\partial Q(t|y)}$ and the last column $\frac{\partial G_{t,y}(Q,\gamma)}{\partial \gamma}$. While we cannot bound the extent of this diagonal approximation, we note that the diagonal terms are also the most significant ones and many off diagonal terms are zero. Once we make the approximation, we can solve Eq. (10) in time linear in L. (See Appendix B for a full development of H and the computation of the orthogonal direction.)

Note that once we find a vector Δ that satisfies Eq. (10), we still need to decide on its length, or the size of the step we want to take in that direction. There are various standard approaches, such as normalizing the direction vector to a predetermined size. However, in our problem, we have a natural measure of progress that stems from the tradeoff defined by the target Lagrangian $\mathcal{L}_{EM}$, where $\boldsymbol{I}(T;Y)$ increases when T captures more and more information about the samples during the annealing procedure. That is, the "interesting" steps in the learning process occur when $\boldsymbol{I}(T;Y)$ grows. These are exactly the points where the balance between the two terms in the Lagrangian changes and the second term grows sufficiently to allow the first term to increase $\boldsymbol{I}(T;Y)$. Using $\boldsymbol{I}(T;Y)$ to gauge the progress of the annealing procedure is appealing since it is a non-parametric measure that does not involve the form of the particular distribution of interest P. In addition, in all runs $\boldsymbol{I}(T;Y)$ starts at 0, and is upper-bounded by the log of the cardinality of T and we are thus given a scale of progress.

With this intuition at hand, we want to normalize the step size by the expected change in $\boldsymbol{I}(T;Y)$. That is, we calibrate our progress with respect to the *actual* amount of regularization applied at the current value of γ. At regions where $\boldsymbol{I}(T;Y)$ is not sensitive to changes in the parameters, we can proceed rapidly. On the other hand, if small changes in the parameters result in significant changes of $\boldsymbol{I}(T;Y)$, then we want to carefully track the solution. Figure 3 illustrates the difference between using a predetermined step of γ and partitioning $\boldsymbol{I}(T;Y)$ in order to determine the step size. It is evident the using $\boldsymbol{I}(T;Y)$ causes the method to concentrate on the region of interest in terms of rapid change of the Lagrangian.

Formally, we compute $\nabla_{Q,\gamma}\boldsymbol{I}(T;Y)$ and rescale the direction vector so that

$$(\nabla_{Q,\gamma}\boldsymbol{I}_Q(T;Y))' \cdot \Delta = \varepsilon, \tag{11}$$

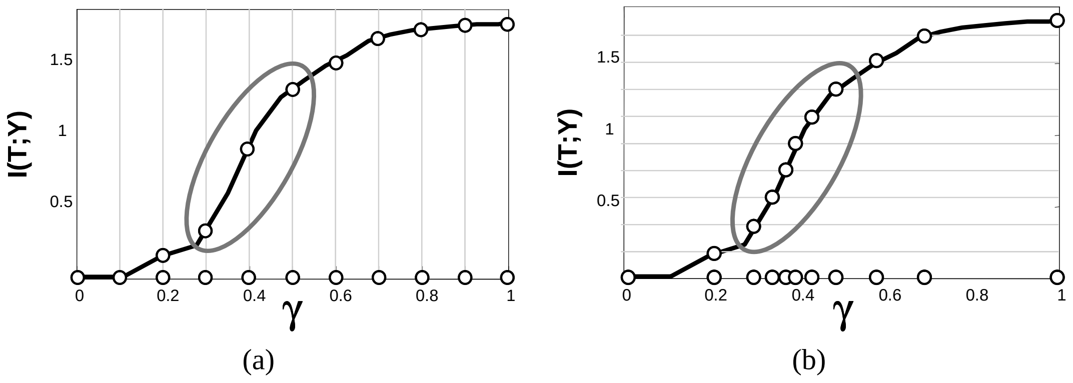


Figure 3: Illustration of the step size calibration process. Both graphs show the change in information between T and Y as a function of γ. The circles denote values of γ to be evaluated. (a) shows naive calibration when fixed steps are taken in the γ range. (b) shows calibration that uses fixed steps in the information range. The grey circle shows the region of dramatic change of the Lagrangian.

where ε is a predetermined step size that is a fraction of $\log|T|$. We also bound the minimal and maximal change in γ so that we do not get trapped in too many steps or alternatively overlook the regions of change.

Finally, although the continuation method takes us in the correct direction, the approximation as well as inherent numerical instability can lead us to a suboptimal path. To cope with this situation, we adopt a commonly used heuristic used in deterministic annealing. At each value of γ, we slightly perturb the current solution and re-iterate the self-consistent equations to converge on a solution. If the perturbation leads to a better value of the Lagrangian, we take it as our current solution.

To summarize, our procedure works as follows: we start with $\gamma = 0$ for which only trivial solutions exists. At each stage we compute the joint direction of γ and $Q(T \mid Y)$ that will leave the fixed point equations intact. We then take a small step in this direction and apply IB-EM iterations to attain the fixed point equilibrium at the new value of γ. We repeat these iterations until we reach $\gamma = 1$.

5. Multiple Hidden Variables

The framework we described in the previous sections can easily accommodate learning networks with multiple hidden variables simply by treating T as a vector of hidden variables. In this case, the distribution $Q(\mathbf{T} \mid Y)$ describes the *joint* distribution of the hidden variables for each value of Y, and $P(\mathbf{T},\mathbf{X})$ describes their joint distribution with the attributes $\mathbf{X}$ in the desired network. Unfortunately, if the number of variables $\mathbf{T}$ is large, the representation of $Q(\mathbf{T} \mid Y)$ grows exponentially, and this approach becomes infeasible.

One strategy to alleviate this problem is to force $Q(\mathbf{T} \mid Y)$ to have a factorized form. This reduces the cost of representing Q and also the cost of performing inference. As an example, we can require that $Q(\mathbf{T} \mid Y)$ is factored as a product $\prod_i Q(T_i \mid Y)$. This assumption is similar to the *mean field variational approximation* (*e.g.*, Jordan et al. (1998)).

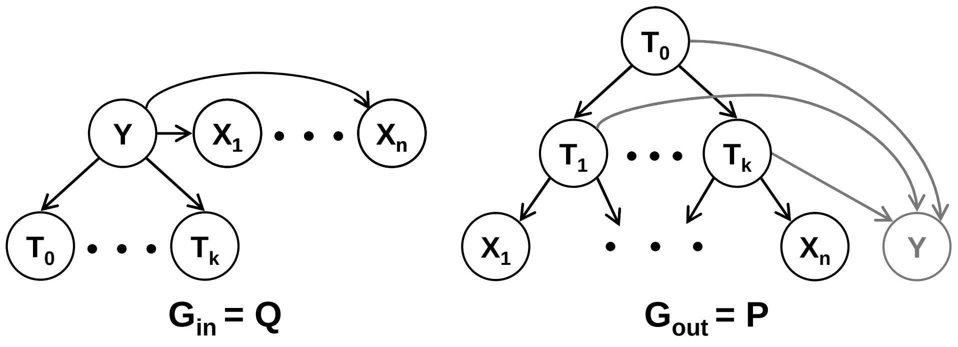


Figure 4: Definition of networks for the multivariate information bottleneck framework with multiple hidden variables. Shown are $\mathcal{G}_{in}$ with the *mean field* assumption, and a possible choice for $\mathcal{G}_{out}$.

In the multivariate information bottleneck framework, different factorizations of $Q(\mathbf{T} \mid Y)$ correspond to different choices of networks $\mathcal{G}_{in}$. For example, the mean field factorization is achieved when $\mathcal{G}_{in}$ is such that the only parent of each T_i is Y, as in Figure 4. In general, we can consider other choices where we introduce edges between the different T_i's. For any such choice of $\mathcal{G}_{in}$, we get exactly the same Lagrangian as in the case of a single hidden variable. The main difference is that since Q has a factorized form, we can decompose $\boldsymbol{I}_Q(\mathbf{T};Y)$. For example, if we use the mean field factorization, we get

$$\boldsymbol{I}_Q(\mathbf{T};Y) = \sum_i \boldsymbol{I}_Q(T_i;Y).$$

Similarly, we can decompose $\boldsymbol{E}_Q[\log P(\mathbf{X},\mathbf{T})]$ into a sum of terms, one for each family in P. These two factorization can lead to tractable computation of the first two terms of the Lagrangian as written in Proposition 1. Unfortunately, the last term $\boldsymbol{E}_Q[\log Q(\mathbf{T})]$ cannot be evaluated efficiently. Thus, we approximate this term as $\sum_i \boldsymbol{E}_Q[\log Q(T_i)]$. For the mean field factorization, the resulting Lagrangian (with this lower bound approximation) has the form

$$\mathcal{L}^{+}_{EM} = \sum_i \boldsymbol{I}_Q(T_i;Y) - \gamma\left(\boldsymbol{E}_Q[\log P(\mathbf{X},T)] - \sum_i \boldsymbol{E}_Q[\log Q(T_i)]\right). \tag{12}$$

The form of $\mathcal{L}^{+}_{EM}$ is valid, if Proposition 1 still holds for the case of multiple hidden variables. This is immediate if we make the following requirements, similar to those made for the case of a single hidden variable:

1. Y is the instance identity;

2. $\mathcal{G}_{in}$ is a Bayesian network structure such that all of the variables $\mathbf{T}$ are independent of X given Y; and

3. $\mathcal{G}_{out}$ is a Bayesian network structure such that Y is a child of $\mathbf{T}$ and has no other parents. This implies that in $\mathcal{G}_{out}$, Y is independent of all $\mathbf{X}$ given $\mathbf{T}$.

The last requirement is needed so that we can set $P(Y \mid T) = Q(Y \mid T)$ in the proof of Proposition 1. As in the case of a single hidden variable, we can now characterize fixed point equations that hold in stationary points of the Lagrangian.

Proposition 6 *Let $\mathcal{L}^{+}_{EM}$ be defined via $\mathcal{G}_{in}$ and $\mathcal{G}_{out}$ as in Eq. (12). Assuming a* mean field *approximation for $Q(\mathbf{T} \mid Y)$, a (local) maximum of $\mathcal{L}^{+}_{EM}$ is achieved by iteratively solving, independently for each hidden variable i, the self-consistent equations*

$$Q(t_i \mid y) \quad = \quad \frac{1}{Z(i,y,\gamma)} Q(t_i)^{1-\gamma} \exp\{\gamma \mathbf{EP}(t_i,y)\},$$

where

$$\mathbf{EP}(t_i,y) \equiv \boldsymbol{E}_{Q(\mathbf{T}|t_i,y)}[\log P(\mathbf{x}[y],\mathbf{T})]$$

and $Z(i,y,\gamma)$ is a normalizing constant that equals to

$$Z(i,y,\gamma) = \sum_{t'_i} Q(t'_i)^{1-\gamma} \exp\left\{\gamma \mathbf{EP}\left(t'_i,y\right)\right\}.$$

See Appendix A for the proof.

The only difference from the case of a single hidden variables is in the form of the expectation $\mathbf{EP}(t_i,y)$. It is easy to see that when a single hidden variable is considered, and $\mathbf{EP}(t_i,y) \equiv \log P(\mathbf{x}[y],t)$, the two forms coincide. It is also easy to see that this term decomposes into a sum of expectations, one for each factor in the factorization of P. We note that only terms that average over factors that involve T_i are of interest in $\mathbf{EP}(t_i,y)$. All other terms do not depend on the value of T_i, and can be absorbed by the normalizing constant. Thus, $\mathbf{EP}(t_i,y)$ can still be computed efficiently.

A more interesting consequence (see theorem below) of this discussion is that when $\gamma = 1$, maximizing $\mathcal{L}^{+}_{EM}$ is equivalent to performing *mean field EM* (Jordan et al., 1998). Thus, by using the modified Lagrangian we generalize this variational learning principle, and as we show below manage to reach better solutions.

The formulation is easily extensible to a general variational approximation of Q where $\mathcal{G}_{in}$ allows, in addition to the dependence of each T_i on Y, dependencies between the different T_i's. In this case, we get

$$\boldsymbol{I}_Q(\mathbf{T};Y) = \sum_i \boldsymbol{I}_Q(T_i;\mathbf{Pa}_{\mathbf{i}}^{\mathcal{G}_{in}}).$$

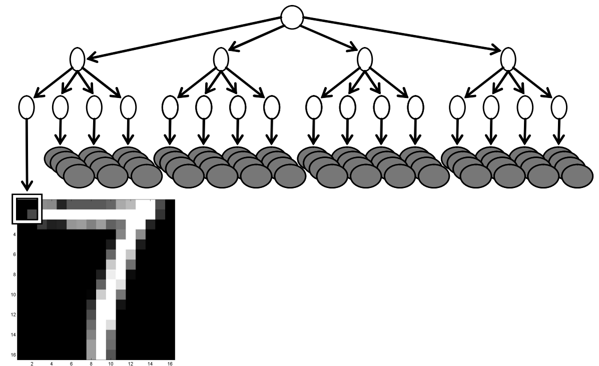
Similarly, $\boldsymbol{E}_Q[\log P(\mathbf{X},T)]$ decomposes according to the *joint* families of T_i in P and in Q. That is, each term in the decomposition depends on T_i, its parents $\mathbf{Pa}_i^{\mathcal{G}_{in}}$ in $\mathcal{G}_{in}$, and its parents $\mathbf{Pa}_i^{\mathcal{G}_{out}}$ in $\mathcal{G}_{out}$. As in the case of the mean field variational approximation, the last term $\boldsymbol{E}_Q[\log Q(\mathbf{T})]$ cannot be evaluated efficiently. We approximate it using a decomposition that follows the structure of $\mathcal{G}_{in}$ as

$$\boldsymbol{E}_Q[\log Q(\mathbf{T})] \approx \sum_i \boldsymbol{E}_Q\left[\log Q(T_i \mid \mathbf{T} \cap \mathbf{Pa}_i^{\mathcal{G}_{in}})\right]. \tag{13}$$

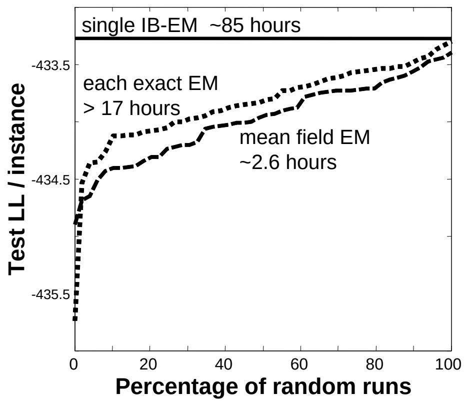
We can now reformulate the results of Theorem 3 for this general case:

Theorem 7 *Let $Q(\mathbf{T} \mid Y)$ decompose according to any structure $\mathcal{G}_{in}$ where all T_i's are descendents of Y and replace $\boldsymbol{E}_Q[\log Q(\mathbf{T})]$ by a decomposition as defined in Eq. (13). Then for the resulting Lagrangian*

$$\mathcal{L}^{+}_{EM} = (1-\gamma) \sum_i \boldsymbol{I}_Q(T_i;\mathbf{Pa}_{\mathbf{i}}^{\mathcal{G}_{in}}) - \gamma \mathcal{F}^{+}[Q,P],$$



(a)



(b)

Figure 5: (a) A quadrant based hierarchy structure with 21 hidden variables for modeling 16×16 images in the Digit domain. (b) Test log-loss of the **IB-EM** algorithm for the model of (a) compared to the cumulative performance of 50 random EM and mean field EM runs.

where $\mathcal{F}^+[Q,P]$ *is defined as in Eq. (5), except that the above decomposition for both* $\boldsymbol{E}_Q[\log P(\mathbf{X},T)]$ *and* $\boldsymbol{H}_Q(T \mid Y)$ *is used.*

Proof: This is a direct result of the fact that in the proof of Theorem 3, no assumptions were made of the form of Q. ∎

The above theorem extends the formal relation of the information bottleneck target Lagrangian and the EM functional for any form of variational approximation encoded by $\mathcal{G}_{in}$. In particular, when $\gamma = 1$, finding a local minimum of $\mathcal{L}^+_{EM}$ is equivalent to finding a local maximum of the likelihood function when the same variational approximation is used in the EM algorithm. Similarly, we can derive the fixed point equations with each for different choices of $\mathcal{G}_{in}$. The change to Proposition 6 is simply a different decomposition for $\mathbf{EP}(i,y)$

To summarize, the IB-EM algorithm of Section 3.2 can be easily generalized to handle multiple hidden variables by simply altering the form of $\mathbf{EP}(t_i,y)$ in the fixed point equations. All other details, such as the continuation method, remain unchanged.

6. Experimental Validation: Parameter Learning

To evaluate the IB-EM method for the task of parameter learning, we examine its generalization performance on several types of models on three real-life data sets. In each architecture, we consider networks with hidden variables of different cardinality, where for now we use the same cardinality for all hidden variables in the same network. We now briefly describe the data sets and the model architectures we use.

- The Stock data set records up/same/down daily changes of 20 major US technology stocks over a period of several years (Boyen et al., 1999). The training set includes 1213 samples and the test set includes 303 instances. We trained a Naive Bayes hidden variable model where the hidden variable is a parent of all the observations.

- The Digits data set contains 7291 training instances and 2007 test instances from the USPS (US Postal Service) data set of handwritten digits (see http://www.kernel-machines.org/data.html). An image is represented by 256 variables, each denoting the gray level of one pixel in a 16×16 matrix. We discretized pixel values into 10 equal bins.

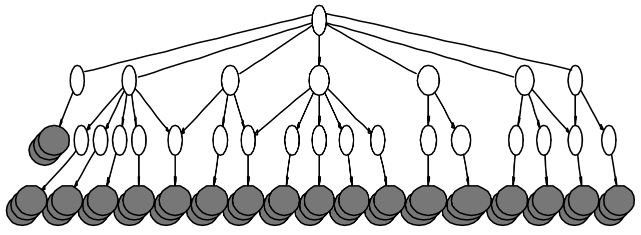
  On this data set we tried several network architectures. The first is a Naive Bayes model with a single hidden variable. In addition, we examined more complex hierarchical models. In these models we introduce a hidden parent to each quadrant of the image recursively. The 3-level hierarchy has a hidden parent to each 8x8 quadrant, and then another hidden variable that is the parent of these four hidden variables. The 4-level hierarchy starts with 4x4 pixel blocks each with a hidden parent. Every 4 of these are joined into an 8x8 quadrant by another level of hidden variables, totaling 21 hidden variables, as illustrated in Figure 5(a).

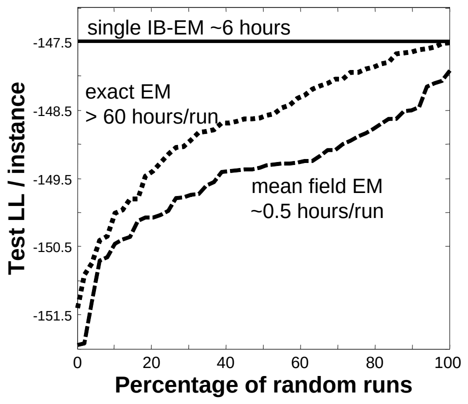
- The Yeast data set contains measurements of the expression of the Baker's yeast genes in 173 experiments (Gasch et al., 2000). These experiments measure the yeast response to changes in its environmental conditions. For each experiment the expression of 6152 genes were measured. We discretized the expression levels of genes into ranges down/same/up by using a threshold of one standard deviation from above and below the gene's mean expression across all experiments. In this data set, we treat each gene as an instance that is described by its behavior in the different experiments. We randomly partitioned the data into 4922 training instances (genes) and 1230 test instances.

  The model we use for this data set has an hierarchical structure with 19 hidden variables in a 4-level hierarchy that was determined by the biological expert based on the nature of the different experiments, as illustrated schematically in Figure 6. In this structure, 5–24 similar conditions (filled nodes) such as different hypo-osmotic shocks are children of a common hidden parent (unfilled nodes). These hidden parents are in their turn children of further abstraction of conditions. For example, the heat shock and heat shock with oxidative stress hidden nodes, are both children of a common more abstract heat node. A root hidden variable is the common parents of these high-level abstractions. Intuitively, each hidden variable encodes how the specific instance (a gene) is altered in the relevant groups of conditions.

As a first sanity check, for each model (and each cardinality of hidden variables) we performed 50 runs of EM with random starting points. The parameter sets learned in these different runs have a wide range of likelihoods both on the training set and the test set. These results (on which we elaborate below), indicate that these learning problems are challenging in the sense that EM runs can be trapped in markedly different local maxima.

Next, we considered the application of IB-EM on these problems. We performed a single IB-EM run on each problem and compared it to the 50 random EM runs, and also to 50 random mean field EM runs. For example, Figure 5 compares the test set performance (log-likelihood per instance) of these runs on the Digit data set with a 4-level hierarchy of 21 hidden variables with 2 states each. The solid line shows the performance of the IB-EM solution at $\gamma = 1$. The two dotted lines show the cumulative performance of the random runs. As we can see, the IB-EM model is superior to all the mean field EM runs, as well as all of the exact EM runs. Figure 6 shows the result for the biological expert constructed hierarchy of Yeast data set with binary variables. As can be seen, in this harder domain, the superiority of the exact EM runs over mean field EM runs is more evident. Yet, the IB-EM run which also use the mean field approximation, is still able to surpass all of the 50 random exact EM runs.


(a)



(b)

Figure 6: (a) A structure constructed by the biological expert for the Yeast data set based on properties of different experiments. 5-24 similar conditions (filled nodes) are aggregated by a common hidden parent (unfilled nodes). These hidden nodes are themselves children of further abstraction nodes of similar experiments, which in their turn are children of the single root node. (b) Comparison of test performance when learning the parameters of the structure of (a) with binary variables. Shown is test log-likelihood per instance of the **IB-EM** algorithm and the cumulative performance of 50 random EM as well as 50 random mean field EM runs.

It is important to note the time required by these runs, all on a Pentium IV 2.4 GHz machine. For the Digit data set, a single mean field EM run requires approximately 2.5 hours, an exact EM run requires roughly 17 hours, and the single IB-EM run requires just over 85 hours. As the IB-EM run reaches a solution that is better than all of this runs, it offers an appealing performance to time tradeoff. This is even more evident for the Yeast data set where the structure is somewhat more complex and the difference between exact learning and the mean field approximation is greater. For this data set, the single IB-EM is still superior and takes significantly less time than a single exact EM.

Figure 7 compares the test log-likelihood per instance performance of our IB-EM algorithms and 50 random EM runs for a range of models for the Stock, Digit and Yeast data sets. In most cases, IB-EM is better than 80% of the EM runs and is often as good or better than the best of them. The advantage of IB-EM is particularly pronounced for the more complex models with higher cardinalities. Table 1 provides more details of these runs including train performance and comparison to 50 random mean field EM runs.

We also compared the IB-EM method to the perturbation method of Elidan et al. (2002). Briefly, their method alters the landscape of the likelihood by perturbing the relative weight of the samples and progressively diminishing this perturbation as a factor of the temperature parameter. In the **Stock** data set, the perturbation method initialized with a starting temperature of 4 and cooling factor of 0.95, had performance similar to that of IB-EM. However, the running time of the perturbation method was an order of magnitude longer. For the other data sets we considered above, running the perturbation method with the same parameters proved to be impractical. When we used more

	Train Log-Likelihood							Test Log-Likelihood						
Model	**IB-EM**	**Random EM**			**Mean Field EM**			**IB-EM**	**Random EM**			**Mean Field EM**		
		%<	100%	80%	%<	100%	80%		%<	100%	80%	%<	100%	80%
Stock														
C=3	-19.91	62%	-19.90	-19.90				-19.90	76%	-19.88	-19.89			
C=4	-19.47	98%	-19.46	-19.52				-19.52	96%	-19.52	-19.62			
C=5	-19.16	94%	-19.15	-19.24				-19.31	98%	-19.30	-19.39			
Digit														
C=5	-429.95	36%	-428.67	-429.11				-439.91	56%	-439.03	-439.47			
C=10	-411.44	100%	-411.72	-413.96				-425.33	100%	-425.36	-427.05			
DigH3														
C=2	-442.02	100%	-442.02	-442.29	100%	-442.03	-442.20	-450.812	92%	-450.76	-450.92	82%	-450.76	-450.84
C=3	-428.77	100%	-428.85	-429.02	100%	-428.83	-429.02	-437.798	98%	-437.74	-438.20	98%	-437.74	-438.04
DigH4														
C=2	-425.43	100%	-425.54	-425.81	100%	-425.61	-425.94	-433.279	100%	-433.30	-433.55	100%	-433.40	-433.71
C=3	-407.60	100%	-407.75	-408.56	100%	-408.49	-408.83	-415.798	100%	-415.88	-416.48	100%	-416.37	-416.77
Yeast														
C=2	-148.13	100%	-148.32	-148.79	100%	-148.89	-149.71	-147.48	100%	-147.51	-147.87	100%	-147.92	-148.78
C=3	-139.44	100%	-139.58	-140.05	100%	-140.09	-140.87	-138.38	100%	-138.57	-139.00	100%	-139.06	-139.92
C=4	-136.36	100%	-136.72	-136.97	100%	-137.72	-138.28	-135.65	100%	-135.96	-136.16	100%	-136.92	-137.34

Table 1: Comparison of the IB-EM algorithm, 50 runs of EM with random starting points, and 50 runs of mean field EM from the same random starting points. Shown are train and test log-likelihood per instance for the best and 80th percentile of the random runs. Also shown is the percentile of the runs that are worse than the IB-EM results. Data sets shown include a Naive Bayes model for the Stock data set and the Digit data set; a 3 and 4 level hierarchical model for the Digit data set (DigH3 and DigH4); and an hierarchical model for the Yeast data set. For each model we show several cardinalities for the hidden variables, shown in the first column.

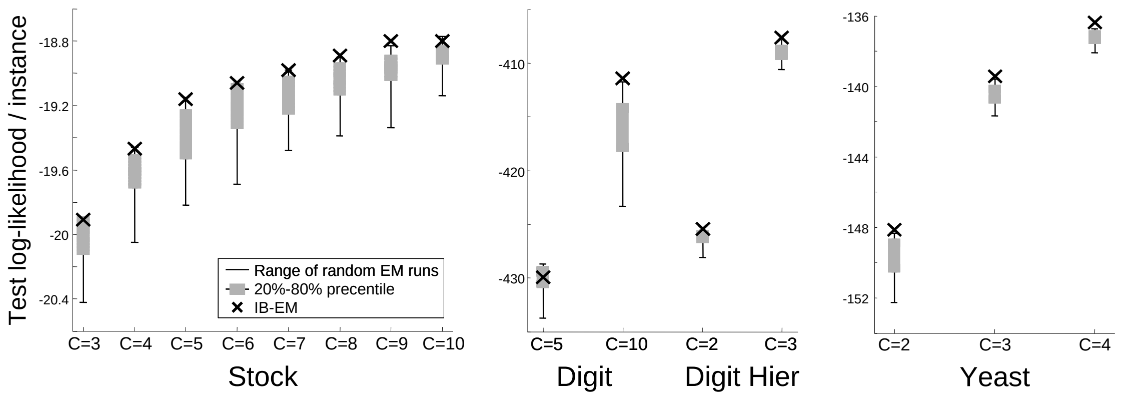


Figure 7: Comparison of log-likelihood per instance test performance of the IB-EM algorithm (black ’X’) and 50 runs of EM with random starting points. The vertical line shows the range of the random runs and boxes mark the 20%-80% range. Data sets shown (x-axis) include a Naive Bayes model for the Stock data set and the Digit data set; a 4 level hierarchical model for the Digit data set (Digit Hier); a hierarchical model for the Yeast data set. For each model we show several cardinalities for the hidden variables, shown in the x-axis.

efficient parameter settings, the perturbation method’s performance was significantly inferior to that of IB-EM. These results do not contradict those of Elidan et al. (2002) who showed some improvement for the case of parameter learning but mainly focused on structure learning, with and without hidden variables.

To demonstrate the effectiveness of the continuation method we examine **IB-EM** during the progress of γ. Figure 8 illustrates the progression of the algorithm on the Stock data set. (a) shows training log-likelihood per instance of parameters in intermediate points in the process. This panel also shows the values of γ evaluated during the continuation process (circles). These were evaluated using the predicted change in $\boldsymbol{I}(T;Y)$ shown in (b). As we can see, the continuation procedure focuses on the region where there are significant changes in $\boldsymbol{I}(T;Y)$ approximately corresponding the areas of significant changes in the likelihood. For both the Stock and Digit data sets, we also tried changing γ naively from 0 to 1 as in standard annealing procedures, without performing continuation. This procedure often “missed” the superior local maxima even when a large number (1000) of γ values were used in the process. In fact, in most runs the results were no better than the average random EM run emphasizing the importance of the continuation in the annealing process.

7. Learning Structure

Up until now, we were only interested in parameter learning. However, in real life it is often not the case that the structure is given. A structure that is too simple will not be able to faithfully capture the distribution, while an overly complex structure will deteriorate our ability to learn. In this section we consider the case where the set of hidden variables is fixed and their cardinalities are known, and we want to learn the network structure. Clearly, this problem is harder than simple

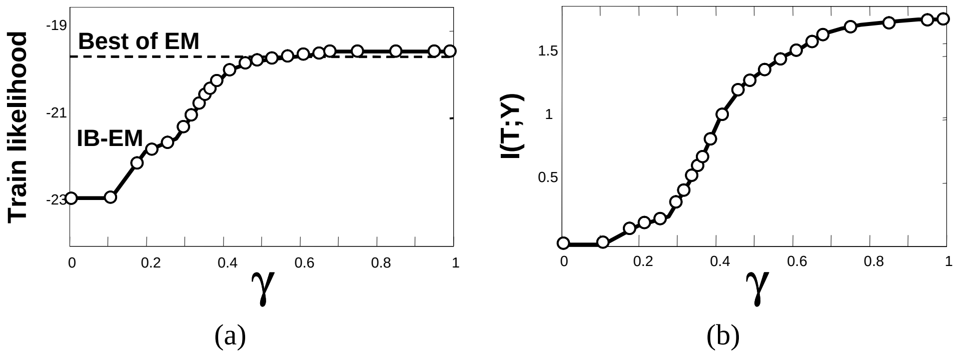


Figure 8: The continuation process for a Naive Bayes model on the Stock data set. (a) Shows the progress of training likelihood as a function of γ compared to the best of 50 EM random runs. Black circles illustrate the progress of the continuation procedure by denoting the value of γ at the end of each continuation step. Calibration is done using information between the hidden variable T and the instance identity Y shown in (b) as a function of γ.

parameter learning, which is just one of the tasks we have to perform in this scenario. The common approach to this model selection task is to use a *score-based approach* where we search for a structure that maximizes some score. Common scores such as the BDe score (Heckerman et al., 1995) balance the likelihood achieved by the model and its complexity. Thus, model selection is achieved independently of the search procedure used (see Section 2.1 for more details).

We now aim to extend the **IB-EM** framework for the task of structure learning using a score-based approach. Naively, we could simply consider different structures and for each one apply the IB-EM procedure to estimate parameters, and then evaluate its generalization ability using the score. Such an approach is extremely inefficient, since it spends a non-trivial amount of time to evaluate each potential candidate structure. In this work we advocate a strategy that is based on the structural EM framework of Friedman (1997). In structural EM, we use the completion distribution Q that is a result of the E-Step to compute *expected sufficient statistics*. That is, instead of Eq. (1), we use

$$\boldsymbol{E}_{Q(\mathbf{T}|Y)}[N(x_i,\mathbf{pa}_i)] = \sum_m \sum_{\mathbf{t}} Q(\mathcal{X}[m] = x_i, \mathbf{Pa}_i = \mathbf{pa}_i, \mathbf{t} \mid Y = m).$$

These statistics are then used in the *M-step* when structure modification steps are evaluated. Thus, instead of assuming that the target structure $\mathcal{G}_{out}$ is fixed, we define the Lagrangian as a function of the pair $(\mathcal{G}_{out},\theta)$. Then, in the M-step, we can consider different choices of $\mathcal{G}_{out}$ and evaluate how each of them changes the score. Given the expected statistics, the problem is identical in form to learning from a fully observed data set and computation of the score is similar. This facilitates an efficient greedy search procedure that uses local edge modification to the network structure. The EM procedure of Section 3.2 is thus revised as follows:

- **E-step** : Maximize $-\mathcal{L}_{EM}$ by varying $Q(T \mid Y)$ while holding P fixed.
- **M-step**: While holding Q fixed:

- Search for the structure $\mathcal{G}_{out}$ of P that maximizes $\text{Score}_{\text{BDe}}(\mathcal{G} : \mathcal{D})$, using the sufficient statistics of Q.
- Maximize $-\mathcal{L}_{EM}$ by varying the parameters of P using the structure $\mathcal{G}_{out}$ selected.

In practice, since the BDe score is not a linear function of the sufficient statistics, we approximate it in the **M-step** using the Cheeseman-Stutz (Cheeseman et al., 1988) approximation. It is important to note the distinction between the optimization of the Lagrangian and that of the score. Specifically, optimizing the Lagrangian involves maximization of the likelihood along with an information theoretic regularization term that does not depend on P. On the other hand, optimization of the structure is performed using the BDe model selection score. This is mathematically valid since each optimization step is ignorant of the inner mechanics of the other step. However, one might wonder why the use of a score is needed at all if regularization is already present in the form of the information theoretic term in the Lagrangian. It is easy to understand the reason for this if we look at the final stage of learning when $\gamma = 1$. At this point, as we have shown, optimizing the Lagrangian is equivalent to optimizing the EM objective. Using the same objective to adapt structure will result in dense structures. In particular, it will be beneficial to add an edge between any two variables that are not perfectly independent in the training data. Thus, while the regularization encoded in the Lagrangian is needed to smooth the parametric EM problem, a model selection regularization via a score is also needed to constrain the network structure.

Using the structural EM framework allows us to apply our framework to structure learning and to use various search operators as simple plug-ins. For general Bayesian networks, for example, one can consider the standard add, delete and reverse edge operators. The only requirement in this case is that a hidden variable is constrained to be non-leaf, in which case it becomes redundant and can be marginalized out. In addition, as in the case of learning parameters, we are still guaranteed to converge for a given value of γ. However, as in parametric EM, convergence is typically to a local maximum. In fact, the problem now has two facets: First, local maxima that result from evaluation of Q in the E-step. Second, local maxima in the discrete structure search space due to the greedy nature of the search algorithm.

Although the method described above applies for any Bayesian network structure, for concreteness we focus on learning *hierarchies* of hidden variables in the following sections. In this sub-class of networks each variable has at most one parent, and that parent has to be a hidden variable. This implies that the hierarchy of hidden variables captures the dependencies between the observed attributes. Since we are dealing with hierarchies we consider search steps that replace the parent of a variable by one of the hidden variables. Such moves preserve the overall hierarchy structure, repositioning a single observed variable, or a sub-hierarchy. We apply these steps in a greedy manner, from the one that leads to the largest improvement, as long as the resulting hierarchy is acyclic.

8. Learning Cardinality

In real life, it is often the case that we do not know the cardinality of a hidden variable. In a clustering application, for example, we typically do not know of a beneficial number of clusters and need to either use some arbitrary choice or spend time evaluating several possibilities. Naively, we might try to set a high cardinality so that we can capture all potential clusters. However, this approach can lead to bad generalization performance due to over-representation. The discussion in Section 4 on the behavior of the model as a function of γ provides insight on the effect of cardinality

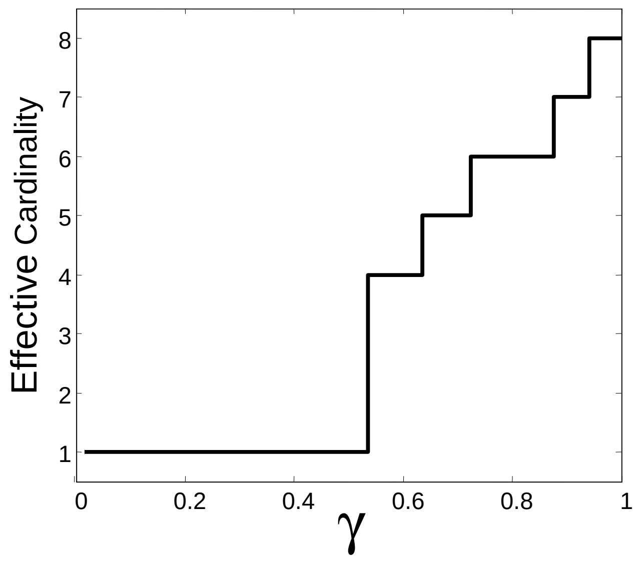


Figure 9: Effective cardinality as a function of γ during the learning process for the Stock data set using a Naive Bayes model. Cardinality is evaluated using local decomposition of the BDe score.

selection. When examining the models during the continuation process, we observe that for lower values of γ the *effective* cardinality of the hidden variable is smaller than its cardinality in the model (we elaborate on how this is measured below). Figure 9 shows an example of this phenomenon for the Naive Bayes model of the Stock data set. Thus, limiting the cardinality of the hidden variable is in effect similar to stopping the continuation process at some $\gamma < 1$. This is, by definition, equivalent to using a regularized version of the EM objective, which can avoid overfitting.

The most straightforward approach to learning the cardinality of a hidden variable is simply to try a few values, and for each value apply IB-EM independently. We can then compare the value of the EM objective (at $\gamma = 1$) corresponding to the different cardinalities. However, models with higher cardinality will achieve a higher likelihood and will thus always be chosen as preferable by the Lagrangian, at the risk of overfitting the training data. In the previous section we discussed the use of a model selection score as a measure for preferring one network structure over another. The same score can also be readily applied for this scenario of cardinality selection. Whether the complexity is a result of a dense structure or an increased number of parameters due to a high cardinality of a variable, all common scores balance the likelihood with the model complexity, either explicitly as in the case of the MDL score (Lam and Bacchus, 1994) or implicitly as in the case of the Bayesian (BDe) score (Heckerman et al., 1995). Thus, similarly to structure learning, we use the Lagrangian when estimating parameters and turn to the score when performing the black-box model selection step. One problem with this simple approach is that it can be extremely time consuming. If we want to try K different cardinalities for each hidden variable, we have to carry out $|H|^K$ independent IB-EM runs, where $|H|$ is the number of hidden variables.

The intuition that the "effective" cardinality of the hidden variable will increase as we consider larger values of γ suggests that we increasing the model complexity during the continuation process.

A simple method is as follows. At each stage allow the model an extra, seemingly redundant, state for the hidden variable. As soon as this state is utilized, we increase the cardinality by adding a new "spare" state. The annealing process, by nature, automatically utilizes this new state when it is beneficial to do so. The task we face is to determine when all the states of a hidden variables are being utilized and therefore a new redundant state is needed. Intuitively, a state of a variable is being used if it captures a distinct behavior that is not captured by other states. That is, for any state i, no other state j is similar.

To determine whether state i is different than all other states, we start by evaluating the cost that we incur due to the merging of state i with another state j. We denote by $\widehat{ij}$ a new state that combines both i and j and alter Q so that

$$Q(T=\widehat{ij} \mid Y=y) = Q(T=i \mid Y=y) + Q(T=j \mid Y=y). \tag{14}$$

We then use this to reestimate the parameters of P in the M-step, and examine the resulting change to the Lagrangian. As shown in Slonim et al. (2002), the difference in the Lagrangian before and after the merge is a sum of Jensen-Shannon divergence terms that measure the difference between the conditional distribution of each child variable given the two states of the hidden variable. This is in fact the change in likelihood of the model resulting from merging the states and can be computed efficiently.

Now that we have the change in the Lagrangian due to the merging of state i with state j, we have to determine whether this change is significant. As already noted, using more states will always improve the likelihood so that the difference in the Lagrangian is not sufficient for model selection. Instead, we can use the BDe score to take into account both the improvement to the likelihood and the change in the model complexity as in Elidan and Friedman (2001). One appealing property of the BDe score is that it is *locally decomposable*. That is, Eq. (2) decomposes according to the different values of each variables. Thus, the difference between the BDe score after and before the merge of states i and j is only in the terms where T appears:

$$\begin{aligned}
&\text{Score}_{\text{BDe}}(\mathcal{G}_{\widehat{ij}} : \mathcal{D}) - \text{Score}_{\text{BDe}}(\mathcal{G}_{i,j} : \mathcal{D}) = \\
&\sum_{pa_t} \left[\log \frac{\Gamma(N^+(T=i,j,pa_t))}{\Gamma(\alpha(T=i,j,pa_t))} - \log \frac{\Gamma(N^+(T=i,pa_t))}{\Gamma(\alpha(T=i,pa_t))} - \log \frac{\Gamma(N^+(T=j,pa_t))}{\Gamma(\alpha(T=j,pa_t))} \right] + \\
&\sum_{C} \sum_{pa_c} \Bigg[\log \frac{\Gamma(\alpha(pa_c,T=i,j))}{\Gamma(N^+(pa_c,T=i,j))} + \sum_c \log \frac{\Gamma(N^+(c,pa_c,T=i,j))}{\Gamma(\alpha(c,pa_c,T=i,j))} \\
&\qquad - \log \frac{\Gamma(\alpha(pa_c,T=i))}{\Gamma(N^+(pa_c,T=i))} - \sum_c \log \frac{\Gamma(N^+(c,pa_c,T=i))}{\Gamma(\alpha(c,pa_c,T=i))} \\
&\qquad - \log \frac{\Gamma(\alpha(pa_c,T=j))}{\Gamma(N^+(pa_c,T=j))} - \sum_c \log \frac{\Gamma(N^+(c,pa_c,T=j))}{\Gamma(\alpha(c,pa_c,T=j))} \Bigg],
\end{aligned}$$

where the first summation correspond to the family of T and its parents, and the second summation is over all $\mathbf{C}$ that are children of T and corresponds to the families of the children of T and their parents. $N^+(x) = N(x) + \alpha(x)$ and correspond to *total* count statistics that include the imaginary prior counts (see Section 2.1). As all the terms are functions of these simple sufficient statistics, the above difference can be computed efficiently. Moreover, as in the case of the likelihood computation, the sufficient statistics needed when merging two states are simply the sum of the statistics needed for scoring the individual states. Thus, we can easily evaluate all pairwise state merges to determine if *any* two states of T are similar.

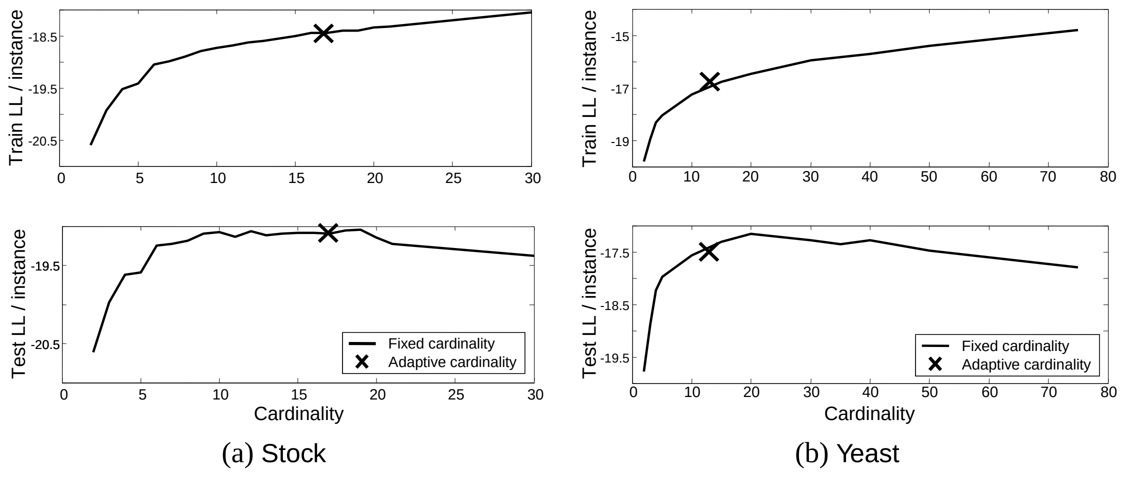


Figure 10: Evaluating adaptive cardinality selection for the Stock and the Yeast data sets with a Naive Bayes model. The 'X' marks the performance of runs with adaptive cardinality selection. The line shows performance of individual runs with a fixed cardinality. The top panel shows training set performance, and the bottom one test set performance.

To summarize, the resulting procedure is as follows. We start with a binary cardinality for the hidden variables at $\gamma = 0$. At each stage, before γ is increased, we determine for each hidden variable if all its states are utilized: For each pair of states we evaluate the BDe score difference between the model with the two states and the model with the states merged. If the difference is positive for all pairs of states then all states are considered utilized and a new state is added. Optimizing the Lagrangian using IB-EM will utilize this new state automatically when it will be beneficial to do so, causing the introduction of a new "spare" state, and so on.

In an early work leading to the formulation of the Information Bottleneck framework, (Pereira et al., 1993) used a similar idea to gauge the effective number of clusters. Briefly, for each cluster a slightly perturbed cluster (twin state) was incorporated in the model allowing each cluster to split into two distinct ones. Similar procedures were used in deterministic annealing (Rose, 1998) and later information bottleneck implementations (Tishby et al., 1999; Slonim et al., 2002). The method we presented in this section differs in two important aspects. First, we use a model selection score to determine when it is beneficial to declare that a redundant cluster is actually being used. This allows us to avoid using an arbitrary distance measure to determine if two clusters diverge. Second, the above allows us to use a single redundant cluster rather than a twin for each state, which significantly reduces the model complexity. While this may not be crucial in standard clustering scenario, it is of great importance for the large models with many hidden variables that we consider in this paper.

9. Experimental Validation: Learning Cardinality

We now want to evaluate the effectiveness of our method for adapting cardinality during the annealing process. For this, we would like to compare the cardinality and model achieved by the method to naive selection of the cardinality. To make this feasible, we look at the context of a Naive Bayes model with a single hidden variable for the Stock and the Yeast data set introduced in Section 6.

We trained the models using the IB-EM algorithm where the hidden variable was assigned a fixed cardinality, and repeated this for different cardinalities. We then applied our adaptive cardinality method to the same model. Figure 10 compares the adaptive cardinality selection run ('X' mark) vs. the fixed cardinality runs for both data sets. As we can see, the adaptive run learns models that generalize nearly as well as the best models learned with fixed cardinality. These results indicate that our method manages to increase cardinality while tracking a high likelihood solution, and that the decision when to add a new state manages to avoid adding spurious states.

A more complex scenario is where, for the Yeast data set, we learn the hierarchy supplied by the biological expert for 62 of the experiments. In this hierarchy there are 6 hidden variables that aggregate similar experiments, a *Heat* node that aggregates 5 of these hidden variables and a root node that is the parent of both *Heat* and the additional *Nitrogen Depletion* node. Figure 11 shows the structure along with the cardinalities of the hidden variables learned by our method and compares the performance of our method to model learned with different fixed cardinalities. As can be seen in (b), the performance of our final model is close to the optimal performance with fixed cardinality. (c) shows that this is achieved with a similar complexity to the simpler of the superior models (at a fixed cardinality of 10).

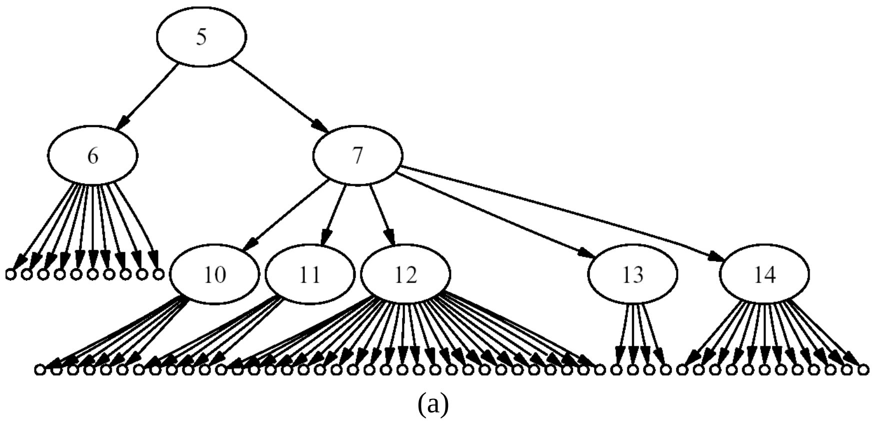
10. Learning New Hidden Variables

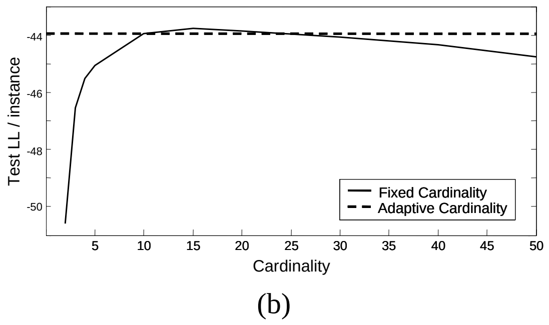
The ideas presented in Section 7 are motivated by the fact that in real life we are typically not given the structure of the Bayesian network. The situation is often even more complex. Hidden variables, as their name implies, are not only unobserved but can also be unknown entities. In this case, we do not even know which variables to include in our model. Thus, we want to determine the number of hidden variables, their cardinality, their relation to the observed variables, and their inter-dependencies. This situation is clearly much more complex than structure learning and might seem hopeless at first. However, as in the case of cardinality adaptation discussed in Section 8, we can use emergent cues of the continuation process to suggest an effective method.

Recall the behavior of our target Lagrangian as a function of γ. For small values of γ, the emphasis of the Lagrangian is on compressing the instance identity, and the hidden variables are (almost) independent of the observed attributes. Thus, at this stage, a simple model would be able to perform just as well as a complex one. In fact, to increase learning robustness, we will want to favor the simpler model and avoid redundant representational complexity. As we increase γ, the hidden variables start capturing properties of the data. In this scenario the need for the more complex structure becomes relevant as it will allow the learning procedure to improve performance.

The above intuition suggests that at small values of γ we start with a simple hierarchy (say, one with only a single hidden variable). When the continuation reaches larger values of γ, the Lagrangian can tolerate more complex structures. Thus, we want to adapt the complexity of the hierarchy as we progress. To do so, we consider a search operator that enriches the structure of hierarchy with a new hidden variable. (This operator is much in the spirit of the "top-down" strategy explored by Adachi and Hasegawa (1996) in learning evolutionary trees.)

Suppose that we want to consider the addition of a new hidden variables into the network structure. For simplicity, consider the scenario shown in Figure 12, where we start with a Naive Bayes network with a hidden variable T_1 as root and want to add a hidden variable T_2 as a parent of a subset $\mathbf{C}$ of T_1's children. Intuitively, we want to select a subset $\mathbf{C}$ that is not "explained well" by T_1 and where we expect to gain a lot by the introduction of T_2. Formally, we evaluate the change in





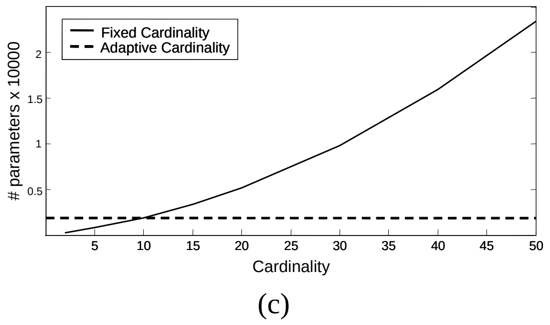


Figure 11: Cardinality learning for the Yeast data set on the structure provided by the biological expert. (a) shows the structure along with the nodes annotated with the cardinality learned by our adaptive approach. (b) shows the test set log-likelihood performance of models learned with different fixed cardinalities (solid line). The horizontal dashed line marks the performance of our adaptive cardinality method. (c) shows plot the number of parameters for each of these models (solid line) with the dashed horizontal line marking the number of parameters of the model learned by our method.

our target Lagrangian as the result of inserting T_2 into the network structure

$$\mathcal{L}_{EM} - \mathcal{L}'_{EM} =$$
$$-\boldsymbol{I}_Q(T_2;Y) + \gamma \boldsymbol{E}_Q[\log P'(T_2 \mid T_1) - \log Q(T_2) + \textstyle\sum_{i \in \mathbf{C}} [\log P'(X_i \mid T_2) - \log P(X_i \mid T_1)]],$$

where P and P' are the models before and after the change to the network, respectively. The term $\log P(X_i \mid T_1)$ can be readily evaluated from the current model for each $X \in \mathbf{C}$ and the terms $\boldsymbol{I}_Q(T_2;Y)$ and $\boldsymbol{E}_Q[\log Q(T_2)]$ can be easily bounded. However, to evaluate $\log P'(T_2 \mid T_1)$ or $\sum_{i \in \mathbf{C}} \log P'(X \mid T_2)$ we need to actually choose $\mathbf{C}$, add T_2 to the current structure and optimize $Q(T_2 \mid Y)$. This can be too costly as the number of possible subsets $\mathbf{C}$ can be large even for a relatively small number of variables. Thus, we want to somehow approximate the above terms efficiently using only the current model. The following bound allows us to do so by bounding the contribution of a hidden variable.

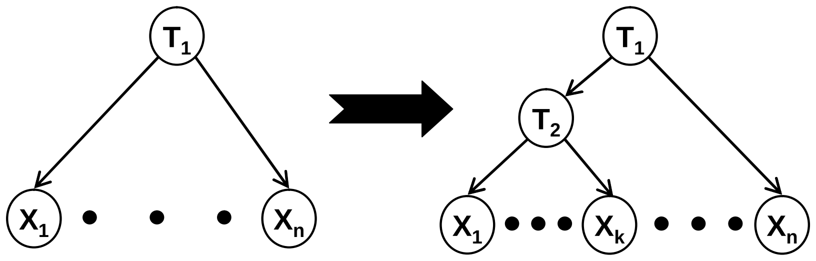


Figure 12: Example of enrichment with new hidden variables T_2 as parent of a subset $\mathbf{C}$ of the observed variables $X_1 \dots X_n$.

Proposition 8 *Let P be a Bayesian network model with a hidden variable T_1 and denote by $\mathbf{C}$ an observed subset of T_1's children. Let P' be the result of replacing T_1 as a parent of $\mathbf{C}$ by T_2, making T_2 a child of T_1 and optimizing the parameters of the model using the IB-EM algorithm for any value of γ. Then*

$$\boldsymbol{E}_Q[\log Q(\mathbf{C} \mid T_1)] \geq \boldsymbol{E}_Q\left[\sum_{i \in \mathbf{C}} \log P'(X_i \mid T_2) + \log P'(T_2 \mid T_1)\right].$$

Proof: Using the chain rule and positivity of entropy, we can write

$$\begin{aligned}
\boldsymbol{E}_Q[\log Q(\mathbf{C} \mid T_1)] &\equiv -\boldsymbol{H}_Q(\mathbf{C} \mid T_1) \\
&= -\Big[\boldsymbol{H}_Q(\mathbf{C}, T_2 \mid T_1) - \boldsymbol{H}_Q(T_2 \mid \mathbf{C}, T_1)\Big] \\
&\geq -\boldsymbol{H}_Q(\mathbf{C}, T_2 \mid T_1) \\
&= -\Big[\boldsymbol{H}_Q(\mathbf{C} \mid T_2, T_1) + \boldsymbol{H}_Q(T_2 \mid T_1)\Big] \\
&= -\Big[\sum_{i \in \mathbf{C}} \boldsymbol{H}_Q(X_i \mid X_1 \dots X_{i-1}, T_2, T_1) + \boldsymbol{H}_Q(T_2 \mid T_1)\Big] \\
&\geq -\Big[\sum_{i \in \mathbf{C}} \boldsymbol{H}_Q(X_i \mid T_2) + \boldsymbol{H}_Q(T_2 \mid T_1)\Big] \\
&\equiv \boldsymbol{E}_Q\left[\sum_{i \in \mathbf{C}} \log P'(X_i \mid T_2) + \log P'(T_2 \mid T_1)\right].
\end{aligned}$$

The last inequality result from the fact that entropy conditioned on less variables can only increase. The final equivalence is a result of the construction of the M-Step of IB-EM, where Q is used when in the optimization of the parameters of P'. ▮

The above proposition provides a bound on the extent to which a hidden variable induces correlations in the marginal distribution. The result is intuitive — the contribution of insertion of a new hidden variable cannot exceed the entropy of its children given their current hidden parent. If we use the bound instead of the original term, we get an over-optimistic estimate of the potential profitability of adding a new hidden variable. However, the scenarios we are interested in are those in which the information between the hidden variable and its children is high and the entropy of

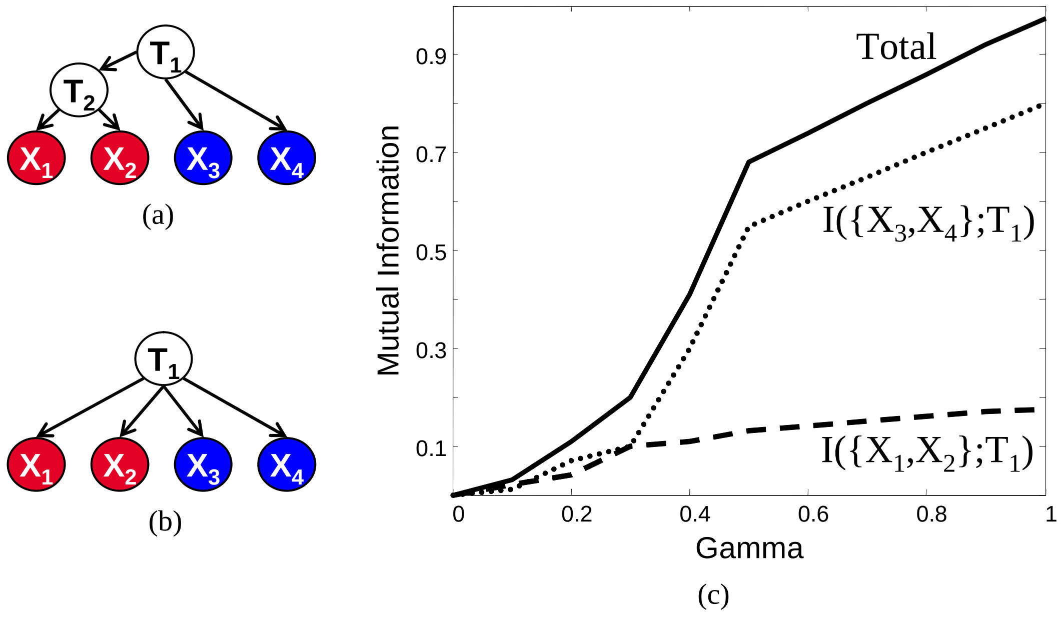


Figure 13: Synthetic example demonstrating the information signal for adding new hidden variables. (a) shows the original structure that generated the samples. (b) shows the structure used in learning without the hidden variable T_2. (c) shows the information as a function of γ between the hidden variables and the observed variables. As learning progresses, the total information rises and the distribution of the direct children of T_1 is captured significantly better (dotted). The information with the original children of T_2 (dashed) remains small.

the hidden variable is low (or there would be no need for it in the network). In such cases, we can expect the bound to be tight in both inequalities.

The above bound provides us with an information signal for putative new hidden variables. In practice, searching for the best subset **C** can be impractical even for relatively small networks. Instead, we use the following greedy approach: first, for each hidden variable, we limit our attention to up to K (we use 20) of its children with the highest entropy individually. We then consider *all* three-node subsets of these children whose entropy level passes some threshold (see details in the experiments below). Intuitively, such seeds will capture the core of the signal needed to attract other nodes when structure change is allowed.[2]

Another complication in using the above signal is a consequence of the annealing process itself. For small values of γ we can expect, and indeed we want, Q to smooth out all statistical signals. This will make most subsets appear equally appealing for adding a hidden variable, since T_1 will not be informative about them. In Section 4, we have shown that $\boldsymbol{I}_Q(Y;T)$ is a natural measure for the progress of the continuation process. To demonstrate the phenomenon in the structure learning scenario, Figure 13 shows a simple synthetic experiment where the samples were generated from

[2] In synthetic experiments for different structures where the network size still made computations feasible, these three node seeds always included two or three variables of the optimal larger subset.

the structure shown in (a) and a Naive Bayes model without T_2 was used when learning. (c) shows the information between the hidden variable T_1 and the observed children (solid), its direct children in the generating distribution (dotted) and the children of T_2 (dashed). Up to some point in the annealing process, the information content of the hidden variable is low and the information with both subsets of variables is low. When the hidden variable starts to capture the distribution of the observed variables, the two subsets diverge and while T_1 captures its original direct children significantly better, the children of T_2 still have high entropy given T_1. Thus, we want to start considering our information "cue" only when the hidden parent becomes meaningful, that is only when $\boldsymbol{I}_Q(Y;T_1)$ passes some threshold.

Finally, we note that although the discussion so far assumed that we have a Naive Bayes model and considered the addition of a single new hidden variable, it is easily generalized for any forms of P where in P' we separate a hidden variables in P from its observed children by introducing a new hidden variable.

To summarize, our approach for learning a new hidden variable T (or several such variables) is as follows: At each value of γ, we first evaluate $\boldsymbol{I}_Q(Y;T)$ to determine if it is above the threshold, signifying that the hidden variable is capturing some of the distribution over the rest of the variables. If this is the case, we greedily search for subsets of children of the hidden variable that have high entropy. These are subsets that are not predicted well by their hidden parent. For the subset with the highest entropy, we suggest a putative new hidden variable that is the parent of the variables in the subset. The purpose of this new variable is to improve the prediction of the subset variables, which are not sufficiently explained by the current model. We then continue with the parameter estimation and structure learning procedure as is. If, after structure search, a hidden variable has at most one child, it is in fact redundant and can be removed from the structure. We iterate the entire procedure until no more hidden variable are added and the structure learning procedure converges.

11. Full Learning — Experimental Validation

We want to evaluate the effectiveness of our method when learning structure with and without the introduction of new hidden variables into the model. We examined two real-life data sets: The Stock data set and the Yeast data set (see Section 6). For the Yeast data set we look at a subset of 62 experiments related to heat conditions and Nitrogen depletion.

In Figure 14 we consider average test set performance on the Stock data set. To create a baseline for hierarchy performance, we train a Naive hierarchy with a single hidden variable and cardinality of 3 totaling 122 parameters. We start by evaluating structure learning without the introduction of new hidden variables. To do this, we generated 25 random hierarchies with 5 binary hidden variables that are parents of the observed variables and a common root parent totaling 91 parameters. We then use structural EM (Friedman, 1997) to adapt the structure by using a *replace-parent* operator where at each local step an observed node can replace its hidden parents. As can be seen in Figure 14, standard structure learning applied to the IB-EM framework significantly improves the model's performance. In fact, many of the 25 random runs with the Search operator surpass the performance of the Naive model using fewer parameters.

Next, we evaluate the ability of the new hidden variable enrichment operator to improve the model. We denote by Enrich the IB-EM run with the automatic enrichment operator. We denote by Enrich+Search the run with this operator augmented with structure search operators in the M-steps. As can be seen in Figure 14, the performance of Enrich by itself was not able to compete with the

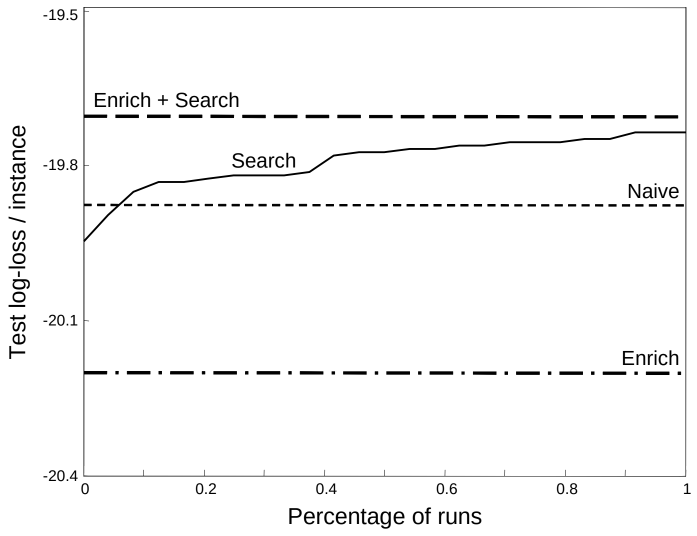


Figure 14: Comparison of performance on the Stock data set of Naive hierarchy (Naive), 25 hierarchies with replace-parent search (Search) , hierarchy learned with enrichment operator (Enrich) and hierarchy learned with enrichment and replace-parent search (Enrich+).

Naive or the Search method. This is not surprising as we cannot expect the information signal to introduce "perfect" hidden variables into the hierarchy. Indeed, when combining the enrichment operator with structure adaptation (Enrich+Search), our method was able to exceed all other runs. The learned hierarchy had only two hidden variables (requiring only 85 parameters). These results show the enrichment operator effectively added useful hidden variables and that the ability to adapt the structure of the network is crucial for utilizing these hidden variables to the best extent.

There are two thresholds used by our algorithm for learning new hidden variables. First, as noted in Section 10, due to the nature of the annealing process we consider adding new hidden variable only when the information $\boldsymbol{I}_Q(Y;T)$ of a hidden variable T in the current structure passes some threshold. In the results presented in this section we use a threshold of 20% of the maximum value the information can reach which is limited by the cardinality of T. Lowering this threshold to as far as 10% or raising it to 40% had negligible effect on the results. We hypothesize that this robustness is caused by the fact that, typically, the cardinality of T will be much lower than Y. Thus, when T undergoes the transition from being redundant to being informative, its information content rises drastically, even if it captures only a small aspect of Y.

The threshold used to limit the number of candidate subsets, however, is more interesting. Recall from Section 10 that the greedy procedure only considers subsets whose entropy passes some threshold. More precisely, we consider only subsets whose entropy passes some percentage of the maximum entropy possible for this subset. Thus, using a lower threshold potentially allows more hidden variables. This is observed empirically in Figure 15(a) for the Yeast data set. A possible concern is that lowering the threshold too much will results in many hidden variables leading to overfitting. However, as is evident in Figure 15(b), even when the number of hidden variables is 20, these new variables are effective in that they improve the generalization performance on unseen test

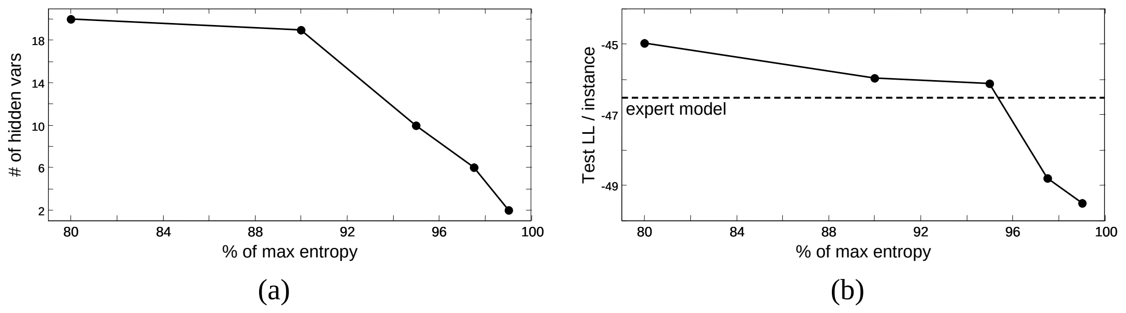


Figure 15: Learning new hidden variables for the Yeast data set. (a) shows the number of variables learned as a function of the threshold on the percentage of entropy of a subset used in the greedy procedure. (b) shows the corresponding test set log-likelihood per instance performance and the performance of the model supplied by the biological expert.

data. In fact, with just a few extra variables, our method successfully surpassed the performance of the structure supplied by the biological expert. Obviously, at some point, having too many variables will lead to overfitting. We could not examine this scenario due to the running time required to learn such large networks.

To qualitatively assess the value of our method, we show in Figure 16 the structure learned for the Stock data set with binary variables and the entropy threshold set at 95% (structures at 92.5% and 97.5% were almost identical for this data set). The emergent structure is evident with the "High-tech giants" and "Internet" group dominating the model. The "Varied" group contains "Canon" and "Sony" that manufacture varied technology products such as electronics, photographic, computer peripheral, etc. The "Japanese" relation of "Toyota" to these companies was interestingly stronger than the relation to the "Car" group.

Finally, we applied runs that combine both automatic cardinality adaptation and enrichment of the structure with new hidden variables. Table 2 shows the train and test performance for the Stock data set. Shown are several runs with the Enrich operator and fixed cardinality. For each run, the number of hidden variables added during the learning process (excluding the initial root node) is noted. Also shown is the automatic cardinality method using the BDe score along with the different cardinalities of the 6 hidden variables introduced into the network structure. The combined method was able to surpass the best of the fixed cardinality models in terms of test set performance with fewer than 70% of the parameters. In addition, the fact that the combined method improves test performance but has worse training likelihood, demonstrates its ability to avoid overfitting.

12. Related Work

To define the IB-EM algorithm, we introduced a formal relation between the information bottleneck (IB) target Lagrangian and the EM functional. This allowed us to formulate an information-theoretic regularization for our learning problem. Given this objective, we used two central ideas to make learning feasible. First, following all annealing methods, we slowly diminish the level of "perturbation" as a way to reach a solution of the hard objective. Second, we use continuation to define a stable traversal from an easy problem to our goal problem.

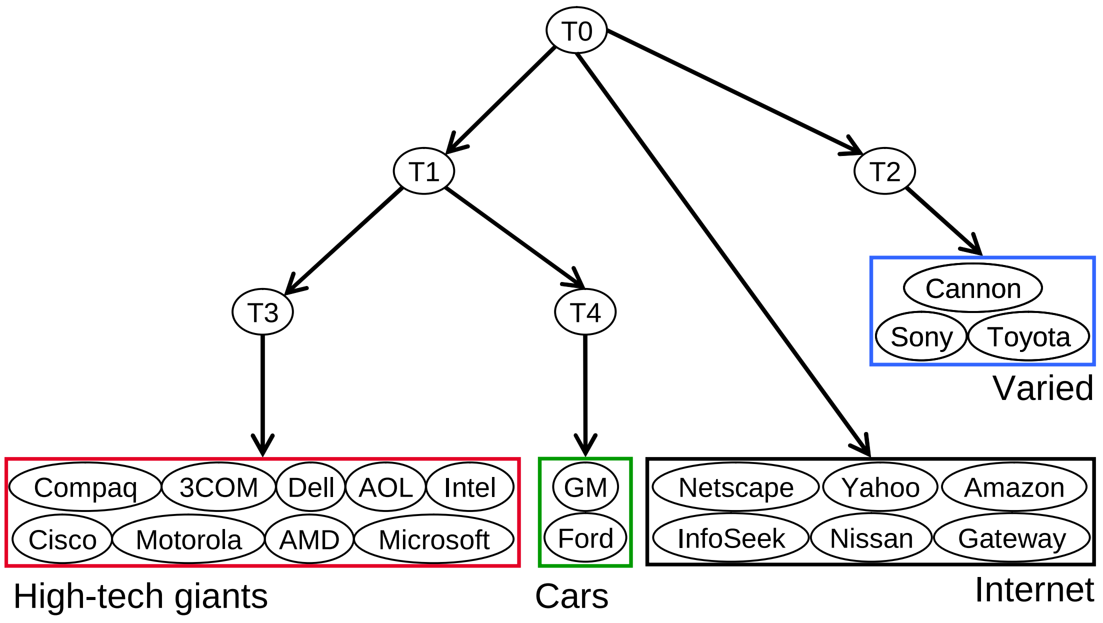


Figure 16: Structure learned for the Stock data set using the enrichment operator augmented with structure search that use the replace-parent operator. All the hidden variables (circles) are binary and the subset entropy threshold was set at 95%. The children of each leaf are annotated with a plausible interpretation.

A multitude of regularization forms are used in machine learning, typically depending on the specific form of the target function (see Bishop (1995) and references within). Information-theoretic regularization has been used for classification with partially labeled data by Szummer and Jaakkola (2002) and for general scenarios in deterministic annealing (Rose, 1998).

Of the annealing methods, the well known *Simulated annealing* (Kirkpatrick et al., 1983) is least similar to ours. Rather than changing the form of the objective function, Simulated annealing allows the search procedure to make "downhill" moves with some diminishing probability. This changes the way the procedure traverses the search space and allows it to potentially reach previously unattainable solutions. Several papers (Heckerman et al., 1994; Chickering, 1996; Elidan et al., 2002) have shown that Simulated annealing is not effective when learning Bayesian networks.

Weight annealing (Elidan et al., 2002), on the other hand, skews the target function directly by perturbing the weights of instances in diminishing magnitudes. Thus, like our method it changes the form of Q directly but does not use an information-theoretic regularization. Weight annealing can actually be applied to a wider variety of problems than our method, including structure search with complete data. However, like other annealing methods, it requires a cooling scheme. For the large problems with hidden variables we explored in this paper, Weight annealing proved inferior with similar running times, and impractical with the settings of Elidan et al. (2002).

Finally, like our method, deterministic annealing (Rose, 1998) alters the problem by explicitly introducing an information-theoretic regularization term. Specifically, following the widely recognized *maximum entropy principle* (Jaynes, 1957), deterministic annealing penalizes the objective

Cardinality	Log-likelihood Train	Test	# of hiddens	# of parameters
2	-19.62	-19.62	5	89
3	-19.32	-19.37	5	146
5	-18.87	-19.04	6	304
10	**-18.53**	**-18.96**	**5**	**769**
20	-18.43	-18.98	5	2340
BDe (9,6,7,7,7,7)	**-18.65**	**-18.94**	**6**	**526**

Table 2: Effect of cardinality when inserting new hidden variables into the network structure with the Enrich operator for the Stock data set. A 95% entropy threshold was used for the hidden variable discovery algorithm. The table shows results for several fixed cardinalities as well as the automatic cardinality method using the BDe score. Shown is the log-likelihood per instance for training as well as test data, the number of hidden variables and the number of parameters in the model. For the automatic method, the cardinalities of each hidden variable is noted.

with a term that is the entropy of the model. A concrete application of deterministic annealing to graphical models was suggested by Ueda and Nakano (1998). However, when learning graphical models, the deterministic annealing was not found to be superior to standard EM (e.g., (Smith and Eisner, 2004)).[3] In particular, Whiley and Titterington (2002); Smith and Eisner (2004)) show why applying deterministic annealing to standard unsupervised learning of Bayesian networks with hidden variables is problematic. One possible explanation for why our method works well for these methods is the difference in motivation of the regularization term. Specifically, our term was motivated by the need for generalization where one want to compress the identity of specific instances. Another important difference between the two methods is that, like Weight annealing, deterministic annealing requires the specification of a cooling policy which makes it potentially impractical for large generative problems. This problem may be avoided using a method similar to the one we used in this work. We leave this prospect as well as the challenge of better understanding the relation between the entropy and information regularization terms for future study.

Continuation methods are a well developed field in mathematics (Watson, 2000). While these methods are used extensively and successfully to solve practical engineering challenges such as complex polynomial systems, they have not been frequently used in machine learning. Recently, Corduneanu and Jaakkola (2002) used continuation to determine a beneficial balance between labeled and unlabeled data. To our knowledge this is the first work in learning graphical models to use continuation to traverse from an easy solution to the desired maximum likelihood problem.

A complementary aspect of our work is the introduction of modification operators for hidden variables. Our method both for learning the cardinality of a hidden variable, and for introducing new hidden variables into the network structure, relies on the annealing process and utilizes emergent signals. The problem of evaluating the cardinality of a hidden variable in a graphical model

[3]Smith and Eisner (2004) also suggest a variant of the deterministic annealing algorithm that appears to work well but is only applicable in the context of semi-supervised learning or when an initial informed starting point for the EM algorithm is at hand.

was explored in several works (*e.g.*, Chang and Fung (1990); Elidan and Friedman (2001)). The work of Stolcke and Omohundro (1993) for HMMs was the first to use evaluation of pairwise state merges to determine adapt the cardinality. In Elidan and Friedman (2001), we extend their method for general Bayesian networks, and Slonim et al. (2002) used a similar approach within the information bottleneck framework. All of these methods start with a large number of states, and then apply bottom-up agglomeration to merge overlaps in the state space and reduce redundancies. By contrast, our method is able to take an "add-when-needed" approach and state mergers are evaluated not to collapse states but rather to determine if a new one is needed. Several papers also explored methods for introducing new hidden variables into the network structure, either for specific classes of Bayesian networks (*e.g.*, Martin and VanLehn (1995); Spirtes et al. (1993); Zhang (2004)) or for general models using a structural signature approach (Elidan et al., 2001). Our contribution in enriching the structure with new hidden variables is twofold. First, we suggested a natural information signature as a "cue" for the presence of a hidden variable. Unlike the structural signature this signature is flexible and is able to weight the influence of different child nodes. Second, we use the enrichment approach in conjunction with the continuation approach for bypassing local maxima. As in cardinality learning, we are able to utilize emergent signals allowing the introduction of new hidden variables into simpler models rendering them more effective.

13. Discussion and Future Work

In this work we addressed the challenge of learning models with hidden variables in real-life scenarios. We presented a general approach for learning the parameters of hidden variables in Bayesian networks and introduced model selection operators that allow learning of new hidden variables and their cardinality. We showed that the method achieves significant improvement on challenging real-life problems.

The contribution of this work is threefold. First, we made a formal connection between the objective functionals of the information bottleneck framework (Tishby et al., 1999; Friedman et al., 2001) and maximum likelihood learning for graphical models. The information bottleneck and its extensions are originally viewed as methods to understand the structure of a distribution. We showed that in some sense the information bottleneck and maximum likelihood estimation are two sides of the same coin. The information bottleneck focuses on the distribution of variables in each instance, while maximum likelihood focuses on the projection of this distribution on the estimated model. This understanding extends to general Bayesian networks the recent results of Slonim and Weiss (2002) that relate the original information bottleneck and maximum likelihood estimation in univariate mixture distributions.

Second, the introduction of the IB-EM principle allowed us to use an approach that starts with a solution at $\gamma = 0$ and progresses toward a solution in the more complex landscape of $\gamma = 1$. This general scheme is common in *deterministic annealing* approaches (Rose, 1998; Ueda and Nakano, 1998). These approaches "flatten" the posterior landscape by raising the likelihood to the power of γ. The main technical difference of our approach is the introduction of a regularization term that is derived from the structure of the approximation of the probability of the latent variables in each instance. This was combined with a continuation method for traversing the path from the trivial solution at $\gamma = 0$ to a solution at $\gamma = 1$. Unlike standard approaches in deterministic annealing and information bottleneck, our procedure can automatically detect important regions where the solution changes drastically and ensure that they are tracked closely. In preliminary experiment the continuation method was clearly superior to standard annealing strategies.

Third, we introduced model enrichment operators for inserting new hidden variables into the network structure and adapting their cardinality. These operators were specifically geared toward utilizing the emergent cues resulting from the annealing procedure. This resulted in models that generalize better and achieve equivalent or better results with a relatively simple model.

The methods presented here can be extended in several directions. First, we can improve the introduction of new hidden variables into the structure by formulating better "signals" that can be efficiently calculated for larger clusters. Second, we can use alternative variational approximations as well as adaptive approximation during the learning process. Third, we want to explore methods for stopping at $\gamma < 1$ as an alternative way for improving generalization performance.

Acknowledgments

We thank R. Bachrach, Y. Barash, G. Chechik, T. Kaplan, D. Koller, M. Ninio, D. Pe'er, A. Regev, T. Tishby, and Y. Weiss for discussions and comments on earlier drafts of this paper. This work was supported, in part, by a grant from the Israeli Ministry of Science and a grant by Intel Corporation. G. Elidan was also supported by the Horowitz fellowship. N. Friedman was also supported by an Alon Fellowship and by the Harry & Abe Sherman Senior Lectureship in Computer Science.

Appendix A. Fixed Point Equations

We now develop the fixed point equations use for solving the target Lagrangian of our approach. We start with the case of a single hidden variables and then address the more general scenario of multiple hidden variables.

A.1 Single Hidden Variable

Proposition 4: *Let $\mathcal{L}_{EM}$ be defined via $\mathcal{G}_{in}$ and $\mathcal{G}_{out}$ as in Proposition 1. $Q(T \mid Y)$ is a stationary point of $\mathcal{L}_{EM}$ with respect to a fixed choice of P if and only if for all values t and y of T and Y, respectively,*

$$Q(t \mid y) = \frac{1}{Z(y,\gamma)} Q(t)^{1-\gamma} P(\mathbf{x}[y],t])^{\gamma},$$

where $Z(y,\gamma)$ is a normalizing constant and equals to

$$Z(y,\gamma) = \sum_{t'} Q(t')^{1-\gamma} P(\mathbf{x}[y],t'])^{\gamma}. \tag{15}$$

To prove the proposition we use the following

Lemma 9 *(El-Hay and Friedman, 2001) Let $Q(\mathbf{X})$ be a joint distribution over a set of random variables $\mathbf{X}$, that decomposes according to $Q(\mathbf{X}) = \prod_i Q(X_i \mid \mathbf{U}_i)$. Then*

$$\frac{\partial \boldsymbol{E}_Q[f(\mathbf{X})]}{\partial Q(x_i \mid \mathbf{u}_i)} = Q(\mathbf{u}_i) \boldsymbol{E}_{Q(\cdot \mid x_i, \mathbf{u}_i)}[f(\mathbf{X})] + \boldsymbol{E}_Q\left[\frac{\partial f(\mathbf{x})}{\partial Q(x_i, \mathbf{u}_i)}\right].$$

The following is an immediate results of that fact that $Q(t) = \sum_{y'} Q(y')Q(t|y')$

$$\frac{\partial Q(T)}{\partial Q(t_0 \mid y_0)} = Q(y_0)1\{T = t_0\}. \tag{16}$$

We use this and an instantiation of the above lemma to prove the following:

Lemma 10

$$\frac{\partial \boldsymbol{I}_Q(T;Y)}{\partial Q(t_0 \mid y_0)} = Q(y_0)\log\frac{Q(t_0|y_0)}{Q(t_0)}.$$

Proof: We define $f(T,Y) \equiv \log\frac{Q(T,Y)}{Q(T)Q(Y)} = \log\frac{Q(T|Y)}{Q(T)}$ so that using Eq. (16), we can write

$$\begin{aligned}\frac{\partial f(T,Y)}{\partial Q(t_0 \mid y_0)} &= \frac{\partial \log Q(T \mid Y)}{\partial Q(t_0 \mid y_0)} - \frac{\partial \log Q(T)}{\partial Q(t_0 \mid y_0)} \\ &= \frac{1}{Q(t_0 \mid y_0)}1\{T = t_0, Y = y_0\} - \frac{Q(y_0)}{Q(t_0)}1\{T = t_0\}.\end{aligned}$$

Plugging this into Lemma 9, we get

$$\begin{aligned}\frac{\partial \boldsymbol{I}_Q(T;Y)}{\partial Q(t_0 \mid y_0)} &= Q(y_0)\boldsymbol{E}_{Q(\cdot|t_0,y_0)}\left[\log\frac{Q(T \mid Y)}{Q(T)}\right] + \boldsymbol{E}_Q\left[\frac{\partial \log\frac{Q(T|Y)}{Q(T)}}{\partial Q(t_0,y_0)}\right] \\ &= Q(y_0)\log\frac{Q(t_0 \mid y_0)}{Q(t_0)} + Q(y_0)\frac{Q(t_0 \mid y_0)}{Q(t_0 \mid y_0)} - \sum_y Q(y)Q(t_0 \mid y)\frac{Q(y_0)}{Q(t_0)} \\ &= Q(y_0)\log\frac{Q(t_0 \mid y_0)}{Q(t_0)} + Q(y_0)\left[1 - \frac{1}{Q(t_0)}\sum_y Q(y)Q(t_0 \mid y)\right] \\ &= Q(y_0)\log\frac{Q(t_0 \mid y_0)}{Q(t_0)} + Q(y_0)[1-1] \\ &= Q(y_0)\log\frac{Q(t_0 \mid y_0)}{Q(t_0)}.\end{aligned}$$

∎

Using Eq. (16) and Lemma 9 with $f(T,Y) \equiv \log Q(T)$, the following is immediate.

Lemma 11

$$\frac{\partial \boldsymbol{E}_Q[\log Q(T)]}{\partial Q(t_0 \mid y_0)} = Q(y_0)\log Q(t_0) + Q(t_0)\frac{1}{Q(t_0)}Q(y_0) = Q(y_0)[\log Q(t_0) + 1].$$

Proof of the proposition: We want to find $Q(T \mid Y)$ that are stationary points of the Lagrangian $\mathcal{L}_{EM}$ and where the constraints $\sum_t Q(t \mid y) = 1$ hold for any y. Thus, using Lagrange multipliers, we want to optimize

$$\mathcal{L} = \boldsymbol{I}_Q(T;Y) - \gamma(\boldsymbol{E}_Q[\log P(\mathbf{X},T)] - \boldsymbol{E}_Q[\log Q(T)]) + \sum_y \lambda_y\left(\sum_{t'} Q(t' \mid y) - 1\right).$$

Since P is fixed, using Lemma 9 with $f(Y,\mathbf{X},T) \equiv \log P(\mathbf{X},T)$, we can write

$$\frac{\partial \boldsymbol{E}_Q[\log P(\mathbf{X},T)]}{\partial Q(t_0 \mid y_0)} = Q(y_0) \log P(\mathbf{x}[y_0], t_0).$$

Combining this with Lemma 10 and Lemma 11, we get

$$\frac{\partial \mathcal{L}_{EM}}{\partial Q(t_0 \mid y_0)} = Q(y_0)\,[\log Q(t_0 \mid y_0) - (1-\gamma)\log Q(t_0) + \gamma - \gamma \log P(\mathbf{x}[y_0], t_0)] + \lambda_{y_0}.$$

Dividing by $Q(y_0)$ and equating to 0, we get after rearranging of terms

$$Q(t_0|y_0) = e^{\lambda_{y_0}/Q(y_0)+\gamma} Q(t_0)^{1-\gamma} P(\mathbf{x}[y_0], t_0)^{\gamma}. \tag{17}$$

This must hold for any value t_0 and y_0. Using $\sum_t Q(t \mid y_0) = 1$ we get

$$e^{\lambda_{y_0}/Q(y_0)+\gamma} = \frac{1}{\sum_t Q(t)^{1-\gamma} P(\mathbf{x}[y_0], t)^{\gamma}}.$$

We get the desired result by plugging this into Eq. (17).

A.2 Multiple Hidden Variables

Proposition 6: *Let $\mathcal{L}^{+}_{EM}$ be defined via $\mathcal{G}_{in}$ and $\mathcal{G}_{out}$ as in Eq. (12). Assuming a* mean field *approximation for $Q(\mathbf{T} \mid Y)$, a (local) maximum of $\mathcal{L}^{+}_{EM}$ is achieved by iteratively solving, independently for each hidden variable i, the self-consistent equations*

$$Q(t_i \mid y) \quad = \quad \frac{1}{Z(i,y,\gamma)} Q(t_i)^{1-\gamma} \exp^{\gamma \mathbf{EP}(t_i,y)},$$

where

$$\mathbf{EP}(t_i,y) \equiv \boldsymbol{E}_{Q(\mathbf{T}|t_i,y)}[\log P(\mathbf{x}[y], \mathbf{T})]$$

and $Z(i,y,\gamma)$ is a normalizing constant that equals to

$$Z(i,y,\gamma) = \sum_{t_i'} Q(t_i')^{1-\gamma} \exp^{\gamma \mathbf{EP}(t_i',y)}.$$

Proof: Using the *mean field* assumption, the information and entropy terms in the Lagrangian decompose as follows

$$\mathcal{L}^{+}_{EM} = \sum_i \boldsymbol{I}_Q(T_i;Y) - \gamma \left(\boldsymbol{E}_Q[\log P(\mathbf{X},T)] - \sum_i \boldsymbol{E}_Q[\log Q(T_i)] \right).$$

When computing the derivative with respect to the parameters of a specific variables T_i, the only change from the case of single hidden variable, is in the derivative of $\boldsymbol{E}_Q[\log P(\mathbf{X},T)]$ given fixed P. Again using Lemma 9 with $f(Y,\mathbf{X},T) \equiv \log P(\mathbf{X},T)$ we get

$$\frac{\partial \boldsymbol{E}_Q[\log P(\mathbf{X},\mathbf{T})]}{\partial Q(t_{i0} \mid y_0)} = \boldsymbol{E}_{Q(T|t_{i0},y_0)}[\log P(\mathbf{x}[y_0], \mathbf{T})],$$

from which we get the change from Proposition 4 to Proposition 6 for the case of multiple hidden variables. ∎

Appendix B. Computing the Continuation Direction

We now develop the precise computations needed to perform continuation as described in Section 4. We start with the case of a single hidden variables T.

B.1 Single Hidden Variable

Consider again Eq. (7), where we now write the normalization term $Z(y,\gamma)$ explicitly:

$$\begin{aligned} G_{t,y}(Q,\gamma) = \; & -\log Q(t \mid y) + (1-\gamma)\log Q(t) + \gamma \log P(\mathbf{x}[y],t) \\ & \underbrace{-\log \sum_{t'} \exp^{(1-\gamma)\log Q(t')+\gamma \log P(\mathbf{x}[y],t')}}_{Z(y,\gamma)} . \end{aligned} \tag{18}$$

We want to compute the derivative of $G_{t,y}(Q,\gamma)$ with respect to the parameters and γ, and and then use the orthogonal direction for continuation. The will follow a direction in which the fix point equations remain unchanged, and the local maximum is tracked. To do so, we start by expressing $\log P(\mathbf{x}[y],t)$ as a function of the parameters Q.

The maximum likelihood parameters of $\log P(\mathbf{X},T)$ for the conditional distribution of the children X_i of T in $\mathcal{G}_{out}$ are

$$\theta_{x_i|\mathbf{pa}_i,t} = \frac{\sum_y Q(y)Q(t|y) 1\{x_i[y]=x_i, \mathbf{pa}_i[y]=\mathbf{pa}_i\} + \alpha(x_i,\mathbf{pa}_i,t)}{\sum_y Q(y)Q(t|y)1\{\mathbf{pa}_i[y]=\mathbf{pa}_i\}+\alpha(\mathbf{pa}_i,t)} \equiv \frac{\mathcal{N}(x_i,\mathbf{pa}_i,t)}{\mathcal{N}(\mathbf{pa}_i,t)}, \tag{19}$$

where $1\{\}$ is the indicator function, $\alpha()$ are the hyper-parameters of the Dirichlet prior distribution (see Section 2.1) and $\mathcal{N}$ are used to denote the total counts (including prior) used for estimation. Similarly the maximum likelihood parameters of the distribution of T given its parents are

$$\theta_{t|\mathbf{pa}_t} = \frac{\sum_y Q(y)Q(t|y)1\{\mathbf{pa}_t[y]=\mathbf{pa}_t\}+\alpha(\mathbf{pa_t},t)}{\sum_y Q(y)1\{\mathbf{pa}_t[y]=\mathbf{pa}_t\}+\alpha(\mathbf{pa}_t)} \equiv \frac{\mathcal{N}(\mathbf{pa}_t,t)}{\mathcal{N}(\mathbf{pa}_t)}. \tag{20}$$

We now consider each term in $G_{t,y}(Q,\gamma)$ and compute its derivative with respect to these parameters of Q.

COMPUTATION OF $\frac{\partial \log P(\mathbf{x}[y],t)}{\partial Q(t_0|y_0)}$

The derivatives of the parameters expressed in Eq. (19) are

$$\begin{aligned} & \frac{\partial \theta_{x_i|\mathbf{pa}_i,t}}{\partial Q(t_0|y_0)} \\ & \quad = \frac{Q(y_0)}{\mathcal{N}(\mathbf{pa}_i,t)^2}\left[\; 1\{x_i[y_0]=x_i,\mathbf{pa}_i[y_0]=\mathbf{pa}_i\}\mathcal{N}(\mathbf{pa}_i,t) - 1\{\mathbf{pa}_i[y_0]=\mathbf{pa}_i\}\mathcal{N}(x_i,\mathbf{pa}_i,t)\;\right] \\ & \quad = \frac{Q(y_0)1\{\mathbf{pa}_i[y_0]=\mathbf{pa}_i\}}{\mathcal{N}(\mathbf{pa}_i,t)^2}\Big(1\{x_i[y_0]=x_i\}\mathcal{N}(\mathbf{pa}_i,t) - \mathcal{N}(x_i,\mathbf{pa}_i,t)\Big) \end{aligned} \tag{21}$$

for $t=t_0$ and are zero otherwise. Similarly, the derivatives of the parameters of Eq. (20) are

$$\frac{\partial \theta_{t|\mathbf{pa}_t}}{\partial Q(t_0 \mid y_0)} = \frac{Q(y_0)}{\mathcal{N}(\mathbf{pa}_t)^2}\left[1\{\mathbf{pa}_t[y_0]=\mathbf{pa}_t\}\mathcal{N}(\mathbf{pa}_t)-0\right] = \frac{Q(y_0)}{\mathcal{N}(\mathbf{pa}_t)}1\{\mathbf{pa}_t[y_0]=\mathbf{pa}_t\} \tag{22}$$

for $t = t_0$, and are zero otherwise. The log-probability of a specific instance can be written as

$$\log P(\mathbf{x}[y],t) = \log\theta_{t|\mathbf{pa}_t}[y] + \sum_{i\in Ch_t}\log\theta_{x_i|\mathbf{pa}_i,t}[y] + \sum_{i\neq t,Ch_t}\log\theta_{x_i|\mathbf{pa}_i}[y], \tag{23}$$

where Ch_t denotes the children of T in $\mathcal{G}_{out}$ and $\theta_{t|\mathbf{pa}_t}[y]$ is the parameter corresponding to the values appearing in instance y. We note that the last summation does not depend on the parameters $Q(t \mid y)$, and by plugging Eq. (21) and Eq. (22) into Eq. (23), we get

$$\begin{aligned}\frac{\partial\log P(\mathbf{x}[y],t)}{\partial Q(t_0 \mid y_0)} &= \frac{1}{\theta_{t|\mathbf{pa}_t}[y]}\frac{\partial\theta_{t|\mathbf{pa}_t}[y]}{\partial Q(t_0\mid y_0)} + \sum_{i\in Ch_t}\frac{1}{\theta_{x_i|\mathbf{pa}_i,t}[y]}\frac{\partial\theta_{x_i|\mathbf{pa}_i,t}[y]}{\partial Q(t_0\mid y_0)}\\ &= Q(y_0)\Bigg[\tfrac{1\{\mathbf{pa}_t[y_0]=\mathbf{pa}_t[y]\}}{\mathcal{N}(\mathbf{pa}_t)\theta_{t|\mathbf{pa}_t}[y_0]}\\ &\qquad + \textstyle\sum_{i\in Ch_t}\tfrac{1\{\mathbf{pa}_i[y_0]=\mathbf{pa}_i[y]\}}{\mathcal{N}(\mathbf{pa}_i,t)^2\theta_{x_i|\mathbf{pa}_i}[y_0]}\Big(1\{x_i[y]=x_i[y_0]\}\mathcal{N}(\mathbf{pa}_i,t) - \mathcal{N}(x_i,\mathbf{pa}_i,t)\Big)\Bigg]\\ &\equiv Q(y_0)\mathcal{D}(y,t),\end{aligned} \tag{24}$$

where in the last line we use $\mathcal{D}(y,t)$ to denote the expression in the square brackets.

COMPUTATION OF $\frac{\partial\log Z(y_0,\gamma)}{\partial Q(t_0|y_0)}$

Using Eq. (16) from Appendix A and the above, we can write

$$\frac{\partial\,(1-\gamma)\log Q(t) + \gamma\log P(\mathbf{x}[y],t)}{\partial Q(t_0\mid y_0)} = Q(y_0)\left[\frac{1-\gamma}{Q(t)} + \gamma\mathcal{D}(y,t)\right]. \tag{25}$$

We can now use Eq. (25) to write the derivative of $Z(y,\gamma)$ since it is a summation over similar expressions

$$\begin{aligned}\tfrac{\partial\log Z(y_0,\gamma)}{\partial Q(t_0|y_0)} &= \tfrac{1}{Z(y_0,\gamma)}\exp^{(1-\gamma)\log Q(t_0)+\gamma\log P(\mathbf{x}[y],t_0)}\,Q(y_0)\left[\tfrac{1-\gamma}{Q(t_0)} + \gamma D(y_0,t_0)\right]\\ &= \tfrac{1}{Z(y_0,\gamma)}Q(y_0)Q(t_0)^{1-\gamma}P(\mathbf{x}[y],t_0)^{\gamma}\left[\tfrac{1-\gamma}{Q(t_0)} + \gamma D(y_0,t_0)\right]\\ &= Q(y_0)Q(t_0\mid y_0)\left[\tfrac{1-\gamma}{Q(t_0)} + \gamma D(y_0,t_0)\right],\end{aligned} \tag{26}$$

where the last equality follows from Proposition 4.

COMPUTATION OF $\frac{\partial G_{t,y}(Q,\gamma)}{\partial Q(t_0|y_0)}$

We combine Eq. (25) and Eq. (26) to write

$$\frac{\partial G_{t,y}(Q,\gamma)}{\partial Q(t_0\mid y_0)} = -1\{y=y_0\} + Q(y_0)\left[1-Q(t_0\mid y_0)\right]\left[\frac{1-\gamma}{Q(t_0)} + \gamma D(y_0,t_0)\right]. \tag{27}$$

COMPUTATION OF $\frac{\partial\log Z(y,\gamma)}{\partial\gamma}$

The only term that is not immediate is the derivative of $Z(y,\gamma)$ with respect to γ

$$\begin{aligned}\frac{\partial\log Z(y,\gamma)}{\partial\gamma} &= \frac{1}{Z(y,\gamma)}\sum_{t'}\exp^{(1-\gamma)\log Q(t')+\gamma\log P(\mathbf{x}[y],t')}\left[-\log Q(t') + \log P(\mathbf{x}[y],t')\right]\\ &= \sum_{t'}\frac{1}{Z(y,\gamma)}Q(t')^{1-\gamma}P(\mathbf{x}[y],t')^{\gamma}\left[-\log Q(t') + \log P(\mathbf{x}[y],t')\right]\\ &= \sum_{t'}Q(t'|y)\left[\log P(\mathbf{x}[y],t') - \log Q(t')\right],\end{aligned}$$

from which follows

$$\frac{\partial G_{t,y}(Q,\gamma)}{\partial \gamma} = \log P(\mathbf{x}[y],t) - \log Q(t) - \sum_{t'} Q(t'|y) \left[\log P(\mathbf{x}[y],t') - \log Q(t')\right]. \tag{28}$$

COMPUTATION OF THE CONTINUATION DIRECTION

We can now compute all the elements of the derivative matrix of Eq. (9)

$$H_{t,y}(Q,\gamma) = \left(\frac{\partial G_{t,y}(Q,\gamma))}{\partial Q(t|y)} \middle| \frac{\partial G_{t,y}(Q,\gamma)}{\partial \gamma} \right).$$

To compute the orthogonal direction to the derivative, we solve Eq. (10)

$$H(Q,\gamma)\Delta = 0.$$

As noted in Section 4, this can be prohibitively expensive and we resort to $H(Q,\gamma)$ with a diagonal approximation for elements of $\frac{\partial G_{t,y}(Q,\gamma)}{\partial Q(t|y)}$ computed in Eq. (27). We denote by $h_{y,t}$ the diagonal entry for $Y = y$ and $T = t$ and $h^{\gamma}_{y,t}$ the corresponding derivative with respect to γ. We then have to solve a set of equations of the form

$$d_{t,y}h_{y,t} + d_{\gamma}h^{\gamma}_{y,t} = 0,$$

where $d_{t,y}$ and d_{γ} are the elements of Δ. Setting $d_{\gamma} = 1$ (an equivalent solution up to scaling) we get the unique solution

$$d_{t,y} = -\frac{h^{\gamma}_{y,t}}{h_{y,t}}.$$

Normalizing Δ using the derivative of $\boldsymbol{I}_Q(T;Y)$ as described in Eq. (11) can now be easily computed given the Lemma 10 in Appendix A.

B.2 Multiple Hidden Variables

When computing the derivative with respect to the parameters associated with a specific hidden variable t_i, the only change in $G_{t,y}(Q,\gamma)$ is that $\log P(\mathbf{x}[y],t)$ is replaced by $\boldsymbol{E}_{Q(\mathbf{T}|t_i,y)}[\log P(\mathbf{x}[y],\mathbf{T})]$. In this case we simply compute the expectation of Eq. (24) over the T's that are in the Markov blanket of t_i. The rest of the details remain the same.

References

J. Adachi and M. Hasegawa. Molphy version 2.3, programs for molecular phylogenetics based on maximum likelihood. Technical report, The Institute of Statistical Mathematics, Tokyo, Japan, 1996.

S. Becker, S. Thrun, and K. Obermayer, editors. *Advances in Neural Information Processing Systems 15*. MIT Press, Cambridge, Mass., 2002.

C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, United Kingdom, 1995.

X. Boyen, N. Friedman, and D. Koller. Discovering the hidden structure of complex dynamic systems. In K. Laskey and H. Prade, editors, *Proc. Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI '99)*, pages 91–100, San Francisco, 1999. Morgan Kaufmann.

J.S. Breese and D. Koller, editors. *Proc. Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI '01)*. Morgan Kaufmann, San Francisco, 2001.

K. Chang and R. Fung. Refinement and coarsening of bayesian networks. In P. P. Bonissone, M. Henrion, L. N. Kanal, and J. F. Lemmer, editors, *Proc. Sixth Annual Conference on Uncertainty Artificial Intelligence (UAI '90)*, pages 475–482, San Francisco, 1990. Morgan Kaufmann.

P. Cheeseman, J. Kelly, M. Self, J. Stutz, W. Taylor, and D. Freeman. Autoclass: a Bayesian classification system. In *Proc. Fifth International Workshop on Machine Learning*, pages 54–64. Morgan Kaufmann, San Francisco, 1988.

D. M. Chickering. Learning equivalence classes of Bayesian network structures. In E. Horvitz and F. Jensen, editors, *Proc. Twelfth Conference on Uncertainty in Artificial Intelligence (UAI '96)*, pages 150–157, San Francisco, 1996. Morgan Kaufmann.

D. M. Chickering and D. Heckerman. Efficient approximations for the marginal likelihood of Bayesian networks with hidden variables. *Machine Learning*, 29:181–212, 1997.

A. Corduneanu and T. Jaakkola. Continuation methods for mixing heterogeneous sources. In A. Darwich and N. Friedman, editors, *Proc. Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI '02)*, pages 111–118, San Francisco, 2002. Morgan Kaufmann.

T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & Sons, New York, 1991.

M. H. DeGroot. *Optimal Statistical Decisions*. McGraw-Hill, New York, 1970.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, B 39:1–39, 1977.

T. El-Hay and N. Friedman. Incorporating expressive graphical models in variational approximations: Chain-graphs and hidden variables. In Breese and Koller (2001), pages 136–143.

G. Elidan and N. Friedman. Learning the dimensionality of hidden variables. In Breese and Koller (2001), pages 144–151.

G. Elidan, N. Lotner, N. Friedman, and D. Koller. Discovering hidden variables: A structure-based approach. In Leen et al. (2001), pages 479–485.

G. Elidan, M. Ninio, N. Friedman, and D. Schuurmans. Data perturbation for escaping local maxima in learning. In *Proc. National Conference on Artificial Intelligence (AAAI '02)*, pages 132–139. AAAI Press, Menlo Park, CA, 2002.

N. Friedman. Learning belief networks in the presence of missing values and hidden variables. In D. Fisher, editor, *Proc. Fourteenth International Conference on Machine Learning*, pages 125–133. Morgan Kaufmann, San Francisco, 1997.

N. Friedman, O. Mosenzon, N. Slonim, and N. Tishby. Multivariate information bottleneck. In Breese and Koller (2001), pages 152–161.

A. P. Gasch, P. T. Spellman, C. M. Kao, O. Carmel-Harel, M. B. Eisen, G. Storz, D. Botstein, and P. O. Brown. Genomic expression program in the response of yeast cells to environmental changes. *Molecular Biology of the Cell*, 11:4241–4257, 2000.

F. Glover and M. Laguna. Tabu search. In C. Reeves, editor, *Modern Heuristic Techniques for Combinatorial Problems*, Oxford, England, 1993. Blackwell Scientific Publishing.

D. Heckerman. A tutorial on learning with Bayesian networks. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer, Dordrecht, Netherlands, 1998.

D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. In R. López de Mantarás and D. Poole, editors, *Proc. Tenth Conference on Uncertainty in Artificial Intelligence (UAI '94)*, pages 293–301, San Francisco, 1994. Morgan Kaufmann.

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. T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106:620–630, 1957.

M. I. Jordan, Z. Ghahramani, T. Jaakkola, and L. K. Saul. An introduction to variational approximations methods for graphical models. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer, Dordrecht, Netherlands, 1998.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220 (4598):671–680, 1983.

W. Lam and F. Bacchus. Learning Bayesian belief networks: An approach based on the MDL principle. *Computational Intelligence*, 10:269–293, 1994.

S. L. Lauritzen. The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, 19:191–201, 1995.

T. K. Leen, T. G. Dietterich, and V. Tresp, editors. *Advances in Neural Information Processing Systems 13*. MIT Press, Cambridge, Mass., 2001.

J. Martin and K. VanLehn. Discrete factor analysis: Learning hidden variables in Bayesian networks. Technical report, Department of Computer Science, University of Pittsburgh, 1995.

M. Meila and M. I. Jordan. Estimating dependency structure as a hidden variable. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 584–590, Cambridge, Mass., 1998. MIT Press.

R. M. Neal and G. E. Hinton. A new view of the EM algorithm that justifies incremental and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer, Dordrecht, Netherlands, 1998.

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

F. Pereira, N. Tishby, and L. Lee. Distributional clustering of English words. In *31st Annual Meeting of the ACL*, pages 183–190, 1993.

K. Rose. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. *Proc. IEEE*, 86:2210–2239, 1998.

N. Slonim, N.Friedman, and T.Tishby. Agglomerative multivariate information bottleneck. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 929–936, Cambridge, Mass., 2002. MIT Press.

N. Slonim and N. Tishby. Agglomerative information bottleneck. In S. A. Solla, T. K. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 617–623, Cambridge, Mass., 2000. MIT Press.

N. Slonim and N. Tishby. Data clustering by markovian relaxation and the information bottleneck method. In Leen et al. (2001), pages 640–646.

N. Slonim and Y. Weiss. Maximum likelihood and the information bottleneck. In Becker et al. (2002), pages 351–358.

N. A. Smith and J. Eisner. Annealing techniques for unsupervised statistical language learning. In *Proc. 42nd Annual Meeting of the Association for Computational Linguistics*, 2004.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction and Search*. Number 81 in Lecture Notes in Statistics. Springer-Verlag, New York, 1993.

A. Stolcke and S. Omohundro. Hidden Markov Model induction by bayesian model merging. In Stephen José Hanson, Jack D. Cowan, and C. Lee Giles, editors, *Advances in Neural Information Processing Systems*, volume 5, pages 11–18. Morgan Kaufmann, San Mateo, CA, 1993.

M. Szummer and T. Jaakkola. Information regularization with partially labeled data. In Becker et al. (2002), pages 640–646.

B. Thiesson. Score and information for recursive exponential models with incomplete data. In D. Geiger and P. Shanoy, editors, *Proc. Thirteenth Conference on Uncertainty in Artificial Intelligence (UAI '97)*, San Francisco, 1997. Morgan Kaufmann.

B. Thiesson, C. Meek, D. M. Chickering, and D. Heckerman. Learning mixtures of Bayesian networks. In G. F. Cooper and S. Moral, editors, *Proc. Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI '98)*, pages 504–513, San Francisco, 1998. Morgan Kaufmann.

N. Tishby, F. Pereira, and W. Bialek. The information bottleneck method. In B. Hajek and R. S. Sreenivas, editors, *Proc. 37th Allerton Conference on Communication, Control and Computation*, pages 368–377. University of Illinois, 1999.

N. Ueda and R. Nakano. Deterministic annealing EM algorithm. *Neural Networks*, 11(2):271–282, 1998.

L. T. Watson. Theory of globally convergent probability-one homotopies for non-linear programming. Technical Report TR-00-04, Department of Computer Science, Virginia Tech, 2000.

M. Whiley and D. M. Titterington. Applying the deterministic annealing expectation maximization algorithm to Naive Bayes networks. Technical Report 02-5, Department of Statistics, University of Glasgow, 2002.

N. L. Zhang. Hierarchical latent class models for cluster analysis. *Journal of Machine Learning Research*, 5:697–723, 2004.

Journal of Machine Learning Research 6 (2005) 129–163 Submitted 1/04; Published 1/05

Diffusion Kernels on Statistical Manifolds

John Lafferty LAFFERTY@CS.CMU.EDU
Guy Lebanon LEBANON@CS.CMU.EDU
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213 USA

Editor: Tommi Jaakkola

Abstract

A family of kernels for statistical learning is introduced that exploits the geometric structure of statistical models. The kernels are based on the heat equation on the Riemannian manifold defined by the Fisher information metric associated with a statistical family, and generalize the Gaussian kernel of Euclidean space. As an important special case, kernels based on the geometry of multinomial families are derived, leading to kernel-based learning algorithms that apply naturally to discrete data. Bounds on covering numbers and Rademacher averages for the kernels are proved using bounds on the eigenvalues of the Laplacian on Riemannian manifolds. Experimental results are presented for document classification, for which the use of multinomial geometry is natural and well motivated, and improvements are obtained over the standard use of Gaussian or linear kernels, which have been the standard for text classification.

Keywords: kernels, heat equation, diffusion, information geometry, text classification

1. Introduction

The use of Mercer kernels for transforming linear classification and regression schemes into nonlinear methods is a fundamental idea, one that was recognized early in the development of statistical learning algorithms such as the perceptron, splines, and support vector machines (Aizerman et al., 1964; Kimeldorf and Wahba, 1971; Boser et al., 1992). The resurgence of activity on kernel methods in the machine learning community has led to the further development of this important technique, demonstrating how kernels can be key components in tools for tackling nonlinear data analysis problems, as well as for integrating data from multiple sources.

Kernel methods can typically be viewed either in terms of an implicit representation of a high dimensional feature space, or in terms of regularization theory and smoothing (Poggio and Girosi, 1990). In either case, most standard Mercer kernels such as the Gaussian or radial basis function kernel require data points to be represented as vectors in Euclidean space. This initial processing of data as real-valued feature vectors, which is often carried out in an *ad hoc* manner, has been called the "dirty laundry" of machine learning (Dietterich, 2002)—while the initial Euclidean feature representation is often crucial, there is little theoretical guidance on how it should be obtained. For example, in text classification a standard procedure for preparing the document collection for the application of learning algorithms such as support vector machines is to represent each document as a vector of scores, with each dimension corresponding to a term, possibly after scaling by an inverse document frequency weighting that takes into account the distribution of terms in the collection (Joachims, 2000). While such a representation has proven to be effective, the statistical justification of such a transform of categorical data into Euclidean space is unclear.

Motivated by this need for kernel methods that can be applied to discrete, categorical data, Kondor and Lafferty (2002) propose the use of discrete diffusion kernels and tools from spectral graph theory for data represented by graphs. In this paper, we propose a related construction of kernels based on the heat equation. The key idea in our approach is to begin with a statistical family that is natural for the data being analyzed, and to represent data as points on the statistical manifold associated with the Fisher information metric of this family. We then exploit the geometry of the statistical family; specifically, we consider the heat equation with respect to the Riemannian structure given by the Fisher metric, leading to a Mercer kernel defined on the appropriate function spaces. The result is a family of kernels that generalizes the familiar Gaussian kernel for Euclidean space, and that includes new kernels for discrete data by beginning with statistical families such as the multinomial. Since the kernels are intimately based on the geometry of the Fisher information metric and the heat or diffusion equation on the associated Riemannian manifold, we refer to them here as *information diffusion kernels*.

One apparent limitation of the discrete diffusion kernels of Kondor and Lafferty (2002) is the difficulty of analyzing the associated learning algorithms in the discrete setting. This stems from the fact that general bounds on the spectra of finite or even infinite graphs are difficult to obtain, and research has concentrated on bounds on the first eigenvalues for special families of graphs. In contrast, the kernels we investigate here are over continuous parameter spaces even in the case where the underlying data is discrete, leading to more amenable spectral analysis. We can draw on the considerable body of research in differential geometry that studies the eigenvalues of the geometric Laplacian, and thereby apply some of the machinery that has been developed for analyzing the generalization performance of kernel machines in our setting.

Although the framework proposed is fairly general, in this paper we focus on the application of these ideas to text classification, where the natural statistical family is the multinomial. In the simplest case, the words in a document are modeled as independent draws from a fixed multinomial; non-independent draws, corresponding to n-grams or more complicated mixture models are also possible. For n-gram models, the maximum likelihood multinomial model is obtained simply as normalized counts, and smoothed estimates can be used to remove the zeros. This mapping is then used as an embedding of each document into the statistical family, where the geometric framework applies. We remark that the perspective of associating multinomial models with individual documents has recently been explored in information retrieval, with promising results (Ponte and Croft, 1998; Zhai and Lafferty, 2001).

The statistical manifold of the n-dimensional multinomial family comes from an embedding of the multinomial simplex into the n-dimensional sphere which is isometric under the the Fisher information metric. Thus, the multinomial family can be viewed as a manifold of constant positive curvature. As discussed below, there are mathematical technicalities due to corners and edges on the boundary of the multinomial simplex, but intuitively, the multinomial family can be viewed in this way as a Riemannian manifold with boundary; we address the technicalities by a "rounding" procedure on the simplex. While the heat kernel for this manifold does not have a closed form, we can approximate the kernel in a closed form using the leading term in the parametrix expansion, a small time asymptotic expansion for the heat kernel that is of great use in differential geometry. This results in a kernel that can be readily applied to text documents, and that is well motivated mathematically and statistically.

We present detailed experiments for text classification, using both the WebKB and Reuters data sets, which have become standard test collections. Our experimental results indicate that the multinomial information diffusion kernel performs very well empirically. This improvement can in part be attributed to the role of the Fisher information metric, which results in points near the boundary of the simplex being given relatively more importance than in the flat Euclidean metric. Viewed differently, effects similar to those obtained by heuristically designed term weighting schemes such as inverse document frequency are seen to arise automatically from the geometry of the statistical manifold.

The remaining sections are organized as follows. In Section 2 we review the relevant concepts that are required from Riemannian geometry, including the heat kernel for a general Riemannian manifold and its parametrix expansion. In Section 3 we define the Fisher metric associated with a statistical manifold of distributions, and examine in some detail the special cases of the multinomial and spherical normal families; the proposed use of the heat kernel or its parametrix approximation on the statistical manifold is the main contribution of the paper. Section 4 derives bounds on covering numbers and Rademacher averages for various learning algorithms that use the new kernels, borrowing results from differential geometry on bounds for the geometric Laplacian. Section 5 describes the results of applying the multinomial diffusion kernels to text classification, and we conclude with a discussion of our results in Section 6.

2. The Heat Kernel

In this section we review the basic properties of the heat kernel on a Riemannian manifold, together with its asymptotic expansion, the parametrix. The heat kernel and its parametrix expansion contains a wealth of geometric information, and indeed much of modern differential geometry, notably index theory, is based upon the use of the heat kernel and its generalizations. The fundamental nature of the heat kernel makes it a natural candidate to consider for statistical learning applications. An excellent introductory account of this topic is given by Rosenberg (1997), and an authoritative reference for spectral methods in Riemannian geometry is Schoen and Yau (1994). In Appendix A we review some of the elementary concepts from Riemannian geometry that are required, as these concepts are not widely used in machine learning, in order to help make the paper more self-contained.

2.1 The Heat Kernel

The Laplacian is used to model how heat will diffuse throughout a geometric manifold; the flow is governed by the following second order differential equation with initial conditions

$$\begin{aligned} \frac{\partial f}{\partial t} - \Delta f &= 0 \\ f(x,0) &= f_0(x) \,. \end{aligned}$$

The value $f(x,t)$ describes the heat at location x at time t, beginning from an initial distribution of heat given by $f_0(x)$ at time zero. The heat or diffusion kernel $K_t(x,y)$ is the solution to the heat equation $f(x,t)$ with initial condition given by Dirac's delta function δ_y. As a consequence of the linearity of the heat equation, the heat kernel can be used to generate the solution to the heat equation with arbitrary initial conditions, according to

$$f(x,t) \;=\; \int_M K_t(x,y)\, f_0(y)\, dy \,.$$

As a simple special case, consider heat flow on the circle, or one-dimensional sphere $M = S^1$, with the metric inherited from the Euclidean metric on $\mathbb{R}^2$. Parameterizing the manifold by angle θ, and letting $f(\theta,t) = \sum_{j=0}^{\infty} a_j(t)\cos(j\theta)$ be the discrete cosine transform of the solution to the heat equation, with initial conditions given by $a_j(0) = a_j$, it is seen that the heat equation leads to the equation

$$\sum_{j=0}^{\infty}\left(\frac{d}{dt}a_j(t) + j^2 a_j(t)\right)\cos(j\theta) = 0\,,$$

which is easily solved to obtain $a_j(t) = e^{-j^2 t}$ and therefore $f(\theta,t) = \sum_{j=0}^{\infty} a_j\, e^{-j^2 t}\cos(j\theta)$. As the time parameter t gets large, the solution converges to $f(\theta,t) \longrightarrow a_0$, which is the average value of f; thus, the heat diffuses until the manifold is at a uniform temperature. To express the solution in terms of an integral kernel, note that by the Fourier inversion formula

$$\begin{aligned} f(\theta,t) &= \sum_{j=0}^{\infty}\langle f, e^{ij\theta}\rangle\, e^{-j^2 t}\, e^{ij\theta} \\ &= \frac{1}{2\pi}\int_{S^1}\sum_{j=0}^{\infty} e^{-j^2 t} e^{ij\theta}\, e^{-ij\phi}\, f_0(\phi)\, d\phi\,, \end{aligned}$$

thus expressing the solution as $f(\theta,t) = \int_{S^1} K_t(\theta,\phi)\, f_0(\phi)\, d\phi$ for the heat kernel

$$K_t(\phi,\theta) = \frac{1}{2\pi}\sum_{j=0}^{\infty} e^{-j^2 t}\cos\left(j(\theta-\phi)\right)\,.$$

This simple example shows several properties of the general solution of the heat equation on a (compact) Riemannian manifold; in particular, note that the eigenvalues of the kernel scale as $\lambda_j \sim e^{-j^{2/d}}$ where the dimension in this case is $d = 1$.

When $M = \mathbb{R}$, the heat kernel is the familiar Gaussian kernel, so that the solution to the heat equation is expressed as

$$f(x,t) = \frac{1}{\sqrt{4\pi t}}\int_{\mathbb{R}} e^{-\frac{(x-y)^2}{4t}}\, f_0(y)\, dy,$$

and it is seen that as $t \longrightarrow \infty$, the heat diffuses out "to infinity" so that $f(x,t) \longrightarrow 0$.

When M is compact, the Laplacian has discrete eigenvalues $0 = \mu_0 < \mu_1 \leq \mu_2 \cdots$ with corresponding eigenfunctions ϕ_i satisfying $\Delta\phi_i = -\mu_i\phi_i$. When the manifold has a boundary, appropriate boundary conditions must be imposed in order for Δ to be self-adjoint. Dirichlet boundary conditions set $\phi_i|_{\partial M} = 0$ and Neumann boundary conditions require $\left.\frac{\partial\phi_i}{\partial\nu}\right|_{\partial M} = 0$ where ν is the outer normal direction. The following theorem summarizes the basic properties for the kernel of the heat equation on M; we refer to Schoen and Yau (1994) for a proof.

Theorem 1 *Let M be a complete Riemannian manifold. Then there exists a function $K \in C^{\infty}(\mathbb{R}_+ \times M \times M)$, called the heat kernel, which satisfies the following properties for all $x,y \in M$, with $K_t(\cdot,\cdot) = K(t,\cdot,\cdot)$*

1. $K_t(x,y) = K_t(y,x)$
2. $\lim_{t\to 0} K_t(x,y) = \delta_x(y)$

3. $\left(\Delta - \frac{\partial}{\partial t}\right) K_t(x,y) = 0$

4. $K_t(x,y) = \int_M K_{t-s}(x,z) K_s(z,y)\, dz$ *for any* $s > 0$.

If in addition M is compact, then K_t *can be expressed in terms of the eigenvalues and eigenfunctions of the Laplacian as* $K_t(x,y) = \sum_{i=0}^{\infty} e^{-\mu_i t} \phi_i(x)\, \phi_i(y)$.

Properties 2 and 3 imply that $K_t(x,y)$ solves the heat equation in x, starting from a point heat source at y. It follows that $e^{t\Delta} f_0(x) = f(x,t) = \int_M K_t(x,y)\, f_0(y)\, dy$ solves the heat equation with initial conditions $f(x,0) = f_0(x)$, since

$$\begin{aligned} \frac{\partial f(x,t)}{\partial t} &= \int_M \frac{\partial K_t(x,y)}{\partial t}\, f_0(y)\, dy \\ &= \int_M \Delta K_t(x,y)\, f_0(y)\, dy \\ &= \Delta \int_M K_t(x,y)\, f_0(y)\, dy \\ &= \Delta f(x,t), \end{aligned}$$

and $\lim_{t\to 0} f(x,t) = \int_M \lim_{t\to 0} K_t(x,y)\, dy = f_0(x)$. Property 4 implies that $e^{t\Delta} e^{s\Delta} = e^{(t+s)\Delta}$, which has the physically intuitive interpretation that heat diffusion for time t is the composition of heat diffusion up to time s with heat diffusion for an additional time $t-s$. Since $e^{t\Delta}$ is a positive operator,

$$\begin{aligned} \int_M \int_M K_t(x,y) g(x)\, g(y)\, dx\, dy &= \int_M f(x)\, e^{t\Delta} g(x)\, dx \\ &= \langle g, e^{t\Delta} g \rangle \geq 0\,. \end{aligned}$$

Thus $K_t(x,y)$ is positive-definite. In the compact case, positive-definiteness follows directly from the expansion $K_t(x,y) = \sum_{i=0}^{\infty} e^{-\mu_i t} \phi_i(x)\, \phi_i(y)$, which shows that the eigenvalues of K_t as an integral operator are $e^{-\mu_i t}$. Together, these properties show that K_t defines a Mercer kernel.

The heat kernel $K_t(x,y)$ is a natural candidate for measuring the similarity between points between $x, y \in M$, while respecting the geometry encoded in the metric g. Furthermore it is, unlike the geodesic distance, a Mercer kernel—a fact that enables its use in statistical kernel machines. When this kernel is used for classification, as in our text classification experiments presented in Section 5, the discriminant function $y_t(x) = \sum_i \alpha_i y_i K_t(x, x_i)$ can be interpreted as the solution to the heat equation with initial temperature $y_0(x_i) = \alpha_i y_i$ on labeled data points x_i, and initial temperature $y_0(x) = 0$ elsewhere.

2.1.1 THE PARAMETRIX EXPANSION

For most geometries, there is no closed form solution for the heat kernel. However, the short time behavior of the solutions can be studied using an asymptotic expansion called the *parametrix expansion*. In fact, the existence of the heat kernel, as asserted in the above theorem, is most directly proven by first showing the existence of the parametrix expansion. In Section 5 we will employ the first-order parametrix expansion for text classification.

Recall that the heat kernel on flat n-dimensional Euclidean space is given by

$$K_t^{\text{Euclid}}(x,y) = (4\pi t)^{-\frac{n}{2}} \exp\left(-\frac{\|x-y\|^2}{4t}\right)$$

where $\|x-y\|^2 = \sum_{i=1}^n |x_i - y_i|^2$ is the squared Euclidean distance between x and y. The parametrix expansion approximates the heat kernel locally as a correction to this Euclidean heat kernel. To begin the definition of the parametrix, let

$$P_t^{(m)}(x,y) = (4\pi t)^{-\frac{n}{2}} \exp\left(-\frac{d^2(x,y)}{4t}\right) (\psi_0(x,y) + \psi_1(x,y)t + \cdots + \psi_m(x,y)t^m) \tag{1}$$

for currently unspecified functions $\psi_k(x,y)$, but where $d^2(x,y)$ now denotes the square of the geodesic distance on the manifold. The idea is to obtain ψ_k recursively by solving the heat equation approximately to order t^m, for small diffusion time t.

Let $r = d(x,y)$ denote the length of the radial geodesic from x to $y \in V_x$ in the normal coordinates defined by the exponential map. For any functions $f(r)$ and $h(r)$ of r, it can be shown that

$$\begin{aligned} \Delta f &= \frac{d^2 f}{dr^2} + \frac{d\left(\log \sqrt{\det g}\right)}{dr} \frac{df}{dr} \\ \Delta(fh) &= f\Delta h + h\Delta f + 2\frac{df}{dr}\frac{dh}{dr}. \end{aligned}$$

Starting from these basic relations, some calculus shows that

$$\left(\Delta - \frac{\partial}{\partial t}\right) P_t^{(m)} = (t^m \Delta\psi_m)(4\pi t)^{-\frac{n}{2}} \exp\left(-\frac{r^2}{4t}\right) \tag{2}$$

when ψ_k are defined recursively as

$$\psi_0 = \left(\frac{\sqrt{\det g}}{r^{n-1}}\right)^{-\frac{1}{2}} \tag{3}$$

$$\psi_k = r^{-k}\psi_0 \int_0^r \psi_0^{-1} (\Delta\phi_{k-1})\, s^{k-1} ds \quad \text{for } k > 0. \tag{4}$$

With this recursive definition of the functions ψ_k, the expansion (1), which is defined only locally, is then extended to all of $M \times M$ by smoothing with a "cut-off function" η, with the specification that $\eta : \mathbb{R}_+ \longrightarrow [0,1]$ is C^∞ and

$$\eta(r) = \begin{cases} 0 & r \geq 1 \\ 1 & r \leq c \end{cases}$$

for some constant $0 < c < 1$. Thus, the order-m parametrix is defined as

$$K_t^{(m)}(x,y) = \eta(d(x,y))\, P_t^{(m)}(x,y).$$

As suggested by equation (2), $K_t^{(m)}$ is an approximate solution to the heat equation, and satisfies $K_t(x,y) = K_t^{(m)}(x,y) + O(t^m)$ for x and y sufficiently close; in particular, the parametrix is not unique. For further details we refer to (Schoen and Yau, 1994; Rosenberg, 1997).

While the parametrix $K_t^{(m)}$ is not in general positive-definite, and therefore does not define a Mercer kernel, it is positive-definite for t sufficiently small. In particular, define the function $f(t) = \min\operatorname{spec}(K_t^m)$, where $\min\operatorname{spec}$ denotes the smallest eigenvalue. Then f is a continuous function with $f(0) = 1$ since $K_0^{(m)} = I$. Thus, there is some time interval $[0,\varepsilon)$ for which $K_t^{(m)}$ is positive-definite in case $t \in [0,\varepsilon)$. This fact will be used when we employ the parametrix approximation to the heat kernel for statistical learning.

3. Diffusion Kernels on Statistical Manifolds

We now proceed to the main contribution of the paper, which is the application of the heat kernel constructions reviewed in the previous section to the geometry of statistical families, in order to obtain kernels for statistical learning.

Under some mild regularity conditions, general parametric statistical families come equipped with a canonical geometry based on the Fisher information metric. This geometry has long been recognized (Rao, 1945), and there is a rich line of research in statistics, with threads in machine learning, that has sought to exploit this geometry in statistical analysis; see Kass (1989) for a survey and discussion, or the monographs by Kass and Vos (1997) and Amari and Nagaoka (2000) for more extensive treatments. The basic properties of the Fisher information metric are reviewed in Appendix B.

We remark that in spite of the fundamental nature of the geometric perspective in statistics, many researchers have concluded that while it occasionally provides an interesting alternative interpretation, it has not contributed new results or methods that cannot be obtained through more conventional analysis. However in the present work, the kernel methods we propose can, arguably, be motivated and derived only through the geometry of statistical manifolds.[1]

The following two basic examples illustrate the geometry of the Fisher information metric and the associated diffusion kernel it induces on a statistical manifold. The spherical normal family corresponds to a manifold of constant negative curvature, and the multinomial corresponds to a manifold of constant positive curvature. The multinomial will be the most important example that we develop, and we report extensive experiments with the resulting kernels in Section 5.

3.1 Diffusion Kernels for Gaussian Geometry

Consider the statistical family given by $\mathfrak{F} = \{p(\cdot\,|\,\theta)\}_{\theta\in\Theta}$ where $\theta = (\mu,\sigma)$ and $p(\cdot\,|\,(\mu,\sigma)) = \mathcal{N}(\mu,\sigma^2 I_{n-1})$, the Gaussian having mean $\mu \in \mathbb{R}^{n-1}$ and variance $\sigma^2 I_{n-1}$, with $\sigma > 0$. Thus, $\Theta = \mathbb{R}^{n-1}\times\mathbb{R}_+$. A derivation of the Fisher information metric for this family is given in Appendix B.1, where it is shown that under coordinates defined by $\theta'_i = \mu_i$ for $1 \le i \le n-1$ and $\theta'_n = \sqrt{2(n-1)}\,\sigma$, the Fisher information matrix is given by

$$g_{ij}(\theta') = \frac{1}{\sigma^2}\,\delta_{ij}\,.$$

Thus, the Fisher information metric gives $\Theta = \mathbb{R}^{n-1}\times\mathbb{R}_+$ the structure of the upper half plane in hyperbolic space. The distance minimizing or geodesic curves in hyperbolic space are straight lines or circles orthogonal to the mean subspace.

In particular, the univariate normal density has hyperbolic geometry. As a generalization in this 2-dimensional case, any location-scale family of densities is seen to have hyperbolic geometry (Kass and Vos, 1997). Such families have densities of the form

$$p(x\,|\,(\mu,\sigma)) = \frac{1}{\sigma} f\left(\frac{x-\mu}{\sigma}\right)$$

where $(\mu,\sigma) \in \mathbb{R}\times\mathbb{R}_+$ and $f : \mathbb{R}\to\mathbb{R}$.

1. By a *statistical manifold* we mean simply a manifold of densities together with the metric induced by the Fisher information matrix, rather than the more general notion of a Riemannian manifold together with a (possibly non-metric) connection, as defined by Lauritzen (1987).

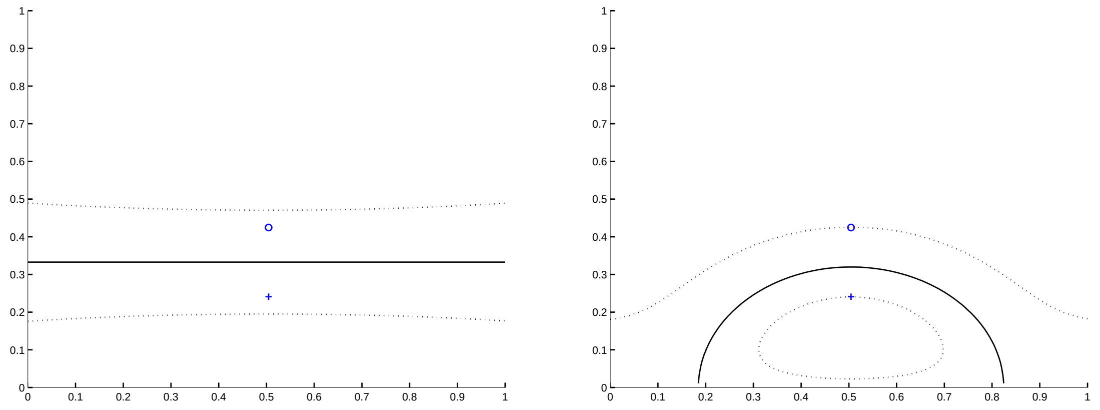


Figure 1: Example decision boundaries for a kernel-based classifier using information diffusion kernels for spherical normal geometry with $d = 2$ (right), which has constant negative curvature, compared with the standard Gaussian kernel for flat Euclidean space (left). Two data points are used, simply to contrast the underlying geometries. The curved decision boundary for the diffusion kernel can be interpreted statistically by noting that as the variance decreases the mean is known with increasing certainty.

The heat kernel on the hyperbolic space $\mathbb{H}^n$ has the following explicit form (Grigor'yan and Noguchi, 1998). For odd $n = 2m+1$ it is given by

$$K_t(x,x') = \frac{(-1)^m}{2^m \pi^m} \frac{1}{\sqrt{4\pi t}} \left(\frac{1}{\sinh r} \frac{\partial}{\partial r} \right)^m \exp\left(-m^2 t - \frac{r^2}{4t} \right), \tag{5}$$

and for even $n = 2m+2$ it is given by

$$K_t(x,x') = \frac{(-1)^m}{2^m \pi^m} \frac{\sqrt{2}}{\sqrt{4\pi t}^3} \left(\frac{1}{\sinh r} \frac{\partial}{\partial r} \right)^m \int_r^\infty \frac{s \exp\left(-\frac{(2m+1)^2 t}{4} - \frac{s^2}{4t} \right)}{\sqrt{\cosh s - \cosh r}} \, \mathrm{d}s, \tag{6}$$

where $r = d(x,x')$ is the geodesic distance between the two points in $\mathbb{H}^n$. If only the mean $\theta = \mu$ is unspecified, then the associated kernel is the standard Gaussian RBF kernel.

A possible use for this kernel in statistical learning is where data points are naturally represented as sets. That is, suppose that each data point is of the form $x = \{x_1, x_2, \dots x_m\}$ where $x_i \in \mathbb{R}^{n-1}$. Then the data can be represented according to the mapping which sends each group of points to the corresponding Gaussian under the MLE: $x \mapsto (\widehat{\mu}(x), \widehat{\sigma}(x))$ where $\widehat{\mu}(x) = \frac{1}{m}\sum_i x_i$ and $\widehat{\sigma}(x)^2 = \frac{1}{m}\sum_i (x_i - \widehat{\mu}(x))^2$.

In Figure 3.1 the diffusion kernel for hyperbolic space $\mathbb{H}^2$ is compared with the Euclidean space Gaussian kernel. The curved decision boundary for the diffusion kernel makes intuitive sense, since as the variance decreases the mean is known with increasing certainty.

Note that we can, in fact, consider M as a manifold with boundary by allowing $\sigma \geq 0$ to be non-negative rather than strictly positive $\sigma > 0$. In this case, the densities on the boundary become singular, as point masses at the mean; the boundary is simply given by $\partial M \cong \mathbb{R}^{n-1}$, which is a manifold without boundary, as required.

3.2 Diffusion Kernels for Multinomial Geometry

We now consider the statistical family of the multinomial over $n+1$ outcomes, given by $\mathfrak{F} = \{p(\cdot\,|\,\theta)\}_{\theta\in\Theta}$ where $\theta = (\theta_1, \theta_2, \ldots, \theta_n)$ with $\theta_i \in (0,1)$ and $\sum_{i=1}^{n} \theta_i < 1$. The parameter space Θ is the open n-simplex $\mathcal{P}_n$ defined in equation (9), a submanifold of $\mathbb{R}^{n+1}$.

To compute the metric, let $x = (x_1, x_2, \ldots, x_{n+1})$ denote one draw from the multinomial, so that $x_i \in \{0,1\}$ and $\sum_i x_i = 1$. The log-likelihood and its derivatives are then given by

$$\begin{aligned}
\log p(x\,|\,\theta) &= \sum_{i=1}^{n+1} x_i \log\theta_i \\
\frac{\partial \log p(x\,|\,\theta)}{\partial\theta_i} &= \frac{x_i}{\theta_i} \\
\frac{\partial^2 \log p(x\,|\,\theta)}{\partial\theta_i \partial\theta_j} &= -\frac{x_i}{\theta_i^2}\,\delta_{ij}\,.
\end{aligned}$$

Since $\mathcal{P}_n$ is an n-dimensional submanifold of $\mathbb{R}^{n+1}$, we can express $u, v \in T_\theta M$ as $(n+1)$-dimensional vectors in $T_\theta\mathbb{R}^{n+1} \cong \mathbb{R}^{n+1}$; thus, $u = \sum_{i=1}^{n+1} u_i e_i$, $v = \sum_{i=1}^{n+1} v_i e_i$. Note that due to the constraint $\sum_{i=1}^{n+1} \theta_i = 1$, the sum of the $n+1$ components of a tangent vector must be zero. A basis for $T_\theta M$ is

$$\left\{ e_1 = (1,0,\ldots,0,-1)^\top, e_2 = (0,1,0,\ldots,0,-1)^\top, \ldots, e_n = (0,0,\ldots,0,1,-1)^\top \right\}.$$

Using the definition of the Fisher information metric in equation (10) we then compute

$$\begin{aligned}
\langle u, v\rangle_\theta &= -\sum_{i=1}^{n+1}\sum_{j=1}^{n+1} u_i v_j E_\theta \left[\frac{\partial^2 \log p(x\,|\,\theta)}{\partial\theta_i\partial\theta_j} \right] \\
&= -\sum_{i=1}^{n+1} u_i v_i E\left\{ -x_i/\theta_i^2 \right\} \\
&= \sum_{i=1}^{n+1} \frac{u_i v_i}{\theta_i}\,.
\end{aligned}$$

While geodesic distances are difficult to compute in general, in the case of the multinomial information geometry we can easily compute the geodesics by observing that the standard Euclidean metric on the surface of the positive n-sphere is the pull-back of the Fisher information metric on the simplex. This relationship is suggested by the form of the Fisher information given in equation (10).

To be concrete, the transformation $F(\theta_1, \ldots, \theta_{n+1}) = (2\sqrt{\theta_1}, \ldots, 2\sqrt{\theta_{n+1}})$ is a diffeomorphism of the n-simplex $\mathcal{P}_n$ onto the positive portion of the n-sphere of radius 2; denote this portion of the sphere as $\mathcal{S}_n^+ = \left\{\theta \in \mathbb{R}^{n+1} : \sum_{i=1}^{n+1} \theta_i^2 = 2,\ \theta_i > 0\right\}$. Given tangent vectors $u = \sum_{i=1}^{n+1} u_i e_i$, $v =$

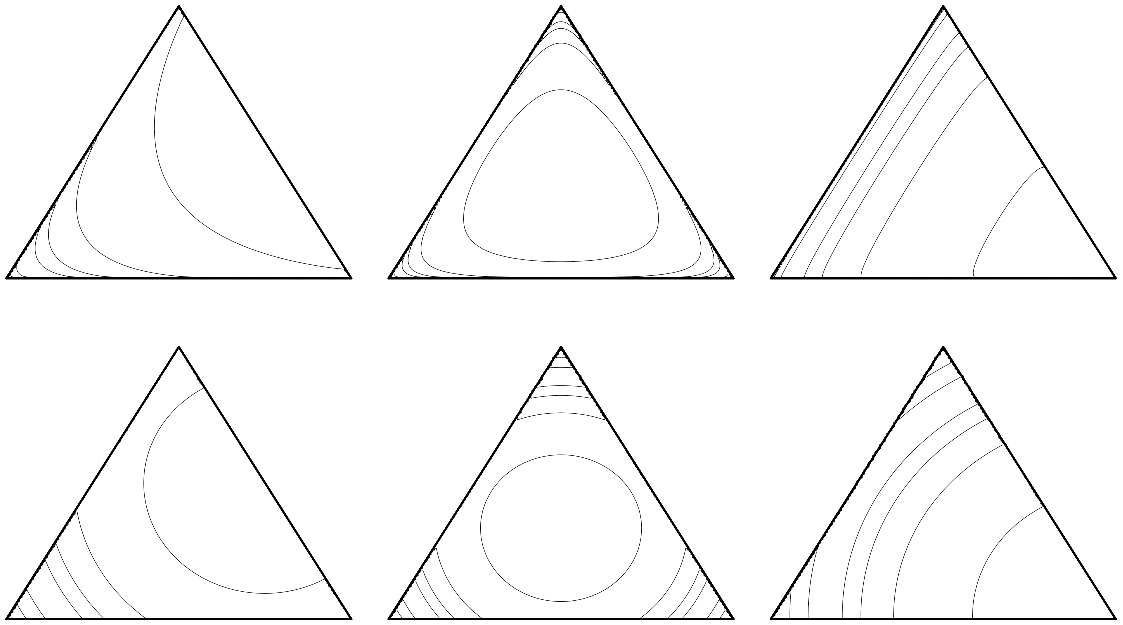


Figure 2: Equal distance contours on $\mathcal{P}_2$ from the upper right edge (left column), the center (center column), and lower right corner (right column). The distances are computed using the Fisher information metric g (top row) or the Euclidean metric (bottom row).

$\sum_{i=1}^{n+1} v_i e_i$, the pull-back of the Fisher information metric through F^{-1} is

$$\begin{aligned}
h_\theta(u,v) &= g_{\theta^2/4}\left(F_*^{-1}\sum_{k=1}^{n+1} u_k e_k, F_*^{-1}\sum_{l=1}^{n+1} v_l e_l\right)\\
&= \sum_{k=1}^{n+1}\sum_{l=1}^{n+1} u_k v_l\, g_{\theta^2/4}(F_*^{-1}e_k, F_*^{-1}e_l)\\
&= \sum_{k=1}^{n+1}\sum_{l=1}^{n+1} u_k v_l \sum_i \frac{4}{\theta_i^2}\,(F_*^{-1}e_k)_i\,(F_*^{-1}e_l)_i\\
&= \sum_{k=1}^{n+1}\sum_{l=1}^{n+1} u_k v_l \sum_i \frac{4}{\theta_i^2}\,\frac{\theta_k\delta_{ki}}{2}\,\frac{\theta_l\delta_{li}}{2}\\
&= \sum_{i=1}^{n+1} u_i v_i\,.
\end{aligned}$$

Since the transformation $F : (\mathcal{P}_n, g) \to (\mathcal{S}_n^+, h)$ is an isometry, the geodesic distance $d(\theta,\theta')$ on $\mathcal{P}_n$ may be computed as the shortest curve on $\mathcal{S}_n^+$ connecting $F(\theta)$ and $F(\theta')$. These shortest curves are portions of great circles—the intersection of a two dimensional plane and $\mathcal{S}_n^+$—and their length is given by

$$d(\theta,\theta') = 2\arccos\left(\sum_{i=1}^{n+1}\sqrt{\theta_i\,\theta_i'}\right)\,. \tag{7}$$

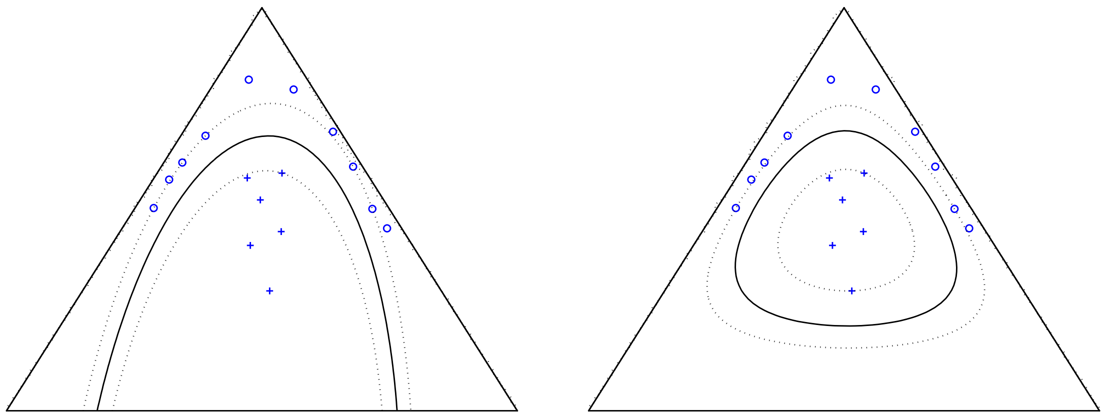


Figure 3: Example decision boundaries using support vector machines with information diffusion kernels for trinomial geometry on the 2-simplex (top right) compared with the standard Gaussian kernel (left).

In Appendix B we recall the connection between the Kullback-Leibler divergence and the information distance. In the case of the multinomial family, there is also a close relationship with the Hellinger distance. In particular, it can easily be shown that the Hellinger distance

$$d_H(\theta,\theta') = \sqrt{\sum_i \left(\sqrt{\theta_i} - \sqrt{\theta'_i}\right)^2}$$

is related to $d(\theta,\theta')$ by

$$d_H(\theta,\theta') = 2\sin\left(d(\theta,\theta')/4\right) .$$

Thus, as $\theta' \to \theta$, d_H agrees with $\frac{1}{2}d$ to second order:

$$d_H(\theta,\theta') = \frac{1}{2}d(\theta,\theta') + O(d^3(\theta,\theta'))$$

The Fisher information metric places greater emphasis on points near the boundary, which is expected to be important for text problems, which typically have sparse statistics. Figure 2 shows equal distance contours on $\mathcal{P}_2$ using the Fisher information and the Euclidean metrics.

While the spherical geometry has been derived as the information geometry for a finite multinomial, the same geometry can be used non-parametrically for an arbitrary subset of probability measures, leading to spherical geometry in a Hilbert space (Dawid, 1977).

3.2.1 THE MULTINOMIAL DIFFUSION KERNEL

Unlike the explicit expression for the Gaussian geometry discussed above, there is not an explicit form for the heat kernel on the sphere, nor on the positive orthant of the sphere. We will therefore resort to the parametrix expansion to derive an approximate heat kernel for the multinomial.

Recall from Section 2.1.1 that the parametrix is obtained according to the local expansion given in equation (1), and then extending this smoothly to zero outside a neighborhood of the diagonal, as defined by the exponential map. As we have just derived, this results in the following parametrix for the multinomial family:

$$P_t^{(m)}(\theta,\theta') = (4\pi t)^{-\frac{n}{2}} \exp\left(-\frac{\arccos^2(\sqrt{\theta}\cdot\sqrt{\theta'})}{t}\right)\left(\psi_0(\theta,\theta') + \cdots + \psi_m(\theta,\theta')t^m\right).$$

The first-order expansion is thus obtained as

$$K_t^{(0)}(\theta,\theta') = \eta(d(\theta,\theta'))\,P_t^{(0)}(\theta,\theta').$$

Now, for the n-sphere it can be shown that the function ψ_0 of (3), which is the leading order correction of the Gaussian kernel under the Fisher information metric, is given by

$$\begin{aligned}\psi_0(r) &= \left(\frac{\sqrt{\det g}}{r^{n-1}}\right)^{-\frac{1}{2}}\\ &= \left(\frac{\sin r}{r}\right)^{-\frac{(n-1)}{2}}\\ &= 1 + \frac{(n-1)}{12}r^2 + \frac{(n-1)(5n-1)}{1440}r^4 + O(r^6)\end{aligned}$$

(Berger et al., 1971). Thus, the leading order parametrix for the multinomial diffusion kernel is

$$P_t^{(0)}(\theta,\theta') = (4\pi t)^{-\frac{n}{2}} \exp\left(-\frac{1}{4t}d^2(\theta,\theta')\right)\left(\frac{\sin d(\theta,\theta')}{d(\theta,\theta')}\right)^{-\frac{(n-1)}{2}}.$$

In our experiments we approximate this kernel further as

$$P_t^{(0)}(\theta,\theta') = (4\pi t)^{-\frac{n}{2}} \exp\left(-\frac{1}{t}\arccos^2(\sqrt{\theta}\cdot\sqrt{\theta'})\right)$$

by appealing to the asymptotic expansion in (8) and the explicit form of the distance given in (7); note that $(\sin r/r)^{-n}$ blows up for large r. In Figure 3 the kernel (3.2.1) is compared with the standard Euclidean space Gaussian kernel for the case of the trinomial model, $d=2$, using an SVM classifier.

3.2.2 ROUNDING THE SIMPLEX

The case of multinomial geometry poses some technical complications for the analysis of diffusion kernels, due to the fact that the open simplex is not complete, and moreover, its closure is not a differentiable manifold with boundary. Thus, it is not technically possible to apply several results from differential geometry, such as bounds on the spectrum of the Laplacian, as adopted in Section 4. We now briefly describe a technical "patch" that allows us to derive all of the needed analytical results, without sacrificing in practice any of the methodology that has been derived so far.

Let $\Delta_n = \overline{\mathcal{P}}_n$ denote the closure of the open simplex; thus Δ_n is the usual probability simplex which allows zero probability for some items. However, it does not form a compact manifold with boundary since the boundary has edges and corners. In other words, local charts $\varphi : U \to \mathbb{R}^{n+}$ cannot be defined to be differentiable. To adjust for this, the idea is to "round the edges" of Δ_n to

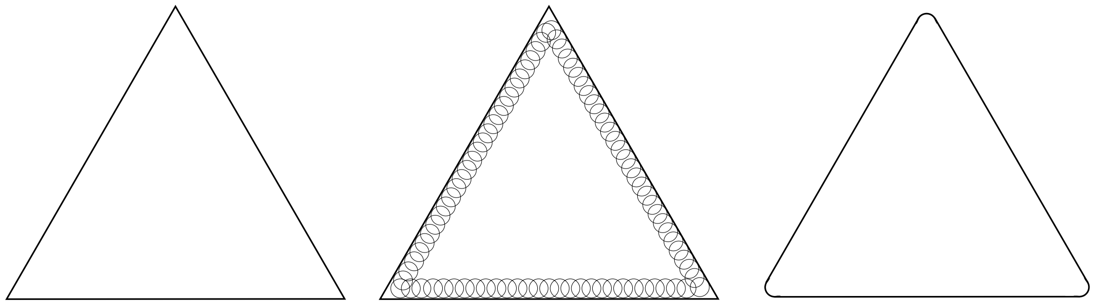


Figure 4: Rounding the simplex. Since the closed simplex is not a manifold with boundary, we carry out a "rounding" procedure to remove edges and corners. The δ-rounded simplex is the closure of the union of all δ-balls lying within the open simplex.

obtain a subset that forms a compact manifold with boundary, and that closely approximates the original simplex.

For $\delta > 0$, let $B_\delta(x) = \{y \mid \|x-y\| < \delta\}$ denote the open Euclidean ball of radius δ centered at x. Denote by $C_\delta(\mathcal{P}_n)$ the δ-*ball centers* of $\mathcal{P}_n$, the points of the simplex whose δ-balls lie completely within the simplex:

$$C_\delta(\mathcal{P}_n) = \{x \in \mathcal{P}_n \,:\, B_\delta(x) \subset \mathcal{P}_n\}\,.$$

Finally, let $\mathcal{P}_n^\delta$ denote the δ-*interior* of $\mathcal{P}_n$, which we define as the union of all δ-balls contained in $\mathcal{P}_n$:

$$\mathcal{P}_n^\delta = \bigcup_{x \in C_\delta(\mathcal{P}_n)} B_\delta(x)\,.$$

The δ-*rounded simplex* Δ_n^δ is then defined as the closure $\Delta_n^\delta = \overline{\mathcal{P}_n^\delta}$.

The rounding procedure that yields Δ_2^δ is suggested by Figure 4. Note that in general the δ-rounded simplex Δ_n^δ will contain points with a single, but not more than one component having zero probability. The set Δ_n^δ forms a compact manifold with boundary, and its image under the isometry $F : (\mathcal{P}_n, g) \to (\mathcal{S}_n^+, h)$ is a compact submanifold with boundary of the n-sphere.

Whenever appealing to results for compact manifolds with boundary in the following, it will be tacitly assumed that the above rounding procedure has been carried out in the case of the multinomial. From a theoretical perspective this enables the use of bounds on spectra of Laplacians for manifolds of non-negative curvature. From a practical viewpoint it requires only smoothing the probabilities to remove zeros.

4. Spectral Bounds on Covering Numbers and Rademacher Averages

We now turn to establishing bounds on the generalization performance of kernel machines that use information diffusion kernels. We first adopt the approach of Guo et al. (2002), estimating covering numbers by making use of bounds on the spectrum of the Laplacian on a Riemannian manifold, rather than on VC dimension techniques; these bounds in turn yield bounds on the expected risk of the learning algorithms. Our calculations give an indication of how the underlying geometry influences the entropy numbers, which are inverse to the covering numbers. We then show how bounds on Rademacher averages may be obtained by plugging in the spectral bounds from differential geometry. The primary conclusion that is drawn from these analyses is that from the point of view of generalization error bounds, diffusion kernels behave essentially the same as the standard Gaussian kernel.

4.1 Covering Numbers

We begin by recalling the main result of Guo et al. (2002), modifying their notation slightly to conform with ours. Let $M \subset \mathbb{R}^d$ be a compact subset of d-dimensional Euclidean space, and suppose that $K : M \times M \longrightarrow \mathbb{R}$ is a Mercer kernel. Denote by $\lambda_1 \geq \lambda_2 \geq \cdots \geq 0$ the eigenvalues of K, that is, of the mapping $f \mapsto \int_M K(\cdot,y)\, f(y)\, dy$, and let $\psi_j(\cdot)$ denote the corresponding eigenfunctions. We assume that $C_K \overset{\text{def}}{=} \sup_j \left\|\psi_j\right\|_\infty < \infty$.

Given m points $x_i \in M$, the kernel hypothesis class for $x = \{x_i\}$ with weight vector bounded by R is defined as the collection of functions on x given by

$$\mathcal{F}_R(x) = \{ f : f(x_i) = \langle w, \Phi(x_i)\rangle \text{ for some } \|w\| \leq R \} ,$$

where $\Phi(\cdot)$ is the mapping from M to feature space defined by the Mercer kernel, and $\langle \cdot,\cdot \rangle$ and $\|\cdot\|$ denote the corresponding Hilbert space inner product and norm. It is of interest to obtain uniform bounds on the covering numbers $\mathcal{N}(\varepsilon, \mathcal{F}_R(x))$, defined as the size of the smallest ε-cover of $\mathcal{F}_R(x)$ in the metric induced by the norm $\|f\|_{\infty,x} = \max_{i=1,\ldots,m} |f(x_i)|$.

Theorem 2 (Guo et al., 2002) *Given an integer $n \in \mathbb{N}$, let j_n^* denote the smallest integer j for which*

$$\lambda_{j+1} < \left(\frac{\lambda_1 \cdots \lambda_j}{n^2}\right)^{\frac{1}{j}}$$

and define

$$\varepsilon_n^* = 6 C_K R \sqrt{\, j_n^* \left(\frac{\lambda_1 \cdots \lambda_{j_n^*}}{n^2}\right)^{\frac{1}{j_n^*}} + \sum_{i=j_n^*}^{\infty} \lambda_i} \,.$$

Then $\sup_{\{x_i\}\in M^m} \mathcal{N}(\varepsilon_n^*, \mathcal{F}_R(x)) \leq n$.

To apply this result, we will obtain bounds on the indices j_n^* using spectral theory in Riemannian geometry.

Theorem 3 (Li and Yau, 1980) *Let M be a compact Riemannian manifold of dimension d with non-negative Ricci curvature, and let $0 < \mu_1 \leq \mu_2 \leq \cdots$ denote the eigenvalues of the Laplacian with Dirichlet boundary conditions. Then*

$$c_1(d) \left(\frac{j}{V}\right)^{\frac{2}{d}} \;\leq\; \mu_j \;\leq\; c_2(d) \left(\frac{j+1}{V}\right)^{\frac{2}{d}}$$

where V is the volume of M and c_1 and c_2 are constants depending only on the dimension.

Note that the manifold of the multinomial model (after δ-rounding) satisfies the conditions of this theorem. Using these results we can establish the following bounds on covering numbers for information diffusion kernels. We assume Dirichlet boundary conditions; a similar result can be proven for Neumann boundary conditions. We include the constant $V = \mathrm{vol}(M)$ and diffusion coefficient t in order to indicate how the bounds depend on the geometry.

Theorem 4 *Let M be a compact Riemannian manifold, with volume V, satisfying the conditions of Theorem 3. Then the covering numbers for the Dirichlet heat kernel K_t on M satisfy*

$$\log \mathcal{N}(\varepsilon, \mathcal{F}_R(x)) = O\left(\left(\frac{V}{t^{\frac{d}{2}}}\right) \log^{\frac{d+2}{2}}\left(\frac{1}{\varepsilon}\right)\right). \tag{8}$$

Proof By the lower bound in Theorem 3, the Dirichlet eigenvalues of the heat kernel $K_t(x,y)$, which are given by $\lambda_j = e^{-t\mu_j}$, satisfy $\log \lambda_j \leq -tc_1(d)\left(\frac{j}{V}\right)^{\frac{2}{d}}$. Thus,

$$-\frac{1}{j}\log\left(\frac{\lambda_1 \cdots \lambda_j}{n^2}\right) \;\geq\; \frac{tc_1}{j}\sum_{i=1}^{j}\left(\frac{i}{V}\right)^{\frac{2}{d}} + \frac{2}{j}\log n \;\geq\; tc_1 \frac{d}{d+2}\left(\frac{j}{V}\right)^{\frac{2}{d}} + \frac{2}{j}\log n,$$

where the second inequality comes from $\sum_{i=1}^{j} i^p \geq \int_0^j x^p\, dx = \frac{j^{p+1}}{p+1}$. Now using the upper bound of Theorem 3, the inequality $j_n^* \leq j$ will hold if

$$tc_2\left(\frac{j+2}{V}\right)^{\frac{2}{d}} \;\geq\; -\log \lambda_{j+1} \;\geq\; tc_1\frac{d}{d+2}\left(\frac{j}{V}\right)^{\frac{2}{d}} + \frac{2}{j}\log n$$

or equivalently

$$\frac{tc_2}{V^{\frac{2}{d}}}\left(j(j+2)^{\frac{2}{d}} - \frac{c_1}{c_2}\frac{d}{d+2} j^{\frac{d+2}{d}}\right) \;\geq\; 2\log n.$$

The above inequality will hold in case

$$j \geq \left\lceil \left(\frac{2V^{\frac{2}{d}}}{t(c_2 - c_1\frac{d}{d+2})}\log n\right)^{\frac{d}{d+2}} \right\rceil \;\geq\; \left\lceil \left(\frac{V^{\frac{2}{d}}(d+2)}{tc_1}\log n\right)^{\frac{d}{d+2}} \right\rceil$$

since we may assume that $c_2 \geq c_1$; thus, $j_n^* \leq \left\lceil \overline{c}_1\left(\frac{V^{\frac{2}{d}}}{t}\log n\right)^{\frac{d}{d+2}} \right\rceil$ for a new constant $\overline{c}_1(d)$. Plugging this bound on j_n^* into the expression for ε_n^* in Theorem 2 and using

$$\sum_{i=j_n^*}^{\infty} e^{-i^{\frac{2}{d}}} = O\left(e^{-j_n^{*\frac{2}{d}}}\right),$$

we have after some algebra that

$$\log\left(\frac{1}{\varepsilon_n}\right) = \Omega\left(\left(\frac{t}{V^{\frac{2}{d}}}\right)^{\frac{d}{d+2}} \log^{\frac{2}{d+2}} n\right).$$

Inverting the above expression in $\log n$ gives equation (8). ■

We note that Theorem 4 of Guo et al. (2002) can be used to show that this bound does not, in fact, depend on m and x. Thus, for fixed t the covering numbers scale as $\log \mathcal{N}(\varepsilon, \mathcal{F}) = O\left(\log^{\frac{d+2}{2}}\left(\frac{1}{\varepsilon}\right)\right)$, and for fixed ε they scale as $\log \mathcal{N}(\varepsilon, \mathcal{F}) = O\left(t^{-\frac{d}{2}}\right)$ in the diffusion time t.

4.2 Rademacher Averages

We now describe a different family of generalization error bounds that can be derived using the machinery of Rademacher averages (Bartlett and Mendelson, 2002; Bartlett et al., 2004). The bounds fall out directly from the work of Mendelson (2003) on computing local averages for kernel-based function classes, after plugging in the eigenvalue bounds of Theorem 3.

As seen above, covering number bounds are related to a complexity term of the form

$$C(n) = \sqrt{j_n^* \left(\frac{\lambda_1 \cdots \lambda_{j_n^*}}{n^2}\right)^{\frac{1}{j_n^*}} + \sum_{i=j_n^*}^{\infty} \lambda_i}\,.$$

In the case of Rademacher complexities, risk bounds are instead controlled by a similar, yet simpler expression of the form

$$C(r) = \sqrt{j_r^* r + \sum_{i=j_r^*}^{\infty} \lambda_i}$$

where now j_r^* is the smallest integer j for which $\lambda_j < r$ (Mendelson, 2003), with r acting as a parameter bounding the error of the family of functions. To place this into some context, we quote the following results from Bartlett et al. (2004) and Mendelson (2003), which apply to a family of loss functions that includes the quadratic loss; we refer to Bartlett et al. (2004) for details on the technical conditions.

Let $(X_1,Y_1),(X_2,Y_2)\ldots,(X_n,Y_n)$ be an independent sample from an unknown distribution P on $\mathcal{X}\times\mathcal{Y}$, where $\mathcal{Y}\subset\mathbb{R}$. For a given loss function $\ell:\mathcal{Y}\times\mathcal{Y}\to\mathbb{R}$, and a family $\mathfrak{F}$ of measurable functions $f:\mathcal{X}\to\mathcal{Y}$, the objective is to minimize the expected loss $E[\ell(f(X),Y)]$. Let $E\ell_{f^*} = \inf_{f\in\mathfrak{F}} E\ell_f$, where $\ell_f(X,Y) = \ell(f(X),Y)$, and let $\hat{f}$ be any member of $\mathfrak{F}$ for which $E_n\ell_{\hat{f}} = \inf_{f\in\mathfrak{F}} E_n\ell_f$ where E_n denotes the empirical expectation. The *Rademacher average* of a family of functions $\mathfrak{G} = \{g:\mathcal{X}\to\mathbb{R}\}$ is defined as the expectation $ER_n\mathfrak{G} = E\left[\sup_{g\in\mathfrak{G}} R_n g\right]$ with $R_n g = \frac{1}{n}\sum_{i=1}^n \sigma_i g(X_i)$, where $\sigma_1,\ldots,\sigma_n$ are independent Rademacher random variables; that is, $p(\sigma_i = 1) = p(\sigma_i = -1) = \frac{1}{2}$.

Theorem 5 (Bartlett et al., 2004) *Let $\mathfrak{F}$ be a convex class of functions and define ψ by*

$$\psi(r) = aER_n\left\{f\in\mathfrak{F} : E(f-f^*)^2 \le r\right\} + \frac{bx}{n}$$

where a and b are constants that depend on the loss function ℓ. Then when $r \ge \psi(r)$,

$$E\left(\ell_{\hat{f}} - \ell_{f^*}\right) \le cr + \frac{dx}{n}$$

with probability at least $1-e^{-x}$, where c and d are additional constants.

Moreover, suppose that K is a Mercer kernel and $\mathfrak{F} = \left\{f\in\mathcal{H}_K : \|f\|_K \le 1\right\}$ is the unit ball in the reproducing kernel Hilbert space associated with K. Then

$$\psi(r) \le a\sqrt{\frac{2}{n}\sum_{j=1}^{\infty}\min\{r,\lambda_j\}} + \frac{bx}{n}\,.$$

Thus, to bound the excess risk for kernel machines in this framework it suffices to bound the term

$$\begin{aligned}\widetilde{\psi}(r) &= \sqrt{\sum_{j=1}^{\infty} \min\{r, \lambda_j\}} \\ &= \sqrt{j_r^* \, r + \sum_{i=j_r^*}^{\infty} \lambda_i}\end{aligned}$$

involving the spectrum. Given bounds on the eigenvalues, this is typically easy to do.

Theorem 6 *Let M be a compact Riemannian manifold, satisfying the conditions of Theorem 3. Then the Rademacher term $\widetilde{\psi}$ for the Dirichlet heat kernel K_t on M satisfies*

$$\widetilde{\psi}(r) \leq C \sqrt{\left(\frac{r}{t^{\frac{d}{2}}}\right) \log^{\frac{d}{2}}\left(\frac{1}{r}\right)},$$

for some constant C depending on the geometry of M.

Proof We have that

$$\begin{aligned}\widetilde{\psi}^2(r) &= \sum_{j=1}^{\infty} \min\{r, \lambda_j\} \\ &= j_r^* \, r + \sum_{j=j_r^*}^{\infty} e^{-t\mu_j} \\ &\leq j_r^* \, r + \sum_{j=j_r^*}^{\infty} e^{-tc_1 j^{\frac{2}{d}}} \\ &\leq j_r^* \, r + C e^{-tc_1 j_r^{*\frac{2}{d}}}\end{aligned}$$

for some constant C, where the first inequality follows from the lower bound in Theorem 3. But $j_r^* \leq j$ in case $\log \lambda_{j+1} > r$, or, again from Theorem 3, if

$$t\, c_2 (j+1)^{\frac{2}{d}} \leq -\log \lambda_j < \log \frac{1}{r}$$

or equivalently,

$$j_r^* \leq \frac{C'}{t^{\frac{d}{2}}} \log^{\frac{d}{2}}\left(\frac{1}{r}\right).$$

It follows that

$$\widetilde{\psi}^2(r) \leq C'' \left(\frac{r}{t^{\frac{d}{2}}}\right) \log^{\frac{d}{2}}\left(\frac{1}{r}\right)$$

for some new constant C''. ■

From this bound, it can be shown that, with high probability,

$$E\left(\ell_{\hat{f}} - \ell_{f^*}\right) = O\left(\frac{\log^{\frac{d}{2}} n}{n}\right),$$

which is the behavior expected of the Gaussian kernel for Euclidean space.

Thus, for both covering numbers and Rademacher averages, the resulting bounds are essentially the same as those that would be obtained for the Gaussian kernel on the flat d-dimensional torus, which is the standard way of "compactifying" Euclidean space to get a Laplacian having only discrete spectrum; the results of Guo et al. (2002) are formulated for the case $d = 1$, corresponding to the circle S^1. While the bounds for diffusion kernels were derived for the case of positive curvature, which apply to the special case of the multinomial, similar bounds for general manifolds with curvature bounded below by a negative constant should also be attainable.

5. Multinomial Diffusion Kernels and Text Classification

In this section we present the application of multinomial diffusion kernels to the problem of text classification. Text processing can be subject to some of the "dirty laundry" referred to in the introduction—documents are cast as Euclidean space vectors with special weighting schemes that have been empirically honed through applications in information retrieval, rather than inspired from first principles. However for text, the use of multinomial geometry is natural and well motivated; our experimental results offer some insight into how useful this geometry may be for classification.

5.1 Representing Documents

Assuming a vocabulary V of size $n+1$, a document may be represented as a sequence of words over the alphabet V. For many classification tasks it is not unreasonable to discard word order; indeed, humans can typically easily understand the high level topic of a document by inspecting its contents as a mixed up "bag of words." Let x_v denote the number of times term v appears in a document. Then $\{x_v\}_{v\in V}$ is the sample space of the multinomial distribution, with a document modeled as independent draws from a fixed model, which may change from document to document. It is natural to embed documents in the multinomial simplex using an embedding function $\widehat{\theta} : \mathbb{Z}_+^{n+1} \to \mathcal{P}_n$. We consider several embeddings $\widehat{\theta}$ that correspond to well known feature representations in text classification (Joachims, 2000). The *term frequency* (tf) representation uses normalized counts; the corresponding embedding is the maximum likelihood estimator for the multinomial distribution

$$\widehat{\theta}_{\text{tf}}(x) = \left(\frac{x_1}{\sum_i x_i}, \ldots, \frac{x_{n+1}}{\sum_i x_i}\right).$$

Another common representation is based on *term frequency, inverse document frequency* (tfidf). This representation uses the distribution of terms across documents to discount common terms; the *document frequency* df_v of term v is defined as the number of documents in which term v appears. Although many variants have been proposed, one of the simplest and most commonly used embeddings is

$$\widehat{\theta}_{\text{tfidf}}(x) = \left(\frac{x_1 \log(D/df_1)}{\sum_i x_i \log(D/df_i)}, \ldots, \frac{x_{n+1} \log(D/df_{n+1})}{\sum_i x_i \log(D/df_i)}\right)$$

where D is the number of documents in the corpus.

We note that in text classification applications the tf and tfidf representations are typically normalized to unit length in the L_2 norm rather than the L_1 norm, as above (Joachims, 2000). For example, the tf representation with L_2 normalization is given by

$$x \mapsto \left(\frac{x_1}{\sum_i x_i^2}, \ldots, \frac{x_{n+1}}{\sum_i x_i^2}\right)$$

and similarly for tfidf. When used in support vector machines with linear or Gaussian kernels, L_2-normalized tf and tfidf achieve higher accuracies than their L_1-normalized counterparts. However, for the diffusion kernels, L_1 normalization is necessary to obtain an embedding into the simplex. These different embeddings or feature representations are compared in the experimental results reported below.

To be clear, we list the three kernels we compare. First, the linear kernel is given by

$$K^{\text{Lin}}(\theta, \theta') = \theta \cdot \theta' = \sum_{v=1}^{n+1} \theta_v \theta'_v .$$

The Gaussian kernel is given by

$$K_\sigma^{\text{Gauss}}(\theta', \theta') \;=\; (2\pi\sigma)^{-\frac{n+1}{2}} \exp\left(-\frac{\|\theta - \theta'\|^2}{2\sigma^2}\right)$$

where $\|\theta - \theta'\|^2 = \sum_{v=1}^{n+1} |\theta_v - \theta'_v|^2$ is the squared Euclidean distance. The multinomial diffusion kernel is given by

$$K_t^{\text{Mult}}(\theta, \theta') = (4\pi t)^{-\frac{n}{2}} \exp\left(-\frac{1}{t} \arccos^2(\sqrt{\theta} \cdot \sqrt{\theta'})\right) ,$$

as derived in Section 3.

5.2 Experimental Results

In our experiments, the multinomial diffusion kernel using the tf embedding was compared to the linear or Gaussian (RBF) kernel with tf and tfidf embeddings using a support vector machine classifier on the WebKB and Reuters-21578 collections, which are standard data sets for text classification.

The WebKb dataset contains web pages found on the sites of four universities (Craven et al., 2000). The pages were classified according to whether they were student, faculty, course, project or staff pages; these categories contain 1641, 1124, 929, 504 and 137 instances, respectively. Since only the student, faculty, course and project classes contain more than 500 documents each, we restricted our attention to these classes. The Reuters-21578 dataset is a collection of newswire articles classified according to news topic (Lewis and Ringuette, 1994). Although there are more than 135 topics, most of the topics have fewer than 100 documents; for this reason, we restricted our attention to the following five most frequent classes: earn, acq, moneyFx, grain and crude, of sizes 3964, 2369, 717, 582 and 578 documents, respectively.

For both the WebKB and Reuters collections we created two types of binary classification tasks. In the first task we designate a specific class, label each document in the class as a “positive” example, and label each document on any of the other topics as a “negative” example. In the second task we designate a class as the positive class, and choose the negative class to be the most frequent remaining class (student for WebKB and earn for Reuters). In both cases, the size of the training set is varied while keeping the proportion of positive and negative documents constant in both the training and test set.

Figure 5 shows the test set error rate for the WebKB data, for a representative instance of the one-versus-all classification task; the designated class was course. The results for the other choices of positive class were qualitatively very similar; all of the results are summarized in Table 1. Similarly,

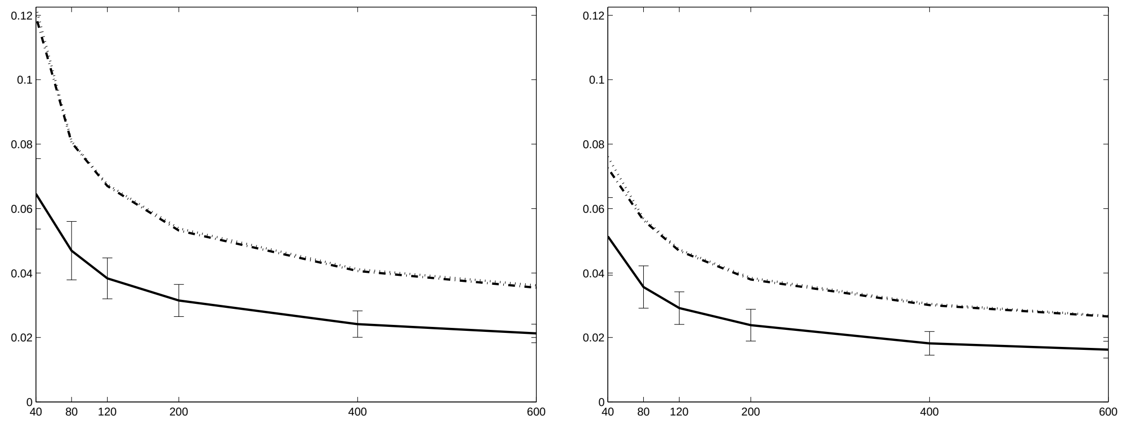


Figure 5: Experimental results on the WebKB corpus, using SVMs for linear (dotted) and Gaussian (dash-dotted) kernels, compared with the diffusion kernel for the multinomial (solid). Classification error for the task of labeling course vs. either faculty, project, or student is shown in these plots, as a function of training set size. The left plot uses tf representation and the right plot uses tfidf representation. The curves shown are the error rates averaged over 20-fold cross validation, with error bars representing one standard deviation. The results for the other "1 vs. all" labeling tasks are qualitatively similar, and are therefore not shown.

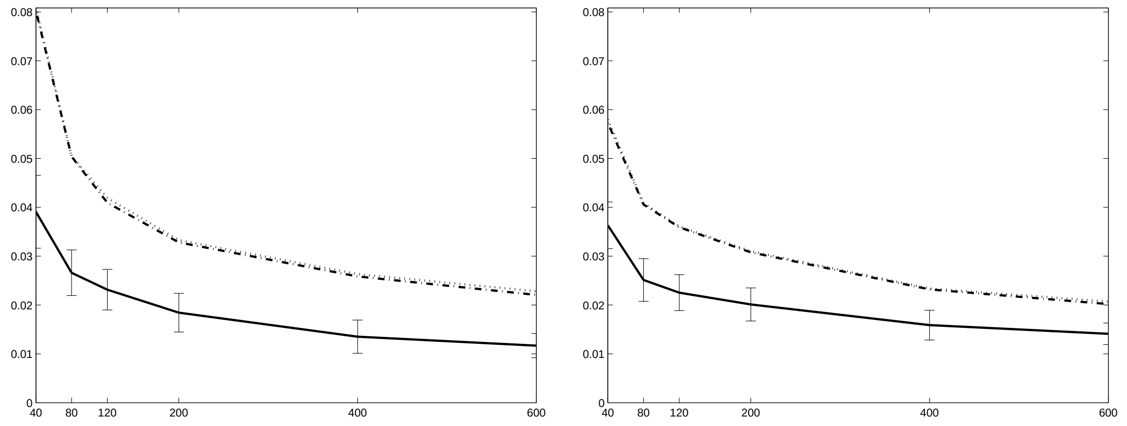


Figure 6: Results on the WebKB corpus, using SVMs for linear (dotted) and Gaussian (dash-dotted) kernels, compared with the diffusion kernel (solid). The course pages are labeled positive and the student pages are labeled negative; results for other label pairs are qualitatively similar. The left plot uses tf representation and the right plot uses tfidf representation.

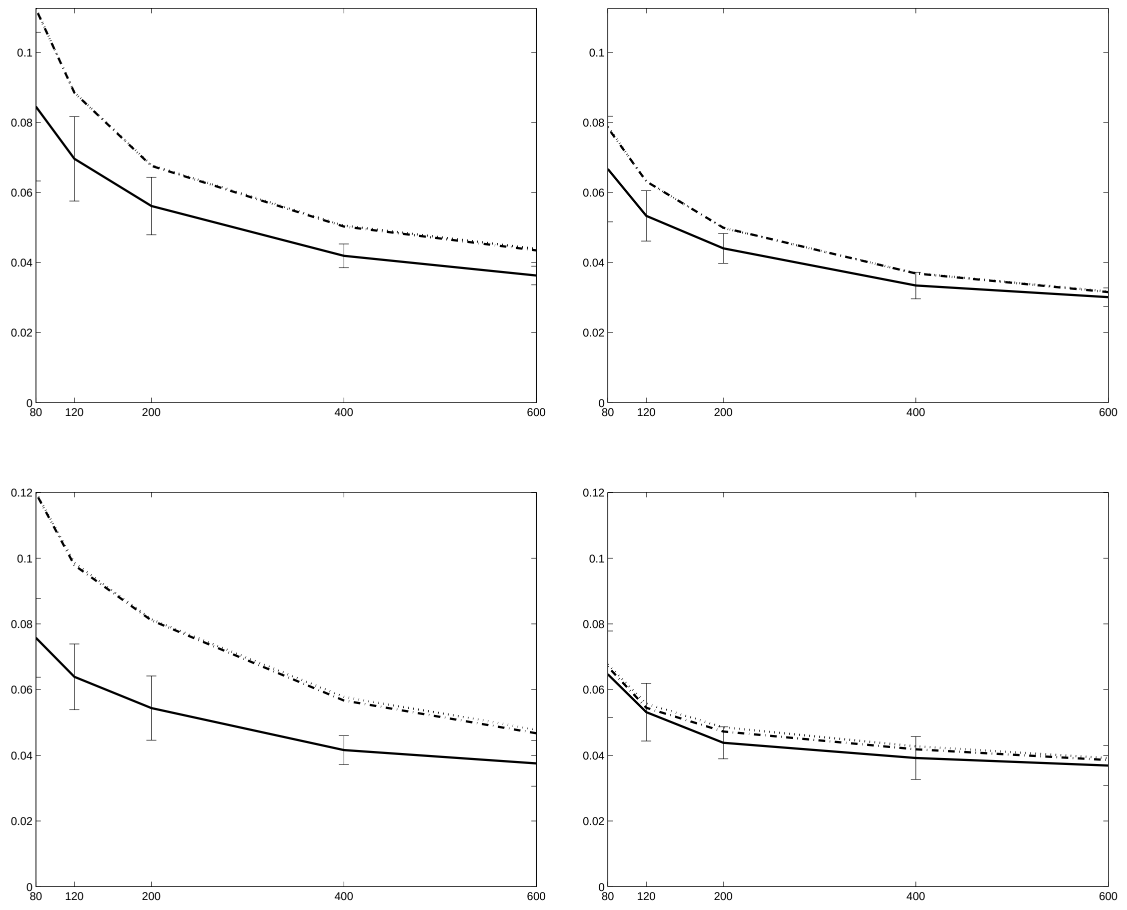


Figure 7: Experimental results on the Reuters corpus, using SVMs for linear (dotted) and Gaussian (dash-dotted) kernels, compared with the diffusion kernel (solid). The classes acq (top), and moneyFx (bottom) are shown; the other classes are qualitatively similar. The left column uses tf representation and the right column uses tfidf. The curves shown are the error rates averaged over 20-fold cross validation, with error bars representing one standard deviation.

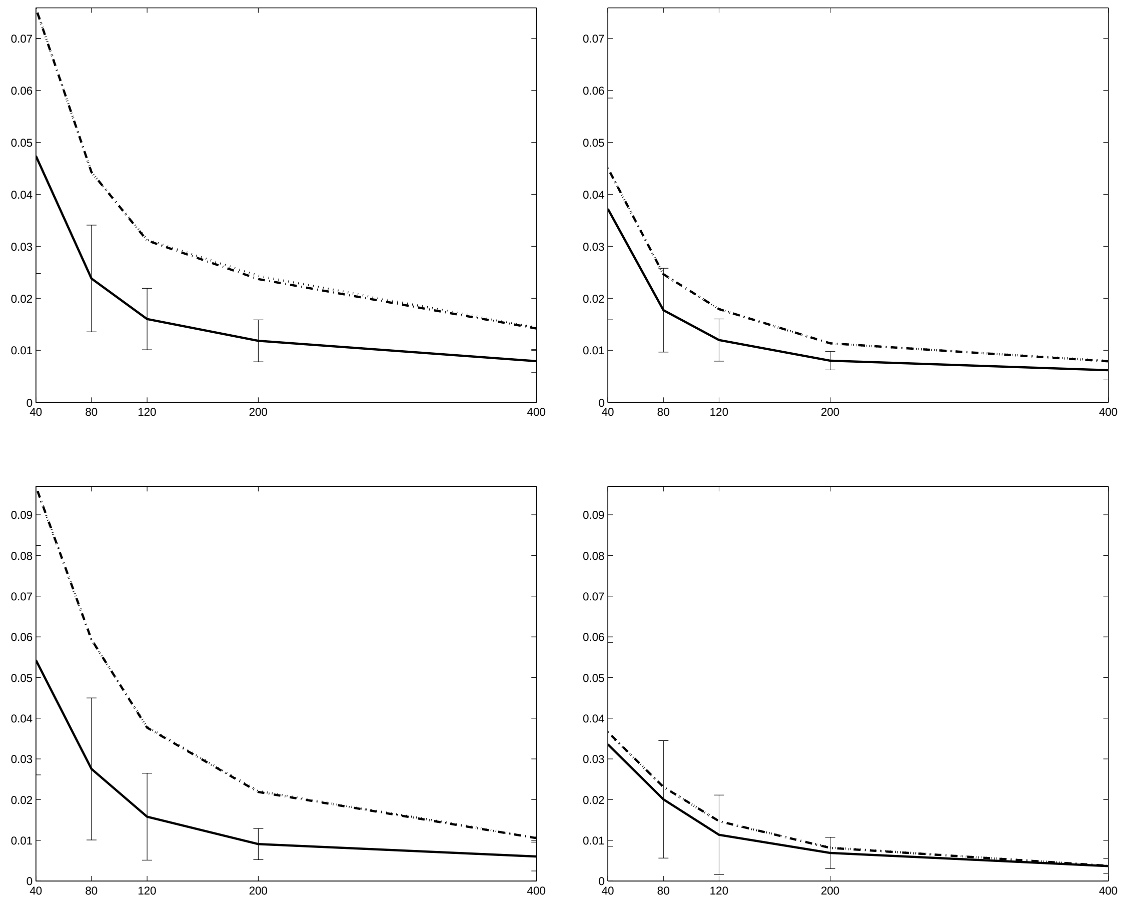


Figure 8: Experimental results on the Reuters corpus, using SVMs for linear (dotted) and Gaussian (dash-dotted) kernels, compared with the diffusion (solid). The classes moneyFx (top) and grain (bottom) are labeled as positive, and the class earn is labeled negative. The left column uses tf representation and the right column uses tfidf representation.

		tf Representation			tfidf Representation		
Task	L	Linear	Gaussian	Diffusion	Linear	Gaussian	Diffusion
course vs. all	40	0.1225	0.1196	**0.0646**	0.0761	0.0726	**0.0514**
	80	0.0809	0.0805	**0.0469**	0.0569	0.0564	**0.0357**
	120	0.0675	0.0670	**0.0383**	0.0473	0.0469	**0.0291**
	200	0.0539	0.0532	**0.0315**	0.0385	0.0380	**0.0238**
	400	0.0412	0.0406	**0.0241**	0.0304	0.0300	**0.0182**
	600	0.0362	0.0355	**0.0213**	0.0267	0.0265	**0.0162**
faculty vs. all	40	0.2336	0.2303	**0.1859**	0.2493	0.2469	**0.1947**
	80	0.1947	0.1928	**0.1558**	0.2048	0.2043	**0.1562**
	120	0.1836	0.1823	**0.1440**	0.1921	0.1913	**0.1420**
	200	0.1641	0.1634	**0.1258**	0.1748	0.1742	**0.1269**
	400	0.1438	0.1428	**0.1061**	0.1508	0.1503	**0.1054**
	600	0.1308	0.1297	**0.0931**	0.1372	0.1364	**0.0933**
project vs. all	40	0.1827	0.1793	**0.1306**	0.1831	0.1805	**0.1333**
	80	0.1426	0.1416	**0.0978**	0.1378	0.1367	**0.0982**
	120	0.1213	0.1209	**0.0834**	0.1169	0.1163	**0.0834**
	200	0.1053	0.1043	**0.0709**	0.1007	0.0999	**0.0706**
	400	0.0785	0.0766	**0.0537**	0.0802	0.0790	**0.0574**
	600	0.0702	0.0680	**0.0449**	0.0719	0.0708	**0.0504**
student vs. all	40	0.2417	0.2411	**0.1834**	0.2100	0.2086	**0.1740**
	80	0.1900	0.1899	**0.1454**	0.1681	0.1672	**0.1358**
	120	0.1696	0.1693	**0.1291**	0.1531	0.1523	**0.1204**
	200	0.1539	0.1539	**0.1134**	0.1349	0.1344	**0.1043**
	400	0.1310	0.1308	**0.0935**	0.1147	0.1144	**0.0874**
	600	0.1173	0.1169	**0.0818**	0.1063	0.1059	**0.0802**

Table 1: Experimental results on the WebKB corpus, using SVMs for linear, Gaussian, and multinomial diffusion kernels. The left columns use tf representation and the right columns use tfidf representation. The error rates shown are averages obtained using 20-fold cross validation. The best performance for each training set size L is shown in boldface. All differences are statistically significant according to the paired t test at the 0.05 level.

Figure 7 shows the test set error rates for two of the one-versus-all experiments on the Reuters data, where the designated classes were chosen to be acq and moneyFx. All of the results for Reuters one-versus-all tasks are shown in Table 3.

Figure 6 and Figure 8 show representative results for the second type of classification task, where the goal is to discriminate between two specific classes. In the case of the WebKB data the results are shown for course vs. student. In the case of the Reuters data the results are shown for moneyFx vs. earn and grain vs. earn. Again, the results for the other classes are qualitatively similar; the numerical results are summarized in Tables 2 and 4.

In these figures, the leftmost plots show the performance of tf features while the rightmost plots show the performance of tfidf features. As mentioned above, in the case of the diffusion kernel we

		tf Representation			tfidf Representation		
Task	L	Linear	Gaussian	Diffusion	Linear	Gaussian	Diffusion
course vs. student	40	0.0808	0.0802	**0.0391**	0.0580	0.0572	**0.0363**
	80	0.0505	0.0504	**0.0266**	0.0409	0.0406	**0.0251**
	120	0.0419	0.0409	**0.0231**	0.0361	0.0359	**0.0225**
	200	0.0333	0.0328	**0.0184**	0.0310	0.0308	**0.0201**
	400	0.0263	0.0259	**0.0135**	0.0234	0.0232	**0.0159**
	600	0.0228	0.0221	**0.0117**	0.0207	0.0202	**0.0141**
faculty vs. student	40	0.2106	0.2102	**0.1624**	0.2053	0.2026	**0.1663**
	80	0.1766	0.1764	**0.1357**	0.1729	0.1718	**0.1335**
	120	0.1624	0.1618	**0.1198**	0.1578	0.1573	**0.1187**
	200	0.1405	0.1405	**0.0992**	0.1420	0.1418	**0.1026**
	400	0.1160	0.1158	**0.0759**	0.1166	0.1165	**0.0781**
	600	0.1050	0.1046	**0.0656**	0.1050	0.1048	**0.0692**
project vs. student	40	0.1434	0.1430	**0.0908**	0.1304	0.1279	**0.0863**
	80	0.1139	0.1133	**0.0725**	0.0982	0.0970	**0.0634**
	120	0.0958	0.0957	**0.0613**	0.0870	0.0866	**0.0559**
	200	0.0781	0.0775	**0.0514**	0.0729	0.0722	**0.0472**
	400	0.0590	0.0579	**0.0405**	0.0629	0.0622	**0.0397**
	600	0.0515	0.0500	**0.0325**	0.0551	0.0539	**0.0358**

Table 2: Experimental results on the WebKB corpus, using SVMs for linear, Gaussian, and multinomial diffusion kernels. The left columns use tf representation and the right columns use tfidf representation. The error rates shown are averages obtained using 20-fold cross validation. The best performance for each training set size L is shown in boldface. All differences are statistically significant according to the paired t test at the 0.05 level.

use L_1 normalization to give a valid embedding into the probability simplex, while for the linear and Gaussian kernels we use L_2 normalization, which works better empirically than L_1 for these kernels. The curves show the test set error rates averaged over 20 iterations of cross validation as a function of the training set size. The error bars represent one standard deviation. For both the Gaussian and diffusion kernels, we test scale parameters ($\sqrt{2}\sigma$ for the Gaussian kernel and $2t^{1/2}$ for the diffusion kernel) in the set $\{0.5, 1, 2, 3, 4, 5, 7, 10\}$. The results reported are for the best parameter value in that range.

We also performed experiments with the popular Mod-Apte train and test split for the top 10 categories of the Reuters collection. For this split, the training set has about 7000 documents and is highly biased towards negative documents. We report in Table 5 the test set accuracies for the tf representation. For the tfidf representation, the difference between the different kernels is not statistically significant for this amount of training and test data. The provided train set is more than enough to achieve outstanding performance with all kernels used, and the absence of cross validation data makes the results too noisy for interpretation.

In Table 6 we report the F1 measure rather than accuracy, since this measure is commonly used in text classification. The last column of the table compares the presented results with the published

		tf Representation			tfidf Representation		
Task	L	Linear	Gaussian	Diffusion	Linear	Gaussian	Diffusion
earn vs. all	80	0.1107	0.1106	**0.0971**	0.0823	0.0827	**0.0762**
	120	0.0988	0.0990	**0.0853**	0.0710	0.0715	**0.0646**
	200	0.0808	0.0810	**0.0660**	0.0535	0.0538	**0.0480**
	400	0.0578	0.0578	**0.0456**	0.0404	0.0408	**0.0358**
	600	0.0465	0.0464	**0.0367**	0.0323	0.0325	**0.0290**
acq vs. all	80	0.1126	0.1125	**0.0846**	0.0788	0.0785	**0.0667**
	120	0.0886	0.0885	**0.0697**	0.0632	0.0632	**0.0534**
	200	0.0678	0.0676	**0.0562**	0.0499	0.0500	**0.0441**
	400	0.0506	0.0503	**0.0419**	0.0370	0.0369	**0.0335**
	600	0.0439	0.0435	**0.0363**	0.0318	0.0316	**0.0301**
moneyFx vs. all	80	0.1201	0.1198	**0.0758**	0.0676	0.0669	**0.0647***
	120	0.0986	0.0979	**0.0639**	0.0557	0.0545	**0.0531***
	200	0.0814	0.0811	**0.0544**	0.0485	0.0472	**0.0438**
	400	0.0578	0.0567	**0.0416**	0.0427	0.0418	**0.0392**
	600	0.0478	0.0467	**0.0375**	0.0391	0.0385	**0.0369***
grain vs. all	80	0.1443	0.1440	**0.0925**	0.0536	**0.0518***	0.0595
	120	0.1101	0.1097	**0.0717**	0.0476	**0.0467***	0.0494
	200	0.0793	0.0786	**0.0576**	0.0430	**0.0420***	0.0440
	400	0.0590	0.0573	**0.0450**	0.0349	**0.0340***	0.0365
	600	0.0517	0.0497	**0.0401**	0.0290	**0.0284***	0.0306
crude vs. all	80	0.1396	0.1396	**0.0865**	0.0502	**0.0485***	0.0524
	120	0.0961	0.0953	**0.0542**	0.0446	**0.0425***	0.0428
	200	0.0624	0.0613	**0.0414**	0.0388	0.0373	**0.0345***
	400	0.0409	0.0403	**0.0325**	0.0345	0.0337	**0.0297**
	600	0.0379	0.0362	**0.0299**	0.0292	0.0284	**0.0264***

Table 3: Experimental results on the Reuters corpus, using SVMs for linear, Gaussian, and multinomial diffusion kernels. The left columns use tf representation and the right columns use tfidf representation. The error rates shown are averages obtained using 20-fold cross validation. The best performance for each training set size L is shown in boldface. An asterisk (*) indicates that the difference is not statistically significant according to the paired t test at the 0.05 level.

results of Zhang and Oles (2001), with a $+$ indicating the diffusion kernel F1 measure is greater than the result published in Zhang and Oles (2001) for this task.

Our results are consistent with previous experiments in text classification using SVMs, which have observed that the linear and Gaussian kernels result in very similar performance (Joachims et al., 2001). However the multinomial diffusion kernel significantly outperforms the linear and Gaussian kernels for the tf representation, achieving significantly lower error rate than the other kernels. For the tfidf representation, the diffusion kernel consistently outperforms the other kernels for the WebKb data and usually outperforms the linear and Gaussian kernels for the Reuters data.

		tf Representation			tfidf Representation		
Task	L	Linear	Gaussian	Diffusion	Linear	Gaussian	Diffusion
acq vs. earn	40	0.1043	0.1043	**0.1021***	0.0829	0.0831	**0.0814***
	80	0.0902	0.0902	**0.0856***	0.0764	0.0767	**0.0730***
	120	0.0795	0.0796	**0.0715**	0.0626	0.0628	**0.0562**
	200	0.0599	0.0599	**0.0497**	0.0509	0.0511	**0.0431**
	400	0.0417	0.0417	**0.0340**	0.0336	0.0337	**0.0294**
moneyFx vs. earn	40	0.0759	0.0758	**0.0474**	0.0451	0.0451	**0.0372***
	80	0.0442	0.0443	**0.0238**	0.0246	0.0246	**0.0177**
	120	0.0313	0.0311	**0.0160**	0.0179	0.0179	**0.0120**
	200	0.0244	0.0237	**0.0118**	0.0113	0.0113	**0.0080**
	400	0.0144	0.0142	**0.0079**	0.0080	0.0079	**0.0062**
grain vs. earn	40	0.0969	0.0970	**0.0543**	0.0365	0.0366	**0.0336***
	80	0.0593	0.0594	**0.0275**	0.0231	0.0231	**0.0201***
	120	0.0379	0.0377	**0.0158**	0.0147	0.0147	**0.0114***
	200	0.0221	0.0219	**0.0091**	0.0082	0.0081	**0.0069***
	400	0.0107	0.0105	**0.0060**	0.0037	0.0037	**0.0037***
crude vs. earn	40	0.1108	0.1107	**0.0950**	**0.0583***	0.0586	0.0590
	80	0.0759	0.0757	**0.0552**	0.0376	0.0377	**0.0366***
	120	0.0608	0.0607	**0.0415**	0.0276	**0.0276***	0.0284
	200	0.0410	0.0411	**0.0267**	**0.0218***	0.0218	0.0225
	400	0.0261	0.0257	**0.0194**	0.0176	**0.0171***	0.0181

Table 4: Experimental results on the Reuters corpus, using SVMs for linear, Gaussian, and multinomial diffusion kernels. The left columns use tf representation and the right columns use tfidf representation. The error rates shown are averages obtained using 20-fold cross validation. The best performance for each training set size L is shown in boldface. An asterisk (*) indicates that the difference is not statistically significant according to the paired t test at the 0.05 level.

The Reuters data is a much larger collection than WebKB, and the document frequency statistics, which are the basis for the inverse document frequency weighting in the tfidf representation, are evidently much more effective on this collection. It is notable, however, that the multinomial information diffusion kernel achieves at least as high an accuracy without the use of any heuristic term weighting scheme. These results offer evidence that the use of multinomial geometry is both theoretically motivated and practically effective for document classification.

6. Discussion and Conclusion

This paper has introduced a family of kernels that is intimately based on the geometry of the Riemannian manifold associated with a statistical family through the Fisher information metric. The metric is canonical in the sense that it is uniquely determined by requirements of invariance (Čencov, 1982), and moreover, the choice of the heat kernel is natural because it effectively encodes a great

Category	Linear	RBF	Diffusion
earn	0.01159	0.01159	**0.01026**
acq	0.01854	0.01854	**0.01788**
money-fx	0.02418	0.02451	**0.02219**
grain	0.01391	0.01391	**0.01060**
crude	0.01755	0.01656	**0.01490**
trade	0.01722	**0.01656**	0.01689
interest	0.01854	0.01854	**0.01689**
ship	0.01324	0.01324	**0.01225**
wheat	0.00894	0.00794	**0.00629**
corn	0.00794	0.00794	**0.00563**

Table 5: Test set error rates for the Reuters top 10 classes using tf features. The train and test sets were created using the Mod-Apte split.

Category	Linear	RBF	Diffusion	$\pm$
earn	0.9781	0.9781	**0.9808**	$-$
acq	0.9626	0.9626	**0.9660**	$+$
money-fx	0.8254	0.8245	**0.8320**	$+$
grain	0.8836	0.8844	**0.9048**	$-$
crude	0.8615	0.8763	**0.8889**	$+$
trade	0.7706	0.7797	**0.8050**	$+$
interest	**0.8263**	**0.8263**	0.8221	$+$
ship	0.8306	0.8404	**0.8827**	$+$
wheat	0.8613	0.8613	**0.8844**	$-$
corn	0.8727	0.8727	**0.9310**	$+$

Table 6: F1 measure for the Reuters top 10 classes using tf features. The train and test sets were created using the Mod-Apte split. The last column compares the presented results with the published results of Zhang and Oles (2001), with a $+$ indicating the diffusion kernel F1 measure is greater than the result published in Zhang and Oles (2001) for this task.

deal of geometric information about the manifold. While the geometric perspective in statistics has most often led to reformulations of results that can be viewed more traditionally, the kernel methods developed here clearly depend crucially on the geometry of statistical families.

The main application of these ideas has been to develop the multinomial diffusion kernel. A related use of spherical geometry for the multinomial has been developed by Gous (1998). Our experimental results indicate that the resulting diffusion kernel is indeed effective for text classification using support vector machine classifiers, and can lead to significant improvements in accuracy compared with the use of linear or Gaussian kernels, which have been the standard for this application. The results of Section 5 are notable since accuracies better or comparable to those obtained using heuristic weighting schemes such as tfidf are achieved directly through the geometric approach. In part, this can be attributed to the role of the Fisher information metric; because of the square root in the embedding into the sphere, terms that are infrequent in a document are effectively up-weighted, and such terms are typically rare in the document collection overall. The primary degree of freedom in the use of information diffusion kernels lies in the specification of the mapping of data to model parameters. For the multinomial, we have used the maximum likelihood mapping. The use of other model families and mappings remains an interesting direction to explore.

While kernel methods generally are "model free," and do not make distributional assumptions about the data that the learning algorithm is applied to, statistical models offer many advantages, and thus it is attractive to explore methods that combine data models and purely discriminative methods. Our approach combines parametric statistical modeling with non-parametric discriminative learning, guided by geometric considerations. In these aspects it is related to the methods proposed by Jaakkola and Haussler (1998). However, the kernels proposed in the current paper differ significantly from the Fisher kernel of Jaakkola and Haussler (1998). In particular, the latter is based on the score $\nabla_\theta \log p(X \,|\, \hat{\theta})$ at a single point $\hat{\theta}$ in parameter space. In the case of an exponential family model it is given by a covariance $K_F(x,x') = \sum_i \left(x_i - E_{\hat{\theta}}[X_i]\right)\left(x'_i - E_{\hat{\theta}}[X_i]\right)$; this covariance is then heuristically exponentiated. In contrast, information diffusion kernels are based on the full geometry of the statistical family, and yet are also invariant under reparameterization of the family. In other conceptually related work, Belkin and Niyogi (2003) suggest measuring distances on the data graph to approximate the underlying manifold structure of the data. In this case the underlying geometry is inherited from the embedding Euclidean space rather than the Fisher geometry.

While information diffusion kernels are very general, they will be difficult to compute in many cases—explicit formulas such as equations (5–6) for hyperbolic space are rare. To approximate an information diffusion kernel it may be attractive to use the parametrices and geodesic distance between points, as we have done for the multinomial. In cases where the distance itself is difficult to compute exactly, a compromise may be to approximate the distance between nearby points in terms of the Kullback-Leibler divergence, using the relation with the Fisher information that is noted in Appendix B. In effect, this approximation is already incorporated into the kernels recently proposed by Moreno et al. (2004) for multimedia applications, which have the form $K(\theta,\theta') \propto \exp(-\alpha D(\theta,\theta')) \approx \exp(-2\alpha d^2(\theta,\theta'))$, and so can be viewed in terms of the leading order approximation to the heat kernel. The results of Moreno et al. (2004) are suggestive that diffusion kernels may be attractive not only for multinomial geometry, but also for much more complex statistical families.

Acknowledgments

We thank Rob Kass, Leonid Kontorovich and Jian Zhang for helpful discussions. This research was supported in part by NSF grants CCR-0122581 and IIS-0312814, and by ARDA contract MDA904-00-C-2106. A preliminary version of this work was published in Advances in Neural Information Processing Systems 15 (Lafferty and Lebanon, 2003).

Appendix A. The Geometric Laplacian

In this appendix we briefly review some of the elementary concepts from Riemannian geometry that are used in the construction of information diffusion kernels, since these concepts are not widely used in machine learning. We refer to Spivak (1979) for details and further background, or Milnor (1963) for an elegant and concise overview; however most introductory texts on differential geometry include this material.

A.1 Basic Definitions

An n-dimensional differentiable manifold M is a set of points that is locally equivalent to $\mathbb{R}^n$ by smooth transformations, supporting operations such as differentiation. Formally, a *differentiable manifold* is a set M together with a collection of *local charts* $\{(U_i, \varphi_i)\}$, where $U_i \subset M$ with $\cup_i U_i = M$, and $\varphi_i : U_i \subset M \longrightarrow \mathbb{R}^n$ is a bijection. For each pair of local charts (U_i, φ_i) and (U_j, φ_j), it is required that $\varphi_j(U_i \cap U_j)$ is open and $\varphi_{ij} = \varphi_i \circ \varphi_j^{-1}$ is a diffeomorphism.

The tangent space $T_pM \cong \mathbb{R}^n$ at $p \in M$ can be be thought of as directional derivatives operating on $C^\infty(M)$, the set of real valued differentiable functions $f : M \to \mathbb{R}$. Equivalently, the tangent space T_pM can be viewed in terms of an equivalence class of curves on M passing through p. Two curves $c_1 : (-\varepsilon, \varepsilon) \longrightarrow M$ and $c_2 : (-\varepsilon, \varepsilon) \longrightarrow M$ are equivalent at p in case $c_1(0) = c_2(0) = p$ and $\varphi \circ c_1$ and $\varphi \circ c_2$ are tangent at p for some local chart φ (and therefore all charts), in the sense that their derivatives at 0 exist and are equal.

In many cases of interest, the manifold M is a submanifold of a larger manifold, often $\mathbb{R}^m$, $m \geq n$. For example, the open n-dimensional simplex, defined by

$$\mathcal{P}_n \;=\; \left\{\theta \in \mathbb{R}^{n+1} : \quad \textstyle\sum_{i=1}^{n+1} \theta_i = 1,\ \theta_i > 0\right\} \tag{9}$$

is a submanifold of $\mathbb{R}^{n+1}$. In such a case, the tangent space of the submanifold T_pM is a subspace of $T_p\mathbb{R}^m$, and we may represent the tangent vectors $v \in T_pM$ in terms of the standard basis of the tangent space $T_p\mathbb{R}^m \cong \mathbb{R}^m$, $v = \sum_{i=1}^m v_i e_i$. The open n-simplex is a differential manifold with a single, global chart.

A *manifold with boundary* is defined similarly, except that the local charts (U, φ) satisfy $\varphi(U) \subset \mathbb{R}^{n+}$, thus mapping a patch of M to the half-space $\mathbb{R}^{n+} = \{x \in \mathbb{R}^n \,|\, x_n \geq 0\}$. In general, if U and V are open sets in $\mathbb{R}^{n+}$ in the topology induced from $\mathbb{R}^n$, and $f : U \longrightarrow V$ is a diffeomorphism, then f induces diffeomorphisms $\mathrm{Int} f : \mathrm{Int} U \longrightarrow \mathrm{Int} V$ and $\partial f : \partial U \longrightarrow \partial V$, where $\partial A = A \cup (\mathbb{R}^{n-1} \times \{0\})$ and $\mathrm{Int} A = A \cup \{x \in \mathbb{R}^n \,|\, x_n > 0\}$. Thus, it makes sense to define the *interior* $\mathrm{Int} M = \cup_U \varphi^{-1}(\mathrm{Int}(\varphi(U)))$ and *boundary* $\partial M = \cup_U \varphi^{-1}(\partial(\varphi(U)))$ of M. Since $\mathrm{Int} M$ is open it is an n-dimensional manifold without boundary, and ∂M is an $(n-1)$-dimensional manifold without boundary.

If $f : M \to N$ is a diffeomorphism of the manifold M onto the manifold N, then f induces a *push-foward mapping* f_* of the associated tangent spaces. A vector field $X \in TM$ is mapped to the push-forward $f_* X \in TN$, satisfying $(f_* X)(g) = X(g \circ f)$ for all $g \in C^\infty(N)$. Intuitively, the push-forward mapping transforms velocity vectors of curves to velocity vectors of the corresponding curves in the new manifold. Such a mapping is of use in transforming metrics, as described next.

A.2 The Laplacian

The construction of our kernels is based on the geometric Laplacian.[2] In order to define the generalization of the familiar Laplacian $\Delta = \frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2} + \cdots + \frac{\partial^2}{\partial x_n^2}$ on $\mathbb{R}^n$ to manifolds, one needs a notion

2. As described by Nelson (1968), "The Laplace operator in its various manifestations is the most beautiful and central object in all of mathematics. Probability theory, mathematical physics, Fourier analysis, partial differential equations, the theory of Lie groups, and differential geometry all revolve around this sun, and its light even penetrates such obscure regions as number theory and algebraic geometry."

of geometry, in particular a way of measuring lengths of tangent vectors. A *Riemannian manifold* (M,g) is a differentiable manifold M with a family of smoothly varying positive-definite inner products $g = g_p$ on T_pM for each $p \in M$. Two Riemannian manifolds (M,g) and (N,h) are *isometric* in case there is a diffeomorphism $f : M \longrightarrow N$ such that

$$g_p(X,Y) = h_{f(p)}(f_*X, f_*Y)$$

for every $X,Y \in T_pM$ and $p \in M$. Occasionally, hard computations on one manifold can be transformed to easier computations on an isometric manifold. Every manifold can be given a Riemannian metric. For example, every manifold can be embedded in $\mathbb{R}^m$ for some $m \geq n$ (the Whitney embedding theorem), and the Euclidean metric induces a metric on the manifold under the embedding. In fact, every Riemannian metric can be obtained in this way (the Nash embedding theorem).

In local coordinates, g can be represented as $g_p(v,w) = \sum_{i,j} g_{ij}(p)\, v_i\, w_j$ where $g(p) = [g_{ij}(p)]$ is a non-singular, symmetric and positive-definite matrix depending smoothly on p, and tangent vectors v and w are represented in local coordinates at p as $v = \sum_{i=1}^n v_i \partial_{i|p}$ and $w = \sum_{i=1}^n w_i \partial_{i|p}$. As an example, consider the open n-dimensional simplex defined in (9). A metric on $\mathbb{R}^{n+1}$ expressed by the symmetric positive-definite matrix $G = [g_{ij}] \in \mathbb{R}^{(n+1)\times(n+1)}$ induces a metric on $\mathcal{P}_n$ as

$$g_p(v,u) = g_p\left(\sum_{i=1}^{n+1} u_i e_i, \sum_{i=1}^{n+1} v_i e_i\right) = \sum_{i=1}^{n+1}\sum_{j=1}^{n+1} g_{ij}\, u_i v_j\,.$$

The metric enables the definition of lengths of vectors and curves, and therefore distance between points on the manifold. The length of a tangent vector at $p \in M$ is given by $\|v\| = \sqrt{\langle v, v\rangle_p}$, $v \in T_pM$ and the length of a curve $c : [a,b] \to M$ is then given by $L(c) = \int_a^b \|\dot{c}(t)\| \mathrm{d}t$ where $\dot{c}(t)$ is the velocity vector of the path c at time t. Using the above definition of lengths of curves, we can define the distance $d(x,y)$ between two points $x,y \in M$ as the length of the shortest piecewise differentiable curve connecting x and y. This *geodesic distance d* turns the Riemannian manifold into a metric space, satisfying the usual properties of positivity, symmetry and the triangle inequality. Riemannian manifolds also support convex neighborhoods. In particular, if $p \in M$, there is an open set U containing p such that any two points of U can be connected by a unique minimal geodesic in U.

A manifold is said to be *geodesically complete* in case every geodesic curve $c(t)$, $t \in [a,b]$, can be extended to be defined for all $t \in \mathbb{R}$. It can be shown (Milnor, 1963), that the following are equivalent: (1) M is geodesically complete, (2) d is a complete metric on M, and (3) closed and bounded subsets of M are compact. In particular, compact manifolds are geodesically complete. The Hopf-Rinow theorem (Milnor, 1963) asserts that if M is complete, then any two points can be joined by a minimal geodesic. This minimal geodesic is not necessarily unique, as seen by considering antipodal points on a sphere. The *exponential map* $\exp_x$ maps a neighborhood V of $0 \in T_xM$ diffeomorphically onto a neighborhood of $x \in M$. By definition, $\exp_x v$ is the point $\gamma_v(1)$ where γ_v is a geodesic starting at x with initial velocity $v = \frac{d\gamma_v}{dt}|_{t=0}$. Any such geodesic satisfies $\gamma_{rv}(s) = \gamma_v(rs)$ for $r > 0$. This mapping defines a local coordinate system on M called *normal coordinates*, under which many computations are especially convenient.

For a function $f : M \longrightarrow \mathbb{R}$, the gradient grad f is the vector field defined by

$$\langle \operatorname{grad} f(p), X\rangle \;=\; X(f)\,.$$

In local coordinates, the gradient is given by

$$(\operatorname{grad} f)_i = \sum_j g^{ij} \frac{\partial f}{\partial x_j}\,,$$

where $\left[g^{ij}(p)\right]$ is the inverse of $[g_{ij}(p)]$. The divergence operator is defined to be the adjoint of the gradient, allowing "integration by parts" on manifolds with special structure. An orientation of a manifold is a smooth choice of orientation for the tangent spaces, meaning that for local charts φ_i and φ_j, the differential $D(\varphi_j \circ \varphi_i)(x) : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ is orientation preserving, so the sign of the determinant is constant. If a Riemannian manifold M is orientable, it is possible to define a *volume form* μ, where if $v_1, v_2, \ldots, v_n \in T_pM$ (positively oriented), then

$$\mu(v_1, \ldots, v_n) \;=\; \sqrt{\det\langle v_i, v_j\rangle}\,.$$

A volume form, in turn, enables the definition of the *divergence* of a vector field on the manifold. In local coordinates, the divergence is given by

$$\operatorname{div} X = \frac{1}{\sqrt{\det g}} \sum_i \frac{\partial}{\partial x_i} \left(\sqrt{\det g}\, X_i\right)$$

where $\det g$ denotes the determinant of the matrix g_{ij}.

Finally, the *Laplace-Beltrami operator* on functions is defined by

$$\Delta = \operatorname{div} \circ \operatorname{grad}\,,$$

which in local coordinates is thus given by

$$\Delta f = \frac{1}{\sqrt{\det g}} \sum_j \frac{\partial}{\partial x_j} \left(\sum_i g^{ij} \sqrt{\det g}\, \frac{\partial f}{\partial x_i} \right)\,.$$

These definitions preserve the familiar intuitive interpretation of the usual operators in Euclidean geometry; in particular, the gradient points in the direction of steepest ascent and the divergence measures outflow minus inflow of liquid or heat.

Appendix B. Fisher Information Geometry

Let $\mathfrak{F} = \{p(\cdot\,|\,\theta)\}_{\theta\in\Theta}$ be an n-dimensional regular statistical family on a set $\mathcal{X}$. Thus, we assume that $\Theta \subset \mathbb{R}^n$ is open, and that there is a σ-finite measure μ on $\mathcal{X}$, such that for each $\theta \in \Theta$, $p(\cdot\,|\,\theta)$ is a density with respect to μ, so that $\int_{\mathcal{X}} p(x\,|\,\theta)\, d\mu(x) = 1$. We identify the manifold M with Θ by assuming that for each $x \in \mathcal{X}$ the mapping $\theta \mapsto p(x\,|\,\theta)$ is C^∞.

Let ∂_i denote $\partial/\partial\theta_i$, and $\ell_\theta(x) = \log p(x\,|\,\theta)$. The *Fisher information metric at* $\theta \in \Theta$ is defined in terms of the matrix $g(\theta) \in \mathbb{R}^{n\times n}$ given by

$$g_{ij}(\theta) \;=\; E_\theta\left[\partial_i \ell_\theta\, \partial_j \ell_\theta\right] \;=\; \int_{\mathcal{X}} p(x\,|\,\theta) \partial_i \log p(x\,|\,\theta)\, \partial_j \log p(x\,|\,\theta)\, d\mu(x)\,.$$

Since the score $s_i(\theta) = \partial_i \ell_\theta$ has mean zero, $g_{ij}(\theta)$ can be seen as the variance of $s_i(\theta)$, and is therefore positive-definite. By assumption, it is smoothly varying in θ, and therefore defines a Riemannian metric on $\Theta = M$.

An equivalent and sometimes more suggestive form of the Fisher information matrix, as will be seen below for the case of the multinomial, is

$$g_{ij}(\theta) \;=\; 4 \int_{\mathcal{X}} \partial_i \sqrt{p(x\,|\,\theta)}\, \partial_j \sqrt{p(x\,|\,\theta)}\, d\mu(x)\,.$$

Yet another equivalent form is $g_{ij}(\theta) = -E_\theta[\partial_j \partial_i \ell_\theta]$. To see this, note that

$$\begin{aligned}
E_\theta[\partial_j \partial_i \ell_\theta] &= \int_{\mathcal{X}} p(x \,|\, \theta) \partial_j \partial_i \log p(x \,|\, \theta)\, d\mu(x) \\
&= -\int_{\mathcal{X}} p(x \,|\, \theta) \frac{\partial_j p(x \,|\, \theta)}{p(x \,|\, \theta)^2} \partial_i p(x \,|\, \theta)\, d\mu(x) - \int_{\mathcal{X}} \partial_j \partial_i p(x \,|\, \theta)\, d\mu(x) \\
&= -\int_{\mathcal{X}} p(x \,|\, \theta) \frac{\partial_j p(x \,|\, \theta)}{p(x \,|\, \theta)} \frac{\partial_i p(x \,|\, \theta)}{p(x \,|\, \theta)}\, d\mu(x) - \partial_j \partial_i \int_{\mathcal{X}} p(x \,|\, \theta)\, d\mu(x) \\
&= -\int_{\mathcal{X}} p(x \,|\, \theta) \partial_j \log p(x \,|\, \theta) \partial_i \log p(x \,|\, \theta)\, d\mu(x) \\
&= -g_{ij}(\theta)\,.
\end{aligned}$$

Since there are many possible choices of metric on a given differentiable manifold, it is important to consider the motivating properties of the Fisher information metric. Intuitively, the Fisher information may be thought of as the amount of information a single data point supplies with respect to the problem of estimating the parameter θ. This interpretation can be justified in several ways, notably through the efficiency of estimators. In particular, the asymptotic variance of the maximum likelihood estimator $\hat{\theta}$ obtained using a sample of size n is $(ng(\theta))^{-1}$. Since the MLE is asymptotically unbiased, the inverse Fisher information represents the asymptotic fluctuations of the MLE around the true value. Moreover, by the Cramér-Rao lower bound, the variance of any unbiased estimator is bounded from below by $(ng(\theta))^{-1}$. Additional motivation for the Fisher information metric is provided by the results of Čencov (1982), which characterize it as the only metric (up to multiplication by a constant) that is invariant with respect to certain probabilistically meaningful transformations called congruent embeddings.

The connection with another familiar similarity measure is worth noting here. If p and q are two densities on $\mathcal{X}$ with respect to μ, the Kullback-Leibler divergence $D(p,q)$ is defined by

$$D(p,q) = \int_{\mathcal{X}} p(x) \log \frac{p(x)}{q(x)}\, d\mu(x)\,.$$

The Kullback-Leibler divergence behaves at nearby points like the square of the information distance. More precisely, it can be shown that

$$\lim_{q \to p} \frac{d^2(p,q)}{2D(p,q)} = 1\,,$$

where the convergence is uniform as $d(p,q) \to 0$. As we comment in the text, this relationship may be of use in approximating information diffusion kernels for complex models.

B.1 Fisher information for the Spherical Gaussian

Here we derive the Fisher information for the special case of the family $\mathfrak{F} = \{p(\cdot \,|\, \theta)\}_{\theta \in \Theta}$ where $\theta = (\mu, \sigma)$ and $p(\cdot \,|\, (\mu, \sigma)) = \mathcal{N}(\mu, \sigma I_{n-1})$, the Gaussian having mean $\mu \in \mathbb{R}^{n-1}$ and variance σI_{n-1}, with $\sigma > 0$. The parameter space is thus $\Theta = \mathbb{R}^{n-1} \times \mathbb{R}_+$.

To compute the Fisher information metric for this family, it is convenient to use the general expression given by equation (10). Let $\partial_i = \partial/\partial\mu_i$ for $i = 1 \ldots n-1$, and $\partial_n = \partial/\partial\sigma$. Then simple calculations yield, for $1 \leq i, j \leq n-1$

$$\begin{aligned} g_{ij}(\theta) &= -\int_{\mathbb{R}^{n-1}} \partial_i \partial_j \left(-\sum_{k=1}^{n-1} \frac{(x_k-\mu_k)^2}{2\sigma^2} \right) p(x \,|\, \theta)\, dx \\ &= \frac{1}{\sigma^2} \delta_{ij} \end{aligned}$$

$$\begin{aligned} g_{ni}(\theta) &= -\int_{\mathbb{R}^{n-1}} \partial_n \partial_i \left(-\sum_{k=1}^{n-1} \frac{(x_k-\mu_k)^2}{2\sigma^2} \right) p(x \,|\, \theta)\, dx \\ &= \frac{2}{\sigma^3} \int_{\mathbb{R}^{n-1}} (x_i - \mu_i)\, p(x \,|\, \theta)\, dx \\ &= 0 \end{aligned}$$

$$\begin{aligned} g_{nn}(\theta) &= -\int_{\mathbb{R}^{n-1}} \partial_n \partial_n \left(-\sum_{k=1}^{n-1} \frac{(x_k-\mu_k)^2}{2\sigma^2} - (n-1)\log\sigma \right) p(x \,|\, \theta)\, dx \\ &= \frac{3}{\sigma^4} \int_{\mathbb{R}^{n-1}} \sum_{k=1}^{n-1} (x_k-\mu_k)^2\, p(x \,|\, \theta)\, dx - \frac{n-1}{\sigma^2} \\ &= \frac{2(n-1)}{\sigma^2}\,. \end{aligned}$$

Letting θ' be new coordinates defined by $\theta'_i = \mu_i$ for $1 \leq i \leq n-1$ and $\theta'_n = \sqrt{2(n-1)}\,\sigma$, it is seen that the Fisher information matrix is given by

$$g_{ij}(\theta') = \frac{1}{\sigma^2} \delta_{ij}\,.$$

Thus, the Fisher information metric gives $\Theta = \mathbb{R}^{n-1} \times \mathbb{R}_+$ the structure of the upper half plane in hyperbolic space.

References

Mark A. Aizerman, Emmanuel M. Braverman, and Lev I. Rozonoér. Theoretical foundations of the potential function method in pattern recognition and learning. *Automation and Remote Control*, 25:821–837, 1964.

Shun-ichi Amari and Hiroshi Nagaoka. *Methods of Information Geometry*, volume 191 of *Translations of Mathematical Monographs*. American Mathematical Society, 2000.

Peter L. Bartlett, Olivier Bousquet, and Shahar Mendelson. Local Rademacher complexities. *The Annals of Statistics*, 2004. To appear.

Peter L. Bartlett and Shahar Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002.

Mikhail Belkin and Partha Niyogi. Using manifold structure for partially labeled classification. In *Advances in Neural Information Processing Systems*, 2003.

Marcel Berger, Paul Gauduchon, and Edmond Mazet. Le spectre d'une varieté Riemannienne. *Lecture Notes in Mathematics*, Vol. 194, Springer-Verlag, 1971.

Bernhard E. Boser, Isabelle Guyon, and Vladimir Vapnik. A training algorithm for optimal margin classifiers. In *Computational Learing Theory*, pages 144–152, 1992.

Nikolai Nikolaevich Čencov. *Statistical Decision Rules and Optimal Inference*, volume 53 of *Translation of Mathematical Monographs*. American Mathematical Society, 1982.

Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew K. McCallum, Tom M. Mitchell, Kamal Nigam, and Seán Slattery. Learning to construct knowledge bases from the World Wide Web. *Artificial Intelligence*, 118(1/2):69–113, 2000.

A. Philip Dawid. Further comments on some comments on a paper by Bradley Efron. *The Annals of Statistics*, 5(6):1249, 1977.

Tom Dietterich. AI Seminar. Carnegie Mellon, 2002.

Alan T. Gous. *Exponential and Spherical Subfamily Models*. PhD thesis, Stanford University, 1998.

Alexander Grigor'yan and Masakazu Noguchi. The heat kernel on hyperbolic space. *Bulletin of the London Mathematical Society*, 30:643–650, 1998.

Ying Guo, Peter L. Bartlett, John Shawe-Taylor, and Robert C. Williamson. Covering numbers for support vector machines. *IEEE Trans. Information Theory*, 48(1), January 2002.

Tommi S. Jaakkola and David Haussler. Exploiting generative models in discriminative classifiers. In *Advances in Neural Information Processing Systems*, volume 11, 1998.

Thorsten Joachims. *The Maximum Margin Approach to Learning Text Classifiers Methods, Theory and Algorithms*. PhD thesis, Dortmund University, 2000.

Thorsten Joachims, Nello Cristianini, and John Shawe-Taylor. Composite kernels for hypertext categorisation. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2001.

Robert E. Kass. The geometry of asymptotic inference. *Statistical Science*, 4(3), 1989.

Robert E. Kass and Paul W. Vos. *Geometrical Foundations of Asymptotic Inference*. Wiley Series in Probability and Statistics. John Wiley & Sons, 1997.

George Kimeldorf and Grace Wahba. Some results on Tchebychean spline functions. *J. Math. Anal. Applic.*, 33:82–95, 1971.

Risi Imre Kondor and John Lafferty. Diffusion kernels on graphs and other discrete input spaces. In C. Sammut and A. Hoffmann, editors, *Proceedings of the International Conference on Machine Learning (ICML)*. Morgan Kaufmann, 2002.

John Lafferty and Guy Lebanon. Information diffusion kernels. In S. Thrun S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 375–382. MIT Press, Cambridge, MA, 2003.

Stefan L. Lauritzen. Statistical manifolds. In S. I. Amari, O. E. Barndorff-Nielsen, R. E. Kass, S. L. Lauritzen, and C. R. Rao, editors, *Differential Geometry in Statistical Inference*, pages 163–216. Institute of Mathematical Statistics, Hayward, CA, 1987.

David D. Lewis and Marc Ringuette. A comparison of two learning algorithms for text categorization. In *Symposium on Document Analysis and Information Retrieval*, pages 81–93, Las Vegas, NV, April 1994. ISRI; Univ. of Nevada, Las Vegas.

Shahar Mendelson. On the performance of kernel classes. *Journal of Machine Learning Research*, 4:759–771, 2003.

John W. Milnor. *Morse Theory*. Princeton University Press, 1963.

Pedro J. Moreno, Purdy P. Ho, and Nuno Vasconcelos. A Kullback-Leibler divergence based kernel for SVM classification in multimedia applications. In *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

Edward Nelson. *Tensor Analysis*. Princeton University Press, 1968.

Tomaso Poggio and Frederico Girosi. Regularization algorithms for learning that are equivalent to multilayer networks. *Science*, 247:978–982, 1990.

Jay Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *Proceedings of the ACM SIGIR*, pages 275–281, 1998.

Calyampudi R. Rao. Information and accuracy attainable in the estimation of statistical parameters. *Bull. Calcutta Math. Soc.*, 37:81–91, 1945.

Steven Rosenberg. *The Laplacian on a Riemannian Manifold*. Cambridge University Press, 1997.

Richard Schoen and Shing-Tung Yau. *Lectures on Differential Geometry*, volume 1 of *Conference Proceedings and Lecture Notes in Geometry and Topology*. International Press, 1994.

Michael Spivak. *Differential Geometry*, volume 1. Publish or Perish, 1979.

Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of SIGIR'2001*, pages 334–342, Sept 2001.

Tong Zhang and Frank J. Oles. Text categorization based on regularized linear classification methods. *Information Retrieval*, 4:5–31, April 2001.

Journal of Machine Learning Research 6 (2005) 165–188 Submitted 2/04; Revised 7/04; Published 1/05

Information Bottleneck for Gaussian Variables

Gal Chechik[*] GAL@ROBOTICS.STANFORD.EDU
Computer Science Department
Stanford University
Stanford CA 94305-9025, USA

Amir Globerson[*] GAMIR@CS.HUJI.AC.IL
Naftali Tishby TISHBY@CS.HUJI.AC.IL
Yair Weiss YWEISS@CS.HUJI.AC.IL
School of Computer Science and Engineering and
The Interdisciplinary Center for Neural Computation
The Hebrew University of Jerusalem
Givat Ram, Jerusalem 91904, Israel

Editor: Peter Dayan

Abstract

The problem of extracting the relevant aspects of data was previously addressed through the *information bottleneck* (IB) method, through (soft) clustering one variable while preserving information about another - *relevance* - variable. The current work extends these ideas to obtain continuous representations that preserve relevant information, rather than discrete clusters, for the special case of multivariate Gaussian variables. While the general continuous IB problem is difficult to solve, we provide an analytic solution for the optimal representation and tradeoff between compression and relevance for the this important case. The obtained optimal representation is a noisy linear projection to eigenvectors of the normalized regression matrix $\Sigma_{x|y}\Sigma_x^{-1}$, which is also the basis obtained in canonical correlation analysis. However, in Gaussian IB, the compression tradeoff parameter uniquely determines the dimension, as well as the scale of each eigenvector, through a cascade of structural phase transitions. This introduces a novel interpretation where solutions of different ranks lie on a continuum parametrized by the compression level. Our analysis also provides a complete analytic expression of the preserved information as a function of the compression (the "information-curve"), in terms of the eigenvalue spectrum of the data. As in the discrete case, the information curve is concave and smooth, though it is made of different analytic segments for each optimal dimension. Finally, we show how the algorithmic theory developed in the IB framework provides an iterative algorithm for obtaining the optimal Gaussian projections.

Keywords: information bottleneck, Gaussian processes, dimensionality reduction, canonical correlation analysis

1. Introduction

Extracting relevant aspects of complex data is a fundamental task in machine learning and statistics. The problem is often that the data contains many structures, which make it difficult to define which of them are relevant and which are not in an unsupervised manner. For example, speech signals may

∗. Both authors contributed equally.

be characterized by their volume level, pitch, or content; pictures can be ranked by their luminosity level, color saturation or importance with regard to some task.

This problem was addressed in a principled manner by the information bottleneck (IB) approach (Tishby et al., 1999). Given the joint distribution of a "source" variable X and another "relevance" variable Y, IB operates to compress X, while preserving information about Y. The variable Y thus implicitly defines what is relevant in X and what is not. Formally, this is cast as the following variational problem

$$\min_{p(t|x)} \mathcal{L} : \mathcal{L} \equiv I(X;T) - \beta I(T;Y) \tag{1}$$

where T represents the compressed representation of X via the conditional distributions $p(t|x)$, while the information that T maintains on Y is captured by the distribution $p(y|t)$. This formulation is general and does not depend on the type of the X,Y distribution. The positive parameter β determines the tradeoff between compression and preserved relevant information, as the Lagrange multiplier for the constrained optimization problem $\min_{p(t|x)} I(X;T) - \beta\,(I(T;Y) - const)$. Since T is a function of X it is independent of Y given X, thus the three variables can be written as the Markov chain $Y - X - T$. From the information inequality we thus have $I(X;T) - \beta I(T;Y) \geq (1-\beta)I(T;Y)$, and therefore for all values of $\beta \leq 1$, the optimal solution of the minimization problem is degenerated $I(T;X) = I(T;Y) = 0$. As we will show below, the range of degenerated solutions is even larger for Gaussian variables and depends on the eigen spectrum of the variables covariance matrices.

The rationale behind the IB principle can be viewed as model-free "looking inside the black-box" system analysis approach. Given the input-output (X,Y) "black-box" statistics, IB aims to construct efficient representations of X, denoted by the variable T, that can account for the observed statistics of Y. IB achieves this using a single tradeoff parameter to represent the tradeoff between the complexity of the representation of X, measured by $I(X;T)$, and the accuracy of this representation, measured by $I(T;Y)$. The choice of mutual information for the characterization of complexity and accuracy stems from Shannon's theory, where information minimization corresponds to optimal compression in Rate Distortion Theory, and its maximization corresponds to optimal information transmission in Noisy Channel Coding.

From a machine learning perspective, IB may be interpreted as regularized generative modeling. Under certain conditions $I(T;Y)$ can be interpreted as an empirical likelihood of a special mixture model, and $I(T;X)$ as penalizing complex models (Slonim and Weiss, 2002). While this interpretation can lead to interesting analogies , it is important to emphasize the differences. First, IB views $I(X;T)$ not as a regularization term, but rather corresponds to the distortion constraint in the original system. As a result, this constraint is useful even when the joint distribution is known exactly, because the goal of IB is to obtain compact representations rather than to estimate density. Interestingly, $I(T;X)$ also characterizes the complexity of the representation T as the expected number of bits needed to specify the t for a given x. In that role it can be viewed as an expected "cost" of the internal representation, as in MDL. As is well acknowledged now source coding with distortion and channel coding with cost are dual problems (see for example Shannon, 1959; Pradhan et al., 2003). In that information theoretic sense, IB is *self dual*, where the resulting source and channel are perfectly matched (as in Gastpar and Vetterli, 2003).

The information bottleneck approach has been applied so far mainly to categorical variables, with a discrete T that represents (soft) clusters of X. It has been proved useful for a range of applications from documents clustering (Slonim and Tishby, 2000) through neural code analysis (Dimitrov and Miller, 2001) to gene expression analysis (Friedman et al., 2001; Sinkkonen and Kaski, 2001) (for a more detailed review of IB clustering algorithms see Slonim (2003)). However, its general information theoretic formulation is not restricted, both in terms of the nature of the variables X and Y, as well as of the compression variable T. It can be naturally extended to nominal, categorical, and continuous variables, as well as to dimension reduction rather than clustering techniques. The goal of this paper is apply the IB for the special, but very important, case of Gaussian processes which has become one of the most important generative classes in machine learning. In addition, this is the first concrete application of IB to dimension reduction with continuous compressed representation, and as such exhibit interesting dimension related phase transitions.

The general solution of IB for continuous T yields the same set of self-consistent equations obtained already in (Tishby et al., 1999), but solving these equations for the distributions $p(t|x)$, $p(t)$ and $p(y|t)$ without any further assumptions is a difficult challenge, as it yields non-linear coupled eigenvalue problems. As in many other cases, however, we show here that the problem turns out to be analytically tractable when X and Y are joint multivariate Gaussian variables. In this case, rather than using the fixed point equations and the generalized Blahut-Arimoto algorithm as proposed in (Tishby et al., 1999), one can explicitly optimize the target function with respect to the mapping $p(t|x)$ and obtain a closed form solution of the optimal dimensionality reduction.

The optimal compression in the Gaussian information bottleneck (GIB) is defined in terms of the compression-relevance tradeoff (also known as the "Information Curve", or "Accuracy-Complexity" tradeoff), determined by varying the parameter β. The optimal solution turns out to be a noisy linear projection to a subspace whose dimensionality is determined by the parameter β. The subspaces are spanned by the basis vectors obtained as in the well known *canonical correlation analysis* (CCA) (Hotelling, 1935), but the exact nature of the projection is determined in a unique way via the parameter β. Specifically, as β increases, additional dimensions are added to the projection variable T, through a series of critical points (structural phase transitions), while at the same time the relative magnitude of each basis vector is rescaled. This process continues until all the relevant information about Y is captured in T. This demonstrates how the IB principle can provide a continuous measure of model complexity in information theoretic terms.

The idea of maximization of relevant information was also taken in the *Imax* framework of Becker and Hinton (Becker and Hinton, 1992; Becker, 1996), which followed Linsker's idea of information maximization (Linsker, 1988, 1992). In the Imax setting, there are two one-layer feed forward networks with inputs X_a, X_b and outputs neurons Y_a, Y_b; the output neuron Y_a serves to define relevance to the output of the neighboring network Y_b. Formally, the goal is to tune the incoming weights of the output neurons, such that their mutual information $I(Y_a;Y_b)$ is maximized. An important difference between *Imax* and the IB setting, is that in the *Imax* setting, $I(Y_a;Y_b)$ is invariant to scaling and translation of the Y's since the compression achieved in the mapping $X_a \rightarrow Y_a$ is not modeled explicitly. In contrast, the IB framework aims to characterize the dependence of the solution on the explicit compression term $I(T;X)$, which is a *scale sensitive* measure when the transformation is noisy. This view of compressed representation T of the inputs X is useful when dealing with neural systems that are stochastic in nature and limited in their responses amplitudes and are thus constrained to finite $I(T;X)$.

The current paper starts by defining the problem of relevant information extraction for Gaussian variables. Section 3 gives the main result of the paper: an analytical characterization of the optimal projections, which is then developed in Section 4. Section 5 develops an analytical expression for the GIB compression-relevance tradeoff - the information curve. Section 6 shows how the general IB algorithm can be adapted to the Gaussian case, yielding an iterative algorithm for finding the optimal projections. The relations to canonical correlation analysis and coding with side-information are discussed in Section 9.

2. Gaussian Information Bottleneck

We now formalize the problem of information bottleneck for Gaussian variables. Let (X,Y) be two jointly multivariate Gaussian variables of dimensions n_x, n_y and denote by Σ_x,Σ_y the covariance matrices of X,Y and by Σ_{xy} their cross-covariance matrix.[1] The goal of GIB is to compress the variable X via a stochastic transformation into another variable $T \in R^{n_x}$, while preserving information about Y. The dimension of T is not explicitly limited in our formalism, since we will show that the effective dimension is determined by the value of β.

It is shown in Globerson and Tishby (2004) that the optimum for this problem is obtained by a variable T which is also jointly Gaussian with X. The formal proof uses the entropy power inequality as in Berger and Zamir (1999), and is rather technical, but an intuitive explanation is that since X and Y are Gaussians, the only statistical dependencies that connect them are bi-linear. Therefore, a linear projection of X is sufficient to capture all the information that X has on Y. The Entropy-power inequality is used to show that a linear projection of X, which is also Gaussian in this case, indeed attains this maximum information.

Since every two centered random variables X and T with jointly Gaussian distribution can be presented through the linear transformation $T = AX + \xi$, where $\xi \sim N(0, \Sigma_\xi)$ is another Gaussian that is independent of X, we formalize the problem using this representation of T, as the minimization

$$\min_{A,\Sigma_\xi} \mathcal{L} \equiv I(X;T) - \beta I(T;Y) \tag{2}$$

over the noisy linear transformations of A, Σ_ξ

$$T = AX + \xi; \quad \xi \sim N(0, \Sigma_\xi). \tag{3}$$

Thus T is normally distributed $T \sim N(0, \Sigma_t)$ with $\Sigma_t = A\Sigma_x A^T + \Sigma_\xi$.

Interestingly, the term ξ can also be viewed as an additive noise term, as commonly done in models of learning in neural networks. Under this view, ξ serves as a regularization term whose covariance determines the scales of the problem. While the goal of GIB is to find the optimal projection parameters A, Σ_ξ jointly, we show below that the problem factorizes such that the optimal projection A does not depend on the noise, which does not carry any information about Y.

3. The Optimal Projection

The first main result of this paper is the characterization of the optimal A,Σ_ξ as a function of β

1. For simplicity we assume that X and Y have zero means and Σ_x, Σ_y are full rank. Otherwise X and Y can be centered and reduced to the proper dimensionality.

Theorem 3.1 *The optimal projection $T = AX + \xi$ for a given tradeoff parameter β is given by $\Sigma_\xi = I_x$ and*

$$A = \left\{ \begin{array}{cc} \left[\mathbf{0}^T; \ldots; \mathbf{0}^T\right] & 0 \leq \beta \leq {\beta^c}_1 \\ \left[\alpha_1 \mathbf{v}_1^T, \mathbf{0}^T; \ldots; \mathbf{0}^T\right] & \beta_1^c \leq \beta \leq {\beta^c}_2 \\ \left[\alpha_1 \mathbf{v}_1^T; \alpha_2 \mathbf{v}_2^T; \mathbf{0}^T; \ldots; \mathbf{0}^T\right] & {\beta^c}_2 \leq \beta \leq {\beta^c}_3 \\ \vdots & \end{array} \right\} \tag{4}$$

where $\{\mathbf{v}_1^T, \mathbf{v}_2^T, \ldots, \mathbf{v}_{n_x}^T\}$ are left eigenvectors of $\Sigma_{x|y}\Sigma_x^{-1}$ sorted by their corresponding ascending eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_{n_x}$, ${\beta^c}_i = \frac{1}{1-\lambda_i}$ are critical β values, α_i are coefficients defined by $\alpha_i \equiv \sqrt{\frac{\beta(1-\lambda_i)-1}{\lambda_i r_i}}$, $r_i \equiv \mathbf{v}_i^T \Sigma_x \mathbf{v}_i$, $\mathbf{0}^T$ is an n_x dimensional row vector of zeros, and semicolons separate rows in the matrix A.

This theorem asserts that the optimal projection consists of eigenvectors of $\Sigma_{x|y}\Sigma_x^{-1}$, combined in an interesting manner: For β values that are smaller than the smallest critical point ${\beta^c}_1$, compression is more important than any information preservation and the optimal solution is the degenerated one $A \equiv 0$. As β is increased, it goes through a series of critical points ${\beta^c}_i$, at each of which another eigenvector of $\Sigma_{x|y}\Sigma_x^{-1}$ is added to A. Even though the rank of A increases at each of these transition points, A changes continuously as a function of β since at each critical point ${\beta^c}_i$ the coefficient α_i vanishes. Thus β parameterizes a sort of "continuous rank" of the projection.

To illustrate the form of the solution, we plot the landscape of the target function $\mathcal{L}$ together with the solution in a simple problem where $X \in R^2$ and $Y \in R$. In this case A has a single non-zero row, thus A can be thought of as a row vector of length 2, that projects X to a scalar $A : X \to R$, $T \in R$. Figure 1 shows the target function $\mathcal{L}$ as a function of the (vector of length 2) projection A. In this example, the largest eigenvalue is $\lambda_1 = 0.95$, yielding ${\beta^c}_1 = 20$. Therefore, for $\beta = 15$ (Figure 1A) the zero solution is optimal, but for $\beta = 100 > \beta^c$ (Figure 1B) the corresponding eigenvector is a feasible solution, and the target function manifold contains two mirror minima. As β increases from 1 to ∞, these two minima, starting as a single unified minimum at zero, split at β^c, and then diverge apart to ∞.

We now turn to prove Theorem 3.1.

4. Deriving the Optimal Projection

We first rewrite the target function as

$$\mathcal{L} = I(X;T) - \beta I(T;Y) = h(T) - h(T|X) - \beta h(T) + \beta h(T|Y) \tag{5}$$

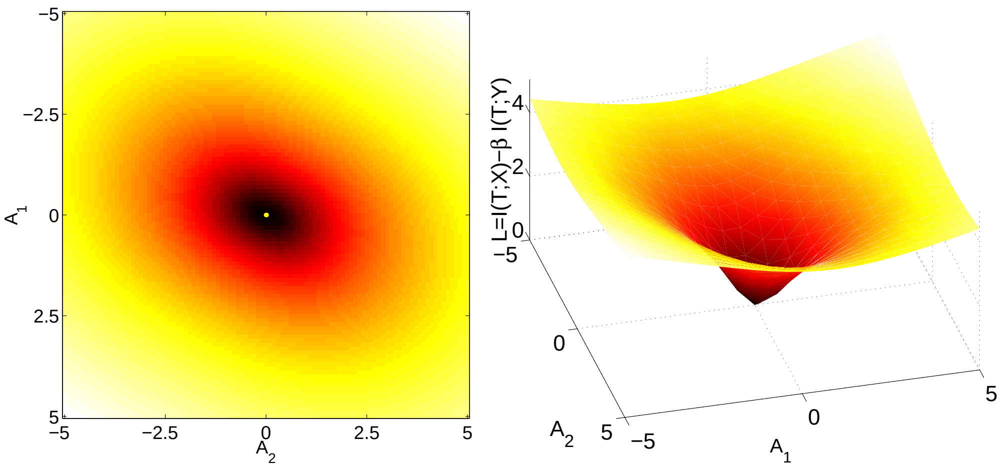
where h is the (differential) entropy of a continuous variable

$$h(X) \equiv -\int_X f(x) \log f(x)\, dx.$$

Recall that the entropy of a d dimensional Gaussian variable is

$$h(X) = \frac{1}{2} \log\left((2\pi e)^d |\Sigma_x|\right)$$

A.



B.

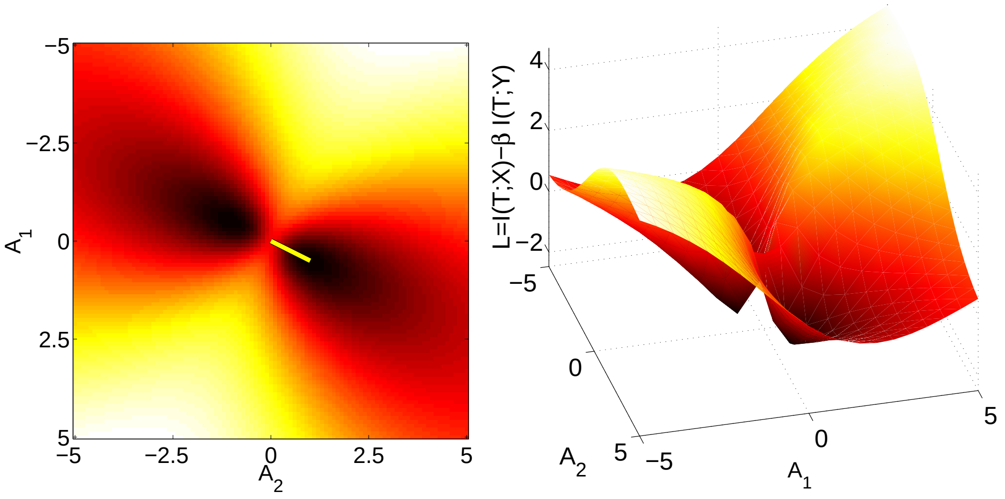


Figure 1: The surface of the target function $\mathcal{L}$ calculated numerically as a function of the optimization parameters in two illustrative examples with a scalar projection $A : R^2 \rightarrow R$. Each row plots the target surface $\mathcal{L}$ both in 2D (left) and 3D (right) as a function of the (two dimensional) projections A. **A.** For $\beta = 15$, the optimal solution is the degenerated solution $A \equiv 0$. **B.** For $\beta = 100$, a non degenerate solution is optimal, together with its mirror solution. The $\Sigma_{x|y}\Sigma_x^{-1}$- eigenvector of smallest eigenvalue, with a norm computed according to Theorem 3.1 is superimposed, showing that it obtains the global minimum of $\mathcal{L}$. Parameters' values $\Sigma_{xy} = [0.1\ 0.2]$, $\Sigma_x = I_2$, $\Sigma_\xi = 0.3I_{2\times 2}$.

where $|x|$ denotes the determinant of x, and Σ_x is the covariance of X. We therefore turn to calculate the relevant covariance matrices. From the definition of T we have $\Sigma_{tx} = A\Sigma_x$, $\Sigma_{ty} = A\Sigma_{xy}$ and $\Sigma_t = A\Sigma_x A^T + \Sigma_\xi$. Now, the conditional covariance matrix $\Sigma_{x|y}$ can be used to calculate the covariance of the conditional variable $T|Y$, using the Schur complement formula (see e.g., Magnus and Neudecker, 1988)

$$\Sigma_{t|y} = \Sigma_t - \Sigma_{ty}\Sigma_y^{-1}\Sigma_{yt} = A\Sigma_{x|y}A^T + \Sigma_\xi.$$

The target function can now be rewritten as

$$\begin{aligned} \mathcal{L} &= \log(|\Sigma_t|) - \log(|\Sigma_{t|x}|) - \beta\log(|\Sigma_t|) + \beta\log(|\Sigma_{t|y}|). \qquad (6)\\ &= (1-\beta)\log(|A\Sigma_x A^T + \Sigma_\xi|) - \log(|\Sigma_\xi|) + \beta\log(|A\Sigma_{x|y}A^T + \Sigma_\xi|) \end{aligned}$$

Although $\mathcal{L}$ is a function of both the noise Σ_ξ *and* the projection A, Lemma A.1 in Appendix A shows that for every pair (A, Σ_ξ), there is another projection $\tilde{A}$ such that the pair $(\tilde{A}, I)$ obtains the same value of $\mathcal{L}$. This is obtained by setting $\tilde{A} = \sqrt{D^{-1}}VA$ where $\Sigma_\xi = VDV^T$, which yields $\mathcal{L}(\tilde{A}, I) = \mathcal{L}(A, \Sigma_\xi)$.[2] This allows us to simplify the calculations by replacing the noise covariance matrix Σ_ξ with the identity matrix I_d.

To identify the minimum of $\mathcal{L}$ we differentiate $\mathcal{L}$ with respect to the projection A using the algebraic identity $\frac{\delta}{\delta A}\log(|ACA^T|) = (ACA^T)^{-1}2AC$ which holds for any symmetric matrix C:

$$\frac{\delta\mathcal{L}}{\delta A} = (1-\beta)(A\Sigma_x A^T + I_d)^{-1}2A\Sigma_x + \beta(A\Sigma_{x|y}A^T + I_d)^{-1}2A\Sigma_{x|y}. \qquad (7)$$

Equating this derivative to zero and rearranging, we obtain necessary conditions for an internal minimum of $\mathcal{L}$, which we explore in the next two sections.

4.1 Scalar Projections

For clearer presentation of the general derivation, we begin with a sketch of the proof by focusing on the case where T is a scalar, that is, the optimal projection matrix A is a now a single row vector. In this case, both $A\Sigma_x A^T$ and $A\Sigma_{x|y}A^T$ are scalars, and we can write

$$\left(\frac{\beta-1}{\beta}\right)\left(\frac{A\Sigma_{x|y}A^T + 1}{A\Sigma_x A^T + 1}\right)A = A\left[\Sigma_{x|y}\Sigma_x^{-1}\right]. \qquad (8)$$

This equation is therefore an eigenvalue problem in which the eigenvalues depend on A. It has two types of solutions depending on the value of β. First, A may be identically zero. Otherwise, A must be the eigenvector of $\Sigma_{x|y}\Sigma_x^{-1}$, with an eigenvalue $\lambda = \frac{\beta-1}{\beta}\frac{A\Sigma_{x|y}A^T+1}{A\Sigma_x A^T+1}$

To characterize the values of β for which the optimal solution does not degenerate, we find when the eigenvector solution is optimal. Denote the norm of Σ_x with respect to A by $r = \frac{A\Sigma_x A^T}{||A||^2}$. When A is an eigenvector of $\Sigma_{x|y}\Sigma_x^{-1}$, Lemma B.1 shows that r is positive and that $A\Sigma_{x|y}\Sigma_x^{-1}\Sigma_x A^T = \lambda r||A||^2$. Rewriting the eigenvalue and isolating $||A||^2$, we have

$$0 < ||A||^2 = \frac{\beta(1-\lambda)-1}{r\lambda}. \qquad (9)$$

2. Although this holds only for full rank Σ_ξ, it does not limit the generality of the discussion since low rank matrices yield infinite values of $\mathcal{L}$ and are therefore suboptimal.

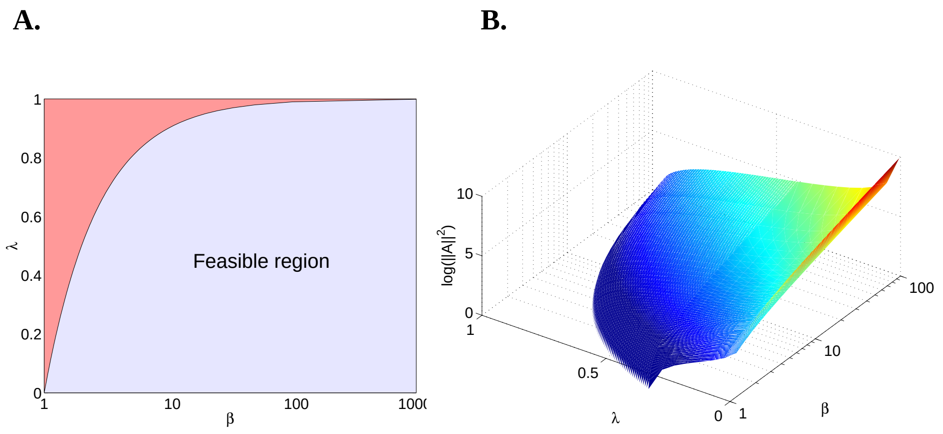


Figure 2: **A.** The regions of (β,λ) pairs that lead to the zero (red) and eigenvector (blue) solutions. **B.** The norm $||A||^2$ as a function of β and λ over the feasible region.

This inequality provides a constraint on β and λ that is required for a non-degenerated type of solution

$$\lambda \leq \frac{\beta-1}{\beta} \quad \text{or} \quad \beta \geq (1-\lambda)^{-1}, \tag{10}$$

thus defining a critical value $\beta^c(\lambda) = (1-\lambda)^{-1}$. For $\beta \leq \beta^c(\lambda)$, the weight of compression is so strong that the solution degenerates to zero and no information is carried about X or Y. For $\beta \geq \beta^c(\lambda)$ the weight of information preservation is large enough, and the optimal solution for A is an eigenvector of $\Sigma_{x|y}\Sigma_x^{-1}$. The feasible regions for non degenerated solutions and the norm $||A||^2$ as a function of β and λ are depicted in Figure 2.

For some β values, several eigenvectors can satisfy the condition for non degenerated solutions of Equation (10). Appendix C shows that the optimum is achieved by the eigenvector of $\Sigma_{x|y}\Sigma_x^{-1}$ with the smallest eigenvalue. Note that this is also the eigenvector of $\Sigma_{xy}\Sigma_y^{-1}\Sigma_{yx}\Sigma_x^{-1}$ with the largest eigenvalue. We conclude that for scalar projections

$$A(\beta) = \begin{cases} \sqrt{\frac{\beta(1-\lambda)-1}{r\lambda}}\, v_1 & 0 < \lambda \leq \frac{\beta-1}{\beta} \\ 0 & \frac{\beta-1}{\beta} \leq \lambda \leq 1 \end{cases} \tag{11}$$

where v_1 is the eigenvector of $\Sigma_{x|y}\Sigma_x^{-1}$ with the smallest eigenvalue.

4.2 The High-Dimensional Case

We now return to the proof of the general, high dimensional case, which follows the same lines as the scalar projection case. Setting the gradient in Equation (7) to zero and reordering we obtain

$$\frac{\beta-1}{\beta}\left[(A\Sigma_{x|y}A^T + I_d)(A\Sigma_x A^T + I_d)^{-1}\right]A = A\left[\Sigma_{x|y}\Sigma_x^{-1}\right]. \tag{12}$$

Equation (12) shows that the multiplication of $\Sigma_{x|y}\Sigma_x^{-1}$ by A must reside in the span of the rows of A. This means that A should be spanned by up to n_t eigenvectors of $\Sigma_{x|y}\Sigma_x^{-1}$. We can therefore represent the projection A as a mixture $A = WV$ where the rows of V are left normalized eigenvectors of $\Sigma_{x|y}\Sigma_x^{-1}$ and W is a mixing matrix that weights these eigenvectors. The form of the mixing matrix W, that characterizes the norms of these eigenvectors, is described in the following lemma, which is proved in Appendix D.

Lemma 4.1 *The optimum of the cost function is obtained with a diagonal mixing matrix W of the form*

$$W = diag\left[\sqrt{\frac{\beta(1-\lambda_1)-1}{\lambda_1 r_1}}, \ldots, \sqrt{\frac{\beta(1-\lambda_k)-1}{\lambda_k r_k}}, 0, \ldots, 0\right] \tag{13}$$

where $\{\lambda_1, \ldots, \lambda_k\}$ are $k \leq n_x$ eigenvalues of $\Sigma_{x|y}\Sigma_x^{-1}$ with critical β values $\beta^c{}_1, \ldots, \beta^c{}_k \leq \beta$. $r_i \equiv \mathbf{v}_i^T \Sigma_x \mathbf{v}_i$ as in Theorem 3.1.

The proof is presented in Appendix D.

We have thus characterized the set of all minima of $\mathcal{L}$, and turn to identify which of them achieve the global minima.

Corollary 4.2
The global minimum of $\mathcal{L}$ is obtained with all λ_i that satisfy $\lambda_i < \frac{\beta-1}{\beta}$.

The proof is presented in Appendix D.

Taken together, these observations prove that for a given value of β, the optimal projection is obtained by taking all the eigenvectors whose eigenvalues λ_i satisfy $\beta \geq \frac{1}{1-\lambda_i}$, and setting their norm according to $A = WV$ with W determined as in Lemma 4.1. This completes the proof of Theorem 3.1.

5. The GIB Information Curve

The information bottleneck is targeted at characterizing the tradeoff between information preservation (accuracy of relevant predictions) and compression. Interestingly, much of the structure of the problem is reflected in the *information curve*, namely, the maximal value of relevant preserved information (accuracy), $I(T;Y)$, as function of the complexity of the representation of X, measured by $I(T;X)$. This curve is related to the rate-distortion function in lossy source coding, as well as to the achievability limit in source coding with side-information (Wyner, 1975; Cover and Thomas, 1991). It was shown to be concave under general conditions (Gilad-Bachrach et al., 2003), but its precise functional form depends on the joint distribution and can reveal properties of the hidden structure of the variables. Analytic forms for the information curve are known only for very special cases, such as Bernoulli variables and some intriguing self-similar distributions. The analytic characterization of the Gaussian IB problem allows us to obtain a closed form expression for the information curve in terms of the relevant eigenvalues.

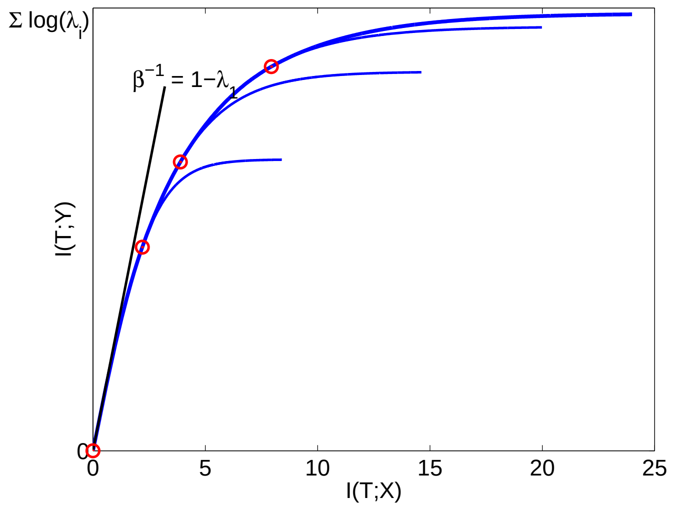


Figure 3: GIB information curve obtained with four eigenvalues $\lambda_i = 0.1, 0.5, 0.7, 0.9$. The information at the critical points are designated by circles. For infinite β, curve is saturated at the log of the determinant $\sum \log \lambda_i$. For comparison, information curves calculated with smaller number of eigenvectors are also depicted (all curves calculated for $\beta < 1000$). The slope of the un-normalized curve at each point is the corresponding β^{-1}. The tangent at zero, with slope $\beta^{-1} = 1 - \lambda_1$, is super imposed on the information curve.

To this end, we substitute the optimal projection $A(\beta)$ into $I(T;X)$ and rewrite them as a function of β

$$\begin{aligned} I_\beta(T;X) &= \frac{1}{2}\log\left(|A\Sigma_x A^T + I_d|\right) \qquad (14) \\ &= \frac{1}{2}\log\left(|(\beta(I-D)-I)D^{-1}|\right) \\ &= \frac{1}{2}\sum_{i=1}^{n(\beta)}\log\left((\beta-1)\frac{1-\lambda_i}{\lambda_i}\right) \\ I_\beta(T;Y) &= I(T;X) - \frac{1}{2}\sum_{i=1}^{n(\beta)}\log\beta(1-\lambda_i), \end{aligned}$$

where D is a diagonal matrix whose entries are the eigenvalues of $\Sigma_{x|y}\Sigma_x^{-1}$ as in Appendix D, and $n(\beta)$ is the maximal index i such that $\beta \geq \frac{1}{1-\lambda_i}$. Isolating β as a function of $I_\beta(T;X)$ in the correct range of n_β and then $I_\beta(T;Y)$ as a function of $I_\beta(T;X)$ we have

$$I(T;Y) = I(T;X) - \frac{n_I}{2}\log\left(\prod_{i=1}^{n_I}(1-\lambda_i)^{\frac{1}{n_I}} + e^{\frac{2I(T;X)}{n_I}}\prod_{i=1}^{n_I}\lambda_i^{\frac{1}{n_I}}\right) \qquad (15)$$

where the products are over the *first* $n_I = n_{\beta(I(T;X))}$ eigenvalues, since these obey the critical β condition, with $c_{n_I} \leq I(T;X) \leq c_{n_I+1}$ and $c_{n_I} = \sum_{i=1}^{n_I-1}\log\frac{\lambda_{n_I}}{\lambda_i}\frac{1-\lambda_i}{1-\lambda_{n_I}}$.

The GIB curve, illustrated in Figure 3, is continuous and smooth, but is built of several segments: as $I(T;X)$ increases additional eigenvectors are used in the projection. The derivative of the curve, which is equal to β^{-1}, can be easily shown to be continuous and decreasing, therefore the information curve is concave everywhere, in agreement with the general concavity of information curve in the discrete case (Wyner, 1975; Gilad-Bachrach et al., 2003). Unlike the discrete case where concavity proofs rely on the ability to use a large number of clusters, concavity is guaranteed here also for segments of the curve, where the number of eigenvectors are limited a-priori.

At each value of $I(T;X)$ the curve is bounded by a tangent with a slope $\beta^{-1}(I(T;X))$. Generally in IB, the data processing inequality yields an upper bound on the slope at the origin, $\beta^{-1}(0) < 1$, in GIB we obtain a tighter bound: $\beta^{-1}(0) < 1 - \lambda_1$. The asymptotic slope of the curve is always zero, as $\beta \to \infty$, reflecting the law of diminishing return: adding more bits to the description of X does not provide higher accuracy about T. This relation between the spectral properties of the covariance matrices raises interesting questions for special cases where the spectrum can be better characterized, such as random-walks and self-similar processes.

6. An Iterative Algorithm

The GIB solution is a set of scaled eigenvectors, and as such can be calculated using standard techniques. For example gradient ascent methods were suggested for learning CCA (Becker, 1996; Borga et al., 1997). An alternative approach is to use the general iterative algorithm for IB problems (Tishby et al., 1999). This algorithm that can be extended to continuous variables and representations, but its practical application for arbitrary distributions leads to a non-linear generalized eigenvalue problem whose general solution can be difficult. It is therefore interesting to explore the form that the iterative algorithm assumes once it is applied to Gaussian variables. Moreover, it may be possible to later extend this approach to more general parametric distributions, such as general exponential forms, for which linear eigenvector methods may no longer be adequate.

The general conditions for the IB stationary points were presented by Tishby et al. (1999) and can be written for a continuous variable x by the following self consistent equations for the unknown distributions $p(t|x)$, $p(y|t)$ and $p(t)$:

$$\begin{aligned} p(t) &= \int_X dx\, p(x)p(t|x) \qquad (16) \\ p(y|t) &= \frac{1}{p(t)} \int_X dx\, p(x,y)p(t|x) \\ p(t|x) &= \frac{p(t)}{Z(\beta)} e^{-\beta D_{KL}[p(y|x)||p(y|t)]} \end{aligned}$$

where $Z(\beta)$ is a normalization factor (partition function) and is independent of x. It is important to realize that those conditions assume nothing about the representation variable T and should be satisfied by *any* fixed point of the IB Lagrangian. When X, Y and T have finite cardinality, those equations can be iterated directly in a Blahut-Arimoto like algorithm,

$$\begin{aligned} p(t_{k+1}|x) &= \frac{p(t_k)}{Z_{k+1}(x,\beta)} e^{-\beta D_{KL}[p(y|x)||p(y|t_k)]} \\ p(t_{k+1}) &= \int_X dx\, p(x) p(t_{k+1}|x) \\ p(y|t_{k+1}) &= \frac{1}{p(t_{k+1})} \int_X dx\, p(x,y) p(t_{k+1}|x). \end{aligned} \tag{17}$$

where each iteration results in a distribution over the variables T_k, X and Y. The second and third equations calculate $p(t_{k+1})$ and $p(y|t_{k+1})$ using standard marginalization, and the Markov property $Y - X - T_k$. These iterations were shown to converge to the optimal T by Tishby et al. (1999).

For the general continuous T such an iterative algorithm is clearly not feasible. We show here, how the fact that we are confined to Gaussian distributions, can be used to turn those equations into an efficient parameter updating algorithm. We conjecture that algorithms for parameters optimizations can be defined also for parametric distribution other than Gaussians, such as other exponential distributions that can be efficiently represented with a small number of parameters.

In the case of Gaussian $p(x,y)$, when $p(t_k|x)$ is Gaussian for some k, so are $p(t_k), p(y|t_k)$ and $p(t_{k+1}|x)$. In other words, the set of Gaussians $p(t|x)$ is invariant under the above iterations. To see why this is true, notice that $p(y|t_k)$ is Gaussian since T_k is jointly Gaussian with X. Also, $p(t_{k+1}|x)$ is Gaussian since $D_{KL}[p(y|x)||p(y|t_k)]$ between two Gaussians contains only second order moments in y and t and thus its exponential is Gaussian. This is in agreement with the general fact that the optima (which are fixed points of 17) are Gaussian (Globerson and Tishby, 2004). This invariance allows us to turn the IB algorithm that iterates over distributions, into an algorithm that iterates over the parameters of the distributions, being the relevant degrees of freedom in the problem.

Denote the variable T at time k by $T_k = A_k X + \xi_k$, where $\xi_k \sim \mathcal{N}(0, \Sigma_{\xi_k})$. The parameters A and Σ at time $k+1$ can be obtained by substituting T_k in the iterative IB equations. As shown in Appendix E, this yields the following update equations

$$\begin{aligned} \Sigma_{\xi_{k+1}} &= \left(\beta \Sigma_{t_k|y}^{-1} - (\beta - 1)\Sigma_{t_k}^{-1}\right)^{-1} \\ A_{k+1} &= \beta \Sigma_{\xi_{k+1}} \Sigma_{t_k|y}^{-1} A_k \left(I - \Sigma_{y|x}\Sigma_x^{-1}\right) \end{aligned} \tag{18}$$

where $\Sigma_{t_k|y}, \Sigma_{t_k}$ are the covariance matrices calculated for the variable T_k.

This algorithm can be interpreted as repeated projection of A_k on the matrix $I - \Sigma_{y|x}\Sigma_x^{-1}$ (whose eigenvectors we seek) followed by scaling with $\beta \Sigma_{\xi_{k+1}} \Sigma_{t_k|y}^{-1}$. It thus has similar form to the power method for calculating the dominant eigenvectors of the matrix $\Sigma_{y|x}\Sigma_x^{-1}$ (Demmel, 1997; Golub and Loan, 1989). However, unlike the naive power method, where only the single dominant eigenvector is preserved, the GIB iterative algorithm maintains several different eigenvectors, and their number is determined by the continuous parameter β and emerges from the iterations: All eigenvectors whose eigenvalues are smaller than the critical β vanish to zero, while the rest are properly scaled. This is similar to an extension of the naive power method known as *Orthogonal Iteration*, in which the projected vectors are renormalized to maintain several non vanishing vectors (Jennings and Stewart, 1975).

Figure 4 demonstrates the operation of the iterative algorithm for a four dimensional X and Y. The tradeoff parameter β was set to a value that leads to two vanishing eigenvectors. The norm of the other two eigenvectors converges to the correct values, which are given in Theorem 3.1.

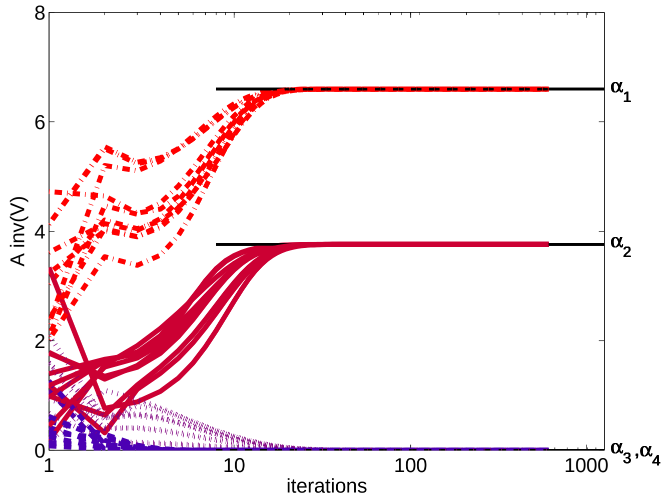


Figure 4: The norm of projection on the four eigenvectors of $\Sigma_{x|y}\Sigma_x^{-1}$, as evolves along the operation of the iterative algorithm. Each line corresponds to the length of the projection of one row of A on the closest eigenvector. The projection on the other eigenvectors also vanishes (not shown). β was set to a value that leads to two non vanishing eigenvectors. The algorithm was repeated 10 times with different random initialization points, showing that it converges within 20 steps to the correct values α_i.

The iterative algorithm can also be interpreted as a regression of X on T via Y. This can be seen by writing the update equation for A_{k+1} as

$$A_{k+1} = \Sigma_{\xi_{k+1}}\Sigma_{t_k|y}^{-1}\left(\Sigma_{yt_k}\Sigma_y^{-1}\right)\left(\Sigma_{yx}\Sigma_x^{-1}\right). \tag{19}$$

Since $\Sigma_{yx}\Sigma_x^{-1}$ describes the optimal linear regressor of X on Y, the operation of A_{k+1} on X can be described by the following diagram

$$X \xrightarrow{\Sigma_{yx}\Sigma_x^{-1}} \mu_{y|x} \xrightarrow{\Sigma_{yt_k}\Sigma_y^{-1}} \mu_{t_k|\mu_{y|x}} \xrightarrow{\Sigma_{\xi_{k+1}}\Sigma_{t_k|y}^{-1}} T_{k+1} \tag{20}$$

where the last step scales and normalizes T.

7. Relation To Other Works

The GIB solutions are related to studies of two main types: studies of eigenvalues based co-projections, and information theoretic studies of continuous compression. We review both below.

7.1 Canonical Correlation Analysis and Imax

The Gaussian information bottleneck projection derived above uses weighted eigenvectors of the matrix $\Sigma_{x|y}\Sigma_x^{-1} = I - \Sigma_{xy}\Sigma_y^{-1}\Sigma_{yx}\Sigma_x^{-1}$. Such eigenvectors are also used in *canonical correlation analysis* (CCA) (Hotelling, 1935; Thompson, 1984; Borga, 2001), a method of descriptive statistics that finds linear relations between two variables. Given two variables X, Y, CCA finds a set of basis vectors for each variable, such that the correlation coefficient between the projection of the variables on the basis vectors is maximized. In other words, it finds the bases in which the correlation matrix is diagonal and the correlations on the diagonal are maximized. The bases are the eigenvectors of the matrices $\Sigma_y^{-1}\Sigma_{yx}\Sigma_x^{-1}\Sigma_{xy}$ and $\Sigma_x^{-1}\Sigma_{xy}\Sigma_y^{-1}\Sigma_{yx}$, and the square roots of their corresponding eigenvalues are the *canonical correlation coefficients*. CCA was also shown to be a special case of continuous Imax (Becker and Hinton, 1992; Becker, 1996), when the Imax networks are limited to linear projections.

Although GIB and CCA involve the spectral analysis of the same matrices, they have some inherent differences. First of all, GIB characterizes not only the eigenvectors but also their norm, in a way that that depends on the trade-off parameter β. Since CCA depends on the correlation coefficient between the compressed (projected) versions of X and Y, which is a *normalized* measure of correlation, it is invariant to a rescaling of the projection vectors. In contrast, for any value of β, GIB will choose one particular rescaling given by Theorem 3.1.

While CCA is symmetric (in the sense that both X and Y are projected), IB is non symmetric and only the X variable is compressed. It is therefore interesting that both GIB and CCA use the same eigenvectors for the projection of X.

7.2 Multiterminal Information Theory

The information bottleneck formalism was recently shown (Gilad-Bachrach et al., 2003) to be closely related to the problem of source coding with side information (Wyner, 1975). In the latter, two *discrete* variables X, Y are encoded separately at rates R_x, R_y, and the aim is to use them to perfectly reconstruct Y. The bounds on the achievable rates in this case were found in (Wyner, 1975) and can be obtained from the IB information curve.

When considering continuous variables, lossless compression at finite rates is no longer possible. Thus, mutual information for continuous variables is no longer interpretable in terms of the actual number of encoding bits, but rather serves as an optimal measure of information between variables. The IB formalism, although coinciding with coding theorems in the discrete case, is more general in the sense that it reflects the tradeoff between compression and information preservation, and is not concerned with exact reconstruction.

Lossy reconstruction can be considered by introducing distortion measures as done for source coding of Gaussians with side information by Wyner (1978) and by Berger and Zamir (1999) (see also Pradhan, 1998), but these focus on the region of achievable rates under constrained distortion and are not relevant for the question of finding the representations which capture the information between the variables. Among these, the formalism closest to ours is that of Berger and Zamir (1999) where the distortion in reconstructing X is assumed to be small (high-resolution scenario). However, their results refer to encoding rates and as such go to infinity as the distortion goes to zero. They also analyze the problem for scalar Gaussian variables, but the one-dimensional setting does not reveal the interesting spectral properties and phase transitions which appear only in the multidimensional case discussed here.

7.3 Gaussian IB with Side Information

When handling real world data, the relevance variable Y often contains multiple structures that are correlated to X, although many of them are actually irrelevant. The information bottleneck with side information (*IBSI*) (Chechik and Tishby, 2002) alleviates this problem using side information in the form of an *irrelevance* variable Y^- about which information is removed. *IBSI* thus aims to minimize

$$\mathcal{L} = I(X;T) - \beta\left(I(T;Y^+) - \gamma I(T;Y^-)\right) \tag{21}$$

This formulation can also be extended to the Gaussian case, in a manner similar to the original GIB functional. Looking at its derivative with respect to the projection A yields

$$\begin{aligned}\frac{\delta\mathcal{L}}{\delta A} = (\quad 1-\beta+\beta\gamma \quad &)(A\Sigma_x A^T + I_d)^{-1} 2A\Sigma_x \\ + \qquad \beta \qquad & \;(A\Sigma_{x|y^+}A^T + I_d)^{-1} 2A\Sigma_{x|y^+} \\ - \qquad \beta\gamma \qquad & \;(A\Sigma_{x|y^-}A^T + I_d)^{-1} 2A\Sigma_{x|y^-}.\end{aligned}$$

While *GIB* relates to an eigenvalue problem of the form $\lambda A = A\Sigma_{x|y}\Sigma_x^{-1}$, GIB with side information (*GIBSI*) requires to solve of a matrix equation of the form $\lambda' A + \lambda^+ A\Sigma_{x|y^+}\Sigma_x^{-1} = \lambda^- A\Sigma_{x|y^-}\Sigma_x^{-1}$, which is similar in form to a generalized eigenvalue problem. However, unlike standard generalized eigenvalue problems, but as in the GIB case analyzed in this paper, the eigenvalues themselves depend on the projection A.

8. Practical Implications

The GIB approach can be viewed as a method for finding the best linear projection of X, under a constraint on $I(T;X)$. Another straightforward way to limit the complexity of the projection is to specify its dimension in advance. Such an approach leaves open the question of the relative weighting of the resulting eigenvectors. This is the approach taken in classical CCA, where the number of eigenvectors is determined according to a statistical significance test, and their weights are then set to $\sqrt{1-\lambda_i}$. This expression is the correlation coefficient between the i^{th} CCA projections on X and Y, and reflects the amount of correlation captured by the i^{th} projection. The GIB weighting scheme is different, since it is derived to preserve maximum information under the compression constraint. To illustrate the difference, consider the case where $\beta = \frac{1}{1-\lambda_3}$, so that only two eigenvectors are used by GIB. The CCA scaling in this case is $\sqrt{1-\lambda_1}$, and $\sqrt{1-\lambda_2}$. The GIB weights are (up to a constant) $\alpha_1 = \sqrt{\frac{\lambda_3-\lambda_1}{\lambda_1 r_1}}, \alpha_2 = \sqrt{\frac{\lambda_3-\lambda_2}{\lambda_2 r_2}}$,, which emphasizes large gaps in the eigenspectrum, and can be very different from the CCA scaling.

This difference between CCA scaling and GIB scaling may have implications on two aspects of learning in practical applications. First, in applications involving compression of Gaussian signals due to limitation on available band-width. This is the case in the growing field of sensor networks in which sensors are often very limited in their communication bandwidth due to energy constraints. In these networks, sensors communicate with other sensors and transmit information about their local measurements. For example, sensors can be used to monitor chemicals' concentrations, temperature or light conditions. Since only few bits can be transmitted, the information has to be compressed in a relevant way, and the relative scaling of the different eigenvectors becomes important (as in transform coding Goyal, 2001). As shown above, GIB describes the optimal transformation of the raw data into information conserving representation.

The second aspect where GIB becomes useful is in interpretation of data. Today, canonical correlation analysis is widely used for finding relations between multi-variate continuous variables, in particular in domains which are inherently high dimensional such as meteorology (von Storch and Zwiers, 1999) chemometrics (Antti et al., 2002) and functional MRI of brains (Friman et al., 2003). Since GIB weights the eigenvectors of the normalized cross correlation matrix in a different way than CCA, it may lead to very different interpretation of the relative importance of factors in these studies.

9. Discussion

We applied the information bottleneck method to continuous jointly Gaussian variables X and Y, with a continuous representation of the compressed variable T. We derived an analytic optimal solution as well as a new general algorithm for this problem (GIB) which is based solely on the spectral properties of the covariance matrices in the problem. The solutions for GIB are characterized in terms of the trade-off parameter β between compression and preserved relevant information, and consist of eigenvectors of the matrix $\Sigma_{x|y}\Sigma_x^{-1}$, continuously adding up vectors as more complex models are allowed. We provide an analytic characterization of the optimal tradeoff between the representation complexity and accuracy - the “information curve” - which relates the spectrum to relevant information in an intriguing manner. Besides its clean analytic structure, GIB offers a way for analyzing empirical multivariate data when only its correlation matrices can be estimated. In that case it extends and provides new information theoretic insight to the classical canonical correlation analysis.

The most intriguing aspect of GIB is in the way the dimensionality of the representation changes with increasing complexity and accuracy, through the continuous value of the trade-off parameter β. While both mutual information values vary continuously on the smooth information curve, the dimensionality of the optimal projection T increases discontinuously through a cascade of structural (second order) phase transitions, and the optimal curve moves from one analytic segment to another. While this transition cascade is similar to the bifurcations observed in the application of IB to clustering through deterministic annealing, this is the first time such dimensional transitions are shown to exist in this context. The ability to deal with all possible dimensions in a single algorithm is a novel advantage of this approach compared to similar linear statistical techniques as CCA and other regression and association methods.

Interestingly, we show how the general IB algorithm which iterates over distributions, can be transformed to an algorithm that performs iterations over the distributions' *parameters*. This algorithm, similar to multi-eigenvector power methods, converges to a solution in which the number of eigenvectors is determined by the parameter β, in a way that emerges from the iterations rather than defined a-priori.

For multinomial variables, the IB framework can be shown to be related in some limiting cases to maximum-likelihood estimation in a latent variable model (Slonim and Weiss, 2002). It would be interesting to see whether the GIB-CCA equivalence can be extended and give a more general understanding of the relation between IB and statistical latent variable models.

While the restriction to a Gaussian joint distribution deviates from the more general distribution independent approach of IB, it provides a precise example to the way representations with different dimensions can appear in the more general case. We believe that this type of dimensionality-transitions appears for more general distributions, as can be revealed in some cases by applying the Laplace method of integration (a Gaussian approximation) to the integrals in the general IB algorithm for continuous T.

The more general exponential forms, can be considered as a kernelized version of IB (see Mika et al., 2000) and appear in other minimum-information methods (such as SDR, Globerson and Tishby, 2003). these are of particular interest here, as they behave like Gaussian distributions in the joint kernel space. The Kernel Fisher-matrix in this case will take the role of the original cross covariance matrix of the variables in GIB.

Another interesting extension of our work is to networks of Gaussian processes. A general framework for that problem was developed in Friedman et al. (2001) and applied for discrete variables. In this framework the mutual information is replaced by multi-information, and the dependencies of the compressed and relevance variables is specified through two Graphical models. It is interesting to explore the effects of dimensionality changes in this more general framework, to study how they induce topological transitions in the related graphical models, as some edges of the graphs become important only beyond corresponding critical values of the tradeoff parameter β.

Acknowledgments

G. C. and A. G. were supported by the Israeli Ministry of Science, the Eshkol Foundation. This work was partially supported by a center of excellence grant of the Israeli Science Foundation (ISF).

Appendix A. Invariance to the Noise Covariance Matrix

Lemma A.1 *For every pair (A,Σ_ξ) of a projection A and a full rank covariance matrix Σ_ξ, there exist a matrix $\tilde{A}$ such that $\mathcal{L}(\tilde{A},I_d) = \mathcal{L}(A,\Sigma_\xi)$, where I_d is the $n_t \times n_t$ identity matrix.*

Proof: Denote by V the matrix which diagonalizes Σ_ξ, namely $\Sigma_\xi = VDV^T$, and by c the determinant $c \equiv |\sqrt{D^{-1}}V| = |\sqrt{D^{-1}}V^T|$. Setting $\tilde{A} \equiv \sqrt{D^{-1}}VA$ we have

$$\begin{aligned}
\mathcal{L}(\tilde{A},I) &= (1-\beta)\log(|\tilde{A}\Sigma_x\tilde{A}^T + I_d|) - \log(|I_d|) + \beta\log(|\tilde{A}\Sigma_{x|y}\tilde{A}^T + I_d|) \qquad (22)\\
&= (1-\beta)\log(c|A\Sigma_x A^T + \Sigma_\xi|c) - \log(c|\Sigma_\xi|c) + \beta\log(c|A\Sigma_{x|y}A^T + \Sigma_\xi|c)\\
&= (1-\beta)\log(|A\Sigma_x A^T + \Sigma_\xi|) - \log(|\Sigma_\xi|) + \beta\log(|A\Sigma_{x|y}A^T + \Sigma_\xi|)\\
&= \mathcal{L}(A,\Sigma_\xi)
\end{aligned}$$

where the first equality stems from the fact that the determinant of a matrix product is the product of the determinants. □

Appendix B. Properties of Eigenvalues of $\Sigma_{x|y}\Sigma_x^{-1}$ and Σ_x

Lemma B.1 *Denote the set of left normalized eigenvectors of $\Sigma_{x|y}\Sigma_x^{-1}$ by $\mathbf{v}_i$ ($||\mathbf{v}_i|| = 1$) and their corresponding eigenvalues by λ_i. Then*

1. *All the eigenvalues are real and satisfy $0 \le \lambda_i \le 1$*
2. *$\exists r_i > 0$ s.t. $\mathbf{v}_i^T\Sigma_x\mathbf{v}_j = \delta_{ij}r_i$.*
3. *$\mathbf{v}_i^T\Sigma_{x|y}\mathbf{v}_j = \delta_{ij}\lambda_i r_i$.*

The proof is standard (see e.g. Golub and Loan, 1989) and is brought here for completeness.
Proof:

1. The matrices $\Sigma_{x|y}\Sigma_x^{-1}$ and $\Sigma_{xy}\Sigma_y^{-1}\Sigma_{yx}\Sigma_x^{-1}$ are positive semi definite (PSD), and their eigenvalues are therefore positive. Since $\Sigma_{x|y}\Sigma_x^{-1} = I - \Sigma_{xy}\Sigma_y^{-1}\Sigma_{yx}\Sigma_x^{-1}$, the eigenvalues of $\Sigma_{x|y}\Sigma_x^{-1}$ are bounded between 0 and 1.

2. Denote by V the matrix whose rows are $\mathbf{v}_i^T$. The matrix $V\Sigma_x^{\frac{1}{2}}$ is the eigenvector matrix of $\Sigma_x^{-\frac{1}{2}}\Sigma_{x|y}\Sigma_x^{-\frac{1}{2}}$ since $\left(V\Sigma_x^{\frac{1}{2}}\right)\Sigma_x^{-\frac{1}{2}}\Sigma_{x|y}\Sigma_x^{-\frac{1}{2}} = V\Sigma_{x|y}\Sigma_x^{-\frac{1}{2}} = \left(V\Sigma_{x|y}\Sigma_x^{-1}\right)\Sigma_x^{\frac{1}{2}} = DV\Sigma_x^{\frac{1}{2}}$. From the fact that $\Sigma_x^{-\frac{1}{2}}\Sigma_{x|y}\Sigma_x^{-\frac{1}{2}}$ is symmetric, $V\Sigma_x^{\frac{1}{2}}$ is orthogonal, and thus $V\Sigma_x V^T$ is diagonal.

3. Follows from 2: $\mathbf{v}_i^T\Sigma_{x|y}\Sigma_x^{-1}\Sigma_x\mathbf{v}_j = \lambda_i\mathbf{v}_i^T\Sigma_x\mathbf{v}_j = \lambda_i\delta_{ij}r_i$.

□

Appendix C. Optimal Eigenvector

For some β values, several eigenvectors can satisfy the conditions for non degenerated solutions (Equation 10). To identify the optimal eigenvector, we substitute the value of $||A||^2$ from Equation (9) $A\Sigma_{x|y}A^T = r\lambda||A||^2$ and $A\Sigma_x A^T = r||A||^2$ into the target function $\mathcal{L}$ of Equation (6), and obtain

$$\mathcal{L} = (1-\beta)\log\left(\frac{(1-\lambda)(\beta-1)}{\lambda}\right) + \beta\log\left(\beta(1-\lambda)\right). \tag{23}$$

Since $\beta \geq 1$, this is monotonically increasing in λ and is minimized by the eigenvector of $\Sigma_{x|y}\Sigma_x^{-1}$ with the smallest eigenvalue. Note that this is also the eigenvector of $\Sigma_{xy}\Sigma_y^{-1}\Sigma_{yx}\Sigma_x^{-1}$ with the largest eigenvalue.

Appendix D. Optimal Mixing Matrix

Lemma D.1 *The optimum of the cost function is obtained with a diagonal mixing matrix W of the form*

$$W = diag\left[\sqrt{\frac{\beta(1-\lambda_1)-1}{\lambda_1 r_1}}, \dots, \sqrt{\frac{\beta(1-\lambda_k)-1}{\lambda_k r_k}}, 0, \dots, 0\right] \tag{24}$$

where $\{\lambda_1, \dots, \lambda_k\}$ are $k \leq n_x$ eigenvalues of $\Sigma_{x|y}\Sigma_x^{-1}$ with critical β values ${\beta^c}_1, \dots, {\beta^c}_k \leq \beta$. $r_i \equiv \mathbf{v}_i^T\Sigma_x\mathbf{v}_i$ as in Theorem 3.1.

Proof: We write $V\Sigma_{x|y}\Sigma_x^{-1} = DV$ where D is a diagonal matrix whose elements are the corresponding eigenvalues, and denote by R the diagonal matrix whose i^{th} element is r_i. When $k = n_x$, we substitute $A = WV$ into Equation (12), and eliminate V from both sides to obtain

$$\frac{\beta-1}{\beta}\left[(WDRW^T + I_d)(WRW^T + I_d)^{-1}\right]W = WD$$

Use the fact that W is full rank to multiply by W^{-1} from the left and by $W^{-1}(WRW^T + I_d)W$ from the right

$$\frac{\beta-1}{\beta}(DRW^TW + I_d) = D(RW^TW + I_d)$$

Rearranging, we have

$$W^T W = [\beta(I-D)-I](DR)^{-1}, \tag{25}$$

which is a diagonal matrix.

While this does not uniquely characterize W, we note that using properties of the eigenvalues from lemma B.1, we obtain

$$|A\Sigma_x A^T + I_d| \quad = \quad |WV\Sigma_x V^T W^T + I_d| = |WRW^T + I_d|.$$

Note that WRW^T has left eigenvectors W^T with corresponding eigenvalues obtained from the diagonal matrix $W^T WR$. Thus if we substitute A into the target function in Equation (6), a similar calculation yields

$$\mathcal{L} = (1-\beta)\sum_{i=1}^{n}\log\left(||\mathbf{w}_i^T||^2 r_i + 1\right) + \beta\sum_{i=1}^{n}\log\left(||\mathbf{w}_i^T||^2 r_i\lambda_i + 1\right) \tag{26}$$

where $||\mathbf{w}_i^T||^2$ is the i^{th} element of the diagonal of $W^T W$. This shows that $\mathcal{L}$ depends only on the norm of the columns of W, and all matrices W that satisfy (25) yield the same target function. We can therefore choose to take W to be the diagonal matrix which is the (matrix) square root of (25)

$$W = \sqrt{[\beta(I-D)-I](DR)^{-1}} \tag{27}$$

which completes the proof of the full rank ($k = n_x$) case.

In the low rank ($k < n_x$) case W does not mix all the eigenvectors, but only k of them. To prove the lemma for this case, we first show that any such low rank matrix is equivalent (in terms of the target function value) to a low rank matrix that has only k non zero rows. We then conclude that the non zero rows should follow the form described in the above lemma.

Consider a $n_x \times n_x$ matrix W of rank $k < n_x$, but without any zero rows. Let U be the set of left eigenvectors of WW^T (that is, $WW^T = U\Lambda U^T$). Then, since WW^T is Hermitian, its eigenvectors are orthonormal, thus $(UW)(WU)^T = \Lambda$ and $W' = UW$ is a matrix with k non zero rows and $n_x - k$ zero lines. Furthermore, W' obtains the same value of the target function, since

$$\begin{aligned}
\mathcal{L} &= (1–\beta)\log(|W'RW'^T + \Sigma_\xi^2|) + \beta\log(|W'DRW'^T + \Sigma_\xi^2|) \qquad (28)\\
&= (1–\beta)\log(|UWRW^T U^T + UU^T\Sigma_\xi^2|) + \beta\log(|UWDRW^T U^T + UU^T\Sigma_\xi^2|)\\
&= (1–\beta)\log(|U||WRW^T + \Sigma_\xi^2||U^T|) + \beta\log(|U||UWDRW^T U^T + \Sigma_\xi^2||U^T|)\\
&= (1–\beta)\log(|WRW^T + \Sigma_\xi^2|) + \beta\log(|WDRW^T T + \Sigma_\xi^2|),
\end{aligned}$$

where we have used the fact that U is orthonormal and hence $|U| = 1$. To complete the proof note that the non zero rows of W' also have $n_x - k$ zero columns and thus define a square matrix of rank k, for which the proof of the full rank case apply, but this time by projecting to a dimension k instead of n_x. □

This provides a characterization of all local minima. To find which is the global minimum, we prove the following corollary.

Corollary D.2
The global minimum of $\mathcal{L}$ is obtained with all λ_i that satisfy $\lambda_i < \frac{\beta-1}{\beta}$.

Proof: Substituting the optimal W of Equation (27) into Equation (26) yields

$$\mathcal{L} = \sum_{i=1}^{k} (\beta - 1)\log\lambda_i + \log(1-\lambda_i) + f(\beta). \tag{29}$$

Since $0 \le \lambda \le 1$ and $\beta \ge \frac{1}{1-\lambda}$, $\mathcal{L}$ is minimized by taking all the eigenvalues that satisfy $\beta > \frac{1}{(1-\lambda_i)}$.
□

Appendix E. Deriving the Iterative Algorithm

To derive the iterative algorithm in Section 6, we assume that the distribution $p(t_k|x)$ corresponds to the Gaussian variable $T_k = A_k X + \xi_k$. We show below that $p(t_{k+1}|x)$ corresponds to $T_{k+1} = A_{k+1}X + \xi_{k+1}$ with $\xi_{k+1} \sim N(0, \Sigma_{\xi_{k+1}})$ and

$$\begin{aligned} \Sigma_{\xi_{k+1}} &= \left(\beta\Sigma_{t_k|y}^{-1} - (\beta-1)\Sigma_{t_k}^{-1}\right)^{-1} \\ A_{k+1} &= \beta\Sigma_{\xi_{k+1}}\Sigma_{t_k|y}^{-1}A_k\left(I - \Sigma_{y|x}\Sigma_x^{-1}\right) \end{aligned} \tag{30}$$

We first substitute the Gaussian $p(t_k|x) \sim N(A_k x, \Sigma_{\xi_k})$ into the equations of (17), and treat the second and third equations. The second equation $p(t_k) = \int_x p(x)p(t_k|x)dx$, is a marginal of the Gaussian $T_k = A_k X + \xi_k$, and yields a Gaussian $p(t_k)$ with zero mean and covariance

$$\Sigma_{t_k} = A_k\Sigma_x A_k^T + \Sigma_{\xi_k} \tag{31}$$

The third equation, $p(y|t_k) = \frac{1}{p(t_k)}\int_x p(x,y)p(t_k|x)dx$ defines a Gaussian with mean and covariance matrix given by:

$$\begin{aligned} \mu_{y|t_k} &= \mu_y + \Sigma_{yt_k}\Sigma_{t_k}^{-1}(t_k - \mu_{t_k}) = \Sigma_{yt_k}\Sigma_{t_k}^{-1}t_k \equiv B_k t_k \\ \Sigma_{y|t_k} &= \Sigma_y - \Sigma_{yt_k}\Sigma_{t_k}^{-1}\Sigma_{t_k y} = \Sigma_y - A_k\Sigma_{xy}\Sigma_{t_k}^{-1}\Sigma_{yx}A_k^T \end{aligned} \tag{32}$$

where we have used the fact that $\mu_y = \mu_{t_k} = 0$, and define the matrix $B_k \equiv \Sigma_{yt_k}\Sigma_{t_k}^{-1}$ as the regressor of t_k on y. Finally, we return to the first equation of (17), that defines $p(t_{k+1}|x)$ as

$$p(t_{k+1}|x) = \frac{p(t_k)}{Z(x,\beta)}e^{-\beta D_{KL}[p(y|x)||p(y|t_k)]}. \tag{33}$$

We now show that $p(t_{k+1}|x)$ is Gaussian and compute its mean and covariance matrix.

The KL divergence between the two Gaussian distributions, in the exponent of Equation (33) is known to be

$$\begin{aligned} 2D_{KL}[p(y|x)||p(y|t_k)] &= \log\frac{|\Sigma_{y|t_k}|}{|\Sigma_{y|x}|} + Tr(\Sigma_{y|t_k}^{-1}\Sigma_{y|x}) \\ &+ (\mu_{y|x} - \mu_{y|t_k})^T\Sigma_{y|t_k}^{-1}(\mu_{y|x} - \mu_{y|t_k}). \end{aligned} \tag{34}$$

The only factor which explicitly depends on the value of t in the above expression is $\mu_{y|t_k}$ derived in Equation (32), is linear in t. The KL divergence can thus be rewritten as

$$D_{KL}[p(y|x)||p(y|t_k)] = c(x) + \frac{1}{2}(\mu_{y|x} - B_k t_k)^T\Sigma_{y|t_k}^{-1}(\mu_{y|x} - B_k t_k).$$

Adding the fact that $p(t_k)$ is Gaussian we can write the log of Equation (33) as a quadratic form in t:

$$\log p(t_{k+1}|x) = Z(x) + (t_{k+1} - \mu_{t_{k+1}|x})^T \Sigma_{\xi_{k+1}} (t_{k+1} - \mu_{t_{k+1}|x}),$$

where

$$\begin{aligned} \Sigma_{\xi_{k+1}} &= \left(\beta B_k^T \Sigma_{y|t_k}^{-1} B_k + \Sigma_{t_k}^{-1}\right)^{-1} \\ \mu_{t_{k+1}|x} &= A_{k+1} x \\ A_{k+1} &= \beta \Sigma_{\xi_{k+1}} B_k^T \Sigma_{y|t_k}^{-1} \Sigma_{yx} \Sigma_x^{-1} x. \end{aligned} \tag{35}$$

This shows that $p(t_{k+1}|x)$ is a Gaussian $T_{k+1} = A_{k+1}x + \xi_{k+1}$, with $\xi \sim N(0, \Sigma_{\xi_{k+1}})$.

To simplify the form of $A_{k+1}, \Sigma_{\xi_{k+1}}$, we use the two following matrix inversion lemmas,[3] which hold for any matrices E, F, G, H of appropriate sizes when E, H are invertible:

$$\begin{aligned} (E - FH^{-1}G)^{-1} &= E^{-1} + E^{-1}F(H - GE^{-1}F)^{-1}GE^{-1} \\ (E - FH^{-1}G)^{-1}FH^{-1} &= E^{-1}F(H - GE^{-1}F)^{-1}. \end{aligned} \tag{36}$$

Using $E \equiv \Sigma_{t_k}$, $F \equiv \Sigma_{yt_k}$, $H \equiv \Sigma_y$, $G \equiv \Sigma_{yt_k}$, $B_k = \Sigma_{yt_k}\Sigma_{t_k}^{-1}$ in the first lemma we obtain

$$\Sigma_{t_k|y}^{-1} = \Sigma_{t_k}^{-1} + B_k^T \Sigma_{y|t_k}^{-1} B_k.$$

Replacing this into the expression for $\Sigma_{\xi_{k+1}}$ in Equation (35) we obtain

$$\Sigma_{\xi_{k+1}} = \left(\beta \Sigma_{t_k|y}^{-1} - (\beta - 1)\Sigma_{t_k}^{-1}\right)^{-1}. \tag{37}$$

Finally, using again $E \equiv \Sigma_{t_k}$, $F \equiv \Sigma_{t_k y}$, $H \equiv \Sigma_y$, $G \equiv \Sigma_{yt_k}$ in the second matrix lemma, we have $\Sigma_{t_k|y}^{-1}\Sigma_{t_k y}\Sigma_y^{-1} = \Sigma_{t_k}^{-1}\Sigma_{t_k y}\Sigma_{y|t_k}^{-1}$, which turns the expression for A_{k+1} in Equation (35) into

$$\begin{aligned} A_{k+1} &= \beta \Sigma_{\xi_{k+1}} \Sigma_{t_k|y}^{-1} \Sigma_{t_k y} \Sigma_y^{-1} \Sigma_{yx} \Sigma_x^{-1} \\ &= \beta \Sigma_{\xi_{k+1}} \Sigma_{t_k|y}^{-1} A_k \Sigma_{xy} \Sigma_y^{-1} \Sigma_{yx} \Sigma_x^{-1} \\ &= \beta \Sigma_{\xi_{k+1}} \Sigma_{t_k|y}^{-1} A_k (I - \Sigma_{x|y} \Sigma_x^{-1}) , \end{aligned} \tag{38}$$

which completes the derivation of the algorithm as described in (17).

References

H. Antti, E. Holmes, and J. Nicholson. Multivariate solutions to metabonomic profiling and functional genomics. part 2 - chemometric analysis, 2002. http://www.acc.umu.se/ tnkjtg/chemometrics/editorial/oct2002.

S. Becker. Mutual information maximization: Models of cortical self-organization. *Network: Computation in Neural Systems*, pages 7,31, 1996.

3. The first equation is the standard inversion lemma (see e.g., Kay, 1993, page 571). The second can be easily verified from the first.

S. Becker and G. E. Hinton. A self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature*, 355(6356):161–163, 1992.

T. Berger and R. Zamir. A semi-continuous version of the berger-yeung problem. *IEEE Transactions on Information Theory*, pages 1520–1526, 1999.

M. Borga. Canonical correlation: a tutorial. http://people.imt.liu.se/magnus/cca, January 2001.

M. Borga, H. Knutsson, and T. Landelius. Learning canonical correlations. In *Proceedings of the 10th Scandinavian Conference on Image Analysis*, Lappeenranta, Finland, June 1997. SCIA.

G. Chechik and N. Tishby. Extracting relevant structures with side information. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, Vancouver, Canada, 2002.

T. M. Cover and J. A. Thomas. *The elements of information theory*. Plenum Press, New York, 1991.

J. W. Demmel. *Applied numerical linear algebra*. Society for Industrial and Applied Mathematics, Philadelphia, 1997. ISBN 0-89871-389-7.

Alexander G. Dimitrov and John P. Miller. Neural coding and decoding: Communication channels and quantization. *Network: Computation in Neural Systems*, 12(4):441–472, 2001. URL `citeseer.nj.nec.com/dimitrov01neural.html`.

N. Friedman, O. Mosenzon, N. Slonim, and N. Tishby. Multivariate information bottleneck. In J. S. Breese and D. Koller, editors, *Uncertainty in Artificial Intelligence: Proceedings of the Seventeenth Conference (UAI-2001)*, pages 152–161, San Francisco, CA, 2001. Morgan Kaufmann.

O. Friman, M. Borga, P. Lundberg, and H. Knutsson. Adaptive analysis of fMRI data. *NeuroImage*, 19(3):837–845, 2003.

Rimoldi Gastpar and Vetterli. To code, or not to code: lossy source-channel communication revisited. *Information Theory*, 49(5):1147–1158, 2003.

R. Gilad-Bachrach, A. Navot, and N. Tishby. An information theoretic tradeoff between complexity and accuracy. In *Proceedings of the COLT*, Washington, 2003.

A. Globerson and N. Tishby. Sufficient dimensionality reduction. *Journal of Macine Learning Research*, 3:1307–1331, 2003. ISSN 1533-7928.

A. Globerson and N. Tishby. Tbd. Technical report, Hebrew University, January 2004.

G. H. Golub and C. F. V. Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, 1989.

V. K. Goyal. Theoretical foundations of transform coding. *Signal Processing Magazine, IEEE*, 18 (5):9–21, 2001.

H. Hotelling. The most predictable criterion. *Journal of Educational Psychology,*, 26:139–142, 1935.

A. Jennings and G. W. Stewart. Simultaneous iteration for partial eigensolution of real matrices. *J. Inst. Math Appl*, 15:351–361, 1975.

S. M. Kay. *Fundamentals of Statistical Signal Processing Volume I Estimation Theory*. Prentice-Hall, 1993.

R. Linsker. Self-organization in a perceptual network. *Computer*, 21(3):105–117, 1988.

R. Linsker. Local synaptic learning rules suffice to maximize mutual information in a linear network. *Neural Computation*, 4:691–702, 1992.

J. R. Magnus and H. Neudecker. *Matrix Differential Calculus with Applications in Statistics and Econometrics*. John Wiley and Sons, 1988.

S. Mika, G. Ratsch, J. Weston, B. Scholkopf, A. Smola, and K. Muller. Invariant feature extraction and classification in kernel spaces. In S. A. Solla, T. K. Leen, and K. R. Muller, editors, *Advances in Neural Information Processing Systems 12*, pages 526–532, Vancouver, Canada, 2000.

S. S. Pradhan. On rate-distortion function of gaussian sources with memory with side information at the decoder. Technical report, Berkeley, 1998.

S. S. Pradhan, J. Chou, and K. Ramchandran. Duality between source coding and channel coding and its extension to the side information case. *Information Theory*, 49(5):1181–1203, 2003.

C. E. Shannon. Coding theorems for a discrete source with a fidelity criterion. In *Institute for Radio Engineers, International Convention Record*, volume 7, part 4, pages 142–163, New York, NY, USA, March 1959.

J. Sinkkonen and S. Kaski. Clustering based on conditional distributions in an auxiliary space. *Neural Computation*, 14:217–239, 2001.

N. Slonim. *Information Bottleneck theory and applications*. PhD thesis, Hebrew University of Jerusalem, 2003.

N. Slonim and N. Tishby. Document clustering using word clusters via the information bottleneck method. In P. Ingwersen N. J. Belkin and M-K. Leong, editors, *Research and Development in Information Retrieval (SIGIR-00)*, pages 208–215. ACM press, new york, 2000.

N. Slonim and Y. Weiss. Maximum likelihood and the information bottleneck. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, Vancouver, Canada, 2002.

B. Thompson. *Canonical correlation analysis: Uses and interpretation.*, volume 47. Thousands Oak, CA Sage publications, 1984.

N. Tishby, F. C. Pereira, and W. Bialek. The information bottleneck method. In *Proc. of 37th Allerton Conference on communication and computation*, 1999.

H. von Storch and F. W. Zwiers. *Statistical Analysis in Climate Research*. Cambridge University Press, 1999.

A. D. Wyner. On source coding with side information at the decoder. *IEEE Transactions on Information Theory*, IT-21:294–300, 1975.

A. D. Wyner. The rate distortion function for source coding with side information at the decoder ii: General sources. *IEEE Transactions on Information Theory*, IT-38:60–80, 1978.

Journal of Machine Learning Research 6 (2005) 189–210 Submitted 6/03; Revised 7/04; Published 2/05

Multiclass Boosting for Weak Classifiers

Günther Eibl GUENTHER.EIBL@UIBK.AC.AT
Karl–Peter Pfeiffer KARL-PETER.PFEIFFER@UIBK.AC.AT
Department of Biostatistics
University of Innsbruck
Schöpfstrasse 41, 6020 Innsbruck, Austria

Editor: Robert Schapire

Abstract

AdaBoost.M2 is a boosting algorithm designed for multiclass problems with weak base classifiers. The algorithm is designed to minimize a very loose bound on the training error. We propose two alternative boosting algorithms which also minimize bounds on performance measures. These performance measures are not as strongly connected to the expected error as the training error, but the derived bounds are tighter than the bound on the training error of AdaBoost.M2. In experiments the methods have roughly the same performance in minimizing the training and test error rates. The new algorithms have the advantage that the base classifier should minimize the confidence-rated error, whereas for AdaBoost.M2 the base classifier should minimize the pseudo-loss. This makes them more easily applicable to already existing base classifiers. The new algorithms also tend to converge faster than AdaBoost.M2.

Keywords: boosting, multiclass, ensemble, classification, decision stumps

1. Introduction

Most papers about boosting theory consider two-class problems. Multiclass problems can be either reduced to two-class problems using error-correcting codes (Allwein et al., 2000; Dietterrich and Bakiri, 1995; Guruswami and Sahai, 1999) or treated more directly using base classifiers for multiclass problems. Freund and Schapire (1996 and 1997) proposed the algorithm AdaBoost.M1 which is a straightforward generalization of AdaBoost using multiclass base classifiers. An exponential decrease of an upper bound of the training error rate is guaranteed as long as the error rates of the base classifiers are less than 1/2. For more than two labels this condition can be too restrictive for weak classifiers like decision stumps which we use in this paper. Freund and Schapire overcame this problem with the introduction of the pseudo-loss of a classifier $h : X \times Y \rightarrow [0,1]$:

$$\varepsilon_t = \frac{1}{2}\left(1 - h_t(x_i,y_i) + \frac{1}{|Y|-1}\sum_{y \neq y_i} h_t(x_i,y)\right).$$

In the algorithm AdaBoost.M2, each base classifier has to minimize the pseudo-loss instead of the error rate. As long as the pseudo-loss is less than 1/2, which is easily reachable for weak base classifiers as decision stumps, an exponential decrease of an upper bound on the training error rate is guaranteed.

In this paper, we will derive two new direct algorithms for multiclass problems with decision stumps as base classifiers. The first one is called GrPloss and has its origin in the gradient descent framework of Mason et al. (1998, 1999). Combined with ideas of Freund and Schapire (1996, 1997) we get an exponential bound on a performance measure which we call pseudo-loss error. The second algorithm was motivated by the attempt to make AdaBoost.M1 work for weak base classifiers. We introduce the maxlabel error rate and derive bounds on it. For both algorithms, the bounds on the performance measures decrease exponentially under conditions which are easy to fulfill by the base classifier. For both algorithms the goal of the base classifier is to minimize the confidence-rated error rate which makes them applicable for a wide range of already existing base classifiers.

Throughout this paper $S = \{(x_i, y_i);\ i = 1, \ldots, N)\}$ denotes the training set where each x_i belongs to some instance or measurement space X and each label y_i is in some label set Y. In contrast to the two-class case, Y can have $|Y| \geq 2$ elements. A boosting algorithm calls a given weak classification algorithm h repeatedly in a series of rounds $t = 1, \ldots, T$. In each round, a sample of the original training set S is drawn according to the weighting distribution D_t and used as training set for the weak classification algorithm h. $D_t(i)$ denotes the weight of example i of the original training set S. The final classifier H is a weighted majority vote of the T weak classifiers h_t where α_t is the weight assigned to h_t. Finally, the elements of a set M that maximize and minimize a function f are denoted $\arg\max\limits_{m \in M} f(m)$ and $\arg\min\limits_{m \in M} f(m)$ respectively.

2. Algorithm GrPloss

In this section we will derive the algorithm GrPloss. Mason et al. (1998, 1999) embedded AdaBoost in a more general theory which sees boosting algorithms as gradient descent methods for the minimization of a loss function in function space. We get GrPloss by applying the gradient descent framework especially for minimizing the exponential pseudo-loss. We first consider slightly more general exponential loss functions. Based on the gradient descent framework, we derive a gradient descent algorithm for these loss functions in a straight forward way in Section 2.1. In contrast to the general framework, we can additionally derive a simple update rule for the sampling distribution as it exists for AdaBoost.M1 and AdaBoost.M2. Gradient descent does not provide a special choice for the "step size" α_t. In Section 2.2, we define the pseudo-loss error and derive α_t by minimization of an upper bound on the pseudo-loss error. Finally, the algorithm is simplified for the special case of decision stumps as base classifiers.

2.1 Gradient Descent for Exponential Loss Functions

First we briefly describe the gradient descent framework for the two-class case with $Y = \{-1, +1\}$. As usual a training set $S = \{(x_i, y_i);\ i = 1, \ldots, N)\}$ is given. We are considering a function space $\mathcal{F} = \mathrm{lin}(\mathcal{H})$ consisting of functions $f : X \to \mathbb{R}$ of the form

$$f(x; \vec{\alpha}, \vec{\beta}) = \sum_{t=1}^{T} \alpha_t h_t(x; \beta_t), \qquad h_t : X \to \{\pm 1\}$$

with $\vec{\alpha} = (\alpha_1, \ldots, \alpha_T) \in \mathbb{R}^T$, $\vec{\beta} = (\beta_1, \ldots, \beta_T)$ and $h_t \in \mathcal{H}$. The parameters β_t uniquely determine h_t therefore $\vec{\alpha}$ and $\vec{\beta}$ uniquely determine f. We choose a loss function

$$L(f) = E_{y,x}[l(f(x), y)] = E_x[E_y[l(yf(x))]] \qquad l : \mathbb{R} \to \mathbb{R}_{\geq 0}$$

where for example the choice of $l(f(x),y) = e^{-yf(x)}$ leads to

$$L(f) = \frac{1}{N}\sum_{i=1}^{N} e^{y_i f(x_i)}.$$

The goal is to find $f^* = \arg\min_{f\in\mathcal{F}} L(f)$.

The gradient in function space is defined as:

$$\nabla L(f)(x) := \frac{\partial L(f + e1_x)}{\partial e}\Big|_{e=0} = \lim_{e\to 0}\frac{L(f+e1_x) - L(f)}{e}$$

where for two arbitrary tuples v and $\tilde{v}$ we denote

$$1_v(\tilde{v}) = \begin{cases} 1 & \tilde{v} = v \\ 0 & \tilde{v} \neq v. \end{cases}$$

A gradient descent method always makes a step in the “direction” of the negative gradient $-\nabla L(f)(x)$. However $-\nabla L(f)(x)$ is not necessarily an element of $\mathcal{F}$, so we replace it by an element h_t of $\mathcal{F}$ which is as parallel to $-\nabla L(f)(x)$ as possible. Therefore we need an inner product $\langle\, ,\, \rangle : \mathcal{F}\times\mathcal{F} \to \mathbb{R}$, which can for example be chosen as

$$\langle f, \tilde{f}\rangle = \frac{1}{N}\sum_{i=1}^{N} f(x_i)\tilde{f}(x_i).$$

This inner product measures the agreement of f and $\tilde{f}$ on the training set. Using this inner product we can set

$$\beta_t := \arg\max_{\beta}\langle -\nabla L(f_{t-1}), h(\cdot\,;\beta)\rangle$$

and $h_t := h(\cdot\,;\beta_t)$. The inequality $\langle -\nabla L(f_{t-1}), h(\beta_t)\rangle \leq 0$ means that we can not find a good “direction” $h(\beta_t)$, so the algorithm stops, when this happens. The resulting algorithm is given in Figure 1.

```
Input: training set S, loss function l, inner product $\langle\, ,\, \rangle : \mathcal{F}\times\mathcal{F} \to \mathbb{R}$, starting value $f_0$.
$t := 1$
Loop: while $\langle -\nabla L(f_{t-1}), h(\beta_t)\rangle > 0$
    • $\beta_t := \arg\max_{\beta}\langle -\nabla L(f_{t-1}), h(\beta)\rangle$
    • $\alpha_t := \arg\min_{\alpha}(L(f_{t-1} + \alpha h_t(\beta_t)))$
    • $f_t = f_{t-1} + \alpha_t h_t(\beta_t)$
Output: $f_t, L(f_t)$
```

Figure 1: Algorithm gradient descent in function space

Now we go back to the multiclass case and modify the gradient descent framework in order to treat classifiers f of the form $f : X\times Y \to \mathbb{R}$, where $f(x,y)$ is a measure of the confidence, that an object with measurements x has the label y. We denote the set of possible classifiers with $\mathcal{F}$. For gradient descent we need a loss function and an inner product on $\mathcal{F}$. We choose

$$\langle f, \hat{f} \rangle := \frac{1}{N} \sum_{i=1}^{N} \sum_{y=1}^{|Y|} f(x_i, y) \hat{f}(x_i, y),$$

which is a straightforward generalization of the definition for the two-class case. The goal of the classification algorithm GrPloss is to minimize the special loss function

$$L(f) := \frac{1}{N} \sum_i l(f,i) \quad \text{with} \quad l(f,i) := \exp\left[\frac{1}{2}\left(1 - f(x_i, y_i) + \sum_{y \neq y_i} \frac{f(x_i, y)}{|Y|-1}\right)\right]. \tag{1}$$

The term

$$-f(x_i, y_i) + \sum_{y \neq y_i} \frac{f(x_i, y)}{|Y|-1}$$

compares the confidence to label the example x_i correctly with the mean confidence of choosing one of the wrong labels. Now we consider slightly more general exponential loss functions

$$l(f,i) = \exp\left[v(f,i)\right] \quad \text{with exponent} - \text{loss } v(f,i) = v_0 + \sum_y v_y(i) f(x_i, y) \; ,$$

where the choice

$$v_0 = \frac{1}{2} \text{ and } v_y(i) = \begin{cases} -\frac{1}{2} & y = y_i \\ \frac{1}{2(|Y|-1)} & y \neq y_i \end{cases}$$

leads to the loss function (1). This choice of the loss function leads to the algorithm given in Figure 2. The properties summarized in Theorem 1 can be shown to hold on this algorithm.

```
Input: training set $S$, maximum number of boosting rounds $T$
Initialisation: $f_0 := 0$, $t := 1$, $\forall i: \; D_1(i) := \frac{1}{N}$.
Loop: For $t = 1, \ldots, T$ do
    • $h_t = \arg\min_h \sum_i D_t(i) v(h,i)$
    • If $\sum_i D_t(i) v(h_t, i) \geq v_0 : \; T := t-1$, goto output.
    • Choose $\alpha_t$.
    • Update $f_t = f_{t-1} + \alpha_t h_t$ and $D_{t+1}(i) = \frac{1}{Z_t} D_t(i) l(\alpha_t h_t, i)$
Output: $f_T, L(f_T)$
```

Figure 2: Gradient descent for exponential loss functions

Theorem 1 *For the inner product*

$$\langle f,h\rangle = \frac{1}{N}\sum_{i=1}^{N}\sum_{y=1}^{|Y|} f(x_i,y)h(x_i,y)$$

and any exponential loss functions $l(f,i)$ of the form

$$l(f,i) = \exp\left[v(f,i)\right] \quad \text{with} \quad v(f,i) = v_0 + \sum_y v_y(i) f(x_i,y)$$

where v_0 and $v_y(i)$ are constants, the following statements hold:
(i) The choice of h_t that maximizes the projection on the negative gradient

$$h_t = \arg\max_h \langle -\nabla L(f_{t-1}), h\rangle$$

is equivalent to that minimizing the weighted exponent-loss

$$h_t = \arg\min_h \sum_i D_t(i) v(h,i)$$

with respect to the sampling distribution

$$D_t(i) := \frac{l(f_{t-1},i)}{\sum_{i'} l(f_{t-1},i')} = \frac{l(f_{t-1},i)}{Z'_{t-1}}.$$

(ii) The stopping criterion of the gradient descent method

$$\langle -\nabla L(f_{t-1}), h(\beta_t)\rangle \leq 0$$

leads to a stop of the algorithm, when the weighted exponent-loss gets positive

$$\sum_i D_t(i) v(h_t,i) \geq v_0.$$

(iii) The sampling distribution can be updated in a similar way as in AdaBoost using the rule

$$D_{t+1}(i) = \frac{1}{Z_t} D_t(i) l(\alpha_t h_t, i),$$

where we define Z_t as a normalization constant

$$Z_t := \sum_i D_t(i) l(\alpha_t h_t, i),$$

which ensures that the update D_{t+1} is a distribution.

In contrast to the general framework, the algorithm uses a simple update rule for the sampling distribution as it exists for the original boosting algorithms. Note that the algorithm does not specify the choice of the step size α_t, because gradient descent only provides an upper bound on α_t. We will derive a special choice for α_t in the next section.

Proof. The proof basically consists of three steps: the calculation of the gradient, the choice for base classifier h_t together with the stopping criterion and the update rule for the sampling distribution.
(i) First we calculate the gradient, which is defined by

$$\nabla L(f)(x,y) := \lim_{k\to 0} \frac{L(f+k1_{(x,y)}) - L(f)}{k}$$

for $1_{(x,y)}(x',y') = \begin{cases} 1 & (x,y)=(x',y') \\ 0 & (x,y)\neq(x',y') \end{cases}$.

So we get for $x = x_i$:

$$L(f+k1_{x_iy}) = \frac{1}{N}\exp\left[v_0 + \sum_{y'} v_{y'}(i) f(x_i,y') + kv_y(i)\right] = \frac{1}{N} l(f,i)e^{kv_y(i)}.$$

Substitution in the definition of $\nabla L(f)$ leads to

$$\nabla L(f)(x_i,y) = \lim_{k\to 0} \frac{l(f,i)(e^{kv_y(i)}-1)}{k} = l(f,i)v_y(i).$$

Thus

$$\nabla L(f)(x,y) = \begin{cases} 0 & x\neq x_i \\ l(f,i)v_y(i) & x = x_i \end{cases}. \tag{2}$$

Now we insert (2) into $\langle -\nabla L(f_{t-1}), h_t\rangle$ and get

$$\langle -\nabla L(f_{t-1}), h_t\rangle = -\frac{1}{N}\sum_i \sum_y l(f_{t-1},i)v_y(i)h(x_i,y) = -\frac{1}{N}\sum_i l(f_{t-1},i)(v(h,i)-v_0). \tag{3}$$

If we define the sampling distribution D_t up to a positive constant C_{t-1} by

$$D_t(i) := C_{t-1} l(f_{t-1},i), \tag{4}$$

we can write (3) as

$$\langle -\nabla L(f_{t-1}), h_t\rangle = -\frac{1}{C_{t-1}N}\sum_i D_t(i)(v(h,i)-v_0) = -\frac{1}{C_{t-1}N}\left(\sum_i D_t(i)v(h,i) - v_0\right). \tag{5}$$

Since we require C_{t-1} to be positive, we get the choice of h_t of the algorithm

$$h_t = \arg\max_h \langle -\nabla L(f_{t-1}), h\rangle = \arg\min_h \sum_i D_t(i)v(h,i).$$

(ii) One can verify the stopping criterion of Figure 2 from (5)

$$\langle -\nabla L(f_{t-1}), h_t\rangle \leq 0 \Leftrightarrow \sum_i D_t(i)v(h_t,i) \geq v_0.$$

(iii) Finally, we show that we can calculate the update rule for the sampling distribution D.

$$\begin{aligned} D_{t+1}(i) &= C_t l(f_t,i) = C_t l(f_{t-1}+\alpha_t h_t, i) \\ &= C_t l(f_{t-1},i) l(\alpha_t h_t, i) = \frac{C_t}{C_{t-1}} D_t(i) l(\alpha_t h_t, i). \end{aligned}$$

This means that the new weight of example i is a constant multiplied with $D_t(i)l(\alpha_t h_t, i)$. By comparing this equation with the definition of Z_t we can determine C_t

$$C_t = \frac{C_{t-1}}{Z_t}.$$

Since l is positive and the weights are positive one can show by induction, that also C_t is positive, which we required before. ■

2.2 Choice of α_t and Resulting Algorithm GrPloss

The algorithm above leaves the step length α_t, which is the weight of the base classifier h_t, unspecified. In this section we define the pseudo-loss error and derive α_t by minimization of an upper bound on the pseudo-loss error.

Definition: A classifier $f : X \times Y \to \mathbb{R}$ makes a pseudo-loss error in classifying an example x with label k, if

$$f(x,k) < \frac{1}{|Y|-1} \sum_{y \neq k} f(x,y).$$

The corresponding training error rate is denoted by $plerr$:

$$plerr := \frac{1}{N} \sum_{i=1}^{N} I\left(f(x_i,y_i) < \frac{1}{|Y|-1} \sum_{y \neq y_i} f(x_i,y) \right).$$

The pseudo-loss error counts the proportion of elements in the training set for which the confidence $f(x,k)$ in the right label is smaller than the *average* confidence in the remaining labels $\sum\limits_{y \neq k} f(x,y)/(|Y|-1)$. Thus it is a weak measure for the performance of a classifier in the sense that it can be much smaller than the training error.

Now we consider the exponential pseudo-loss. The constant term of the pseudo-loss leads to a constant factor which can be put into the normalizing constant. So with the definition

$$u(f,i) := f(x_i,y_i) - \frac{1}{|Y|-1} \sum_{y \neq y_i} f(x_i,y)$$

the update rule can be written in the shorter form

$$D_{t+1}(i) = \frac{1}{Z_t} D_t(i) e^{-\alpha_t u(h_t,i)/2}, \text{ with } Z_t := \sum_{i=1}^{N} D_t(i) e^{-\alpha_t u(h_t,i)/2}.$$

We present our next algorithm, GrPloss, in Figure 3, which we will derive and justify in what follows.

(i) Similar to Schapire and Singer (1999) we first bound $plerr$ by the product of the normalization constants

$$plerr \leq \prod_{t=1}^{T} Z_t. \tag{6}$$

To prove (6), we first notice that

$$plerr \leq \frac{1}{N} \sum_i e^{-u(f_T,i)/2}. \tag{7}$$

```
**Input:** training set $S = \{(x_1,y_1),\ldots,(x_N,y_N); x_i \in X, y_i \in Y\}$,
    $Y = \{1,\ldots,|Y|\}$, weak classification algorithm with output $h : X \times Y \to [0,1]$
    Optionally $T$: maximal number of boosting rounds

**Initialization:** $D_1(i) = \frac{1}{N}$.

**For** $t = 1,\ldots,T$**:**

    • Train the weak classification algorithm $h_t$ with distribution $D_t$, where $h_t$ should maximize
      $U_t := \sum_i D_t(i)u(h_t,i)$.

    • If $U_t \leq 0$: goto output with $T := t-1$

    • Set
          $\alpha_t = \ln\left(\frac{1+U_t}{1-U_t}\right)$.

    • Update D:
          $D_{t+1}(i) = \frac{1}{Z_t} D_t(i) e^{-\alpha_t u(h_t,i)/2}$.
      where $Z_t$ is a normalization factor (chosen so that $D_{t+1}$ is a distribution)

**Output:** final classifier $H(x)$:
    $H(x) = \arg\max_{y \in Y} f(x,y) = \arg\max_{y \in Y} \left(\sum_{t=1}^{T} \alpha_t h_t(x,y)\right)$
```

Figure 3: Algorithm GrPloss

Now we unravel the update rule

$$\begin{aligned}
D_{T+1}(i) &= \frac{1}{Z_T} e^{-\alpha_T u(h_T,i)/2} D_T(i) \\
&= \frac{1}{Z_T Z_{T-1}} e^{-\alpha_T u(h_T,i)/2} e^{-\alpha_{T-1} u(h_{T-1},i)/2} D_{T-1}(i) \\
&= \ldots = D_1(i) \prod_{t=1}^{T} e^{-\alpha_t u(h_t,i)/2} \frac{1}{Z_t} \\
&= \frac{1}{N} \exp\left(-\sum_{t=1}^{T} \alpha_t u(h_t,i)/2\right) \prod_{t=1}^{T} \frac{1}{Z_t} \\
&= \frac{1}{N} e^{-u(f_T,i)/2} \prod_{t=1}^{T} \frac{1}{Z_t}
\end{aligned}$$

where the last equation uses the property that u is linear in h. Since

$$1 = \sum_i D_{T+1}(i) = \sum_i \frac{1}{N} e^{-u(f_T,i)/2} \prod_{t=1}^{T} \frac{1}{Z_T}$$

we get Equation (6) by using (7) and the equation above

$$plerr \leq \frac{1}{N}\sum_i e^{-u(f_T,i)/2} = \prod_{t=1}^{T} Z_t.$$

(ii) Derivation of α_t:
Now we derive α_t by minimizing the upper bound (6). First, we plug in the definition of Z_t

$$\prod_{t=1}^{T} Z_t = \prod_{t=1}^{T}\left(\sum_i D_t(i)e^{-\alpha_t u(h_t,i)/2}\right).$$

Now we get an upper bound on this product using the convexity of the function $e^{-\alpha_t u}$ between -1 and +1 (from $h(x,y) \in [0,1]$ it follows that $u \in [-1,+1]$) for positive α_t:

$$\prod_{t=1}^{T} Z_t \leq \prod_{t=1}^{T}\left(\sum_i D_t(i)\frac{1}{2}[(1-u(h_t,i))e^{+\frac{1}{2}\alpha_t} + (1+u(h_t,i))e^{-\frac{1}{2}\alpha_t}]\right). \tag{8}$$

Now we choose α_t in order to minimize this upper bound by setting the first derivative with respect to α_t to zero. To do this, we define

$$U_t := \sum_i D_t(i)u(h_t,i).$$

Since each α_t occurs in exactly one factor of the bound (8) the result for α_t only depends on U_t and not on $U_s,\ s \neq t$, more specifically

$$\alpha_t = \ln\left(\frac{1+U_t}{1-U_t}\right).$$

Note that U_t has its values in the interval $[-1,1]$, because $u_t \in [-1,+1]$ and D_t is a distribution.
(iii) Derivation of the upper bound of the theorem:
Now we substitute α_t back in (8) and get after some straightforward calculations

$$\prod_{t=1}^{T} Z_t \leq \prod_{t=1}^{T}\sqrt{1-U_t^2}.$$

Using the inequality $\sqrt{1-x} \leq (1-\frac{1}{2}x) \leq e^{-x/2}$ for $x \in [0,1]$ we can get an exponential bound on $\prod_t Z_t$

$$\prod_{t=1}^{T} Z_t \leq \exp\left[\sum_{t=1}^{T} -U_t^2/2\right].$$

If we assume that each classifier h_t fulfills $U_t \geq \delta$, we finally get

$$\prod_{t=1}^{T} Z_t \leq e^{-\delta^2 T/2}.$$

(iv) Stopping criterion:
The stopping criterion of the slightly more general algorithm directly results in the new stopping criterion to stop, when $U_t \leq 0$. However, note that the bound depends on the square of U_t instead of U_t leading to a formal decrease of the bound even when $U_t > 0$.

We summarize the foregoing argument as a theorem.

Theorem 2 *If for all base classifiers $h_t : X \times Y \rightarrow [0,1]$ of the algorithm GrPloss given in Figure 3*

$$U_t := \sum_i D_t(i) u(h_t, i) \geq \delta$$

holds for $\delta > 0$ then the pseudo-loss error of the training set fulfills

$$plerr \leq \prod_{t=1}^{T} \sqrt{1 - U_t^2} \leq e^{-\delta^2 T/2}. \tag{9}$$

2.3 GrPloss for Decision Stumps

So far we have considered classifiers of the form $h : X \times Y \rightarrow [0,1]$. Now we want to consider base classifiers that have additionally the normalization property

$$\sum_{y \in Y} h(x,y) = 1 \tag{10}$$

which we did not use in the previous section for the derivation of α_t. The decision stumps we used in our experiments find an attribute a and a value v which are used to divide the training set into two subsets. If attribute a is continuous and its value on x is at most v then x belongs to the first subset; otherwise x belongs to the second subset. If attribute a is categorical the two subsets correspond to a partition of all possible values of a into two sets. The prediction $h(x,y)$ is the proportion of examples with label y belonging to the same subset as x. Since proportions are in the interval $[0,1]$ and for each of the two subsets the sum of proportions is one our decision stumps have both the former and the latter property (10). Now we use these properties to minimize a tighter bound on the pseudo-loss error and further simplify the algorithm.

(i) Derivation of α_t:
To get α_t we can start with

$$plerr \leq \prod_{t=1}^{T} Z_t = \prod_{t=1}^{T} \left(\sum_i D_t(i) e^{-\alpha_t u(h_t,i)/2} \right)$$

which was derived in part (i) of the proof of the previous section. First, we simplify $u(h,i)$ using the normalization property and get

$$u(h,i) = \frac{|Y|}{|Y|-1} h(x_i,y_i) - \frac{1}{|Y|-1} \tag{11}$$

In contrast to the previous section, $u(h,i) \in [-\frac{1}{|Y|-1}, 1]$ for $h(x_i,y_i) \in [0,1]$, which we will take into account for the convexity argument:

$$plerr \leq \prod_{t=1}^{T} \sum_{i=1}^{N} D_t(i) \left(h(x_i,y_i)\, \mathrm{e}^{-\alpha_t/2} + (1 - h_t(x_i,y_i))\, \mathrm{e}^{\alpha_t/(2(|Y|-1))} \right) \tag{12}$$

Setting the first derivative with respect to α_t to zero leads to

$$\alpha_t = \frac{2(|Y|-1)}{|Y|} \ln\left(\frac{(|Y|-1)r_t}{1-r_t}\right),$$

where we defined

$$r_t := \sum_{i=1}^{N} D_t(i) h_t(x_i, y_i).$$

(ii) Upper bound on the pseudo-loss error:
Now we plug α_t in (12) and get

$$plerr \leq \prod_{t=1}^{T} \left(r_t \left(\frac{1-r_t}{r_t(|Y|-1)}\right)^{(|Y|-1)/|Y|} + (1-r_t)\left(\frac{r_t(|Y|-1)}{1-r_t}\right)^{1/|Y|} \right). \tag{13}$$

(iii) Stopping criterion:
As expected for $r_t = 1/|Y|$ the corresponding factor is 1. The stopping criterion $U_t \leq 0$ can be directly translated into $r_t \geq 1/|Y|$. Looking at the first and second derivative of the bound one can easily verify that it has a unique maximum at $r_t = 1/|Y|$. Therefore, the bound drops as long as $r_t > 1/|Y|$. Note again that since $r_t = 1/|Y|$ is a unique maximum we get a formal decrease of the bound even when $r_t > 1/|Y|$.
(iv) Update rule:
Now we simplify the update rule using (11) and insert the new choice of α_t and get

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \mathrm{e}^{-\tilde{\alpha}_t(h_t(x_i,y_i)-1/|Y|)} \quad \text{for} \quad \tilde{\alpha}_t := \ln\left(\frac{(|Y|-1)r_t}{1-r_t}\right)$$

Also the goal of the base classifier can be simplified, because maximizing U_t is equivalent to maximizing r_t.

We will see in the next section, that the resulting algorithm is a special case of the algorithm BoostMA of the next chapter with $c = 1/|Y|$.

3. BoostMA

The aim behind the algorithm BoostMA was to find a simple modification of AdaBoost.M1 in order to make it work for weak base classifiers. The original idea was influenced by a frequently used argument for the explanation of ensemble methods. Assuming that the individual classifiers are uncorrelated, majority voting of an ensemble of classifiers should lead to better results than using one individual classifier. This explanation suggests that the weight of classifiers that perform better than random guessing should be positive. This is not the case for AdaBoost.M1. In AdaBoost.M1 the weight of a base classifier α is a function of the error rate, so we tried to modify this function so that it gets positive, if the error rate is less than the error rate of random guessing. The resulting classifier AdaBoost.M1W showed good results in experiments (Eibl and Pfeiffer, 2002). Further theoretical considerations led to the more elaborate algorithm which we call BoostMA which uses confidence-rated classifiers and also compares the base classifier with the uninformative rule.

In AdaBoost.M2, the sampling weights are increased for instances for which the pseudo-loss exceeds 1/2. Here we want to increase the weights for instances, where the base classifier h :

$X \times Y \to [0,1]$ performs worse than the uninformative or what we call the maxlabel rule. The maxlabel rule labels each instance as the most frequent label. As a confidence-rated classifier, the uninformative rule has the form

$$\text{maxlabel rule} : X \times Y \to [0,1] : h(x,y) := \frac{N_y}{N},$$

where N_y is the number of instances in the training set with label y. So it seems natural to investigate a modification where the update of the sampling distribution has the form

$$D_{t+1}(i) = D_t(i)\frac{e^{-\alpha_t(h_t(x_i,y_i)-c)}}{Z_t}, \text{ with } Z_t := \sum_{i=1}^{N} D_t(i)e^{-\alpha_t(h_t(x_i,y_i)-c)},$$

where c measures the performance of the uninformative rule. Later we will set

$$c := \sum_{y\in Y}\left(\frac{N_y}{N}\right)^2$$

and justify this setting. But up to that point we let the choice of c open and just require $c \in (0,1)$. We now define a performance measure which plays the same role as the pseudo-loss error.

Definition 1 *Let c be a number in $(0,1)$. A classifier $f : X \times Y \to [0,1]$ makes a maxlabel error in classifying an example x with label k, if*

$$f(x,k) < c.$$

The maxlabel error for the training set is called mxerr:

$$mxerr := \frac{1}{N}\sum_{i=1}^{N} I\left(f(x_i,y_i) < c\right).$$

The maxlabel error counts the proportion of elements of the training set for which the confidence $f(x,k)$ in the right label is smaller than c. The number c must be chosen in advance. The higher c is, the higher is the maxlabel error for the *same* classifier f; therefore to get a weak error measure we set c very low. For BoostMA we choose c as the accuracy for the uninformative rule. When we use decision stumps as base classifiers we have the property $h(x,y) \in [0,1]$. By normalizing $\alpha_1,\ldots,\alpha_T$, so that they sum to one, we ensure $f(x,y) \in [0,1]$ (Equation 15).

We present the algorithm BoostMA in Figure 4 and in what follows we justify and establish some properties about it. As for GrPloss the modus operandi consists of finding an upper bound on *mxerr* and minimizing the bound with respect to α.

(i) Bound of *mxerr* in terms of the normalization constants Z_t:

Similar to the calculations used to bound the pseudo-loss error we begin by bounding *mxerr* in terms of the normalization constants Z_t: We have

$$\begin{aligned} 1 &= \sum_i D_{t+1}(i) = \sum_i D_t(i)\frac{e^{-\alpha_t(h_t(x_i,y_i)-c)}}{Z_t} = \ldots \\ &= \frac{1}{\prod_s Z_s}\frac{1}{N}\sum_i\prod_{s=1}^{t} e^{-\alpha_s(h_s(x_i,y_i)-c)} = \frac{1}{\prod_s Z_s}\frac{1}{N}\sum_i e^{-(f(x_i,y_i)-c\sum_s \alpha_s)}. \end{aligned}$$

```
Input: training set $S = \{(x_1,y_1),\ldots,(x_N,y_N); x_i \in X, y_i \in Y\}$,
    $Y = \{1,\ldots,|Y|\}$, weak classification algorithm of the form $h : X \times Y \to [0,1]$.
    Optionally $T$: number of boosting rounds
Initialization: $D_1(i) = \frac{1}{N}$.
For $t = 1,\ldots,T$:
    • Train the weak classification algorithm $h_t$ with distribution $D_t$, where $h_t$ should maximize
          $r_t = \sum_i D_t(i) h_t(x_i,y_i)$
    • If $r_t \leq c$: goto output with $T := t-1$
    • Set
          $\alpha_t = \ln\left(\frac{(1-c)r_t}{c(1-r_t)}\right)$.
    • Update D:
          $D_{t+1}(i) = D_t(i)\frac{e^{-\alpha_t(h_t(x_i,y_i)-c)}}{Z_t}$.
      where $Z_t$ is a normalization factor (chosen so that $D_{t+1}$ is a distribution)
Output: Normalize $\alpha_1,\ldots,\alpha_T$ and set the final classifier $H(x)$:
    $H(x) = \arg\max_{y\in Y} f(x,y) = \arg\max_{y\in Y}\left(\sum_{t=1}^{T}\alpha_t h_t(x,y)\right)$
```

Figure 4: Algorithm BoostMA

So we get

$$\prod_t Z_t = \frac{1}{N}\sum_i e^{-\left(f(x_i,y_i)-c\sum_t \alpha_t\right)}. \tag{14}$$

Using

$$\frac{f(x_i,y_i)}{\sum_t \alpha_t} < c \quad \Rightarrow \quad e^{-\left(f(x_i,y_i)-c\sum_t \alpha_t\right)} > 1 \tag{15}$$

and (14) we get

$$mxerr \leq \prod_t Z_t. \tag{16}$$

(ii) Choice of α_t:
Now we bound $\prod_t Z_t$ and then we minimize it, which leads us to the choice of α_t. First we use the definition of Z_t and get

$$\prod_t Z_t = \prod_t\left(\sum_i D_t(i) e^{-\alpha_t(h_t(x_i,y_i)-c)}\right). \tag{17}$$

Now we use the convexity of $e^{-\alpha_t(h_t(x_i,y_i)-c)}$ for $h_t(x_i,y_i)$ between 0 and 1 and the definition

$$r_t := \sum_i D_t(i) h_t(x_i,y_i)$$

and get

$$\begin{aligned} mxerr &\leq \prod_t \sum_i D_t(i) \left(h_t(x_i,y_i) e^{-\alpha_t(1-c)} + (1-h_t(x_i,y_i)) e^{\alpha_t c} \right) \\ &= \prod_t \left(r_t e^{-\alpha_t(1-c)} + (1-r_t) e^{\alpha_t c} \right). \end{aligned}$$

We minimize this by setting the first derivative with respect to α_t to zero, which leads to

$$\alpha_t = \ln\left(\frac{(1-c)r_t}{c(1-r_t)} \right).$$

(iii) First bound on *mxerr*:
To get the bound on *mxerr* we substitute our choice for α_t in (17) and get

$$mxerr \leq \prod_t \left(\left(\frac{(1-c)r_t}{c(1-r_t)} \right)^c \sum_i D_t(i) \left(\frac{c(1-r_t)}{(1-c)r_t} \right)^{h_t(x_i,y_i)} \right). \tag{18}$$

Now we bound the term $\left(\frac{c(1-r_t)}{(1-c)r_t}\right)^{h_t(x_i,y_i)}$ by use of the inequality

$$x^a \leq 1 - a + ax \quad \text{for } x \geq 0 \text{ and } a \in [0,1],$$

which comes from the convexity of x^a for a between 0 and 1 and get

$$\left(\frac{c(1-r_t)}{(1-c)r_t} \right)^{h_t(x_i,y_i)} \leq 1 - h_t(x_i,y_i) + h_t(x_i,y_i) \frac{c(1-r_t)}{(1-c)r_t}.$$

Substitution in (18) and simplifications lead to

$$mxerr \leq \prod_t \left(\frac{r_t^c (1-r_t)^{1-c}}{(1-c)^{1-c} c^c} \right). \tag{19}$$

The factors of this bound are symmetric around $r_t = c$ and take their maximum of 1 there. Therefore if $r_t > c$ is valid the bound on *mxerr* decreases.
(iv) Exponential decrease of *mxerr*:
To prove the second bound we set $r_t = c + \delta$ with $\delta \in (0, 1-c)$ and rewrite (19) as

$$mxerr \leq \prod_t \left(1 - \frac{\delta}{1-c} \right)^{1-c} \left(1 + \frac{\delta}{c} \right)^c.$$

We can bound both terms using the binomial series: all terms of the series of the first term are negative, we stop after the terms of first order and get

$$\left(1 - \frac{\delta}{1-c} \right)^{1-c} \leq 1 - \delta.$$

The series of the second term has both positive and negative terms, we stop after the positive term of first order and get

$$\left(1+\frac{\delta}{c}\right)^c \leq 1+\delta.$$

Thus

$$mxerr \leq \prod_t (1-\delta^2).$$

Using $1+x \leq e^x$ for $x \leq 0$ leads to

$$mxerr \leq e^{-\delta^2 T}.$$

We summarize the foregoing argument as a theorem.

Theorem 3 *If all base classifiers h_t with $h_t(x,y) \in [0,1]$ fulfill*

$$r_t := \sum_i D_t(i) h_t(x_i, y_i) \geq c+\delta$$

for $\delta \in (0, 1-c)$ (and the condition $c \in (0,1)$) then the maxlabel error of the training set for the algorithm in Figure 4 fulfills

$$mxerr \leq \prod_t \left(\frac{r_t^c\,(1-r_t)^{1-c}}{(1-c)^{1-c}\,c^c}\right) \leq e^{-\delta^2 T}. \tag{20}$$

Remarks: 1.) Choice of c for BoostMA: since we use confidence-rated base classification algorithms we choose the training accuracy for the confidence-rated uninformative rule for c, which leads to

$$c = \frac{1}{N}\sum_{i=1}^{N} \frac{N_{y_i}}{N} = \frac{1}{N}\sum_y \sum_{i;y_i=y} \frac{N_y}{N} = \sum_{y\in Y}\left(\frac{N_y}{N}\right)^2. \tag{21}$$

2.) For base classifiers with the normalization property (10) we can get a simpler expression for the pseudo-loss error. From

$$\sum_{y\neq k} f(x,y) = \sum_{y\neq k}\sum_t \alpha_t h_t(x,y) = \sum_t \alpha_t (1-h_t(x,k)) = \sum_t \alpha_t - f(x,k)$$

we get

$$f(x,k) < \frac{1}{|Y|-1}\sum_{y\neq k} f(x,y) \;\Leftrightarrow\; \frac{f(x,k)}{\sum_t \alpha_t} < \frac{1}{|Y|}. \tag{22}$$

That means that if we choose $c = 1/|Y|$ for BoostMA the maxlabel error is the same as the pseudo-loss error. For the choice (21) of c this is the case when the group proportions are balanced, because then

$$c = \sum_{y\in Y}\left(\frac{N_y}{N}\right)^2 = \sum_{y\in Y}\left(\frac{1}{|Y|}\right)^2 = |Y|\frac{1}{|Y|^2} = \frac{1}{|Y|}.$$

For this choice of c the update rule of the sampling distribution for BoostMA gets

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \mathrm{e}^{-\alpha_t(h_t(x_i,y_i)-1/|Y|)} \quad \text{and} \quad \alpha_t = \ln\left(\frac{(|Y|-1)r_t}{1-r_t}\right),$$

which is just the same as the update rule GrPloss for decision stumps. Summarizing these two results we can say that for base classifiers with the normalization property, the choice (21) for c of BoostMA and data sets with balanced labels, the two algorithms GrPloss and BoostMA and their error measures are the same.
3.) In contrast to GrPloss the algorithm does not change when the base classifier additionally fulfills the normalization property (10) because the algorithm only uses $h_t(x_i, y_i)$.

4. Experiments

In our experiments we focused on the derived bounds and the practical performance of the algorithms.

4.1 Experimental Setup

To check the performance of the algorithms experimentally we performed experiments with 12 data sets, most of which are available from the UCI repository (Blake and Merz, 1998). To get reliable estimates for the expected error rate we used relatively large data sets consisting of about 1000 cases or more. The expected classification error was estimated either by a test error rate or 10-fold cross-validation. A short overview of the data sets is given in Table 1.

Database	N	♯ Labels	♯ Variables	Error Estimation	Labels
car *	1728	4	6	10-CV	unbalanced
digitbreiman	5000	10	7	test error	balanced
letter	20000	26	16	test error	balanced
nursery *	12960	4	8	10-CV	unbalanced
optdigits	5620	10	64	test error	balanced
pendigits	10992	10	16	test error	balanced
satimage *	6435	6	34	test error	unbalanced
segmentation	2310	7	19	10-CV	balanced
waveform	5000	3	21	test error	balanced
vehicle	846	4	18	10-CV	balanced
vowel	990	11	10	test error	balanced
yeast *	1484	10	9	10-CV	unbalanced

Table 1: Properties of the databases

For all algorithms we used boosting by resampling with decision stumps as base classifiers. We used AdaBoost.M2 by Freund and Schapire (1997), BoostMA with $c = \sum_{y \in Y} (N_y/N)^2$ and the algorithm GrPloss for decision stumps of Section 2.3 which corresponds to BoostMA with $c = 1/|Y|$. For only four databases the proportions of the labels are significantly unbalanced so that GrPloss and BoostMA should have greater differences only for these four databases (marked with a *). As discussed by Bauer and Kohavi (1999) the individual sampling weights $D_t(i)$ can get very small. Similar to was done there, we set the weights of instances which were below 10^{-10} to 10^{-10}.

We also set a maximum number of 2000 boosting rounds to stop the algorithm if the stopping criterion is not satisfied.

4.2 Results

The experiments have two main goals. From the theoretical point of view one is interested in the derived bounds. For the practical use of the algorithms, it is important to look at the training and test error rates and the speed of the algorithms.

4.2.1 DERIVED BOUNDS

First we look at the bounds on the error measures. For the algorithm AdaBoost.M2, Freund and Schapire (1997) derived the upper bound

$$(|Y|-1)2^{T-1}\prod_{t=1}^{T}\sqrt{\varepsilon_t(1-\varepsilon_t)} \tag{23}$$

on the training error. We have three different bounds on the pseudo-loss error of Grploss: the term

$$\prod_t Z_t \tag{24}$$

which was derived in the first part of the proof of Theorem 2, the tighter bound (9) of Theorem 2 and the bound (13) for the special case of decision stumps as base classifiers. In Section 3, we derived two upper bounds on the maxlabel error for BoostMA: term (24) and the tighter bound (20) of Theorem 3.

For all algorithms their respective bounds hold for all time steps and for all data sets. Bound (23) on the training error of AdaBoost.M2 is very loose – it even exceeds 1 for eight of the 12 data sets, which is possible due to the factor $|Y|-1$ (Table 2). In contrast to the bound on the training error of AdaBoost.M2, the bounds on the pseudo-loss error of GrPloss and the maxlabel error of BoostMA are below 1 for all data sets and all boosting rounds. In that sense, they are tighter than the bounds on the training error of AdaBoost.M2.

As expected, bound (13) derived for the special case of decision stumps as base classifiers on the pseudo-loss error is smaller than bound (9) of Theorem 2 which doesn't use the normalization property (10) of the decision stumps.

For both GrPloss and BoostMA, bound (24) is the smallest bound since it contains the fewest approximations. For BoostMA, term (24) is a bound on the maxlabel error and for GrPloss term (24) is a bound on the pseudo-loss error. For unbalanced data sets, the maxlabel error is the "harder" error measure than the pseudo-loss error, so for these data sets bound (24) is higher for BoostMA than for GrPloss. For balanced data sets the maxlabel error and the pseudo-loss error are the same. Bound (9) for GrPloss is higher for these data sets than bound (20) of BoostMA. This suggests that bound (9) for GrPloss could be improved by more sophisticated calculations.

4.2.2 COMPARISON OF THE ALGORITHMS

Now we wish to compare the algorithms with one another. Since GrPloss and BoostMA differ only for the four unbalanced data sets, we focus on the comparison of GrPloss with AdaBoost.M2 and make only a short comparison of GrPloss and BoostMA. For the subsequent comparisons we take

	AdaBoost.M2 training error [%]		GrPloss pseudo-loss error [%]				BoostMA maxlabel error [%]		
Database	trerr	BD23	plerr	BD24	BD13	BD9	mxerr	BD24	BD20
car *	0	33.9	0	3.4	11.1	31.9	7.8	63.1	71.9
digitbreiman	25.5	327.4	0.5	3.7	19.9	81.0	1.0	11.9	35.6
letter	46.1	1013.1	0.4	7.2	27.8	94.3	0.4	8.1	29.5
nursery *	14.2	78.7	0	0	0.5	11.1	0	0.8	7.6
optdigits	0	421.1	0	0	2.0	51.4	0	0	0.1
pendigits	13.8	190.2	0	0	0.1	42.6	0	0	0.1
satimage *	15.9	118.5	0.1	1.8	13.2	62.3	3.8	26.0	50.1
segmentation	7.5	96.2	0	0.4	2.8	30.5	0	0.4	3.5
vehicle	26.5	101.2	0.1	2.8	14.7	50.0	0.1	3.3	16.5
vowel	30.9	273.8	0	0	0.1	40.4	0	0.1	3.0
waveform	12.5	48.4	0	0.5	6.3	23.3	0	0.4	6.0
yeast *	60.2	365.0	0.4	6.6	26.0	83.6	49.2	99.2	99.6

Table 2: Performance measures and their bounds in percent at the boosting round with minimal training error. trerr, BD23: training error of AdaBoost and its bound (23); plerr, BD24 ,BD13 ,BD9: pseudo-loss error of GrPloss and its bounds (24), (13) and (9); mxerr, BD24, BD20: maxlabel error of BoostMA and its bounds (24) and (20).

all error rates at the boosting round with minimal training error rate as was done by Eibl and Pfeiffer (2002).

First we look at the minimum achieved training and test error rates. The theory suggests AdaBoost.M2 to work best in minimizing the training error. However, GrPloss seems to have roughly the same performance with maybe AdaBoost.M2 leading by a slight edge (Tables 3 and 4, Figure 5). The difference in the training error mainly carries over to the difference in the test error. Only for the data sets digitbreiman and yeast do the training and the test error favor different algorithms (Table 4). Both the training and the test error favor AdaBoost.M2 for six data sets and GrPloss for four data sets with two draws (Table 4).

While GrPloss and AdaBoost.M2 were quite close for the training and test error rates, this is not the case for the pseudo-loss error. Here, GrPloss is the clear winner against AdaBoost.M1 with eight wins and four draws (Table 4). The reason for this might be the fact that bound (13) on the pseudo-loss error of GrPloss is tighter than bound (23) on the training error of AdaBoost.M2 (Table 2). For the data set nursery, bound (13) on the pseudo-loss error of GrPloss (0.5%) is smaller than the pseudo-loss error of AdaBoost.M2 (1.9%). So for this data set, bound (13) can explain the superiority of GrPloss in minimizing the pseudo-loss error.

Due to the fact that only four data sets are significantly unbalanced, it is not easy to assess the difference between GrPloss and BoostMA. GrPloss seems to have a lead regarding the training and test error rates (Tables 3 and 5). For the experiments, the constant c of BoostMA was chosen as the training accuracy for the confidence-rated uninformative rule (21). For the unbalanced data sets, this c exceeds $1/|Y|$, which is the corresponding choice for GrPloss (22). A change of c – maybe even adaptively during the run – could possibly improve the performance. We wish to make further

	training error			test error		
Database	AdaM2	GrPloss	BoostMA	AdaM2	GrPloss	BoostMA
car *	0	0	**7.75**	0	0	**7.75**
digitbreiman	25.49	25.63	25.63	27.51	27.13	27.38
letter	**46.07**	40.02	40.14	**47.18**	41.70	41.70
nursery *	*14.16*	12.37	12.63	*14.27*	12.35	12.67
optdigits	0	0	0	0	0	0
pendigits	13.82	*17.17*	*17.20*	18.61	*20.44*	*20.75*
satimage *	15.85	15.69	16.87	18.25	17.80	18.90
segmentation	7.49	*9.05*	8.90	8.40	9.31	9.48
vehicle	26.46	*30.15*	*30.19*	35.34	*38.16*	*36.87*
vowel	30.87	**41.67**	**42.23**	54.33	**67.32**	**67.32**
waveform	12.45	*14.55*	*14.49*	16.63	*18.17*	17.72
yeast *	60.18	59.31	60.61	60.65	61.99	*62.47*

Table 3: Training and test error at the boosting round with minimal training error; bold and italic numbers correspond to high(>5%) and medium(>1.5%) differences to the smallest of the three error rates

	GrPloss vs. AdaM2			
Database	trerr	testerr	plerr	speed
car *	o	o	o	+
digitbreiman	-	+	+	+
letter	+	+	+	+
nursery *	+	+	+	+
optdigits	o	o	o	-
pendigits	-	-	+	+
satimage *	+	+	+	+
segmentation	-	-	o	+
vehicle	-	-	+	+
vowel	-	-	o	+
waveform	-	-	+	+
yeast *	+	-	+	-
total	4-2-6	4-2-6	8-4-0	10-0-2

Table 4: Comparison of GrPloss with AdaBoost.M2: win-loss-table for the training error, test error, pseudo-loss error and speed of the algorithm (+/o/-: win/draw/loss for GrPloss)

investigations about a systematic choice of c for BoostMA. Both algorithms seem to be better in the minimization of their corresponding error measure (Table 5). The small differences between GrPloss and BoostMA occurring for the nearly balanced data sets can not only come from the small

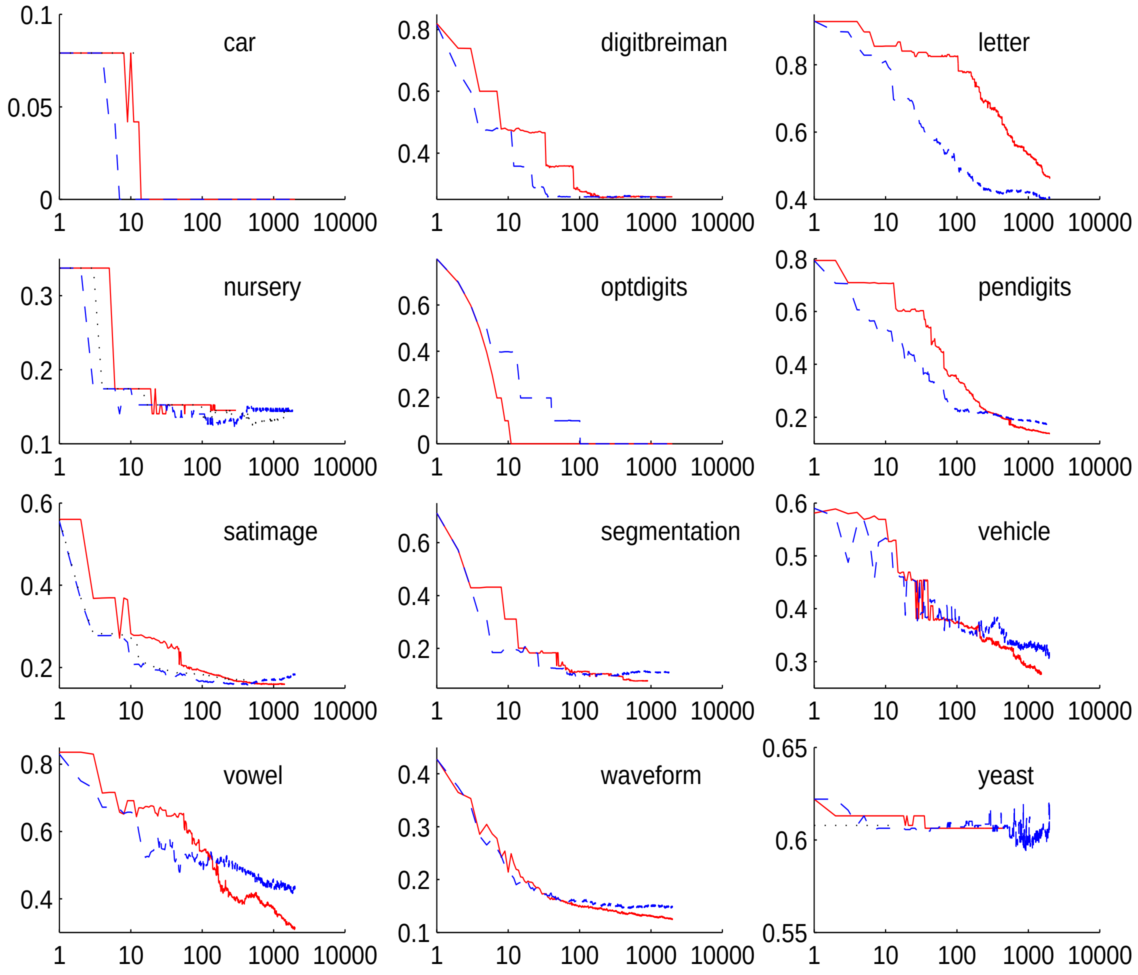


Figure 5: Training error curves: solid: AdaBoost.M2, dashed: GrPloss, dotted: BoostMA

differences in the group proportions, but also from differences in the resampling step and from the partition of a balanced data set into unbalanced training and test sets during cross-validation.

Performing a boosting algorithm is a time consuming procedure, so the speed of an algorithm is an important topic. Figure 5 indicates that the training error rate of GrPloss is decreasing faster than the training error rate of AdaBoost.M2. To be more precise, we look at the number of boosting rounds needed to achieve 90% of the total decrease of the training error rate. For 10 of the 12 data sets, AdaBoost.M2 needs more boosting rounds than GrPloss, so GrPloss seems to lead to a faster decrease in the training error rate (Table 4). Besides the number of boosting rounds, the time for the algorithm is also heavily influenced by the time needed to construct a base classifier. In our program, it took longer to construct a base classifier for AdaBoost.M2 because the minimization of the pseudo-loss which is required for AdaBoost.M2 is not as straightforward as the maximization of r_t required for GrPloss and BoostMA. However, the time needed to construct a base classifier strongly depends on programming details, so we do not wish to over-emphasize this aspect.

	GrPloss vs. BoostMA				
Database	trerr	testerr	plerr	mxerr	speed
car *	+	+	+	o	-
nursery *	+	+	o	o	+
satimage *	+	+	+	-	o
yeast *	+	+	+	-	-
total	4-0-0	4-0-0	3-1-0	0-2-2	1-0-2

Table 5: Comparison of GrPloss with BoostMA for the unbalanced data sets: win-loss-table for the training error, test error, pseudo-loss error, maxlabel error and speed of the algorithm (+/o/-: win/draw/loss for GrPloss)

5. Conclusion

We proposed two new algorithms GrPloss and BoostMA for multiclass problems with weak base classifiers. The algorithms are designed to minimize the pseudo-loss error and the maxlabel error respectively. Both have the advantage that the base classifier minimizes the confidence-rated error instead of the pseudo-loss. This makes them easier to use with already existing base classifiers. Also the changes to AdaBoost.M1 are very small, so one can easily get the new algorithms by only slight adaption of the code of AdaBoost.M1. Although they are not designed to minimize the training error, they have comparable performance as AdaBoost.M2 in our experiments. As a second advantage, they converge faster than AdaBoost.M2. AdaBoost.M2 minimizes a bound on the training error. The other two algorithms have the disadvantage of minimizing bounds on performance measures which are not connected so strongly to the expected error. However the bounds on the performance measures of GrPloss and BoostMA are tighter than the bound on the training error of AdaBoost.M2, which seems to compensate for this disadvantage.

References

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

Eric Bauer, Ron Kohavi. An empirical comparison of voting classification algorithms: bagging, boosting and variants. *Machine Learning*, 36:105–139, 1999.

Catherine Blake, Christopher J. Merz. UCI Repository of machine learning databases [http://www.ics.uci.edu/ mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science, 1998

Thomas G. Dietterrich, Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research* 2:263–286, 1995.

Günther Eibl, Karl–Peter Pfeiffer. Analysis of the performance of AdaBoost.M2 for the simulated digit-recognition-example. *Machine Learning: Proceedings of the Twelfth European Conference*, 109–120, 2001.

Günther Eibl, Karl–Peter Pfeiffer. How to make AdaBoost.M1 work for weak classifiers by changing only one line of the code. *Machine Learning: Proceedings of the Thirteenth European Conference*, 109–120, 2002.

Yoav Freund, Robert E. Schapire. Experiments with a new boosting algorithm. *Machine Learning: Proceedings of the Thirteenth International Conference,* 148–56, 1996.

Yoav Freund, Robert E. Schapire. A decision-theoretic generalization of online-learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

Venkatesan Guruswami, Amit Sahai. Multiclass learning, boosting, and error-correcting codes. *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, 145–155, 1999.

Llew Mason, Peter L. Bartlett, Jonathan Baxter. Direct optimization of margins improves generalization in combined classifiers. *Proceedings of NIPS 98*, 288–294, 1998.

Llew Mason, Peter L. Bartlett, Jonathan Baxter, Marcus Frean. Functional gradient techniques for combining hypotheses. *Advances in Large Margin Classifiers*, 221–246, 1999.

Ross Quinlan. Bagging, boosting, and C4.5. *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 725–730, 1996.

Gunnar Rätsch, Bernhard Schölkopf, Alex J. Smola, Sebastian Mika, Takashi Onoda, Klaus R. Müller. Robust ensemble learning. *Advances in Large Margin Classifiers*, 207–220, 2000a.

Gunnar Rätsch, Takashi Onoda, Klaus R. Müller. Soft margins for AdaBoost. *Machine Learning* 42(3):287–320, 2000b.

Robert E. Schapire, Yoav Freund, Peter L. Bartlett, Wee Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics*, 26(5):1651–1686, 1998.

Robert E. Schapire, Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning* 37:297-336, 1999.

Journal of Machine Learning Research 6 (2005) 211–232 Submitted 11/04; Published 3/05

A Classification Framework for Anomaly Detection

Ingo Steinwart INGO@LANL.GOV
Don Hush DHUSH@LANL.GOV
Clint Scovel JCS@LANL.GOV
Modeling, Algorithms and Informatics Group, CCS-3
Los Alamos National Laboratory
Los Alamos, NM 87545, USA

Editor: Bernhard Schölkopf

Abstract

One way to describe anomalies is by saying that anomalies are not concentrated. This leads to the problem of finding level sets for the data generating density. We interpret this learning problem as a binary classification problem and compare the corresponding classification risk with the standard performance measure for the density level problem. In particular it turns out that the empirical classification risk can serve as an empirical performance measure for the anomaly detection problem. This allows us to compare different anomaly detection algorithms *empirically*, i.e. with the help of a test set. Furthermore, by the above interpretation we can give a strong justification for the well-known heuristic of artificially sampling "labeled" samples, provided that the sampling plan is well chosen. In particular this enables us to propose a support vector machine (SVM) for anomaly detection for which we can easily establish universal consistency. Finally, we report some experiments which compare our SVM to other commonly used methods including the standard one-class SVM.

Keywords: unsupervised learning, anomaly detection, density levels, classification, SVMs

1. Introduction

Anomaly (or novelty) detection aims to detect anomalous observations from a system. In the machine learning version of this problem we cannot directly model the normal behaviour of the system since it is either unknown or too complex. Instead, we have some sample observations from which the normal behaviour is to be learned. This anomaly detection learning problem has many important applications including the detection of e.g. anomalous jet engine vibrations (see Nairac et al., 1999; Hayton et al., 2001; King et al., 2002), abnormalities in medical data (see Tarassenko et al., 1995; Campbell and Bennett, 2001), unexpected conditions in engineering (see Desforges et al., 1998) and network intrusions (see Manikopoulos and Papavassiliou, 2002; Yeung and Chow, 2002; Fan et al., 2001). For more information on these and other areas of applications as well as many methods for solving the corresponding learning problems we refer to the recent survey of Markou and Singh (2003a,b).

It is important to note that a typical feature of these applications is that only unlabeled samples are available, and hence one has to make some a-priori assumptions on anomalies in order to be able to distinguish between normal and anomalous future oberservations. One of the most common ways to define anomalies is by saying that *anomalies are not concentrated* (see e.g. Ripley, 1996;

Schölkopf and Smola, 2002). To make this precise let Q be our *unknown data-generating distribution* on the input space X which has a density h with respect to a *known reference distribution* μ on X. Obviously, the density level sets $\{h > \rho\}$, $\rho > 0$, describe the concentration of Q. Therefore to define anomalies in terms of the concentration one only has to fix a threshold level $\rho > 0$ so that a sample $x \in X$ is considered to be anomalous whenever $h(x) \leq \rho$. Consequently, our aim is to find the set $\{h \leq \rho\}$ to detect anomalous observations, or equivalently, the ρ-*level set* $\{h > \rho\}$ to describe normal observations.

We emphasize that given the data-generating distribution Q the choice of μ determines the density h, and consequently *anomalies are actually modeled by both* μ *and* ρ. Unfortunately, many popular algorithms are based on density estimation methods that implicitly assume μ to be the uniform distribution (e.g. Gaussian mixtures, Parzen windows and k-nearest neighbors density estimates) and therefore for these algorithms defining anomalies is restricted to the choice of ρ. With the lack of any further knowledge one might feel that the uniform distribution is a reasonable choice for μ, however there are situations in which a different μ is more appropriate. In particular, this is true if we consider a modification of the anomaly detection problem where μ is not known but can be sampled from. We will see that unlike many others our proposed method can handle both problems.

Finding level sets of an unknown density is also a well known problem in statistics which has some important applications different from anomaly detection. For example, it can be used for the problem of cluster analysis as described in by Hartigan (1975) and Cuevas et al. (2001), and for testing of multimodality (see e.g. Müller and Sawitzki, 1991; Sawitzki, 1996). Some other applications including estimation of non-linear functionals of densities, density estimation, regression analysis and spectral analysis are briefly described by Polonik (1995). Unfortunately, the algorithms considered in these articles cannot be used for the anomaly detection problem since the imposed assumptions on h are often tailored to the above applications and are in general unrealistic for anomalies.

One of the main problems of anomaly detection—or more precisely density level detection—is the lack of an empirical performance measure which allows us to compare the generalization performance of different algorithms by test samples. By interpreting the density level detection problem as binary classification with respect to an appropriate measure, we show that the corresponding empirical classification risk can serve as such an empirical performance measure for anomaly detection. Furthermore, we compare the excess classification risk with the standard performance measure for the density level detection problem. In particular, we show that both quantities are asymptotically equivalent and that simple inequalities between them are possible under mild conditions on the density h.

A well-known heuristic (see e.g. Fan et al., 2001; González and Dagupta, 2003; Yu et al., 2004; Theiler and Cai., 2003) for anomaly detection is to generate a labeled data set by assigning one label to the original unlabeled data and another label to a set of artificially generated data, and then apply a binary classification algorithm. By interpreting the density level detection problem as a binary classification problem we can show that this heuristic can be strongly justified provided that the sampling plan for the artificial samples is chosen in a certain way and the used classification algorithm is well-adopted to this plan. We will work out this justification in detail by showing how to modify the standard support vector machine (SVM) for classification, and establishing a consistency result for this modification. Finally we report some experiments comparing the modified SVM with some other commonly used algorithms including Gaussian maximum-likelihood methods, and the standard one-class SVM proposed by Schölkopf et al. (2001).

2. Detecting Density Levels is a Classification Problem

We begin with rigorously defining the density level detection (DLD) problem. To this end let $(X,\mathcal{A})$ be a measurable space and μ a *known* distribution on $(X,\mathcal{A})$. Furthermore, let Q be an *unknown* distribution on $(X,\mathcal{A})$ which has an *unknown* density h with respect to μ, i.e. $dQ = hd\mu$. Given a $\rho > 0$ the set $\{h > \rho\}$ is called the ρ-*level set* of the density h. Throughout this work we assume that $\{h = \rho\}$ is a μ-zero set and hence it is also a Q-zero set. For the density level detection problem and related tasks this is a common assumption (see e.g. Polonik, 1995; Tsybakov, 1997).

Now, the goal of the DLD problem is to find an estimate of the ρ-level set of h. To this end we need some information which in our case is given to us by a training set $T = (x_1,\dots,x_n) \in X^n$. We will assume in the following that T is i.i.d. drawn from Q. With the help of T a DLD algorithm constructs a function $f_T : X \to \mathbb{R}$ for which the set $\{f_T > 0\}$ is an estimate of the ρ-level set $\{h > \rho\}$ of interest. Since in general $\{f_T > 0\}$ does not excactly coincide with $\{h > \rho\}$ we need a *performance measure* which describes how well $\{f_T > 0\}$ approximates the set $\{h > \rho\}$. Probably the best known performance measure (see e.g. Tsybakov, 1997; Ben-David and Lindenbaum, 1997, and the references therein) for measurable functions $f : X \to \mathbb{R}$ is

$$\mathcal{S}_{\mu,h,\rho}(f) \ := \ \mu\Big(\{f > 0\} \vartriangle \{h > \rho\}\Big)\,,$$

where $\vartriangle$ denotes the symmetric difference. Obviously, the smaller $\mathcal{S}_{\mu,h,\rho}(f)$ is, the more $\{f > 0\}$ coincides with the ρ-level set of h and a function f minimizes $\mathcal{S}_{\mu,h,\rho}$ if and only if $\{f > 0\}$ is μ-almost surely identical to $\{h > \rho\}$. Furthermore, for a sequence of functions $f_n : X \to \mathbb{R}$ with $\mathcal{S}_{\mu,h,\rho}(f_n) \to 0$ we easily see that sign $f_n(x) \to 1_{\{h>\rho\}}(x)$ for μ-almost all $x \in X$, and since Q is absolutely continuous with respect to μ the same convergence holds Q-almost surely. Finally, it is important to note, that the performance measure $\mathcal{S}_{\mu,h,\rho}$ is insensitive to μ-zero sets. Since we cannot detect μ-zero sets using a training set T drawn from Q^n this feature is somehow natural for our model.

Although $\mathcal{S}_{\mu,h,\rho}$ seems to be well-adapted to our model, it has a crucial disadvantage in that we cannot compute $\mathcal{S}_{\mu,h,\rho}(f)$ since $\{h > \rho\}$ is unknown to us. Therefore, we have to *estimate* it. In our model the only information we can use for such an estimation is a *test set* $W = (\hat{x}_1,\dots,\hat{x}_m)$ which is i.i.d. drawn from Q. Unfortunately, there is no method known to estimate $\mathcal{S}_{\mu,h,\rho}(f)$ from W with *guaranteed* accuracy in terms of m, f, μ and ρ, and we strongly believe that such a method cannot exist. Because of this lack, we cannot *empirically* compare different algorithms in terms of the performance measure $\mathcal{S}_{\mu,h,\rho}$.

Let us now describe another performance measure which has merits similar to $\mathcal{S}_{\mu,h,\rho}$ but additionally has an empirical counterpart, i.e. a method to estimate its value with guaranteed accuracy by only using a test set. This performance measure is based on interpreting the DLD problem as a binary classification problem in which T is assumed to be positively labeled and infinitely many negatively labeled samples are available by the knowledge of μ. To make this precise we write $Y := \{-1,1\}$ and define

Definition 1 *Let μ and Q be probability measures on X and $s \in (0,1)$. Then the probability measure $Q \ominus_s \mu$ on $X \times Y$ is defined by*

$$Q \ominus_s \mu(A) \ := \ s\mathbb{E}_{x\sim Q}1_A(x,1) + (1-s)\mathbb{E}_{x\sim\mu}1_A(x,-1)$$

for all measurable subsets $A \subset X \times Y$. Here we used the shorthand $1_A(x,y) := 1_A((x,y))$.

Roughly speaking, the distribution $Q \ominus_s \mu$ measures the "1-slice" of $A \subset X \times Y$ by sQ and the "-1-slice" by $(1-s)\mu$. Moreover, the measure $P := Q \ominus_s \mu$ can obviously be associated with a binary classification problem in which positive samples are drawn from sQ and negative samples are drawn from $(1-s)\mu$. Inspired by this interpretation let us recall that the binary classification risk for a measurable function $f : X \to \mathbb{R}$ and a distribution P on $X \times Y$ is defined by

$$\mathcal{R}_P(f) \;=\; P\bigl(\{(x,y) : \operatorname{sign} f(x) \neq y\}\bigr)\,,$$

where we define $\operatorname{sign} t := 1$ if $t > 0$ and $\operatorname{sign} t = -1$ otherwise. Furthermore, the *Bayes risk* $\mathcal{R}_P$ of P is the smallest possible classification risk with respect to P, i.e.

$$\mathcal{R}_P \;:=\; \inf\Bigl\{\mathcal{R}_P(f)\,\Big|\, f : X \to \mathbb{R} \text{ measurable}\Bigr\}\,.$$

We will show in the following that learning with respect to $\mathcal{S}_{\mu,h,\rho}$ is equivalent to learning with respect to $\mathcal{R}_P(.)$. To this end we begin by computing the marginal distribution P_X and the *supervisor* $\eta(x) := P(y=1|x)$, $x \in X$, of $P := Q \ominus_s \mu$:

Proposition 2 *Let μ and Q be probability measures on X such that Q has a density h with respect to μ, and let $s \in (0,1)$. Then the marginal distribution of $P := Q \ominus_s \mu$ on X is $P_X = sQ + (1-s)\mu$. Furthermore, we P_X-a.s. have*

$$P(y=1|x) = \frac{sh(x)}{sh(x)+1-s}\,.$$

Proof As recalled in the appendix, $P(y=1|x)$, $x \in X$, is a regular conditional probability and hence we only have to check the condition of Corollary 19. To this end we first observe by the definition of $P := Q \ominus_s \mu$ that for all non-negative, measurable functions $f : X \times Y \to \mathbb{R}$ we have

$$\int_{X\times Y} f\,dP \;=\; s\int_X f(x,1)Q(dx) + (1-s)\int_X f(x,-1)\mu(dx)\,.$$

Therefore, for $A \in \mathcal{A}$ we obtain

$$\begin{aligned}
&\int_{A\times Y} \frac{sh(x)}{sh(x)+1-s} P(dx,dy)\\
=\;& s\int_A \frac{sh(x)}{sh(x)+1-s} h(x)\mu(dx) + (1-s)\int_A \frac{sh(x)}{sh(x)+1-s}\mu(dx)\\
=\;& \int_A sh(x)\mu(dx)\\
=\;& s\int_A 1_{X\times\{1\}}(x,1)Q(dx) + (1-s)\int_A 1_{X\times\{1\}}(x,-1)\mu(dx)\\
=\;& \int_{A\times Y} 1_{X\times\{1\}}\,dP\,.
\end{aligned}$$

■

Note that the formula for the marginal distribution P_X in particular shows that the μ-zero sets of X are exactly the P_X-zero sets of X. As an immediate consequence of the above proposition we additionally obtain the following corollary which describes the ρ-level set of h with the help of the supervisor η:

Corollary 3 *Let μ and Q be probability measures on X such that Q has a density h with respect to μ. For $\rho > 0$ we write $s := \frac{1}{1+\rho}$ and define $P := Q \ominus_s \mu$. Then for $\eta(x) := P(y = 1|x)$, $x \in X$, we have*

$$\mu\Big(\{\eta > 1/2\} \,\triangle\, \{h > \rho\}\Big) \;=\; 0\,,$$

i.e. $\{\eta > 1/2\}$ μ-almost surely coincides with $\{h > \rho\}$.

Proof By Proposition 2 we see that $\eta(x) > \frac{1}{2}$ is μ-almost surely equivalent to $\frac{sh(x)}{sh(x)+1-s} > \frac{1}{2}$ which is equivalent to $h(x) > \frac{1-s}{s} = \rho$. ■

The above results in particular show that every distribution $P := Q \ominus_s \mu$ with $dQ := hd\mu$ and $s \in (0,1)$ determines a triple (μ, h, ρ) with $\rho := (1-s)/s$ and vice-versa. In the following we therefore use the shorthand $\mathcal{S}_P(f) := \mathcal{S}_{\mu,h,\rho}(f)$.

Let us now compare $\mathcal{R}_P(.)$ with $\mathcal{S}_P(.)$. To this end recall, that binary classification aims to discriminate $\{\eta > 1/2\}$ from $\{\eta < 1/2\}$. In view of the above corollary it is hence no surprise that $\mathcal{R}_P(.)$ can serve as a surrogate for $\mathcal{S}_P(.)$ as the following theorem shows:

Theorem 4 *Let μ and Q be probability measures on X such that Q has a density h with respect to μ. Let $\rho > 0$ be a real number which satisfies $\mu(\{h = \rho\}) = 0$. We write $s := \frac{1}{1+\rho}$ and define $P := Q \ominus_s \mu$. Then for all sequences (f_n) of measurable functions $f_n : X \to \mathbb{R}$ the following are equivalent:*

i) $\mathcal{S}_P(f_n) \to 0$.

ii) $\mathcal{R}_P(f_n) \to \mathcal{R}_P$.

In particular, for a measurable function $f : X \to \mathbb{R}$ we have $\mathcal{S}_P(f) = 0$ if and only if $\mathcal{R}_P(f) = \mathcal{R}_P$.

Proof For $n \in \mathbb{N}$ we define $E_n := \{f_n > 0\} \,\triangle\, \{h > \rho\}$. Since by Corollary 3 we know $\mu(\{h > \rho\} \,\triangle\, \{\eta > \frac{1}{2}\}) = 0$ it is easy to see that the classification risk of f_n can be computed by

$$\mathcal{R}_P(f_n) \;=\; \mathcal{R}_P + \int_{E_n} |2\eta - 1| dP_X\,. \tag{1}$$

Now, $\{|2\eta - 1| = 0\}$ is a μ-zero set and hence a P_X-zero set. The latter implies that the measures $|2\eta - 1|dP_X$ and P_X are absolutely continuous with respect to each other, and hence we have

$$|2\eta - 1|dP_X(E_n) \to 0 \qquad \text{if and only if} \qquad P_X(E_n) \to 0\,.$$

Furthermore, we have already observed after Proposition 2 that P_X and μ are absolutely continuous with respect to each other, i.e. we also have

$$P_X(E_n) \to 0 \qquad \text{if and only if} \qquad \mu(E_n) \to 0\,.$$

Therefore, the assertion follows from $\mathcal{S}_P(f_n) = \mu(E_n)$. ■

Theorem 4 shows that instead of using $\mathcal{S}_P$ as a performance measure for the density level detection problem one can alternatively use the classification risk $\mathcal{R}_P(.)$. Therefore, we will establish some basic properties of this performance measure in the following. To this end we write $I(y,t) := \mathbf{1}_{(-\infty,0]}(yt)$, $y \in Y$ and $t \in \mathbb{R}$, for the standard classification loss function. With this notation we can compute $\mathcal{R}_P(f)$:

Proposition 5 *Let μ and Q be probability measures on X. For $\rho > 0$ we write $s := \frac{1}{1+\rho}$ and define $P := Q \ominus_s \mu$. Then for all measurable $f : X \to \mathbb{R}$ we have*

$$\mathcal{R}_P(f) \;=\; \frac{1}{1+\rho}\mathbb{E}_Q I(1, \operatorname{sign} f) + \frac{\rho}{1+\rho}\mathbb{E}_\mu I(-1, \operatorname{sign} f)\,.$$

Furthermore, for the Bayes risk we have

$$\mathcal{R}_P \;\leq\; \min\Big\{\frac{1}{1+\rho}, \frac{\rho}{1+\rho}\Big\}$$

and

$$\mathcal{R}_P \;=\; \frac{1}{1+\rho}\mathbb{E}_Q 1_{\{h\leq\rho\}} + \frac{\rho}{1+\rho}\mathbb{E}_\mu 1_{\{h>\rho\}}\,.$$

Proof The first assertion directly follows from

$$\begin{aligned}
\mathcal{R}_P(f) &= P\big(\{(x,y) : \operatorname{sign} f(x) \neq y\}\big) \\
&= P\big(\{(x,1) : \operatorname{sign} f(x) = -1\}\big) + P\big(\{(x,-1) : \operatorname{sign} f(x) = 1\}\big) \\
&= sQ\big(\{\operatorname{sign} f = -1\}\big) + (1-s)\mu\big(\{\operatorname{sign} f = 1\}\big) \\
&= s\mathbb{E}_Q I(1, \operatorname{sign} f) + (1-s)\mathbb{E}_\mu I(-1, \operatorname{sign} f)\,.
\end{aligned}$$

The second assertion directly follows from $\mathcal{R}_P \leq \mathcal{R}_P(1_X) \leq s$ and $\mathcal{R}_P \leq \mathcal{R}_P(-1_X) \leq 1-s$. Finally, for the third assertion recall that $f = 1_{\{h>\rho\}} - 1_{\{h\leq\rho\}}$ is a function which realizes the Bayes risk. ■

As described at the beginning of this section our main goal is to find a performance measure for the density level detection problem which has an empirical counterpart. In view of Proposition 5 the choice of an empirical counterpart for $\mathcal{R}_P(.)$ is rather obvious:

Definition 6 *Let μ be a probability measure on X and $\rho > 0$. Then for $T = (x_1, \ldots, x_n) \in X^n$ and a measurable function $f : X \to \mathbb{R}$ we define*

$$\mathcal{R}_T(f) \;:=\; \frac{1}{(1+\rho)n}\sum_{i=1}^{n} I(1, \operatorname{sign} f(x_i)) + \frac{\rho}{1+\rho}\mathbb{E}_\mu I(-1, \operatorname{sign} f)\,.$$

If we identify T with the corresponding empirical measure it is easy to see that $\mathcal{R}_T(f)$ is the classification risk with respect to the measure $T \ominus_s \mu$ for $s := \frac{1}{1+\rho}$. Then for measurable functions $f : X \to \mathbb{R}$, e.g. Hoeffding's inequality shows that $\mathcal{R}_T(f)$ approximates the true classification risk $\mathcal{R}_P(f)$ in a fast and controllable way.

It is highly interesting that the classification risk $\mathcal{R}_P(.)$ is strongly connected with another approach for the density level detection problem which is based on the so-called *excess mass* (see e.g. Hartigan, 1987; Müller and Sawitzki, 1991; Polonik, 1995; Tsybakov, 1997, and the references therein). To be more precise let us first recall that the excess mass of a measurable function $f : X \to \mathbb{R}$ is defined by

$$\mathcal{E}_P(f) \;:=\; Q(\{f > 0\}) - \rho\mu(\{f > 0\})\,,$$

where Q, ρ and μ have the usual meaning. The following proposition shows that $\mathcal{R}_P(.)$ and $\mathcal{E}_P(.)$ are essentially the same:

Proposition 7 *Let μ and Q be probability measures on X. For $\rho > 0$ we write $s := \frac{1}{1+\rho}$ and define $P := Q \ominus_s \mu$. Then for all measurable $f : X \to \mathbb{R}$ we have*

$$\mathcal{E}_P(f) \;=\; 1 - (1+\rho)\mathcal{R}_P(f)\,.$$

Proof We obtain the assertion by the following simple calculation:

$$\begin{aligned}
\mathcal{E}_P(f) &= Q(\{f > 0\}) - \rho\mu(\{f \geq 0\}) \\
&= 1 - Q(\{f \leq 0\}) - \rho\mu(\{f > 0\}) \\
&= 1 - Q\big(\{\operatorname{sign} f = -1\}\big) - \rho\mu\big(\{\operatorname{sign} f = 1\}\big) \\
&= 1 - (1+\rho)\mathcal{R}_P(f)\,.
\end{aligned}$$

■

If Q is an empirical measure based on a training set T in the definition of $\mathcal{E}_P(.)$ then we obtain am empirical performance measure which we denote by $\mathcal{E}_T(.)$. By the above proposition we have

$$\mathcal{E}_T(f) = 1 - \frac{1}{n}\sum_{i=1}^{n} I(1, \operatorname{sign} f(x_i)) - \rho\mathbb{E}_\mu I(-1, \operatorname{sign} f) = 1 - (1+\rho)\mathcal{R}_T(f) \tag{2}$$

for all measurable $f : X \to \mathbb{R}$. Now, given a class $\mathcal{F}$ of measurable functions from X to $\mathbb{R}$ the (empirical) excess mass approach considered e.g. by Hartigan (1987); Müller and Sawitzki (1991); Polonik (1995); Tsybakov (1997), chooses a function $f_T \in \mathcal{F}$ which maximizes $\mathcal{E}_T(.)$ within $\mathcal{F}$. By equation (2) we see that this approach is actually a type of empirical risk minimization (ERM). Surprisingly, this connection has not been observed, yet. In particular, the excess mass has only been considered as an algorithmic tool, but not as a performance measure. Instead, the papers dealing with the excess mass approach measures the performance by $\mathcal{S}_P(.)$. In their analysis an additional assumption on the behaviour of h around the level ρ is required. Since this condition can also be used to establish a quantified version of Theorem 4 we will recall it now:

Definition 8 *Let μ be a distribution on X and $h : X \to [0,\infty)$ be a measurable function with $\int h d\mu = 1$, i.e. h is a density with respect to μ. For $\rho > 0$ and $0 \leq q \leq \infty$ we say that h has* ρ-exponent *q if there exists a constant $C > 0$ such that for all sufficiently small $t > 0$ we have*

$$\mu\big(\{|h - \rho| \leq t\}\big) \;\leq\; Ct^q\,. \tag{3}$$

Condition (3) was first considered by Polonik (1995, Thm. 3.6). This paper also provides an example of a class of densities on $\mathbb{R}^d$, $d \geq 2$, which has exponent $q = 1$. Later, Tsybakov (1997, p. 956) used (3) for the analysis of a density level detection method which is based on a localized version of the empirical excess mass approach.

Interestingly, condition (3) is closely related to a concept for binary classification called the Tsybakov noise exponent (see e.g. Mammen and Tsybakov, 1999; Tsybakov, 2004; Steinwart and Scovel, 2004) as the following proposition proved in the appendix shows:

Proposition 9 *Let μ and Q be distributions on X such that Q has a density h with respect to μ. For $\rho > 0$ we write $s := \frac{1}{1+\rho}$ and define $P := Q \ominus_s \mu$. Then for $0 < q \leq \infty$ the following are equivalent:*

i) *h has* ρ-exponent *q*.

ii) *P has Tsybakov noise exponent q, i.e. there exists a constant* $C > 0$ *such that for all sufficiently small* $t > 0$ *we have*

$$P_X\big(|2\eta - 1| \leq t\big) \ \leq \ C \cdot t^q \tag{4}$$

In recent years Tsybakov's noise exponent has played a crucial role for establishing learning rates faster than $n^{-\frac{1}{2}}$ for certain algorithms (see e.g. Mammen and Tsybakov, 1999; Tsybakov, 2004; Steinwart and Scovel, 2004). Remarkably, it was already observed by Mammen and Tsybakov (1999), that the classification problem can be analyzed by methods originally developed for the DLD problem. However, to our best knowledge the exact relation between the DLD problem and binary classification has not been presented, yet. In particular, it has not been observed yet, that this relation opens a *non-heuristic* way to use classification methods for the DLD problem as we will discuss in the next section.

As already announced we can also establish inequalities between $\mathcal{S}_P$ and $\mathcal{R}_P(.)$ with the help of the ρ-exponent. This is done in the following theorem:

Theorem 10 *Let* $\rho > 0$ *and* μ *and* Q *be probability measures on* X *such that* Q *has a density* h *with respect to* μ*. For* $s := \frac{1}{1+\rho}$ *we write* $P := Q \ominus_s \mu$*. Then the following statements hold:*

i) *If* h *is bounded then there exists a constant* $c > 0$ *such that for all measurable* $f : X \to \mathbb{R}$ *we have*

$$\mathcal{R}_P(f) - \mathcal{R}_P \ \leq \ c\,\mathcal{S}_P(f)\,.$$

ii) *If* h *has* ρ*-exponent* $q \in (0,\infty]$ *then there exists a constant* $c > 0$ *such that for all measurable* $f : X \to \mathbb{R}$ *we have*

$$\mathcal{S}_P(f) \ \leq \ c\big(\mathcal{R}_P(f) - \mathcal{R}_P\big)^{\frac{q}{1+q}}\,.$$

Proof The first assertion directly follows from (1) and Proposition 2. The second assertion follows from Proposition 9 and a result of Tsybakov (2004, Prop. 1). ■

Remark 11 *We note that many of the results of this section can be generalized to the case where* Q *is not absolutely continuous with respect to* μ*. Indeed, select an auxilliary measure* ν *such that both* Q *and* μ *are absolutely continuous with respect to* ν*. For example one could choose* $\nu = \frac{Q+\mu}{2}$*. Consequently we have* $Q = h_1\nu$ *and* $\mu = h_2\nu$ *for some real valued functions* h_1 *and* h_2*. Then Proposition 2 holds with* $h(x) := \frac{h_1(x)}{h_2(x)}$*, where one defines the righthand side to be* 0 *when* $h_1(x) = h_2(x) = 0$*. One can also show that* h *is* P_X*-a.s. independent of the choice of* ν*. Corollary 3 holds where the measure of the symmetric difference is evaluated with either* Q *or* μ*. However it appears that only the "*$\mathcal{R}_P(f_n) \to \mathcal{R}_P \Rightarrow \mathcal{S}_P(f_n) \to 0$*" assertion of Theorem 4 holds instead of equivalence. Finally, Propositions 5 and 7 hold, Proposition 9 holds with a suitable generalization of Definition 8 of* ρ*-exponent, and the second assertion of Theorem 10 holds.*

3. A Support Vector Machine for Density Level Detection

In the previous section we have shown that the DLD problem can be interpreted as a binary classification problem in which one conditional class probability is known. We now show that this interpretation has far reaching algorithmic consequences. To this end let us assume that we give each sample of our training set $T = (x_1,\dots,x_n)$ drawn from Q the label 1. Additionally we generate a second training set $T' = (x'_1,\dots,x'_{n'})$ from μ and label each sample of it with -1. Merging these labeled sample sets gives a new training set which then can be used by a binary classification algorithm. By our considerations in the previous section it seems reasonable to expect that the used classification algorithm actually learns the DLD problem provided that the algorithm is well-adjusted to the sample set sizes and the parameter ρ.

In the following we work this high-level approach out by constructing an SVM for the DLD problem. To this end let $k : X \times X \to \mathbb{R}$ be a positive definite kernel with reproducing kernel Hilbert space (RKHS) H. Furthermore, let μ be a known probability measure on X and $l : Y \times \mathbb{R} \to [0,\infty)$ be the *hinge* loss function, i.e. $l(y,t) := \max\{0, 1-yt\}$, $y \in Y$, $t \in \mathbb{R}$. Then for a training set $T = (x_1,\dots,x_n) \in X^n$, a regularization parameter $\lambda > 0$, and $\rho > 0$ we initially define

$$f_{T,\mu,\lambda} \;:=\; \arg\min_{f\in H} \lambda \|f\|_H^2 + \frac{1}{(1+\rho)n}\sum_{i=1}^{n} l(1, f(x_i)) + \frac{\rho}{1+\rho}\mathbb{E}_{x\sim\mu} l(-1, f(x))\,, \tag{5}$$

and

$$(\tilde f_{T,\mu,\lambda}, \tilde b_{T,\mu,\lambda}) \;:=\; \arg\min_{\substack{f\in H\\ b\in R}} \lambda \|f\|_H^2 + \frac{1}{(1+\rho)n}\sum_{i=1}^{n} l(1, f(x_i)+b) + \frac{\rho}{1+\rho}\mathbb{E}_{x\sim\mu} l(-1, f(x)+b)\,. \tag{6}$$

The decision function of the *SVM without offset* is $f_{T,\mu,\lambda} : X \to \mathbb{R}$ and analogously, the *SVM with offset* has the decision function $\tilde f_{T,\mu,\lambda} + \tilde b_{T,\mu,\lambda} : X \to \mathbb{R}$.

Although the measure μ is known, almost always the expectation $\mathbb{E}_{x\sim\mu} l(-1, f(x))$ can only be numerically computed which requires finitely many function evaluations of f. If the integrand of this expectation was smooth we could use some known deterministic methods to choose these function evaluations efficiently. However, since the hinge loss is not differentiable there is no such method known to us. According to our above plan we will therefore use points $T' := (x'_1,\dots,x'_{n'})$ which are randomly sampled from μ to approximate $\mathbb{E}_{x\sim\mu} l(-1, f(x))$ by $\frac{1}{n'}\sum_{i=1}^{n'} l(-1, f(x'_i))$. We denote the corresponding approximate solutions of (5) and (6) by $f_{T,T',\lambda}$ and $(\tilde f_{T,T',\lambda}, \tilde b_{T,T',\lambda})$, respectively. Furthermore, in these cases the formulations (5) and (6) are identical to the standard L1-SVM formulations besides the weighting factors in front of the empirical error terms. Therefore, the derivation of the corresponding dual problems is straightforward. For example, the dual problem for (6) can be written as follows:

$$\begin{aligned}
\max\quad & \sum_{i=1}^{n}\alpha_i + \sum_{i=1}^{n'}\alpha'_i - \tfrac12\sum_{i,j=1}^{n}\alpha_i\alpha_j k(x_i,x_j) - \tfrac12\sum_{i,j=1}^{n'}\alpha'_i\alpha'_j k(x'_i,x'_j) + \sum_{i,j=1}^{n,n'}\alpha_i\alpha'_j k(x_i,x'_j)\\
\text{s.t.}\quad & \sum_{i=1}^{n}\alpha_i - \sum_{i=1}^{n'}\alpha'_i = 0,\\
& 0 \le \alpha_i \le \tfrac{2}{\lambda(1+\rho)n}, \qquad i = 1,\dots,n,\\
& 0 \le \alpha'_i \le \tfrac{2\rho}{\lambda(1+\rho)n'}, \qquad i = 1,\dots,n'.
\end{aligned} \tag{7}$$

The fact that the SVM for DLD essentially coincides with the standard L1-SVM also allows us to modify many known results for these algorithms. For simplicity we will only state a consistency result which describes the case where we use $n' = n$ random samples from μ in order to approximate the expectation with respect to μ. However, it is straight forward to extend the result to the more general case of $n' = rn$ samples for some positive $r \in \mathbb{Q}$. In order to formulate the result we have to recall the notion of universal kernels (see Steinwart, 2001). To this end let X be a compact metric space, say a closed and bounded subset of $\mathbb{R}^d$. We denote the space of all continuous functions on X by $C(X)$. As usual, this space is equipped with the supremum norm $\|.\|_\infty$. Then the RKHS H of a continuous kernel k on X is embedded into $C(X)$, i.e. $H \subset C(X)$, where the inclusion is continuous. We say that the kernel k is *universal*, if in addition H is dense in $C(X)$, i.e. for every $f \in C(X)$ and every $\varepsilon > 0$ there exists a $g \in H$ with $\|f - g\|_\infty < \varepsilon$. Some examples of universal kernels including the Gaussian RBF kernels were presented by Steinwart (2001).

Now we can formulate the announced result:

Theorem 12 (Universal consistency) *Let X be a compact metric space and k be a universal kernel on X. Furthermore, let $\rho > 0$, and μ and Q be probability measures on X such that Q has a density h with respect to μ. For $s := \frac{1}{1+\rho}$ we write $P := Q \ominus_s \mu$. Then for all sequences (λ_n) of positive numbers with $\lambda_n \to 0$ and $n\lambda_n^2 \to \infty$ and for all $\varepsilon > 0$ we have*

$$(Q \otimes \mu)^n \Big((T,T') \in X^n \times X^n : \mathcal{R}_P(f_{T,T',\lambda_n}) \leq \mathcal{R}_P + \varepsilon \Big) \to 0\,,$$

for $n \to \infty$. The same result holds for the SVM with offset if one replaces the condition $n\lambda_n^2 \to \infty$ by the slightly stronger assumption $n\lambda_n^2 / \log n \to \infty$. Finally, for both SVMs it suffices to assume $n\lambda_n^{1+\delta} \to \infty$ for some $\delta > 0$ if one uses a Gaussian RBF kernel.

Sketch of the Proof Let us introduce the shorthand $\nu = Q \otimes \mu$ for the product measure of Q and μ. Moreover, for a measurable function $f : X \to \mathbb{R}$ we define the function $l \odot f : X \times X \to \mathbb{R}$ by

$$l \odot f(x,x') := \frac{1}{1+\rho} l(1, f(x)) + \frac{\rho}{1+\rho} l(-1, f(x'))\,, \qquad x,x' \in X.$$

Furthermore, we write $l \circ f(x,y) := l(y, f(x))$, $x \in X$, $y \in Y$. Then it is easy to check that we always have $\mathbb{E}_\nu l \odot f = \mathbb{E}_P l \circ f$. Analogously, we see $\mathbb{E}_{T \otimes T'} l \odot f = \mathbb{E}_{T \ominus_s T'} l \circ f$, if $T \otimes T'$ denotes the product measure of the empirical measures based on T and T'. Now, using Hoeffding's inequality for ν it is easy to establish a concentration inequality in the sense of Steinwart (2005, Lem. III.5). The rest of the proof is analogous to the steps of Steinwart (2005). ■

Recall that by Theorem 4 consistency with respect to $\mathcal{R}_P(.)$ is equivalent to consistency with respect to $\mathcal{S}_P(.)$. Therefore we immediately obtain the following corollary

Corollary 13 *Under the assumptions of Theorem 12 both the DLD-SVM with offset and without offset are universally consistent with respect to $\mathcal{S}_P(.)$, i.e. $\mathcal{S}_P(\tilde{f}_{T,\mu,\lambda} + \tilde{b}_{T,\mu,\lambda}) \to 0$ and $\mathcal{S}_P(f_{T,T',\lambda_n}) \to 0$ in probability.*

Remark 14 *We have just seen that our DLD-SVM whose design was based on the plan described in the beginning of this section can learn arbitrary DLD problems. It should be almost clear that* *a similar approach and analysis is possible for many other classification algorithms. This gives a strong justification for the well-known heuristic of adding artificial samples to anomaly detection problems with unlabeled data. However, it is important to note that this justification only holds for the above sampling plan and suitably adjusted classification algorithms, and that other, heuristic sample plans may actually lead to bad learning performance (cf. the second part of Section 5)*

4. Experiments

We present experimental results for anomaly detection problems where the set X is a subset of $\mathbb{R}^d$. A total of four different learning algorithms are used to produce functions f which declare the set $\{x : f(x) \leq 0\}$ anomalous. A distinct advantage of the formulation in Section 2 is that it allows us to make *quantitative* comparisons of different functions by comparing estimates of their risk $\mathcal{R}_P(f)$ which can be computed from sample data. In particular consider a data set pair (S, S') where S contains samples drawn from Q and S' contains samples drawn from μ (in what follows (S, S') is either training data, validation data, or test data). Based on Definition 6 we define the empirical risk of f with respect to (S, S') to be

$$\mathcal{R}_{(S,S')}(f) = \frac{1}{(1+\rho)|S|}\sum_{x\in S} I(1, \operatorname{sign} f(x)) + \frac{\rho}{(1+\rho)|S'|}\sum_{x\in S'} I(-1, \operatorname{sign} f(x)). \tag{8}$$

A smaller risk indicates a better solution to the DLD problem. Since the risk $\mathcal{R}_P(\cdot)$ depends explicitly on ρ additional insight into the performance of f can be obtained from the two error terms. Specifically the quantity $\frac{1}{|S|}\sum_{x\in S} I(1, \operatorname{sign} f(x))$ is an estimate of $Q(\{f \leq 0\})$ which we call the *alarm rate* (i.e. the rate at which samples will be labeled anomalous by f), and the quantity $\frac{1}{|S'|}\sum_{x\in S'} I(-1, \operatorname{sign} f(x))$ is an estimate of $\mu(\{f > 0\})$ which we call the *volume* of the predicted normal set. There is an obvious trade-off between these two quantities, i.e. for the optimal solutions for fixed ρ smaller alarm rates correspond to larger volumes and vice versa. Also, from the expression for the risk in Proposition 5 it is clear that for any two functions with the same alarm rate we prefer the function with the smaller volume and vice versa. More generally, when comparing different solution methods it is useful to consider the values of these quantities that are achieved by varying the value of ρ in the design process. Such *performance curves* are presented in the comparisons below.

We consider three different anomaly detection problems, two are synthetic and one is an application in cybersecurity. In each case we define a problem instance to be a triplet consisting of samples from Q, samples from μ, and a value for the density level ρ. We compare four learning algorithms that accept a problem instance and automatically produce a function f: the density level detection support vector machine (DLD–SVM), the one-class support vector machine (1CLASS–SVM), the Gaussian maximum-likelihood (GML) method, the mixture of Gaussians maximum-likelihood (MGML) method.[1] The first is the algorithm introduced in this paper, the second is an algorithm based on the the one-class support vector machine introduced by Schölkopf et al. (2001) and the others (including the Parzen windows method) are based on some of the most common parametric and non-parametric statistical methods for density-based anomaly detection in $\mathbb{R}^d$. Each of the four learning algorithms is built on a core procedure that contains one or more free parameters. The availability of a computable risk estimate makes it possible to determine values for these parameters

1. We also experimented with a Parzen windows method, but do not include the results because they were substantially worse than the other methods in every case.

using a principled approach that is applied uniformly to all four core procedures. In particular this is accomplished as follows in our experiments. The data in each problem instance is partitioned into three pairs of sets; the training sets (T,T'), the validation sets (V,V') and the test sets (W,W'). The core procedures are run on the training sets and the values of the free parameters are chosen to minimize the empirical risk (8) on the validation sets. The test sets are used to estimate performance. We now describe the four learning algorithms in detail.

In the DLD–SVM algorithm we employ the SVM *with offset* described in Section 3 with a Gaussian RBF kernel

$$k(x,x') = e^{-\sigma^2\|x-x'\|^2}.$$

With λ and σ^2 fixed and the expected value $\mathbb{E}_{x\sim\mu}l(-1, f(x)+b)$ in (6) replaced with an empirical estimate based on T' this formulation can be solved using, for example, the `C-SVC` option in the `LIBSVM` software (see Chang and Lin, 2004) by setting $C=1$ and setting the class weights to $w_1 = 1/\big(\lambda|T|(1+\rho)\big)$ and $w_{-1} = \rho/\big(\lambda|T'|(1+\rho)\big)$. The regularization parameters λ and σ^2 are chosen to (approximately) minimize the empirical risk $\mathcal{R}_{(V,V')}(f)$ on the validation sets. This is accomplished by employing a grid search over λ and a combined grid/iterative search over σ^2. In particular, for each value of λ from a fixed grid we seek a minimizer over σ^2 by evaluating the validation risk at a coarse grid of σ^2 values and then performing a Golden search over the interval defined by the two σ^2 values on either side of the coarse grid minimum.[2] As the overall search proceeds the (λ,σ^2) pair with the smallest validation risk is retained.

The 1CLASS–SVM algorithm is based on the one-class support vector machine introduced by Schölkopf et al. (2001). Recall that this method neither makes the assumption that there is a reference distribution μ nor uses T' in the production of its decision function f. Consequently it may be harder to compare the empirical results of the 1CLASS–SVM with those of the other methods in a fair way. Again we employ the Gaussian RBF kernel with width parameter σ^2. The one-class formulation of Schölkopf et al. (2001) contains a parameter ν which controls the size of the set $\{x\in T : f(x)\le 0\}$ (and therefore controls the measure $Q(\{f\le 0\})$ through generalization). With ν and σ^2 fixed a solution can be obtained using the `one-class-SVM` option in the `LIBSVM` software. To use this 1-class algorithm to solve an instance of the DLD problem we determine ν automatically as a function of ρ. In particular both ν and σ^2 are chosen to (approximately) minimize the validation risk using the search procedure described above for the DLD–SVM where the grid search for λ is replaced by a Golden search (over $[0,1]$) for ν.

The GML algorithm produces a function $f = g - t$ where t is an offset and g is a Gaussian probability density function whose mean and inverse covariance are determined from maximum likelihood estimates formed from the training data T (see e.g. Duda et al., 2000). In particular the inverse covariance takes the form $(\Sigma+\lambda I)^{-1}$ where Σ is the maximum likelihood covariance estimate and the regularization term λI is a scaled identity matrix which guarantees that the inverse is well-defined and numerically stable. Once the parameters of g are determined the offset t is chosen to minimize the training risk $\mathcal{R}_{(T,T')}$. The regularization parameter λ is chosen to (approximately) minimize the validation risk by searching a fixed grid of λ values.

The MGML algorithm is essentially the same as the GML method except that g is a mixture of K Gaussians whose maximum likelihood parameter estimates are determined using the Expectation-Maximization (EM) algorithm of Dempster et al. (1977). The same regularization parameter is used

2. If the minimum occurs at more than one grid point or at an end point the Golden search interval is defined by the nearest grid points that include all minimal values.

	Train	Validate	Test
Number of Q samples	1000	500	100,000
Number of μ samples	2000	2000	100,000

λ grid (DLD–SVM/GML/MGML)	1.0, 0.5, 0.1, 0.05, 0.01, ..., 0.0000005, 0.0000001
σ^2 grid (DLD–SVM/1CLASS–SVM)	0.001, 0.01, 0.05, 0.1, 0.5, 1.0, 5.0, 10.0, 100.0

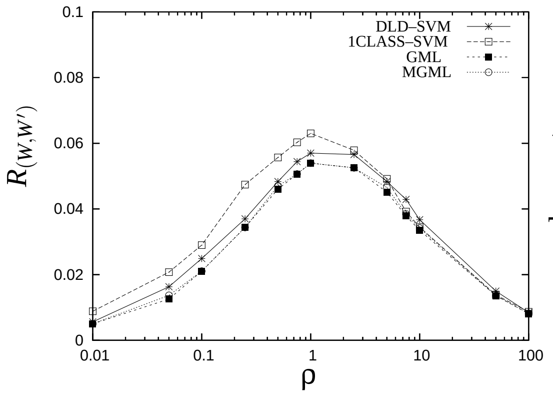
Table 1: Parameters for experiments 1 and 2.

for all inverse covariance estimates and both λ and K are chosen to (approximately) minimize the validation risk by searching a fixed grid of (λ, K) values.

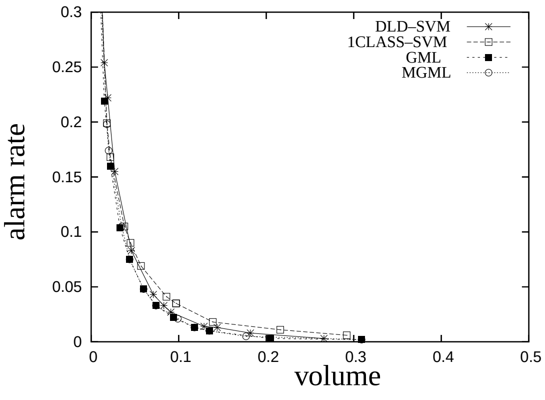
Data for the first experiment are generated using an approach designed to mimic a type of real problem where x is a feature vector whose individual components are formed as linear combinations of raw measurements and therefore the central limit theorem is used to invoke a Gaussian assumption for Q. Specifically, samples of the random variable $x \sim Q$ are generated by transforming samples of a random variable u that is uniformly distributed over $[0,1]^{27}$. The transform is $x = Au$ where A is a 10–by–27 matrix whose rows contain between $m = 2$ and $m = 5$ non-zero entries with value $1/m$ (i.e. each component of x is the average of m uniform random variables). Thus Q is approximately Gaussian with mean $(0.5, 0.5)$ and support $[0,1]^{10}$. Partial overlap in the nonzero entries across the rows of A guarantee that the components of x are partially correlated. We chose μ to be the uniform distribution over $[0,1]^{10}$. Data for the second experiment are identical to the first except that the vector $(0,0,0,0,0,0,0,0,0,1)$ is added to the samples of x with probability 0.5. This gives a bi-modal distribution Q that approximates a mixture of Gaussians. Also, since the support of the last component is extended to $[0,2]$ the corresponding component of μ is also extended to this range. A summary of the data and algorithm parameters for experiments 1 and 2 is shown in Table 1. Note that the test set sizes are large enough to provide very accurate estimates of the risk.

The four learning algorithms were applied for values of ρ ranging from .01 to 100 and the results are shown in Figure 1. Figures 1(a) and 1(c) plot the empirical risk $R_{(W,W')}$ versus ρ while Figures 1(b) and 1(d) plot the corresponding performance curves. Since the data is approximately Gaussian it is not surprising that the best results are obtained by GML (first experiment) and MGML (both experiments). However, for most values of ρ the next best performance is obtained by DLD–SVM (both experiments). The performance of 1CLASS–SVM is clearly worse than the other three at smaller values of ρ (i.e. larger values of the volume), and this difference is more substantial in the second experiment. In addition, although we do not show it, this difference is even more pronounced (in both experiments) at smaller training and validation set sizes. These results are significant because values of ρ substantially larger than one appear to have little utility here since they yield alarm rates that do not conform to our notion that anomalies are rare events. In addition $\rho \gg 1$ appears to have little utility in the general anomaly detection problem since it defines anomalies in regions where the concentration of Q is much larger than the concentration of μ, which is contrary to our premise that anomalies are not concentrated.

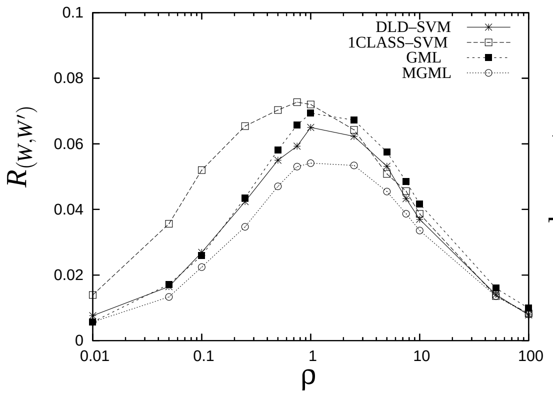
The third experiment considers an application in cybersecurity. The goal is to monitor the network traffic of a computer and determine when it exhibits anomalous behavior. The data for this experiment was collected from an active computer in a normal working environment over the course of 16 months. The features in Table 2 were computed from the outgoing network traffic.



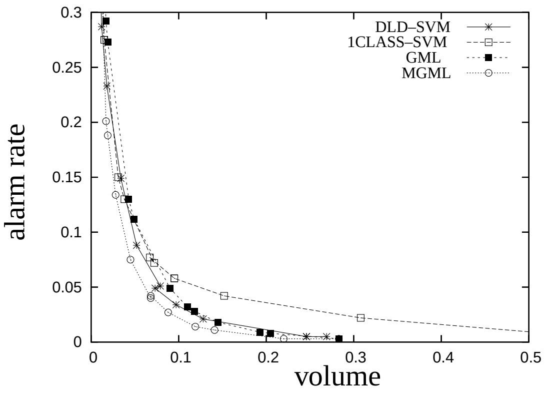
(a) Risk curves for $Q \approx$ Gaussian.



(b) Performance curves for $Q \approx$ Gaussian.



(c) Risk curves for $Q \approx$ Gaussian mixture.



(d) Performance curves for $Q \approx$ Gaussian mixture.

Figure 1: Synthetic data experiments.

The averages were computed over one hour time windows giving a total of 11664 feature vectors. The feature values were normalized to the range $[0,1]$ and treated as samples from Q. Thus Q has support in $[0,1]^{12}$. Although we would like to choose μ to capture a notion of anomalous behavior for this application, only the DLD–SVM method allows such a choice. Thus, since both GML and MGML define densities with respect to a uniform measure and we wish to compare with these methods, we chose μ to be the uniform distribution over $[0,1]^{12}$. A summary of the data and algorithm parameters for this experiment is shown in Table 3. Again, we would like to point out that this choice may actually penalize the 1CLASS-SVM since this method is not based on the notion of a reference measure. However, we currently do not know any other approach which treats the 1CLASS-SVM with its special strucure in a fairer way.

The four learning algorithms were applied for values of ρ ranging from .005 to 50 and the results are summarized by the empirical risk curve in Figure 2(a) and the corresponding performance curve in Figure 2(b). The empirical risk values for DLD–SVM and MGML are nearly identical except for $\rho = 0.05$ where the MGML algorithm happened to choose $K = 1$ to minimize the validation risk

Feature Number	**Description**
1	Number of sessions
2	Average number of source bytes per session
3	Average number of source packets per session
4	Average number of source bytes per packet
5	Average number of destination bytes per session
6	Average number of destination packets per session
7	Average number of destination bytes per packet
8	Average time per session
9	Number of unique destination IP addresses
10	Number of unique destination ports
11	Number of unique destination IP addresses divided by total number of sessions
12	Number of unique destination ports divided by total number of sessions

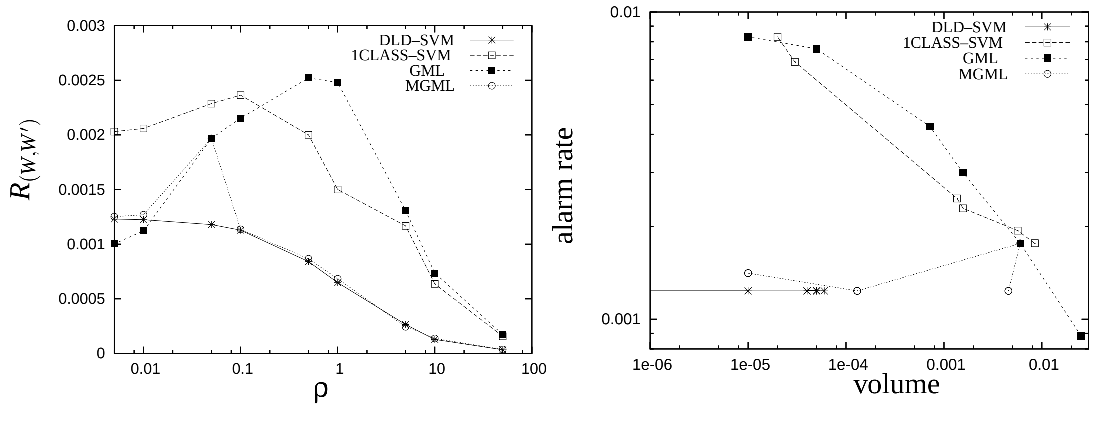
Table 2: Outgoing network traffic features.

	Train	Validate	Test
Number of Q samples	4000	2000	5664
Number of μ samples	10,000	100,000	100,000

λ grid (DLD–SVM/GML/MGML)	0.1, 0.01, 0.001, . . . , 0.0000001
σ^2 grid (DLD–SVM/1CLASS–SVM)	0.001, 0.01, 0.05, 0.1, 0.5, 1.0, 5.0, 10.0, 100.0

Table 3: Parameters for cybersecurity experiment.

(i.e. the MGML and GML solutions are identical at $\rho = 0.05$). Except for this case the empirical risk values for DLD–SVM and MGML are much better than 1CLASS–SVM and GML at nearly all values of ρ. The performance curves confirm the superiority of DLD–SVM and MGML, but also reveal differences not easily seen in the empirical risk curves. For example, all four methods produced some solutions with identical performance estimates for different values of ρ which is reflected by the fact that the performance curves show fewer points than the corresponding empirical risk curves.



(a) Risk curves. (b) Performance curves.

Figure 2: Cybersecurity experiment.

5. Discussion

A review of the literature on anomaly detection suggests that there are many ways to characterize anomalies (see e.g. Markou and Singh, 2003a,b). In this work we assumed that anomalies are not concentrated. This assumption can be specified by choosing a reference measure μ which determines a density and a level value ρ. The density then quantifies the degree of concentration and the density level ρ establishes a threshold on the degree that determines anomalies. Thus, μ and ρ play key roles in the *definition* of anomalies. In practice the user chooses μ and ρ to capture some notion of anomaly that he deems relevant to the application.

This paper advances the existing state of "density based" anomaly detection in the following ways.

- Most existing algorithms make an implicit choice of μ (usually the Lebesgue measure) whereas our approach allows μ to be any measure that defines a density. Therefore we accommodate a larger class of anomaly detection problems. This flexibility is in particular important when dealing with e.g. categorical data. In addition, it is the key ingredient when dealing with *hidden classification problems*, which we will discuss below.

- Prior to this work there have been no methods known to rigorously estimate the performance based on *unlabeled* data. Consequently, it has been difficult to compare different methods for anomaly detection in practice. We have introduced an empirical performance measure, namely the empirical classification risk, that enables such a comparision. In particular, it can be used to perform a model selection based on cross validation. Furthermore, the infinite sample version of this empirical performance measure is asymptotically equivalent to the standard performance measure for the DLD problem and under mild assumptions inequalities between them have been obtained.

- By interpreting the DLD problem as a binary classification problem we can use well-known classification algorithms for DLD if we generate artificial samples from μ. We have demonstrated this approach which is a rigorous variant of a well-known heuristic for anomaly detection in the formulation of the DLD-SVM.

These advances have created a situation in which much of the knowledge on classification can now be used for anomaly detection. Consequently, we expect substantial advances in anomaly detection in the future.

Finally let us consider a different learning scenario in which anomaly detection methods are also commonly employed. In this scenario we are interested in solving a binary classification problem given only unlabeled data. More precisely, suppose that there is a distribution ν on $X \times \{-1,1\}$ and the samples are obtained from the *marginal* distribution ν_X on X. Since labels exist but are hidden from the user we call this a *hidden classification problem (HCP)*. Hidden classification problems for example occur in network intrusion detection problems where it is impractical to obtain labels. Obviously, solving a HCP is intractable if no assumptions are made on the labeling process. One such assumption is that one class consists of anomalous, lowly concentrated samples (e.g. intrusions) while the other class reflects normal behaviour. Making this assumption rigorous requires the specification of a reference measure μ and a threshold ρ. Interestingly, when ν_X is absolutely continuous[3] with respect to $\nu(\,.\,|y=1)$ solving the DLD problem with

$$\begin{aligned} Q &:= \nu_X \\ \mu &:= \nu(\,.\,|y=1) \\ \rho &:= 2\nu(X\times\{1\}) \end{aligned}$$

gives the Bayes classifier for the binary classification problem associated with ν. Therefore, in principle the DLD formalism can be used to solve the binary classification problem. In the HCP however, although information about $Q=\nu_X$ is given to us by the samples, we must rely entirely on first principle knowledge to *guess* μ and ρ. Our inability to choose μ and ρ correctly determines the *model error* that establishes the limit on how well the classification problem associated with ν can be solved with unlabeled samples. This means for example that when an anomaly detection method is used to produce a classifier f for a HCP its anomaly detection performance $\mathcal{R}_P(f)$ with $P := Q \ominus_s \mu$ and $s := \frac{1}{1+\rho}$ may be very different from its hidden classification performance $\mathcal{R}_\nu(f)$. In particular $\mathcal{R}_P(f)$ may be very good, i.e. very close to $\mathcal{R}_P$, while $\mathcal{R}_\nu(f)$ may be very poor, i.e. far above $\mathcal{R}_\nu$. Another consequence of the above considerations is that the common practice of measuring the performance of anomaly detection algorithms on (hidden) binary classification problems is problematic. Indeed, the obtained classification errors depend on the model error and thus they provide an inadequate description how well the algorithms solve the anomaly detection problem.

3. This assumption is actually superfluous by Remark 11.

Furthermore, since the model error is strongly influenced by the particular HCP it is almost impossible to generalize from the reported results to more general statements on the hidden classification performance of the considered algorithms.

In conclusion although there are clear similiarities between the use of the DLD formalism for anomaly detection and its use for the HCP there is also an important difference. In the first case the specification of μ and ρ determines the *definition* of anomalies and therefore there is no model error, whereas in the second case the model error is determined by the choice of μ and ρ.

Acknowledgments

We would like to thank J. Theiler who inspired this work when giving a talk on a recent paper (Theiler and Cai., 2003).

Appendix A. Regular Conditional Probabilities

In this apendix we recall some basic facts on conditional probabilities and regular conditional probabilities. We begin with

Definition 15 *Let $(X,\mathcal{A},P)$ be a probability space and $\mathcal{C}\subset\mathcal{A}$ a sub-σ-algebra. Furthermore, let $A\in\mathcal{A}$ and $g:(X,\mathcal{C})\to\mathbb{R}$ be $P_{|\mathcal{C}}$-integrable. Then g is called a conditional probability of A with respect to $\mathcal{C}$ if*

$$\int_C 1_A dP \;=\; \int_C g dP$$

for all $C\in\mathcal{C}$. In this case we write $P(A|\mathcal{C}) := g$.

Furthermore we need the notion of regular conditional probabilities. To this end let $(X,\mathcal{A})$ and $(Y,\mathcal{B})$ be measurable spaces and P be a probability measure on $(X\times Y,\mathcal{A}\otimes\mathcal{B})$. Denoting the projection of $X\times Y$ onto X by π_X we write $\pi_X^{-1}(\mathcal{A})$ for the sub-σ-Algebra of $\mathcal{A}\otimes\mathcal{B}$ which is induced by π_X. Recall, that this sub-σ-Algebra is generated by the collection of the sets $A\times Y$, $A\in\mathcal{A}$. For later purpose, we also notice that this collection is obviously stable against finite intersections. Finally, P_X denotes the marginal distribution of P on X, i.e. $P_X(A)=P(\pi_X^{-1}(A))$ for all $A\in\mathcal{A}$.

Now let us recall the definition of regular conditional probabilities:

Definition 16 *A map $P(\,.\,|\,.\,):\mathcal{B}\times X\to[0,1]$ is called a* regular conditional probability *of P if the following conditions are satisfied:*

i) $P(\,.\,|x)$ is a probability measure on $(Y,\mathcal{B})$ for all $x\in X$.

ii) $x\mapsto P(B|x)$ is $\mathcal{A}$-measurable for all $B\in\mathcal{B}$.

iii) For all $A\in\mathcal{A}$, $B\in\mathcal{B}$ we have

$$P(A\times B) \;=\; \int_A P(B|x)P_X(dx)\,.$$

Under certain conditions such regular conditional probabilities exist. To be more precise, recall that a topological space is called *Polish* if its topology is metrizable by a complete, separable metric. The following theorem in the book of Dudley (2002, Thm. 10.2.2) gives a sufficient condition for the existence of a regular conditional probability:

Theorem 17 *If Y is a Polish space then a regular conditional probability $P(\,.\,|\,.\,) : \mathcal{B} \times X \to [0,1]$ of P exists.*

Regular conditional probabilities play an important role in binary classification problems. Indeed, given a probability measure P on $X \times \{-1,1\}$ the aim in classification is to approximately find the set $\{P(y=1|x) > \frac{1}{2}\}$, where "approximately" is measured by the classification risk.

Let us now recall the connection between conditional probabilities and regular conditional probabilities (see Dudley, 2002, p. 342 and Thm. 10.2.1):

Theorem 18 *If a conditional probability $P(\,.\,|\,.\,) : \mathcal{B} \times X \to [0,1]$ of P exists then we P-a.s. have*

$$P(B|x) \;=\; P\big(X \times B | \pi_X^{-1}(\mathcal{A})\big)(x,y)\,.$$

As an immediate consequence of this theorem we can formulate the following "test" for regular conditional probabilities.

Corollary 19 *Let $B \in \mathcal{B}$ and $f : X \to [0,1]$ be $\mathcal{A}$-measurable. Then $f(x) = P(B|x)$ P_X-a.s. if*

$$\int_{A\times Y} f \circ \pi_X \, dP = \int_{A \times Y} 1_{X\times B} \, dP$$

for all $A \in \mathcal{A}$.

Proof The assertion follows from Theorem 18, the definition of conditional probabilities and the fact that the collection of the sets $A \times Y$, $A \in \mathcal{A}$ is stable against finite intersections. ■

Appendix B. Proof of Proposition 9

Proof of Proposition 9 By Proposition 2 we have $|2\eta - 1| = \left|\frac{h-\rho}{h+\rho}\right|$ and hence we observe

$$\begin{aligned}\big\{|2\eta-1| \le t\big\} &= \big\{|h-\rho| \le (h+\rho)t\big\} \\ &= \big\{-(h+\rho)t \le h-\rho \le (h+\rho)t\big\} \\ &= \left\{\frac{1-t}{1+t}\rho \le h \le \frac{1+t}{1-t}\rho\right\},\end{aligned}$$

whenever $0 < t < 1$.

Now let us first assume that P has Tsybakov exponent $q > 0$ with some constant $C > 0$. Then using

$$\big\{|h-\rho| \le t\rho\big\} \;=\; \big\{(1-t)\rho \le h \le (1+t)\rho\big\} \;\subset\; \left\{\frac{1-t}{1+t}\rho \le h \le \frac{1+t}{1-t}\rho\right\}$$

we find

$$P_X\big(\big\{|h-\rho| \le t\rho\big\}\big) \;\le\; P_X\big(\big\{|2\eta-1| \le t\big\}\big) \;\le\; Ct^q\,,$$

which by $P_X = \frac{1}{\rho+1}Q + \frac{\rho}{\rho+1}\mu$ shows that h has ρ-exponent q .

Now let us conversely assume that h has ρ-exponent q with some constant $C > 0$. Then for $0 < t < 1$ we have

$$\begin{aligned} Q\big(\{|h-\rho| \le t\}\big) &= \int_X 1_{\{|h-\rho|\le t\}} h d\mu \\ &= \int_{\{h\le 1+\rho\}} 1_{\{|h-\rho|\le t\}} h d\mu \\ &\le (1+\rho) \int_{\{h\le 1+\rho\}} 1_{\{|h-\rho|\le t\}} d\mu \\ &= (1+\rho)\mu\big(\{|h-\rho| \le t\}\big) . \end{aligned}$$

Using $P_X = \frac{1}{\rho+1} Q + \frac{\rho}{\rho+1}\mu$ we hence find

$$P_X\big(\{|h-\rho| \le t\}\big) \le 2\mu\big(\{|h-\rho| \le t\}\big) \le 2Ct^q$$

for all sufficiently small $t \in (0,1)$. Let us now define $t_l := \frac{2t}{1+t}$ and $t_r := \frac{2t}{1-t}$. This immediately gives $1-t_l = \frac{1-t}{1+t}$ and $1+t_r = \frac{1+t}{1-t}$. Furthermore, we obviously also have $t_l \le t_r$. Therefore we find

$$\begin{aligned} \Big\{\frac{1-t}{1+t}\rho \le h \le \frac{1+t}{1-t}\rho\Big\} &= \big\{(1-t_l)\rho \le h \le (1+t_r)\rho\big\} \\ &\subset \big\{(1-t_r)\rho \le h \le (1+t_r)\rho\big\} \\ &= \big\{|h-\rho| \le t_r\rho\big\} . \end{aligned}$$

Hence for all sufficiently small $t > 0$ with $t < \frac{1}{1+2\rho}$, i.e. $t_r\rho < 1$, we obtain

$$P_X\big(\{|2\eta - 1| \le t\}\big) \le P_X\big(\{|h-\rho| \le t_r\rho\}\big) \le 2C(t_r\rho)^q \le 2C(1+2\rho)^q t^q .$$

From this we easily get the assertion. ■

References

S. Ben-David and M. Lindenbaum. Learning distributions by their density levels: a paradigm for learning without a teacher. *J. Comput. System Sci.*, 55:171–182, 1997.

C. Campbell and K. P. Bennett. A linear programming approach to novelty detection. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 395–401. MIT Press, 2001.

C.-C. Chang and C.-J. Lin. LIBSVM: a library for support vector machines, 2004.

A. Cuevas, M. Febrero, and R. Fraiman. Cluster analysis: a further approach based on density estimation. *Computat. Statist. Data Anal.*, 36:441–459, 2001.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statist. Soc. Ser. B (methodology)*, 39:1–38, 1977.

M. J. Desforges, P. J. Jacob, and J. E. Cooper. Applications of probability density estimation to the detection of abnormal conditions in engineering. *Proceedings of the Institution of Mechanical Engineers, Part C—Mechanical engineering science*, 212:687–703, 1998.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley, New York, 2000.

R. M. Dudley. *Real Analysis and Probability*. Cambridge University Press, 2002.

W. Fan, M. Miller, S. J. Stolfo, W. Lee, and P. K. Chan. Using artificial anomalies to detect unknown and known network intrusions. In *IEEE International Conference on Data Mining (ICDM'01)*, pages 123–130. IEEE Computer Society, 2001.

F. González and D. Dagupta. Anomaly detection using real-valued negative selection. *Genetic Programming and Evolvable Machines*, 4:383–403, 2003.

J. A. Hartigan. *Clustering Algorithms*. Wiley, New York, 1975.

J. A. Hartigan. Estimation of a convex density contour in 2 dimensions. *J. Amer. Statist. Assoc.*, 82: 267–270, 1987.

P. Hayton, B. Schölkopf, L. Tarassenko, and P. Anuzis. Support vector novelty detection applied to jet engine vibration spectra. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 946–952. MIT Press, 2001.

S. P. King, D. M. King, P. Anuzis, K. Astley, L. Tarassenko, P. Hayton, and S. Utete. The use of novelty detection techniques for monitoring high-integrity plant. In *IEEE International Conference on Control Applications*, pages 221–226. IEEE Computer Society, 2002.

E. Mammen and A. Tsybakov. Smooth discrimination analysis. *Ann. Statist.*, 27:1808–1829, 1999.

C. Manikopoulos and S. Papavassiliou. Network intrusion and fault detection: a statistical anomaly approach. *IEEE Communications Magazine*, 40:76–82, 2002.

M. Markou and S. Singh. Novelty detection: a review—Part 1: statistical approaches. *Signal Processing*, 83:2481–2497, 2003a.

M. Markou and S. Singh. Novelty detection: a review—Part 2: neural network based approaches. *Signal Processing*, 83:2499–2521, 2003b.

D. W. Müller and G. Sawitzki. Excess mass estimates and tests for multimodality. *J. Amer. Statist. Assoc.*, 86:738–746, 1991.

A. Nairac, N. Townsend, R. Carr, S. King, P. Cowley, and L. Tarassenko. A system for the analysis of jet engine vibration data. *Integrated Computer-Aided Engineering*, 6:53–56, 1999.

W. Polonik. Measuring mass concentrations and estimating density contour clusters—an excess mass aproach. *Ann. Statist.*, 23:855–881, 1995.

B. D. Ripley. *Pattern recognition and neural networks*. Cambridge University Press, 1996.

G. Sawitzki. The excess mass approach and the analysis of multi-modality. In W. Gaul and D. Pfeifer, editors, *From data to knowledge: Theoretical and practical aspects of classification, data analysis and knowledge organization*, Proc. 18th Annual Conference of the GfKl, pages 203–211. Springer, 1996.

B. Schölkopf, J. C. Platt, J. Shawe-Taylor, and A. J. Smola. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13:1443–1471, 2001.

B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2002.

I. Steinwart. On the influence of the kernel on the consistency of support vector machines. *J. Mach. Learn. Res.*, 2:67–93, 2001.

I. Steinwart. Consistency of support vector machines and other regularized kernel machines. *IEEE Trans. Inform. Theory*, 51:128–142, 2005.

I. Steinwart and C. Scovel. Fast rates for support vector machines using Gaussian kernels. *Ann. Statist.*, submitted, 2004. `http://www.c3.lanl.gov/~ingo/publications/ann-04a.pdf`.

L. Tarassenko, P. Hayton, N. Cerneaz, and M. Brady. Novelty detection for the identification of masses in mammograms. In *4th International Conference on Artificial Neural Networks*, pages 442–447, 1995.

J. Theiler and D. M. Cai. Resampling approach for anomaly detection in multispectral images. In *Proceedings of the SPIE 5093*, pages 230–240, 2003.

A. B. Tsybakov. On nonparametric estimation of density level sets. *Ann. Statist.*, 25:948–969, 1997.

A. B. Tsybakov. Optimal aggregation of classifiers in statistical learning. *Ann. Statist.*, 32:135–166, 2004.

D. Y. Yeung and C. Chow. Parzen-window network intrusion detectors. In *Proceedings of the 16th International Conference on Pattern Recognition (ICPR'02) Vol. 4*, pages 385–388. IEEE Computer Society, 2002.

H. Yu, J. Hen, and K. C. Chang. PEBL: Web page classification without negative examples. *IEEE. Trans. on Knowledge and Data Engineering*, 16:70–81, 2004.

Journal of Machine Learning Research 6 (2005) 233–272 Submitted 3/04; Revised 12/04; Published 3/05

Denoising Source Separation

Jaakko Särelä JAAKKO.SARELA@HUT.FI
Neural Networks Research Centre
Helsinki University of Technology
P. O. Box 5400
FI-02015 HUT, Espoo, Finland

Harri Valpola HARRI.VALPOLA@HUT.FI
Laboratory of Computational Engineering
Helsinki University of Technology
P. O. Box 9203
FI-02015 HUT, Espoo, Finland

Editor: Michael Jordan

Abstract

A new algorithmic framework called denoising source separation (DSS) is introduced. The main benefit of this framework is that it allows for the easy development of new source separation algorithms which can be optimised for specific problems. In this framework, source separation algorithms are constructed around denoising procedures. The resulting algorithms can range from almost blind to highly specialised source separation algorithms. Both simple linear and more complex nonlinear or adaptive denoising schemes are considered. Some existing independent component analysis algorithms are reinterpreted within the DSS framework and new, robust blind source separation algorithms are suggested. The framework is derived as a one-unit equivalent to an EM algorithm for source separation. However, in the DSS framework it is easy to utilise various kinds of denoising procedures which need not be based on generative models. In the experimental section, various DSS schemes are applied extensively to artificial data, to real magnetoencephalograms and to simulated CDMA mobile network signals. Finally, various extensions to the proposed DSS algorithms are considered. These include nonlinear observation mappings, hierarchical models and over-complete, nonorthogonal feature spaces. With these extensions, DSS appears to have relevance to many existing models of neural information processing.

Keywords: blind source separation, BSS, prior information, denoising, denoising source separation, DSS, independent component analysis, ICA, magnetoencephalograms, MEG, CDMA

1. Introduction

In recent years, source separation of linearly mixed signals has attracted a wide range of researchers. The focus of this research has been on developing algorithms that make minimal assumptions about the underlying process, thus approaching blind source separation (BSS). Independent component analysis (ICA) (Hyvärinen et al., 2001b) clearly follows this tradition. This blind approach gives the algorithms a wide range of possible applications. ICA has been a valuable tool, in particular, in testing certain hypotheses in magnetoencephalogram (MEG) and electroencephalogram (EEG) analysis (see Vigário et al., 2000).

Nearly always, however, there is further information available in the experimental setup, other design specifications or from accumulated knowledge due to scientific research. For example in biomedical signal analysis (see Gazzaniga, 2000; Rangayyan, 2002), careful design of experimental setups provides us with presumed signal characteristics. In man-made technology, such as a CDMA mobile system (see Viterbi, 1995), the transmitted signals are even more restricted.

The Bayesian approach provides a sound framework for including prior information into inferences about the signals. Recently, several Bayesian ICA algorithms have been suggested (see Knuth, 1998; Attias, 1999; Lappalainen, 1999; Miskin and MacKay, 2001; Choudrey and Roberts, 2001; d. F. R. Højen-Sørensen et al., 2002; Chan et al., 2003). They offer accurate estimations for the linear model parameters. For instance, universal density approximation using a mixture of Gaussians (MoG) may be used for the source distributions. Furthermore, hierarchical models can be used for incorporating complex prior information (see Valpola et al., 2001). However, the Bayesian approach does not always result in simple or computationally efficient algorithms.

FastICA (Hyvärinen, 1999) provides a set of algorithms for performing ICA based on optimising easily calculable contrast functions. The algorithms are fast but often more accurate results can be achieved by computationally more demanding algorithms (Giannakopoulos et al., 1999), for example by the Bayesian ICA algorithms. Valpola and Pajunen (2000) analysed the factors behind the speed of FastICA. The analysis suggested that the nonlinearity used in FastICA can be interpreted as denoising and taking Bayesian noise filtering as the nonlinearity resulted in fast Bayesian ICA.

Denoising corresponds to procedural knowledge while in most approaches to source separation, the algorithms are derived from explicit objective functions or generative models. This corresponds to declarative knowledge. Algorithms are procedural, however. Thus declarative knowledge has to be translated into procedural form, which may result in complex and computationally demanding algorithms.

In this paper, we generalise the denoising interpretation of Valpola and Pajunen (2000) and introduce a source separation framework called denoising source separation (DSS). We show that it is actually possible to construct the source separation algorithms around the denoising methods themselves. Fast and accurate denoising will result in a fast and accurate separation algorithm. We suggest that various kinds of prior knowledge can be easily formulated in terms of denoising. In some cases a denoising scheme has been used to post-process the results after separation (see Vigneron et al., 2003), but in the DSS framework this denoising can be used for the source separation itself.

The paper is organised as follows: After setting the general problem of linear source separation in Sec. 2, we review an expectation-maximisation (EM) algorithm as a solution to a generative linear model and a one-unit version of it (Sec. 2.1). We interpret the nonlinearity as denoising and call this one-unit algorithm DSS. Equivalence of the linear DSS and a power method is shown in Sec. 2.2. In Sec. 2.3, the convergence of the DSS algorithms is analysed. The linear DSS is analysed via the power method. To shed light on the convergence of the nonlinear DSS, we define local eigenvalues, giving analysis similar to the linear case. The applicability of two common extensions of the power method—deflation and spectral shift—are discussed in the rest of the section. In Sec. 3, we suggest an approximation for an objective function that is maximised by the DSS algorithms. We then introduce some practical denoising functions in Sec. 4. These denoising functions are extensively applied to artificial mixtures (Sec. 5.1) and to MEG recordings (Secs. 5.2 and 5.3). We also apply a DSS algorithm to bit-stream recovery in a simulated CDMA network (Sec. 5.4). Finally, in Sec. 6, we discuss extensions to the DSS framework and their connections to models of neural information processing.

2. Source Separation by Denoising

Consider a linear instantaneous mixing of sources:

$$\mathbf{X} = \mathbf{AS} + \nu\,, \tag{1}$$

where

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_M \end{bmatrix}, \quad \mathbf{S} = \begin{bmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \\ \vdots \\ \mathbf{s}_N \end{bmatrix}.$$

The source matrix $\mathbf{S}$ consists of N sources. Each individual source $\mathbf{s}_i$ consists of T samples, that is, $\mathbf{s}_i = [s_i(1) \dots s_i(t) \dots s_i(T)]$. Note that in order to simplify the notation throughout the paper, we have defined each source to be a row vector instead of the more traditional column vector. The symbol t often stands for time, but other possibilities include, *e.g.*, space. For the rest of the paper, we refer to t as time, for convenience. The observations $\mathbf{X}$ consist of M mixtures of the sources, that is, $\mathbf{x}_i = [x_i(1) \dots x_i(t) \dots x_i(T)]$. Usually it is assumed that $M \geq N$. The linear mapping $\mathbf{A} = [\mathbf{a}_1\, \mathbf{a}_2 \cdots \mathbf{a}_N]$ consists of the mixing vectors $\mathbf{a}_i = [a_{1i}\, a_{2i} \dots a_{Mi}]^T$, and is usually called the mixing matrix. In the model, there is some Gaussian noise ν, too. The sources, the noise and hence also the mixtures can be assumed to have zero mean without losing generality because the mean can always be removed from the data.

If the sources are assumed i.i.d. Gaussian, this is a general, linear factor analysis model with rotational invariance. There are several ways to fix the rotation, *i.e.*, to separate the original sources $\mathbf{S}$. Some approaches assume structure for the mixing matrix. If no structure is assumed, the solution to this problem is usually called blind source separation (BSS). Note that this approach is not really blind, since one always needs some information to be able to fix the rotation. One such piece of information is the non-Gaussianity of the sources, which leads to the recently popular ICA methods (see Hyvärinen et al., 2001b). The temporal structure of the sources may be used too, as in Tong et al. (1991); Molgedey and Schuster (1994); Belouchrani et al. (1997); Ziehe and Müller (1998); Pham and Cardoso (2001).

The rest of this section is organised as follows: first we review an EM algorithm for source separation and a one-unit version derived from it in Sec. 2.1. The E- and M-steps have natural interpretations as denoising of the sources and re-estimation of the mixing vector, respectively, and the derived algorithm provides the starting point for the DSS framework. In Sec. 2.2, we show that a Gaussian source model leads to linear denoising. Such a DSS is equivalent to PCA of suitably filtered data, implemented by the classical power method. The convergence of the DSS algorithms are discussed in Sec. 2.3. For the linear DSS algorithms, the well-known convergence results of the power method are used. Furthermore, the same results may be exploited for the nonlinear case by defining local eigenvalues. They play a similar role as the (global) eigenvalues in the linear case. Deflation and symmetric method for extracting several sources are reviewed in Sec. 2.4. Sec. 2.5 discusses a speedup technique called spectral shift.

2.1 One-Unit Algorithm for Source Separation

The EM algorithm (Dempster et al., 1977) is a method for performing maximum likelihood estimation when part of the data is missing. One way to perform EM estimation in the case of linear models is to assume that the missing data consists of the sources and that the mixing matrix needs to be estimated. In the following, we review one such EM algorithm by Bermond and Cardoso (1999) and a derivation of a one-unit version of it by Hyvärinen et al. (2001b).

The algorithm proceeds by alternating two steps: 1) E-step and 2) M-step. In the E-step, the posterior distribution for the sources is calculated based on the known data and the current estimate of the mixing matrix using Bayes' theorem. In the M-step, the mixing matrix is fitted to the new source estimates. In other words:

$$\mathrm{E-step: compute}\, q(\mathbf{S}) = p(\mathbf{S}|\mathbf{A},\mathbf{X}) = p(\mathbf{X}|\mathbf{A},\mathbf{S})p(\mathbf{S})/p(\mathbf{X}|\mathbf{A}) \tag{2}$$

$$\mathrm{M-step: find}\, \mathbf{A}_{\mathrm{new}} = \arg\max_{\mathbf{A}} \mathrm{E}_{q(\mathbf{S})}[\log p(\mathbf{S},\mathbf{X}|\mathbf{A})] = \mathbf{C}_{\mathbf{XS}}\mathbf{C}_{\mathbf{SS}}^{-1}\,. \tag{3}$$

The covariances are computed as expectations over $q(\mathbf{S})$:

$$\mathbf{C}_{\mathbf{XS}} = \frac{1}{T}\sum_{t=1}^{T} \mathrm{E}[\mathbf{x}(t)\mathbf{s}(t)^T|\mathbf{X},\mathbf{A}] = \frac{1}{T}\sum_{t=1}^{T}\mathbf{x}(t)\,\mathrm{E}[\mathbf{s}(t)^T|\mathbf{X},\mathbf{A}] \tag{4}$$

$$\mathbf{C}_{\mathbf{SS}} = \frac{1}{T}\sum_{t=1}^{T} \mathrm{E}[\mathbf{s}(t)\mathbf{s}(t)^T|\mathbf{X},\mathbf{A}], \tag{5}$$

where $\mathbf{x}(t) = [x_1(t)\cdots x_i(t)\cdots x_M(t)]^T$ and $\mathbf{s}(t) = [s_1(t)\cdots s_j(t)\cdots s_N(t)]^T$ are used to denote the values of all of the mixtures and the sources at the time instance t, respectively.

Many source separation algorithms preprocess the data by normalising the covariance to the unit matrix, *i.e.*, $\mathbf{C}_{\mathbf{XX}} = \mathbf{X}\mathbf{X}^T/T = \mathbf{I}$. This is referred to as sphering or whitening and its result is that any signal obtained by projecting the sphered data on any unit vector has zero mean and unit variance. Furthermore, orthogonal projections yield uncorrelated signals. Sphering is often combined with reducing the dimension of the data by selecting a principal subspace which contains most of the energy of the original data.

Because of the indeterminacy of scale in linear models, it is necessary to fix either the variance of the sources or the norm of the mixing matrix. It is usual to fix the variance of the sources to unity: $\mathbf{S}\mathbf{S}^T/T = \mathbf{I}$. Then, assuming that the linear independent-source model holds and there is an infinite amount of data, with Gaussian noise, the covariance of the sphered data is $\mathbf{A}\mathbf{S}\mathbf{S}^T\mathbf{A}^T/T + \Sigma_\nu = \mathbf{A}\mathbf{A}^T + \Sigma_\nu = \mathbf{I}$, *i.e.*, a unit matrix because of the sphering. If the noise variance is proportional to the covariance of the data that is due to the sources, *i.e.*, $\Sigma_\nu \propto \mathbf{A}\mathbf{A}^T$, it holds that $\mathbf{A}\mathbf{A}^T \propto \mathbf{I}$, which means that the mixing matrix $\mathbf{A}$ is orthogonal for sphered data. Furthermore, the likelihood $L(\mathbf{S}) = p(\mathbf{X}|\mathbf{A},\mathbf{S})$ of $\mathbf{S}$ can be factorised:

$$L(\mathbf{S}) = C\prod_i L_i(\mathbf{s}_i)\,, \tag{6}$$

where the constant C is independent of $\mathbf{S}$. The constant C reflects the fact that likelihoods do not normalise the same way as probability densities. The above factorisation still becomes unique if $L_i(\mathbf{s}_i)$ are appropriately normalised. In the case of a linear model with Gaussian noise, a convenient normalisation is to require the maximum of $L_i(\mathbf{s}_i)$ to equal one. The terms can then be shown to equal

$$L_i(\mathbf{s}_i) = \exp\left(-\frac{1}{2}(\mathbf{s}_i - \mathbf{a}_i^{-1}\mathbf{X})\,\Sigma_{\mathbf{s},\nu}^{-1}\,(\mathbf{s}_i - \mathbf{a}_i^{-1}\mathbf{X})^T\right), \tag{7}$$

where $\mathbf{a}_i^{-1}$ is the ith row vector of $\mathbf{A}^{-1}$ and $\Sigma_{\mathbf{s},\nu} \propto \mathbf{I}$ is a diagonal matrix with the diagonal elements equalling $\sigma_\nu^2/(\mathbf{a}_i^T\mathbf{a}_i)$.

Since the prior $p(\mathbf{S})$ factorises, too, the sources are independent in the posterior $q(\mathbf{S})$ and the covariance $\mathbf{C_{SS}}$ is diagonal. This means that $\mathbf{C_{SS}}^{-1}$ reduces to scaling of individual sources in the M-step (3).

Noisy estimates of the sources can be recovered by $\mathbf{S} = \mathbf{A}^{-1}\mathbf{X}$ which is the mode of the likelihood. Since $\mathbf{A}^{-1} \propto \mathbf{A}^T$ because of the sphering and the posterior $q(\mathbf{S})$ depends on the data only through the likelihood $L(\mathbf{S})$, the expectation $\mathrm{E}[\mathbf{S}|\mathbf{X},\mathbf{A}]$ is a function of $\mathbf{A}^T\mathbf{X}$, or for individual sources, $\mathrm{E}[\mathbf{s}_i|\mathbf{X},\mathbf{A}] = \mathbf{f}(\mathbf{a}_i^T\mathbf{X})$. In the case of Gaussian source model $p(\mathbf{S})$, this function is linear (further discussion in Sec. 2.2). The expectation can be computed exactly in some other cases, too, *e.g.*, when the source distributions are mixtures of Gaussians (MoG).[1] In other cases the expectation can be approximated for instance by $\mathrm{E}_{q(\mathbf{S})}[\mathbf{S}] = \mathbf{S} + \varepsilon\,\partial \log p(\mathbf{S})/\partial\mathbf{S}$, where the constant ε depends on the noise variance.

In the EM algorithm, all the components are estimated simultaneously. However, pre-sphering renders it possible to extract the sources one-by-one (see Hyvärinen et al., 2001b, for a similarly derived algorithm):

$$\mathbf{s} = \mathbf{w}^T\mathbf{X} \tag{8}$$

$$\mathbf{s}^+ = \mathbf{f}(\mathbf{s}) \tag{9}$$

$$\mathbf{w}^+ = \mathbf{X}\mathbf{s}^{+T} \tag{10}$$

$$\mathbf{w}_{\text{new}} = \frac{\mathbf{w}^+}{||\mathbf{w}^+||}. \tag{11}$$

In this algorithm, the first step (8) calculates the noisy estimate of one source and corresponds to the mode of the likelihood. It is a convention to denote the mixing vector $\mathbf{a}$, which in this case is also the separating vector, by $\mathbf{w}$. The second step (9) corresponds to the expectation of $\mathbf{s}$ over $q(\mathbf{S})$ and can be seen as denoising based on the model of the sources. Note that $\mathbf{f}(\mathbf{s})$ is a row-vector-valued function of a row-vector argument. The re-estimation step (10) calculates the new ML estimate of the mixing vector and the M-step (3) is completed by normalisation (11). This prevents the norm of the mixing vector from diverging. Although this algorithm separates only one component, it has been shown that the original sources correspond to stable fixed points of the algorithm under quite general conditions (see Theorem 8.1, Hyvärinen et al., 2001b), provided that the independent-source model holds.

In this paper, we interpret the step (9) as denoising. While this interpretation is not novel, it allows for the development of new algorithms that are not derived starting from generative models. We call all of the algorithms where Eq. (9) can be interpreted as denoising and that have the form (8)–(11) DSS algorithms.

1. MoG as the source distributions would lead to ICA.

2.2 Linear DSS

In this section, we show that separation of Gaussian sources using the DSS algorithm results in linear denoising. This is called linear DSS and it converges to the eigenvector of a data matrix that has been suitably filtered. The algorithm is equivalent to the classical power method applied to the covariance of the filtered data.

First, let us assume the Gaussian source to have an autocovariance matrix $\Sigma_{\mathbf{ss}}$. The prior probability density function for a Gaussian source is given by

$$p(\mathbf{s}) = \frac{1}{\sqrt{|2\pi\Sigma_{\mathbf{ss}}|}} \exp\left(-\frac{1}{2}\mathbf{s}\Sigma_{\mathbf{ss}}^{-1}\mathbf{s}^T\right),$$

where $\Sigma_{\mathbf{ss}}$ is the autocovariance matrix of the source and $|\Sigma_{\mathbf{ss}}|$ is its determinant. Furthermore, as noted in Eq. (7), the likelihood $L(\mathbf{s})$ is an unnormalised Gaussian with a diagonal covariance $\Sigma_{\mathbf{s},\nu}$:

$$L(\mathbf{s}) = \exp\left(-\frac{1}{2}(\mathbf{s}-\mathbf{w}^T\mathbf{X})\Sigma_{\mathbf{s},\nu}^{-1}(\mathbf{s}-\mathbf{w}^T\mathbf{X})^T\right).$$

After some algebraic manipulation, the Gaussian posterior is reached:

$$q(\mathbf{s}) = \frac{1}{\sqrt{|2\pi\Sigma|}} \exp\left(-\frac{1}{2}(\mathbf{s}-\mu)\Sigma^{-1}(\mathbf{s}-\mu)^T\right),$$

with mean $\mu = \mathbf{w}^T\mathbf{X}\left(\mathbf{I}+\sigma_\nu^2\Sigma_{\mathbf{ss}}^{-1}\right)^{-1}$, and variance $\Sigma^{-1} = \frac{1}{\sigma_\nu^2} + \Sigma_{\mathbf{ss}}^{-1}$. Hence, the denoising step (9) becomes

$$\mathbf{s}^+ = \mathbf{f}(\mathbf{s}) = \mathbf{s}\left(\mathbf{I}+\sigma_\nu^2\Sigma_{\mathbf{ss}}^{-1}\right)^{-1} = \mathbf{sD}, \tag{12}$$

which corresponds to linear denoising. The denoising step in the DSS algorithm $\mathbf{s}^+ = \mathbf{f}(\mathbf{s})$ is thus equivalent to multiplying the current source estimate $\mathbf{s}$ with a constant matrix $\mathbf{D}$.

To gain more intuition about the denoising, it is useful to consider the eigenvalue decomposition of $\mathbf{D}$. It turns out that $\mathbf{D}$ and $\Sigma_{\mathbf{ss}}$ have the same eigenvectors and the eigenvalue decompositions are

$$\Sigma_{\mathbf{ss}} = \mathbf{V}\Lambda_\Sigma\mathbf{V}^T \tag{13}$$

$$\mathbf{D} = \mathbf{V}\Lambda_D\mathbf{V}^T, \tag{14}$$

where $\mathbf{V}$ is an orthonormal matrix with the eigenvectors as columns and Λ is a diagonal matrix with the corresponding eigenvalues on the diagonal. The eigenvalues are related as

$$\lambda_{D,i} = \frac{1}{1+\frac{\sigma_\nu^2}{\lambda_{\Sigma,i}}}.$$

Note that $\lambda_{D,i}$ is a monotonically increasing function of $\lambda_{\Sigma,i}$. Those directions of $\mathbf{s}$ are suppressed the most which have the smallest variances according to the prior model of $\mathbf{s}$.

Now, let us pack the different phases of the algorithm (8), (12), (10) together:

$$\mathbf{w}^+ = \mathbf{X}\mathbf{s}^{+T} = \mathbf{X}\mathbf{D}\mathbf{s}^T = \mathbf{X}\mathbf{D}\mathbf{X}^T\mathbf{w}.$$

The transpose was dropped from $\mathbf{D}$ since it is symmetric. By writing $\Lambda_D = \Lambda_D^{\frac{1}{2}}\Lambda_D^{\frac{1}{2}T} = \Lambda^*\Lambda^{*T}$ and adding $\mathbf{V}^T\mathbf{V} = \mathbf{I}$ in the middle, we may split the denoising matrix into two parts:

$$\mathbf{D} = \mathbf{D}^*\mathbf{D}^{*T},$$

where $\mathbf{D}^* = \mathbf{V}\Lambda^*\mathbf{V}^T$. Further, let us denote $\mathbf{Z} = \mathbf{X}\mathbf{D}^*$. This brings the DSS algorithm for estimating one separating vector into the form

$$\mathbf{w}^+ = \mathbf{Z}\mathbf{Z}^T\mathbf{w}. \tag{15}$$

This is the classical *power method* (see Wilkinson, 1965) implementation for principal component analysis (PCA). Note that $\mathbf{Z}\mathbf{Z}^T$ is the unnormalised covariance matrix. The algorithm converges to the *fixed point* $\mathbf{w}^*$ satisfying

$$\lambda\mathbf{w}^* = \mathbf{Z}\mathbf{Z}^T/T\,\mathbf{w}^*, \tag{16}$$

where λ corresponds to the principal eigenvalue of the covariance matrix $\mathbf{Z}\mathbf{Z}^T/T$ and $\mathbf{w}^*$ is the principal direction. The asterisk is used to stress the fact that the estimate is at the fixed point.

The operation of the linear DSS algorithm is depicted in Fig. 1. Figure 1a shows two sources that have been mixed into Fig. 1b. The mixing vectors have been plotted by the dashed lines. The curve shows the standard deviation of the data projected in different directions. It is evident that the principal eigenvector (solid line) does not separate any of the sources. For that two things would be needed: 1) The mixing vectors should be orthogonal. 2) The eigenvalues should differ. After sphering in Fig. 1c, the basis and sphered mixing vectors are roughly orthogonal. However, any unit-length projection yields unit variance, and PCA still cannot separate the sources. The first source has a somewhat slower temporal evolution and low-pass filtering retains more of that signal, giving it a larger eigenvalue. This is evident in Fig. 1d which shows the denoised data and the first eigenvector, which is now aligned with the (sphered) mixing vector of the slow source. The sources can then be recovered by $\mathbf{s} = \mathbf{w}^T\mathbf{X}$.

There are other algorithms for separating Gaussian sources (Tong et al., 1991; Molgedey and Schuster, 1994; Belouchrani et al., 1997; Ziehe and Müller, 1998) and, although functionally different, they yield similar results for the example given above. All these algorithms assume that the autocovariance structure of the sources is time-invariant corresponding to Toeplitz autocovariance and filtering matrices $\Sigma_{\mathbf{ss}}$ and $\mathbf{D}$. In our analysis, $\Sigma_{\mathbf{ss}}$ can be any covariance matrix, and only one out of four examples in Sec. 4.1 has the Toeplitz form.

2.3 Convergence Analysis

In this section, we analyse the convergence properties of DSS algorithms. In the case of linear denoising, we will refer to well-known convergence properties of the power method (*e.g.*, Wilkinson, 1965). The analysis extends to nonlinear denoising under the assumptions that the mixing model holds and there is an infinite amount of data.

Linear DSS is equivalent to the power method whose convergence is governed by the eigenvalues λ_i corresponding to the fixed points $\mathbf{w}_i^*$. If some of the eigenvalues are equal ($\lambda_i = \lambda_j$, $i \neq j$), the fixed points are degenerate and there are subspaces of fixed points. In any case, it is possible to choose an orthonormal basis spanned by $\mathbf{w}_i^*$. This means that any $\mathbf{w}$ can be represented as

$$\mathbf{w} = \sum_i c_i \mathbf{w}_i^*, \tag{17}$$

where $c_i = \mathbf{w}^T\mathbf{w}_i^*$. With a linear denoising function $\mathbf{f}_{\text{lin}}$, the unnormalised estimate $\mathbf{w}^+$ is

$$\mathbf{w}^+ = \mathbf{X}\mathbf{f}_{\text{lin}}^T\left(\sum_i c_i\mathbf{s}_i^*\right) = \mathbf{X}\sum_i c_i\mathbf{f}_{\text{lin}}^T(\mathbf{s}_i^*) = \sum_i c_i\mathbf{X}\mathbf{f}_{\text{lin}}^T(\mathbf{s}_i^*) = T\sum_i c_i\lambda_i\mathbf{w}_i^*, \tag{18}$$

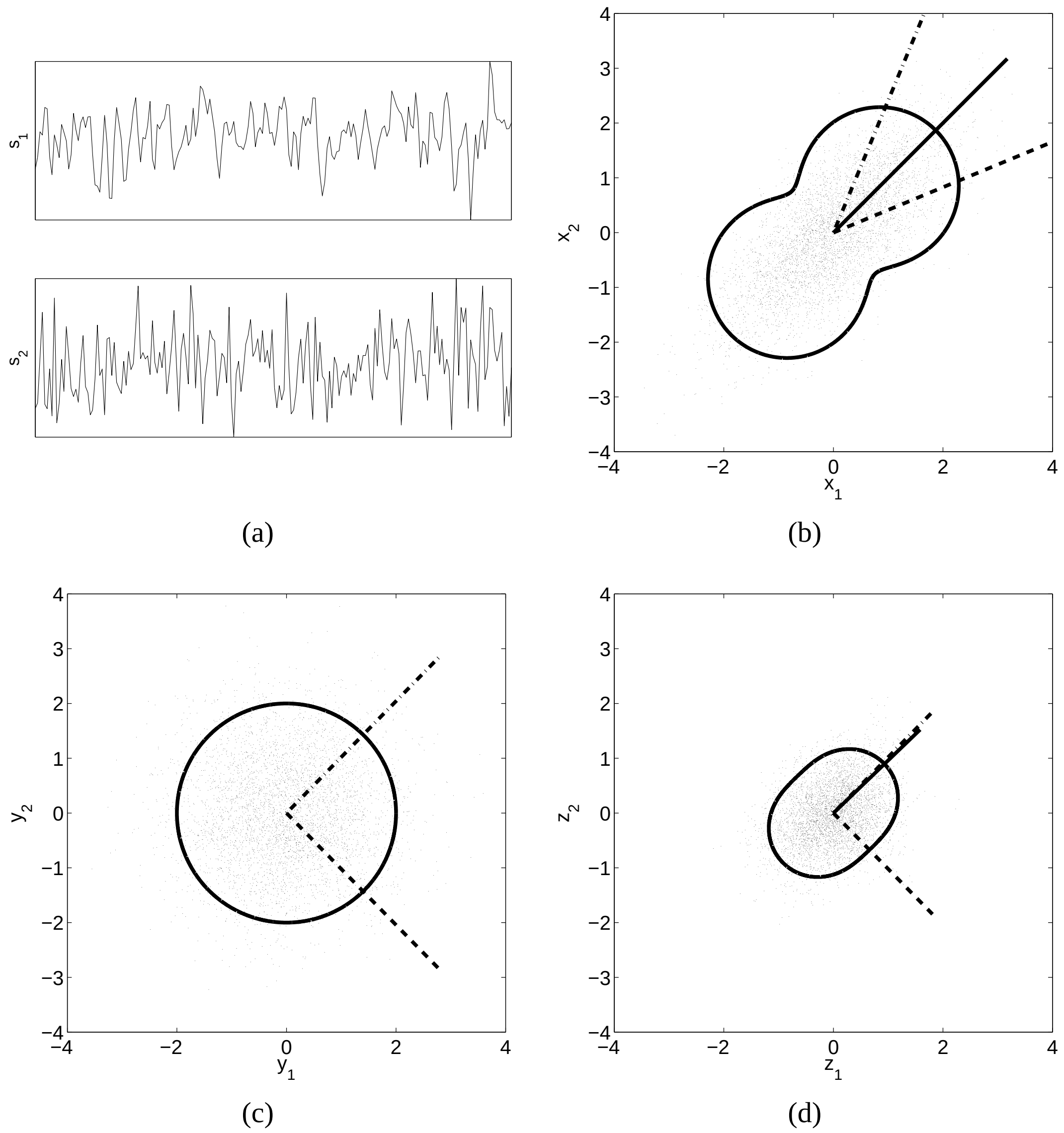


Figure 1: *(a) Original sources, (b) scatter-plot of the mixtures, (c) sphered data* $\mathbf{X}$ *and (d) denoised data* $\mathbf{Z} = \mathbf{X}\mathbf{D}^*$. *The dashed lines depict the mixing vectors and the solid lines the largest eigenvector. The curves denote the standard deviation of the projection of the data in different directions.*

where λ_i is the ith eigenvalue corresponding to $\mathbf{w}_i^*$ and $\mathbf{s}_i^* = \mathbf{w}_i^{*T}\mathbf{X}$. The normalisation step (11) changes the contributions of the fixed points by equal fractions. After n iterations, the relative contributions of the fixed points thus change from $\frac{c_i}{c_j}$ into $\frac{c_i\lambda_i^n}{c_j\lambda_j^n}$.

If there are two fixed points $\mathbf{w}_i^*$ and $\mathbf{w}_j^*$ that have identical eigenvalues $\lambda_i = \lambda_j$, the linear DSS cannot separate between the two. This means, for instance, that it is not possible to separate Gaussian sources that have identical autocovariance matrices, *i.e.*, $\Sigma_{\mathbf{s}_i\mathbf{s}_i} = \Sigma_{\mathbf{s}_j\mathbf{s}_j}$ or in other words sources whose time structures do not differ. Otherwise, as long as $c_i \neq 0$, the algorithm converges globally to the source with the largest eigenvalue.

The speed of convergence in the power method (hence in linear DSS) depends linearly on the log-ratio of the largest (absolute) eigenvalues $\log|\lambda_1|/|\lambda_2|$, where $|\lambda_1| \geq |\lambda_2| \geq |\lambda_i|$, $i = 3, \ldots, N$. Note that absolute values of the eigenvalues have been used. While the eigenvalues are usually positive, there are cases where negative eigenvalues may exist, for instance in the case of complex data or when using the so-called *spectral shift*, which is discussed in Sec. 2.5.

The above analysis for linear denoising functions makes no assumptions about the data-generating process. As such it does not extend to nonlinear denoising functions because there can be more or less fixed points than the dimensionality of the data, and the fixed points $\mathbf{w}_i^*$ are not, in general, orthogonal. We shall therefore assume that the data are generated by independent sources by the model (1) and the assumptions discussed in Sec. 2.1 hold, *i.e.*, the mixing vectors are orthogonal after sphering. Under these assumptions, the orthonormal basis spanned by the mixing vectors corresponds to fixed points of the DSS algorithm. This holds because from the independence of different sources $\mathbf{s}_i$ it follows that

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} s_j(t) f_t(\mathbf{s}_i) = 0 \tag{19}$$

for $i \neq j$.

In the linear power method, eigenvalues λ_i govern the rate of relative changes of the contributions of individual basis vectors in the estimate. We shall define *local eigenvalues* $\lambda_i(\mathbf{s})$ which play similar roles in nonlinear DSS. Unlike the constant eigenvalues λ_i, the local eigenvalues have different values depending on the current source estimate. The formal definition is as follows. Assume that the current weight vector and the subsequent unnormalised new weight vector are

$$\mathbf{w} = \sum_i c_i(\mathbf{s})\mathbf{w}_i^* \tag{20}$$

$$\mathbf{w}^+ = \sum_i \gamma_i(\mathbf{s})\mathbf{w}_i^* \,. \tag{21}$$

The local eigenvalue is defined to be the relative change in the contribution:

$$\gamma_i(\mathbf{s}) = Tc_i(\mathbf{s})\lambda_i(\mathbf{s}) \Leftrightarrow \lambda_i(\mathbf{s}) = \frac{\gamma_i(\mathbf{s})}{Tc_i(\mathbf{s})} \,. \tag{22}$$

The idea of the DSS framework is that the user can tailor the denoising function to the task at hand. The denoising can but need not be based on the E-step (2) derived from a generative model. The purpose of defining the local eigenvalues is to draw attention to the factors influencing separation quality and convergence speed.

The first thing to consider is whether the algorithm converges at all. It is possible to view the nonlinear denoising as linear denoising which is constantly adapted to the source estimate. This means that different sources can have locally the largest eigenvalue. If the adaptation is consistent, *i.e.*, $\lambda_i(\mathbf{s})$ grows monotonically with c_i, all stable fixed points correspond to the original sources. In general, the best separation quality and the fastest convergence is achieved when $\lambda_i(\mathbf{s})$ is very large compared to all $\lambda_j(\mathbf{s})$ with $j \neq i$ in the vicinity of $\mathbf{s}_i^*$.

Sometimes it may be sufficient to separate a signal subspace. Then it is enough for the denoising function to make the eigenvalues corresponding to this subspace large compared to the rest but the eigenvalues do not need to differ within the subspace.

If the mixture model (1) holds and there is an infinite amount of data, the sources can usually be separated even in the linear case because minute differences in the eigenvalues of the sources are sufficient for separation. In practice, the separation is based on a finite number of samples and the ICA model only holds approximately. Conceptually, we can think that there are true eigenvalues and mixing vectors but the finite sample size introduces noise to the eigenvalues and leakage between mixing vectors. In practice the separation quality is therefore much better if the local eigenvalues differ significantly around the fixed points and this is often easiest to achieve with nonlinear denoising which utilises a lot of prior information.

2.4 Deflation

The classical power method has two common extensions: deflation and spectral shift. They are readily available for the linear DSS since it is equivalent to the power method applied to filtered data via Eq. (2.2). It is also relatively straightforward to apply them in the nonlinear case.

Linear DSS algorithms converge globally to the source whose eigenvalue has the largest magnitude. Nonlinear DSS algorithms may have several fixed points but even then it is useful to guarantee that the algorithm converges to a source estimate which has not been extracted yet. The deflation method is a procedure which allows one to estimate several sources by iteratively applying the DSS algorithm several times. The convergence to previously extracted sources is prevented by making their eigenvalues zero: $\mathbf{w}_{\mathrm{orth}} = \mathbf{w} - \mathbf{A}\mathbf{A}^T\mathbf{w}$ (Luenberger, 1969), where $\mathbf{A}$ now contains the already estimated mixing vectors.

Note that in this deflation scheme, it is possible to use different kinds of denoising procedures when the sources differ in characteristics. Also, if more than one source is estimated simultaneously, the symmetric orthogonalisation methods proposed for symmetric FastICA (Hyvärinen, 1999) can be used. It should be noted, however, that such symmetric orthogonalisation cannot separate sources with linear denoising where the eigenvalues of the sources are globally constant.

2.5 Spectral Shift

As discussed in Sec. 2.2, the matrix multiplication (15) in the power method does not promote the largest eigenvalue effectively compared to the second largest eigenvalue if they have comparable values. The convergence speed in such cases can be increased by so-called spectral shift[2] (Wilkinson, 1965) which modifies the eigenvalues without changing the fixed points. At the fixed point of the DSS algorithm,

$$\lambda \mathbf{w}^* = \mathbf{X}\mathbf{f}^T(\mathbf{s}^*)/T \,. \tag{23}$$

2. The set of the eigenvalues is often called the eigenvalue spectrum.

If the denoising function is multiplied by a scalar, the convergence of the algorithm does not change in any way because the scaling will be overruled by the normalisation step (11). All eigenvalues will be scaled but their ratios, which are what count in convergence, are not affected.

Adding a multiple of $\mathbf{s}$ into $\mathbf{f}(\mathbf{s})$ does not affect the fixed points because $\mathbf{X}\mathbf{s}^T \propto \mathbf{w}$. However the ratios of the eigenvalues get affected and hence the convergence speed. In summary, $\mathbf{f}(\mathbf{s})$ can be replaced by

$$\alpha(\mathbf{s})[\mathbf{f}(\mathbf{s}) + \beta(\mathbf{s})\mathbf{s}]\,, \tag{24}$$

where $\alpha(\mathbf{s})$ and $\beta(\mathbf{s})$ are scalars. The multiplier $\alpha(\mathbf{s})$ is overruled by the normalisation step (11) and has no effect on the algorithm. The term $\beta(\mathbf{s})\mathbf{s}$ is turned into $T\beta(\mathbf{s})\mathbf{w}$ in the re-estimation step (8) and does affect the convergence speed but not the fixed points (however, it can turn a stable fixed point unstable or vice versa). This is because all eigenvalues are shifted by $\beta(\mathbf{s})$:

$$\mathbf{X}[\mathbf{f}(\mathbf{s}^*) + \beta(\mathbf{s}^*)\mathbf{s}^*]^T/T = \lambda\mathbf{w}^* + \beta(\mathbf{s}^*)\mathbf{w}^* = [\lambda + \beta(\mathbf{s}^*)]\mathbf{w}^*\,.$$

The spectral shift using $\beta(\mathbf{s})$ modifies the ratios of the eigenvalues and the ratio of the two largest eigenvalues[3] becomes $|[\lambda_1 + \beta(\mathbf{s})]/[\lambda_2 + \beta(\mathbf{s})]| > |\lambda_1/\lambda_2|$, provided that $\beta(\mathbf{s})$ is negative but not much smaller than $-\lambda_2$. This procedure can greatly accelerate convergence.

For very negative $\beta(\mathbf{s})$, some eigenvalues will become negative. In fact, if $\beta(\mathbf{s})$ is small enough, the absolute value of the originally smallest eigenvalue will exceed that of the originally largest eigenvalue. Iterations of linear DSS will then minimise the eigenvalue rather than maximise it.

We suggest that it is often reasonable to shift the eigenvalue corresponding to the Gaussian signal ν to zero. Some eigenvalues may then become negative and the algorithms can converge to fixed points corresponding to these eigenvalues rather than the positive ones. In many cases, this is perfectly acceptable because, as will be further discussed in Sec. 3.3, any deviation from the Gaussian eigenvalue is indicative of signal. A side effect of a negative eigenvalue is that the estimate $\mathbf{w}$ changes its sign at each iteration. This is not a problem but needs to be kept in mind when determining the convergence.

Since the convergence of the nonlinear DSS is governed by local eigenvalues, the spectral shift needs to be adapted to the changing local eigenvalues to achieve optimal convergence speed. In practice, the eigenvalue λ_ν of a Gaussian signal can be estimated by linearising $\mathbf{f}(\mathbf{s})$ around the current source estimate $\mathbf{s}$:

$$\mathbf{f}(\mathbf{s} + \Delta\mathbf{s}) \approx \mathbf{f}(\mathbf{s}) + \Delta\mathbf{s}\mathbf{J}(\mathbf{s}) \tag{25}$$

$$\lambda_\nu(\mathbf{s}) \approx \frac{\mathbf{f}(\mathbf{s} + \varepsilon\nu) - \mathbf{f}(\mathbf{s})}{\varepsilon}\nu^T/T \approx \frac{\varepsilon\nu\mathbf{J}(\mathbf{s})}{\varepsilon}\nu^T/T = \nu\mathbf{J}(\mathbf{s})\nu^T/T \tag{26}$$

$$\beta(\mathbf{s}) = E[-\lambda_\nu(\mathbf{s})] \approx -\operatorname{tr}\mathbf{J}(\mathbf{s})/T \tag{27}$$

The last step follows from the fact that the elements of ν are mutually uncorrelated and have zero mean and unit variance. Here $\mathbf{J}(\mathbf{s})$ denotes the Jacobian matrix of $\mathbf{f}(\mathbf{s})$ computed at $\mathbf{s}$. For linear denoising $\mathbf{J}(\mathbf{s}) = \mathbf{D}$ and hence β does not depend on $\mathbf{s}$. If denoising is instantaneous, *i.e.*, $\mathbf{f}(\mathbf{s}) = [f_1(s(1))\ f_2(s(2))\ \ldots]$, the shift can be written as $\beta(\mathbf{s}) = -\sum_t f_t'(s(t))/T$. This is the spectral shift used in FastICA (Hyvärinen, 1999), but it has been justified as an approximation to Newton's method and our analysis thus provides a novel interpretation.

3. Since the denoising operation presumably preserves some of the signal and noise, it is reasonable to assume that all eigenvalues are originally positive.

Sometimes the spectral shift turns out to be either too modest or too strong, leading to slow convergence or lack of convergence, respectively. For this reason, we suggest a simple stabilisation rule, henceforth called 179-rule: instead of updating $\mathbf{w}$ into $\mathbf{w}_{\text{new}}$ defined by Eq. (11), it is updated into

$$\mathbf{w}_{\text{adapted}} = \text{orth}(\mathbf{w} + \gamma\Delta\mathbf{w}) \tag{28}$$

$$\Delta\mathbf{w} = \mathbf{w}_{\text{new}} - \mathbf{w}\,, \tag{29}$$

where γ is the step size and the orthogonalisation has been added in case several sources are to be extracted. Originally $\gamma = 1$, but if the consecutive steps are taken in nearly opposite directions, *i.e.*, the angle between $\Delta\mathbf{w}$ and $\Delta\mathbf{w}_{\text{old}}$ is greater than 179°, then $\gamma = 0.5$ for the rest of the iterations. A stabilised version of FastICA has been proposed by Hyvärinen (1999) as well and a procedure similar to the one above has been used. The different speedup techniques considered above, and some additional ones, are studied further by Valpola and Särelä (2004).

Sometimes there are several signals with similar large eigenvalues. It may then be impossible to use spectral shift to accelerate their separation significantly because of small eigenvalues that would assume very negative values exceeding the signal eigenvalues in magnitude. In that case, it may be beneficial to first separate the subspace of the signals with large eigenvalues from the smaller ones. Spectral shift will then be useful in the signal subspace.

3. Approximation for the Objective Function

The virtue of the DSS framework is that it allows one to develop procedural source separation algorithms without referring to an exact objective function or a generative model. However, in many cases an approximation of the underlying objective function is nevertheless useful. In this section, we propose such an approximation (Sec. 3.1) and discuss its uses, including monitoring (Sec. 3.2) and acceleration of convergence (Sec. 3.3) as well as analysis of separation results (Sec. 3.4).

3.1 The Objective Function of DSS

The power-method version of the linear DSS algorithm maximises the variance $||\mathbf{w}^T\mathbf{Z}||^2$. When the denoising is performed for the source estimates $\mathbf{f}(\mathbf{s}) = \mathbf{s}D$, the equivalent objective function is $g(\mathbf{s}) = \mathbf{s}D\mathbf{s}^T = \mathbf{s}\mathbf{f}^T_{\text{lin}}(\mathbf{s})$. We propose this formula as an approximation $\hat{g}$ for the objective function for nonlinear DSS as well:

$$\hat{g}(\mathbf{s}) = \mathbf{s}\mathbf{f}^T(\mathbf{s})\,. \tag{30}$$

There is, however, an important caveat to be made. Note that Eq. (24) includes the scalar functions $\alpha(\mathbf{s})$ and $\beta(\mathbf{s})$. This means that functionally equivalent DSS algorithms can be implemented with slightly different denoising functions $\mathbf{f}(\mathbf{s})$ and while they would converge exactly to the same results, the approximation (30) might yield completely different values. In fact, by tuning $\alpha(\mathbf{s})$, $\beta(\mathbf{s})$ or both, the approximation $\hat{g}(\mathbf{s})$ could be made to yield any function which need not have any correspondence to the true $g(\mathbf{s})$.

Due to $\alpha(\mathbf{s})$ and $\beta(\mathbf{s})$, it seems virtually impossible to write down a simple approximation of $g(\mathbf{s})$ that could not go wrong with a malevolent choice of $\mathbf{f}(\mathbf{s})$. In the following, however, we argue that Eq. (30) is in most cases a good approximation and it is usually easy to check whether it behaves as desired—yields values which are monotonic in signal-to-noise ratio (SNR). If it does not, $\alpha(s)$ and $\beta(s)$ can be easily tuned to correct this.

Let us first check what would be the DSS algorithm maximising $\hat{g}(\mathbf{s})$. Obviously, the approximation is good if the algorithm turns out to use a denoising similar to $\mathbf{f}(\mathbf{s})$. The following Lagrange equation holds at the optimum:

$$\nabla_{\mathbf{w}}[\hat{g}(\mathbf{s}) - \xi^T \mathbf{h}(\mathbf{w})] = 0, \tag{31}$$

where $\mathbf{h}$ denotes the constraints under which the optimisation is performed and ξ are the corresponding Lagrange multipliers. In this case, only unit-length projection vectors $\mathbf{w}$ are considered, *i.e.*, $h(\mathbf{w}) = \mathbf{w}^T\mathbf{w} - 1 = 0$, and it thus follows that

$$\mathbf{X}\nabla_{\mathbf{s}}\hat{g}^T(\mathbf{s}) - 2\xi\mathbf{w} = 0. \tag{32}$$

Substituting 2ξ with the appropriate normalising factor which guarantees $||\mathbf{w}|| = 1$ results in the following fixed point:

$$\mathbf{w} = \frac{\mathbf{X}\nabla_{\mathbf{s}}\hat{g}^T(\mathbf{s})}{||\mathbf{X}\nabla_{\mathbf{s}}\hat{g}^T(\mathbf{s})||}. \tag{33}$$

Using $\mathbf{s} = \mathbf{w}^T\mathbf{X}$ and (30), and omitting normalisation yields

$$\mathbf{w}^+ = \mathbf{X}[\mathbf{f}^T(\mathbf{s}) + \mathbf{J}^T(\mathbf{s})\mathbf{s}^T], \tag{34}$$

where $\mathbf{J}$ is the Jacobian of $\mathbf{f}$. This should conform with the corresponding steps (9) and (10) in the nonlinear DSS which uses $\mathbf{f}(\mathbf{s})$ for denoising. This is true if the two terms in the square brackets have the same form, *i.e.*, $\mathbf{f}(\mathbf{s}) \propto \mathbf{s}\,\mathbf{J}(\mathbf{s})$.

As expected, in the linear case the two algorithms are exactly the same because the Jacobian is a constant matrix and $\mathbf{f}(\mathbf{s}) = \mathbf{s}\mathbf{J}$. The denoised sources are also proportional to $\mathbf{s}\,\mathbf{J}(\mathbf{s})$ in some special nonlinear cases, for instance, when $\mathbf{f}(\mathbf{s}) = \mathbf{s}^n$.

3.2 Negentropy Ordering

The approximation (30) can be readily used for monitoring the convergence of DSS algorithms. It is also easy to use it for ordering the sources based on their SNR if several sources are estimated using DSS with the same $\mathbf{f}(\mathbf{s})$. However, simple ordering based on Eq. (30) is not possible if different denoising functions are used for different sources because the approximation does not provide a universal scaling.

In these cases it is useful to order the source estimates by their negentropy which is a normalised measure of structure in the signal. Differential entropy H of a random variable is a measure of disorder and is dependent on the variance of the variable. Negentropy is a normalised quantity measuring the difference between the differential entropy of the component and a Gaussian component with the same variance. Negentropy is zero for the Gaussian distribution and non-negative for all distributions since among the distributions with a given variance, the Gaussian distribution has the highest entropy.

Calculation of the differential entropy assumes the distribution to be known. Usually this is not the case and estimation of the distribution is often difficult and computationally demanding. Following Hyvärinen (1998), we approximate the negentropy $N(\mathbf{s})$ by

$$N(\mathbf{s}) = H(\nu) - H(\mathbf{s}) \approx \eta_g[\hat{g}(\mathbf{s}) - \hat{g}(\nu)]^2, \tag{35}$$

where ν is a normally distributed variable. The reasoning behind Eq. (35) is that $\hat{g}(\mathbf{s})$ carries information about the distribution of $\mathbf{s}$. If $\hat{g}(\mathbf{s})$ equals $\hat{g}(\nu)$, there is no evidence of the negentropy to be greater than zero, so this is when $N(\mathbf{s})$ should be minimised. A Taylor series expansion of $N(\mathbf{s})$ w.r.t. $\hat{g}(\mathbf{s})$ around $\hat{g}(\nu)$ yields the approximation (35) as the first non-zero term.

Comparison of signals extracted with different optimisation criteria presumes that the weighting constants η_g are known. We propose that η_g can be calibrated by generating a signal with a known, nonzero negentropy. Negentropy ordering is most useful for signals which have a relatively poor SNR—the signals with a good SNR will most likely be selected in any case. Therefore we choose our calibration signal to have SNR of 0 dB, *i.e.*, it contains equal amounts of signal and noise in terms of energy: $\mathbf{s}_s = (\nu + \mathbf{s}_{\text{opt}})/\sqrt{2}$, where $\mathbf{s}_{\text{opt}}$ is a pure signal having no noise. It obeys fully the signal model implicitly defined by the corresponding denoising function $\mathbf{f}$. Since $\mathbf{s}_{\text{opt}}$ and ν are uncorrelated, $\mathbf{s}_s$ has unit variance. The entropy of $\nu/\sqrt{2}$ is

$$H(\nu/\sqrt{2}) = H(\nu) + \log 1/\sqrt{2} = H(\nu) - 1/2 \log 2\,.$$

Since the entropy can only increase by adding a second, independent signal $\mathbf{s}_{\text{opt}}$, $H(\mathbf{s_s}) \geq H(\nu) - 1/2\log 2$. It thus holds $N(\mathbf{s}_s) = H(\nu) - H(\mathbf{s}_s) \leq 1/2 \log 2$. One can usually expect that $\mathbf{s}_{\text{opt}}$ has a lot of structure, *i.e.*, its entropy is low. Then its addition to $\nu/\sqrt{2}$ does not significantly increase the entropy. It is therefore often reasonable to approximate

$$N(\mathbf{s}_s) \approx 1/2 \log 2 = 1/2\,\text{bit}, \tag{36}$$

where we chose base-2 logarithm yielding bits. Depending on $\mathbf{s}_{\text{opt}}$, it may also be possible to compute the negentropy $N(\mathbf{s}_s)$ exactly. This can then be used instead of the approximation (36).

The coefficients η_g in Eq. (35) can now be solved by requiring that the approximation (35) yields Eq. (36) for $\mathbf{s}_s$. This results in

$$\eta_g = \frac{1}{2(\hat{g}(\mathbf{s}_s) - \hat{g}(\nu))^2}\text{bit} \tag{37}$$

and finally, substitution of the approximation of the objective function (30) and Eq. (37) into Eq. (35) yields the calibrated approximation of the negentropy:

$$N(\mathbf{s}) \approx \frac{\left[\mathbf{s}\,\mathbf{f}^T(\mathbf{s}) - \nu\,\mathbf{f}^T(\nu)\right]^2}{2\left[\mathbf{s}_s\,\mathbf{f}^T(\mathbf{s}_s) - \nu\,\mathbf{f}^T(\nu)\right]^2}\,\text{bit}. \tag{38}$$

3.3 Spectral Shift Revisited

In Sec. 2.5, we suggested that a reasonable spectral shift is to move the eigenvalue corresponding to a Gaussian signal ν to zero. This leads to minimising $g(\mathbf{s})$, when the largest absolute eigenvalue is negative. It does not seem very useful to minimise $g(\mathbf{s})$, a function that measures the SNR of the sources, but as we saw with negentropy and its approximation (35), values $g(\mathbf{s}) < g(\nu)$ are, in fact, indicative of signal. A reasonable selection for β is thus $-\lambda_\nu$ given by (27) which leads linear DSS to extremise $g(\mathbf{s}) - g(\nu)$ or, equivalently, to maximise the negentropy approximation (35).

A well known example where the spectral shift by the eigenvalue of a Gaussian signal is useful is the mixture of both super- and sub-Gaussian distributions. A DSS algorithm designed for super-Gaussian distributions would lead to $\lambda > \lambda_\nu$ for super-Gaussian and $\lambda < \lambda_\nu$ for sub-Gaussian distributions, λ_ν being the eigenvalue of the Gaussian signal. By shifting the eigenvalue spectrum by $-\lambda_\nu$, the most non-Gaussian distributions will result in the largest absolute eigenvalues regardless of whether the distribution is super- or sub-Gaussian. By using the spectral shift it is therefore possible to extract both super- and sub-Gaussian distributions with a denoising scheme which is designed for one type of distribution only.

Consider for instance $\mathbf{f}(\mathbf{s}) = \tanh \mathbf{s}$ which can be used as a denoising function for sub-Gaussian signals while, as will be further discussed in Sec. 4.2.3, $\mathbf{s} - \tanh \mathbf{s} = -(\tanh \mathbf{s} - \mathbf{s})$ is a suitable denoising for super-Gaussian signals. This shows that depending on the choice of β, DSS can find either sub-Gaussian ($\beta = 0$) or super-Gaussian ($\beta = -1$) sources. With the FastICA spectral shift (27), β will always lie in the range $-1 < \beta \leq \tanh^2 1 - 1 \approx -0.42$. In general, β will be closer to -1 for super-Gaussian sources which shows that FastICA is able to adapt its spectral shift to the source distribution.

3.4 Detection of Overfitting

In exploratory data analysis, DSS is very useful for giving better insight into the data using a linear factor model. However, it is possible that DSS extracts structures that are due to noise, *i.e.*, the results may be overfits.

Overfitting in ICA has been extensively studied by Särelä and Vigário (2003). It was observed that it typically results in signals that are mostly inactive, except for a single spike. In DSS the type of the overfitted results depends on the denoising criterion.

To detect an overfitted result, one should know what it looks like. As a first approximation, DSS can be performed with the same amount of i.i.d. Gaussian data. Then all the results present cases of overfitting. An even better characterisation of the overfitting results can be obtained by mimicking the actual data characteristics as closely as possible. In that case it is important to make sure that the structure assumed by the signal model has been broken. Both the Gaussian overfitting test and the more advanced test are used throughout the experiments in Secs. 5.2–5.3.

Note that in addition to visual test, the methods described above provide us with a quantitative measure as well. Using the negentropy approximation (38), we can set a threshold under which the sources are very likely overfits and do not carry much real structure. In the simple case of linear DSS, the negentropy can be approximated easily using the corresponding eigenvalue.

4. Denoising Functions in Practice

DSS is a framework for designing source separation algorithms. The idea is that the algorithms differ in the denoising function $\mathbf{f}(\mathbf{s})$ while the other parts of the algorithm remain mostly the same. Denoising is useful as such and therefore there is a wide literature of sophisticated denoising methods to choose from (see Anderson and Moore, 1979). Moreover, one usually has some knowledge about the signals of interest and thus possesses the information needed for denoising. In fact, quite often the signals extracted by BSS techniques would be post-processed to reduce noise in any case (see Vigneron et al., 2003). In the DSS framework, the available denoising methods can be directly applied to source separation, producing better results than purely blind techniques. There are also very general noise reduction techniques such as wavelet denoising (Donoho et al., 1995; Vetterli and Kovacevic, 1995) or median filtering (Kuosmanen and Astola, 1997) which can be applied in exploratory data analysis.

In this section, we discuss denoising functions ranging from simple but powerful linear ones to sophisticated nonlinear ones with the goal of inspiring others to try out their own denoising methods. The range of applicability of the examples spans from cases where knowledge about the signals is relatively specific to almost blind source separation. Many of the denoising functions discussed in this section are applied in experiments in Sec. 5.

The DSS framework has been implemented in an open-source and publicly available MATLAB package (DSS, 2004). The package contains the denoising functions and speedups discussed in this paper and in another paper (Valpola and Särelä, 2004). It is modular and allows for custom-made functions (denoising, spectral shift, and other parts) to be nested in the core program.

Before proceeding to examples of denoising functions, we note that DSS would not be very useful if very exact denoising would be needed. Fortunately, this is usually not the case and it is enough for the denoising function $\mathbf{f}(\mathbf{s})$ to remove more noise than signal (see Hyvärinen et al., 2001b, Theorem 8.1), assuming that the independent source model holds. This is because the re-estimation steps (10) and (11) constrain the source $\mathbf{s}$ to the subspace spanned by the data. Even if the denoising discards parts of the signal or creates nonexistent signals, re-estimation steps restore them.

If there is no detailed knowledge about the characteristics of the signals to start with, it is useful to bootstrap the denoising functions. This can be achieved by starting with relatively general signal characteristics and then tuning the denoising functions based on analyses of the structure in the noisy signals extracted in the first phase. In fact, some of the nonlinear DSS algorithms can be regarded as linear DSS algorithms where a linear denoising function is adapted to the sources, leading to nonlinear denoising.

4.1 Detailed Linear Denoising Functions

In this section, we consider several detailed, simple but powerful, linear denoising schemes. We introduce the denoisings using the denoising matrix $\mathbf{D}$ when feasible. We consider efficient implementation of the denoisings as well.

The eigenvalue decomposition (14) shows that any denoising in linear DSS can be implemented as an orthonormal rotation followed by a point-wise scaling of the samples and rotation back to the original space. The eigenvalue decomposition of the denoising matrix $\mathbf{D}$ often offers good intuitive insight into the denoising function as well as practical means for its implementation.

4.1.1 ON/OFF-DENOISING

Consider designed experiments, *e.g.*, in the fields of psychophysics or biomedicine. It is usual to control them by having periods of activity and non-activity. In such experiments, the denoising can be simply implemented by

$$\mathbf{D} = \mathrm{diag}(\mathbf{m}), \tag{39}$$

where $\mathbf{D}$ refers to the linear denoising matrix in Eq. (9) and

$$\mathbf{m} = \begin{cases} 1, \text{for the active parts} \\ 0, \text{for the inactive parts} \end{cases} \tag{40}$$

This amounts to multiplying the source estimate $\mathbf{s}$ by a binary mask,[4] where ones represent the active parts and zeroes the non-active parts. Notice that this masking procedure actually satisfies $\mathbf{D} = \mathbf{D}\mathbf{D}^T$. This means that DSS is equivalent to the PCA applied to denoised $\mathbf{Z} = \mathbf{X}\mathbf{D}$ even with

4. By masking we refer to point-wise multiplication of a signal or a transformation of a signal.

exactly the same filtering. In practice this DSS algorithm could be implemented by PCA applied to the active parts of the data with the sphering stage would still involving the whole data set.

4.1.2 DENOISING BASED ON FREQUENCY CONTENT

If, on the other hand, signals are characterised by having certain frequency components, one can transform the source estimate to a frequency space, mask the spectrum, *e.g.*, with a binary mask, and inverse transform to obtain the denoised signal:

$$\mathbf{D} = \mathbf{V}\Lambda_D\mathbf{V}^T ,$$

where $\mathbf{V}$ is the transform, Λ_D is the matrix with the mask on its diagonal, and $\mathbf{V}^T$ is the inverse transform. The transform $\mathbf{V}$ can be implemented for example with the Fourier transform[5] or by discrete cosine transform (DCT). After the transform, the signal is filtered using the diagonal matrix Λ, *i.e.*, by a point-wise scaling of the frequency bins. Finally the signal is inverse transformed using $\mathbf{V}^T$. In the case of linear time-invariant (LTI) filtering, the filtering matrix has a Toeplitz structure and the denoising characteristics are manifested only in the diagonal matrix Λ_D, while the transforming matrix $\mathbf{V}$ represents a constant rotation. When this is the case, the algorithm can be further simplified by imposing the transformation on the sphered data $\mathbf{X}$. Then the iteration can be performed in the transformed basis. This trick has been exploited in the first experiment of Sec. 5.2.

4.1.3 SPECTROGRAM DENOISING

Often a signal is well characterised by what frequencies occur at what times. This is evident, *e.g.*, in oscillatory activity in the brain where oscillations often occur in bursts. An example of source separation in such data is studied in Sec. 5.2. The time-frequency behaviour can be described by calculating DCT in short windows in time. This results in a combined time and frequency representation, i.e., a spectrogram, where the masking can be applied.

There is a known dilemma in the calculation of the spectrogram: detailed description of the frequency content does not allow detailed information of the activity in time and vice versa. In other words, a large amount of different frequency bins T_f will result in a small amount of time locations T_t. Wavelet transforms (Donoho et al., 1995; Vetterli and Kovacevic, 1995) have been suggested to overcome this problem. There an adaptive or predefined basis, different from the pure sinusoids used in Fourier transform or DCT, is used to divide the resources of time and frequency behaviour optimally in some sense. Another possibility is to use the so-called multitaper technique (Percival and Walden, 1993, Ch. 7).

Here we apply an overcomplete-basis approach related to the above methods. Instead of having just one spectrogram, we use several time-frequency analyses with different T_t's and T_f's. Then the new estimate of the projection $\mathbf{w}^+$ is achieved by summing the new estimates $\mathbf{w}_i^+$ of each of the time-frequency analyses: $\mathbf{w}^+ = \sum_i \mathbf{w}_i^+$.

4.1.4 DENOISING OF QUASIPERIODIC SIGNALS

As a final example of denoising based on detailed source characteristics, consider Fig. 2a. Let us assume to be known beforehand that the signal $\mathbf{s}$ has a repetitive structure and that the average

5. Note that the eigenvalue decomposition contains real rotations instead of complex, but Fourier transform is usually seen as a complex transformation. To keep the theory simple, we consider real Fourier transform where the corresponding sine and cosine terms have been separated in different elements.

repetition rate is known. The quasi-periodicity of the signal can be used to perform DSS to get a better estimate. The denoising proceeds as follows:

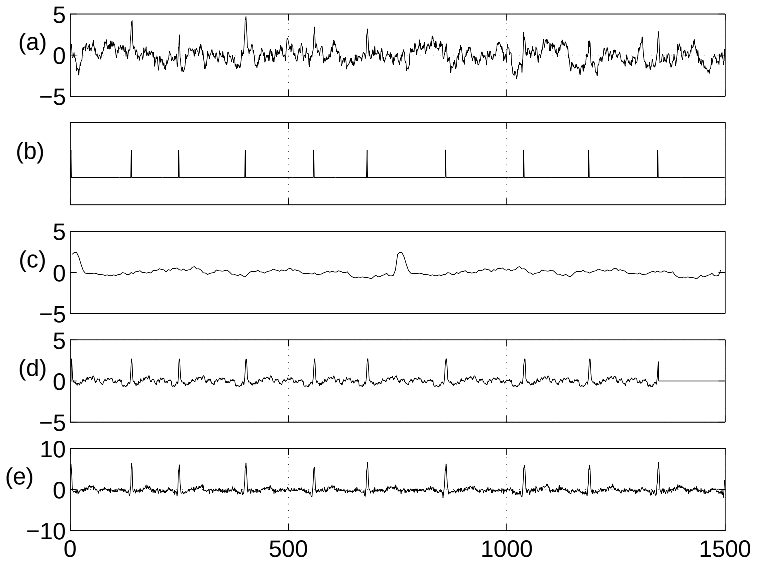


Figure 2: *a) Current source estimate* **s** *of a quasiperiodic signal b) Peak estimates c) Average signal* s_{ave} *(two periods are shown for clarity). d) Denoised source estimate* $\mathbf{s}^+$*. e) Source estimate corresponding to the re-estimated* $\mathbf{w}_{\text{new}}$.

1. Estimate the locations of the peaks of the current source estimate **s** (Fig. 2b).
2. Chop each period from peak to peak.
3. Dilate each period to a fixed length L (linearly or nonlinearly).
4. Average the dilated periods (Fig. 2c).
5. Let the denoised source estimate $\mathbf{s}^+$ be a signal where each period has been replaced by the averaged period dilated back to its original length (Fig. 2d).

The re-estimated signal in Fig. 2e, based on the denoised signal $\mathbf{s}^+$, shows significantly better SNR compared to the original source estimate **s**, in Fig. 2a.

This averaging is a form of linear denoising since it can be implemented as matrix multiplication. Furthermore, it presents another case in addition to the binary masking, where DSS is equivalent to the power method even with exactly the same filtering. It would not be easy to see from the denoising matrix $\mathbf{D}$ itself that $\mathbf{D} = \mathbf{D}\mathbf{D}^T$. However, this becomes evident should one consider the averaging of source estimate $\mathbf{s}^+$ (Fig. 2d) that is already averaged.

Note that there are cases where chopping from peak to peak does not guarantee the best result. This is especially true when the periods do not span the whole section from peak to peak, but there are parts where the response is silent. Then there is a need to estimate the lengths of the periods separately.

4.2 Denoising Based on Estimated Signal Variance

In the previous section, several denoising schemes were introduced. In all of them, the details of the denoising were assumed to be known. It is as well possible to estimate the denoising specifications from the data. This makes the denoising nonlinear or adaptive. In this section, we consider a particular ICA algorithm in the DSS framework, suggesting modifications which improve separation results and robustness.

4.2.1 KURTOSIS-BASED ICA

Consider one of the best known BSS approaches, ICA by optimisation of the sample kurtosis of the sources. The objective function is then $g(\mathbf{s}) = \sum s^4(t)/T - 3\left(\sum s^2(t)/T\right)^2$. Since the source variance has been fixed to unity, we can simply use $g(\mathbf{s}) = \sum s^4(t)/T$ and derive the function $\mathbf{f}(\mathbf{s})$ from gradient ascend. This yields $\nabla_{\mathbf{s}} g(\mathbf{s}) = 4/T\,\mathbf{s}^3$, where $\mathbf{s}^3 = [s^3(1)\, s^3(2) \ldots]$. Selecting $\alpha(\mathbf{s}) = T/4$ and $\beta(\mathbf{s}) = 0$ in Eq. (24) then result in

$$\mathbf{f}(\mathbf{s}) = \mathbf{s}^3 . \tag{41}$$

This implements an ICA algorithm with nonlinear denoising. So far, we have not referred to denoising, but a closer examination of Eq. (41) reveals that one can, in fact, interpret $\mathbf{s}^3$ as being $\mathbf{s}$ masked by $\mathbf{s}^2$, the latter being a somewhat naïve estimate of signal variance and thus relating to SNR.

Kurtosis as an objective function is notorious for being prone to overfitting and producing very spiky source estimates (Särelä and Vigário, 2003; Hyvärinen, 1998). For illustration of this consider Fig. 3. There one iteration of DSS using kurtosis-based denoising is shown. Assume that via some means, the source estimate shown in Fig. 3a has been reached. The source seems to contain increased activity in three portions (around time instances 1000, 2300 and 6000). As well, it contains a peak roughly at time instance 4700. The signal variance estimate, *i.e.*, the mask is shown in Fig. 3b. While it has boosted somewhat the broad activity compared to the silent parts, the magnification of the peak is far greater. Thus the denoised source estimate $\mathbf{s}^+$ (Fig. 3c) has nearly nothing else except the peak. The new source estimate $\mathbf{s}_{\text{new}}$, based on the new projection $\mathbf{w}_{\text{new}}$, is a clear spike having little left of the broad activity.

The denoising interpretation suggests that the failure to extract the broad activity is due to a poor estimate of SNR.

4.2.2 BETTER ESTIMATE FOR THE SIGNAL VARIANCE

Let us now consider a related but better founded estimate. Assume that $\mathbf{s}$ is composed of Gaussian noise with a constant variance σ_n^2 and of a Gaussian signal with non-stationary variance $\sigma_s^2(t)$. From Eq. (12) it follows that

$$s^+(t) = s(t)\frac{\sigma_s^2(t)}{\sigma_{\text{tot}}^2(t)} , \tag{42}$$

where $\sigma_{\text{tot}}^2(t) = \sigma_s^2(t) + \sigma_n^2$ is the total variance of the observation. This is also the maximum-a-posteriori (MAP) estimate.

The kurtosis-based DSS (41) can be obtained from this MAP estimate if the signal variance is assumed to be far smaller than the total variance. In that case it is reasonable to assume σ_{tot}^2 to be constant and $\sigma_s^2(t)$ can be estimated by $s^2(t) - \sigma_n^2$. Subtraction of σ_n^2 does not affect the fixed points as it can be embedded in the term $\beta(\mathbf{s}) = -\sigma_n^2$ in Eq. (24). Likewise, the division by $\sigma_{\text{tot}}^2(t)$ is absorbed by $\alpha(\mathbf{s})$.

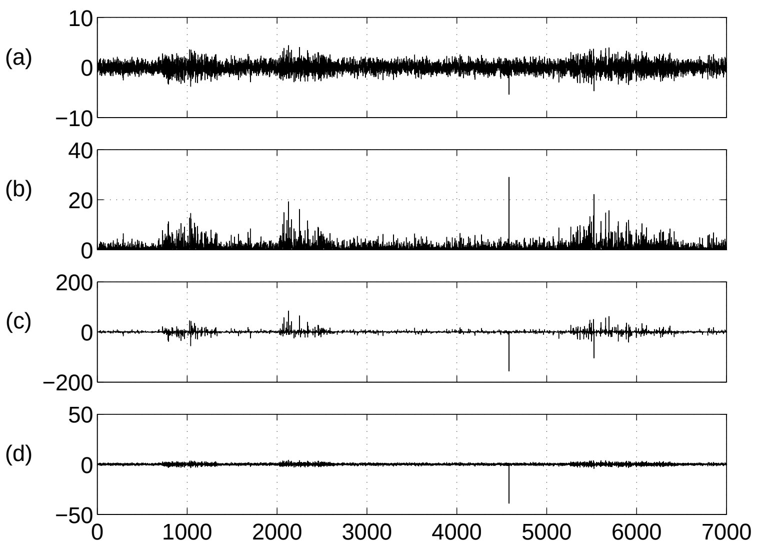


Figure 3: *a) Source estimate* $\mathbf{s}$ *b) Mask* $s^2(t)$ *c) Denoised source estimate* $\mathbf{s}^+ = \mathbf{f}(\mathbf{s}) = \mathbf{s}^3$ *d) Source estimate corresponding to the re-estimated* $\mathbf{w}_{\text{new}}$.

Comparison of Eq. (42) and Eq. (41) immediately suggests improvements to the kurtosis-based DSS. For instance, it is clear that if $s^2(t)$ is large enough, it is not reasonable to assume that $\sigma_s^2(t)$ is small compared to $\sigma_n^2(t)$. Instead, the mask should saturate for large $s^2(t)$. This already improves robustness against outliers and alleviates the tendency to produce spiky source estimates.

We suggest the following improvements over the kurtosis-based denoising function (41):

1. The estimates of signal variance and total variance are based on several observations. The rationale of smoothing is the assumption of smoothness of the signal variance. In practice this can be achieved by low-pass filtering the variance of the time, frequency or time-frequency description of $s(t)$, yielding the approximation of total variance.

2. The noise variance is likewise estimated from the data. It should be some kind of soft minimum of the estimated total variances because the estimate can be expected to have random fluctuations. We suggest the following formula:

$$\sigma_n^2 = C\left(\exp\left\{E\left[\log\left(\sigma_{\text{tot}}^2(t) + \sigma_n^2\right)\right]\right\} - \sigma_n^2\right) . \tag{43}$$

The noise variance σ_n^2 appears on both sides of the equation, but at the right-hand side, it appears only to prevent rare small values of σ_{tot}^2 from spoiling the estimate. Hence, we suggest to use the previously estimated value on the right-hand side. The constant C is tuned such that the formula gives a consistent estimate of the noise variance if the source estimate is, in fact, nothing but Gaussian noise.

3. The signal variance should be close to the estimate of the total variance minus the estimate of the noise variance. Since a variance cannot be negative and the estimate of the total variance

has fluctuations, we use a formula which yields zero only when the total variance is zero but which asymptotically approaches $\sigma^2_{\text{tot}}(t) - \sigma^2_n$ for large values of the total variance:

$$\sigma^2_s(t) = \sqrt{\sigma^4_{\text{tot}}(t) + \sigma^4_n} - \sigma^2_n\,. \tag{44}$$

As an illustration of these improvements consider Fig. 4 where one iteration of DSS using the MAP estimate is shown. The first two subplots (Fig. 4a and b) are identical to the ones using kurtosis-based denoising. In Fig. 4c, the variance estimate is smoothed using low-pass filtering. Note that the broad activity has been magnified when compared to the spike around time instance 4700. The noise level σ^2_n, calculated using Eq. (43), is shown using a dashed line. Corresponding masking (Fig. 4d) results in a denoised source estimate using Eq. (42), shown in Fig. 4e. Finally, the new source estimate $\mathbf{s}_{\text{new}}$ is shown after five iterations of DSS in Fig. 4f. DSS using the MAP-based denoising has clearly removed a considerable amount of background noise as well as the lonely spike.

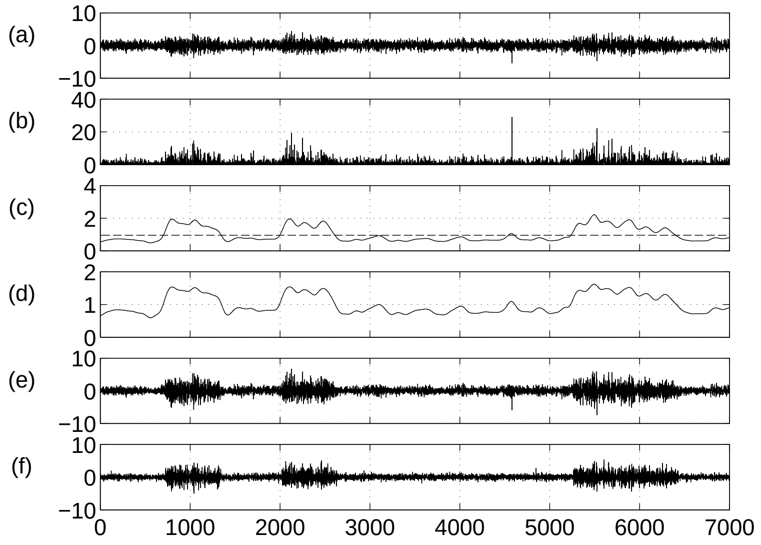


Figure 4: *a) Source estimate* $\mathbf{s}$ *b)* $s^2(t)$ *c) Smoothed total variance with the noise level in dashed line d) Denoising mask e) Denoised source estimate* $\mathbf{s}^+$ *f) Source estimate after five iterations of DSS.*

The exact details of these improvements are not crucial, but we wanted to show that the denoising interpretation of Eq. (41) can carry us quite far. The above estimates plugged into Eq. (42) yield a DSS algorithm which is far more robust against overfitting, does not produce the spiky signal estimates and in general yields signals with better SNRs than kurtosis.

Despite the merits of the DSS algorithm described above, there is still one problem with it. While the extracted signals have excellent SNR, they do not necessarily correspond to independent sources, *i.e.*, the sources may remain mixed. This is because there is nothing in the denoising which could discard other sources. In terms of eigenvalues, when $\mathbf{s}$ is in the vicinity of one of the fixed points $\mathbf{s}_i^*$, the local eigenvalue $\lambda_i(\mathbf{s}_i^*)$ is much larger than λ_ν, as it should, but $\lambda_j(\mathbf{s}_i^*)$ may be large, too, which means that the iterations do not remove the contribution of the weaker sources efficiently.

Assume, for instance, that two sources have clear-cut and non-overlapping times of strong activity ($\sigma_s^2(t) \gg 0$) and remain silent for most of the time ($\sigma_s^2(t) = 0$). Suppose that one source is present for some time at the beginning of the data and another at the end. If the current source estimate is a mixture of both, the mask will have values close to one at the beginning and at the end of the signal. Denoising can thus clean the noise from the signal estimate, but it cannot decide between the two sources.

In this respect, kurtosis actually works better than DSS based on the above improvements. This is because the mask never saturates and small differences in the strengths of the relative contributions of two original sources in the current source estimate will be amplified. This problem only occurs in the saturated regime of the mask and we therefore suggest a simple modification of the MAP estimate (42):

$$\mathbf{f}_t(\mathbf{s}) = s(t)\frac{\sigma_s^{2\mu}(t)}{\sigma_{\text{tot}}^2(t)}, \tag{45}$$

where μ is a constant slightly greater or equal to one. Note that this modification is usually needed at the beginning of the iterations only. Once the source estimate is dominated by one of the original sources and the contributions of the other sources fall closer to the noise level, the values of the mask are smaller for the other original sources possibly still present in the estimated source.

Another approach is based on the observation that orthogonalising the mixing vectors $\mathbf{A}$ cancels only the linear correlations between different sources. Higher-order correlations may still exist. It can be assumed that competing sources contribute to the current variance estimate: $\sigma_{\text{tot}}^2(t) = \sigma_s^2(t) + \sigma_n^2 + \sigma_{\text{others}}^2(t)$, where $\sigma_{\text{others}}^2(t)$ stands for the estimate of total leakage of variance from the other sources. Valpola and Särelä (2004) showed that decorrelating the variance-based masks actively promotes the separation of the sources. This bares resemblance to proposals of the role of divisive normalisation on cortex (Schwartz and Simoncelli, 2001) and to the classical ICA method called JADE (Cardoso, 1999).

The problems related to kurtosis are well known and several other improved nonlinear functions $\mathbf{f}(\mathbf{s})$ have been proposed. However, some aspects of the above denoising, especially smoothing of the total-variance estimate $s^2(t)$, have not been suggested previously although they arise quite naturally from the denoising interpretation.

4.2.3 TANH-NONLINEARITY INTERPRETED AS SATURATED VARIANCE ESTIMATE

A popular replacement of the kurtosis-based nonlinearity (41) is the hyperbolic tangent $\tanh(\mathbf{s})$ operating point-wise on the sources. It is generally considered to be more robust against overfitted and spiky source estimates than kurtosis. By selecting $\alpha(\mathbf{s}) = -1$ and $\beta(\mathbf{s}) = -1$, we arrive at

$$\mathbf{f}_t(\mathbf{s}) = s(t) - \tanh[s(t)] = s(t)\left(1 - \frac{\tanh[s(t)]}{s(t)}\right). \tag{46}$$

Now the term multiplying $s(t)$ can be interpreted as a mask related to SNR. Unlike the naïve mask $s^2(t)$ resulting from kurtosis, the tanh-based mask (46) saturates, though not very fast.

The variance based mask (45) with the improvements considered above offers a new interpretation for the robustness of the tanh-mask. Parameter values $\sigma_n^2 = 1$ and $\mu = 1.08$ give an excellent fit between the masks as shown in Fig. 5. The advantages of the denoising we propose are that σ_n^2

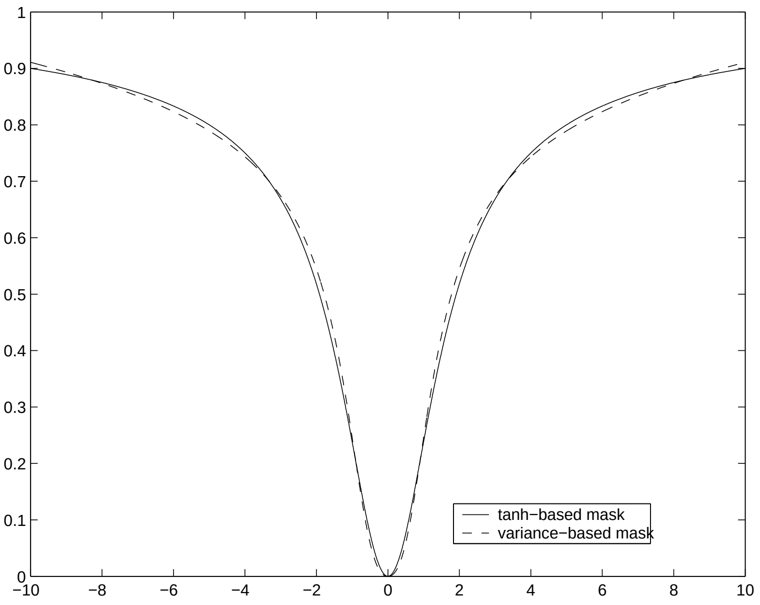


Figure 5: *The tanh-based denoising mask* $1-\tanh(s)/s$ *is shown together with the variance-based denoising mask proposed here. The parameters in the proposed mask were* $\sigma_n^2 = 1$ *and* $\mu = 1.08$. *We have scaled the proposed mask to match the scale of the tanh-based mask.*

can be tuned to the source estimate, μ can be controlled during the iterations and the estimate of the signal variance can be smoothed. These features contribute to the resistance against overfitting and spiky source estimates.

4.3 Other Denoising Functions

There are cases where the system specification itself suggests some denoising schemes. One such case, CDMA transmission, is described in Sec. 5.4. Another example is source separation with a microphone array combined with speech recognition. Many speech recognition systems rely on generative models which can be readily used to denoise the speech signals.

Often it would be useful to be able to separate the sources online, *i.e.*, in real time. Since there exists online sphering algorithms (see Douglas and Cichocki, 1997; Oja, 1992), real time DSS can be considered as well. One simple case of online denoising is presented by moving-average filters. Such online filters are typically not symmetric and the eigenvalues (14) of the matrix $\mathbf{XDX}^T$ may be complex numbers. These eigenvalues come in conjugate pairs and are analogous to sine-cosine pairs. The resulting DSS algorithm converges to a 2-D subspace corresponding to the eigenvalues with largest absolute magnitude, but fails to converge within the subspace. Consider, for example, a case of two harmonic oscillatory sources. It has a rotational invariance in a space defined by the corresponding sine-cosine pair. Batch DSS algorithms with temporally symmetric denoising would converge to some particular rotation, but non-symmetric on-line denoising by $\mathbf{f}(s(t)) = s(t-1)$ would keep oscillating between sine and cosine components.

The above is a special case of subspace analysis and there are several other examples where the sources can be grouped to form interesting subspaces. This can be the case, *e.g.*, when all the sources are not independent of each others, but form subspaces that are mutually independent. It may be desirable to use the information in all sources $\mathbf{S}$ for denoising any particular source $\mathbf{s}_i$. This leads to the following denoising function: $\mathbf{s}_i^+ = \mathbf{f}_i(\mathbf{S})$. Some form of subspace rules can be used to guide the extraction of interesting subspaces in DSS. It is possible to further relax the independence criterion at the borders of the subspaces. This can be achieved by incorporating a neighbourhood denoising rule in DSS, resulting in a topographic ordering of the sources. This suggests a fast fixed-point algorithm that can be used instead of the gradient-descent-based topographic ICA (Hyvärinen et al., 2001a).

It is also possible to combine various denoising functions when the sources are characterised by more than one type of structure. Note that the combination order might be crucial for the outcome. This is simply because, in general, $\mathbf{f}_i(\mathbf{f}_j(\mathbf{s})) \neq \mathbf{f}_j(\mathbf{f}_i(\mathbf{s}))$ where $\mathbf{f}_i$ and $\mathbf{f}_j$ present two different linear or nonlinear denoisings. As an example, consider the combination of the linear on/off-mask (39) and (40), and the nonlinear variance-based mask (45): the noise estimation becomes significantly more accurate when the on/off-masking is performed only after the nonlinear denoising.

Finally, a source might be almost completely known. Then it is possible to apply a detailed matched filter to estimate the mixing coefficients or the noise level. Detailed matched filters have been used in Sec. 5.1 to get an upper limit of the SNRs of the source estimates.

4.4 Spectral Shift and Approximation of the Objective Function with Mask-Based Denoisings

In Sec. 3.1, it was mentioned that a DSS algorithm may work perfectly fine but (30) may still fail to approximate the true objective function if $\alpha(\mathbf{s})$ and $\beta(\mathbf{s})$ are not selected suitably. As an example, consider the mask-based denoisings where denoising is implemented by multiplying the source point-wise by a mask. Without loss of generality, it can be assumed that the data has been rotated with $\mathbf{V}$ and the masking operates directly on the source. According to Eq. (30), $g(\mathbf{s}) = \sum_t s^2(t)m(t)$, where $m(t)$ is the mask. If the mask is constant w.r.t. $\mathbf{s}$, denoising is linear and Eq. (30) is an exact formula, but let us assume that the mask is computed based on the current source estimate $\mathbf{s}$.

In some cases it may be useful to normalise the mask and this could be implemented in several ways. Some possibilities that may come to mind are to normalise the maximum value or the sum of squared values of the mask. While this type of normalisation has no effect on the behaviour of DSS, it can render the approximation (30) useless. This is because a maximally flat mask usually corresponds to a source with a low SNR. However, after normalisation, the sum of values in the mask would be greatest for a maximally flat mask and this tends to produce high values of the approximation of $g(\mathbf{s})$ conflicting with the low SNR.

As a simple example, consider the mask to be $m(t) = s^2(t)$. This corresponds to the kurtosis-based denoising (41). Now the sum of squared values of the mask is $\sum s^4(t)$, but so is $\mathbf{s}\mathbf{f}^T(\mathbf{s})$. If the mask were normalised by dividing by the sum of squares, the approximation (30) would always yield a constant value of one, totally independent of $\mathbf{s}$.

A better way of normalising a mask is to normalise the sum of the values. Then Eq. (30) should always yield approximately the same value if the mask and source estimate are unrelated, but the value would be greater for cases where the magnitude of the source is correlated with the value of the mask. This is usually a sign of a structured source and a high SNR.

The above normalisation also has the benefit that the eigenvalue of a Gaussian signal can be expected to be roughly constant. Assuming that the mask $m(t)$ does not depend very much on the source estimate, the Jacobian matrix $\mathbf{J}(\mathbf{s})$ of $\mathbf{f}(\mathbf{s})$ is roughly diagonal with $m(t)$ as the elements on the diagonal. The trace of $\mathbf{J}(\mathbf{s})$ needed for the estimate of the eigenvalue of a Gaussian signal in (27) is then $\sum_t m(t)$ and the appropriate spectral shift is

$$\beta = -\frac{1}{T}\sum_t m(t)\,. \tag{47}$$

The spectral shift can thus be approximated to be constant due to the normalisation.

5. Experiments

In this section, we demonstrate the separation capabilities of the algorithms presented earlier. The experiments can be carried out using the publicly available MATLAB package (DSS, 2004).

The experimental section contains the following experiments: First, in Sec. 5.1, we separate artificial signals with different DSS schemes, some of which can be implemented by FastICA (1998); Hyvärinen (1999). Furthermore, we compare the results to one standard ICA algorithm, JADE (1999); Cardoso (1999). In Secs. 5.2–5.3, linear and nonlinear DSS algorithms are applied extensively in the study of magnetoencephalograms (MEG). Finally, in Sec. 5.4, recovery of CDMA signals is demonstrated. In each experiment after the case of artificial sources, we first discuss the nature of the expected underlying sources. Then we describe this knowledge in the form of denoising.

5.1 Artificial Signals

Artificial signals were mixed to compare different DSS schemes and JADE (Cardoso, 1999). Ten mixtures of the five sources were produced and independent white noise was added with different SNRs ranging from nearly noiseless mixtures of 50dB to -10dB, a very noisy case. The original sources and the mixtures are shown in Figs. 6a and 6b respectively. The mixtures shown have SNR of 50 dB.

5.1.1 LINEAR DENOISING

In this section, we show how the simple linear denoising schemes described in Sec. 4.1 can be used to separate the artificial sources. These schemes require prior knowledge about the source characteristics.

The base frequencies of the first two signals were assumed to be known. Thus two band-pass filtering masks were constructed around these base frequencies. The third and fourth source estimates were known to have periods of activity and non-activity. The third was known to be active in the second quadrant and the fourth a definite period in the latter half. They were denoised using binary masks in the time domain. Finally, the fifth source had a known quasi-periodic repetition rate and was denoised using the averaging procedure described in Sec. 4.1.4 and Fig. 2. Since all the five denoisings are linear, five separate filtered data sets were produced and PCA was used to recover the principal components. The separation results are described in Sec. 5.1.3 together with the results of other DSS schemes and JADE.

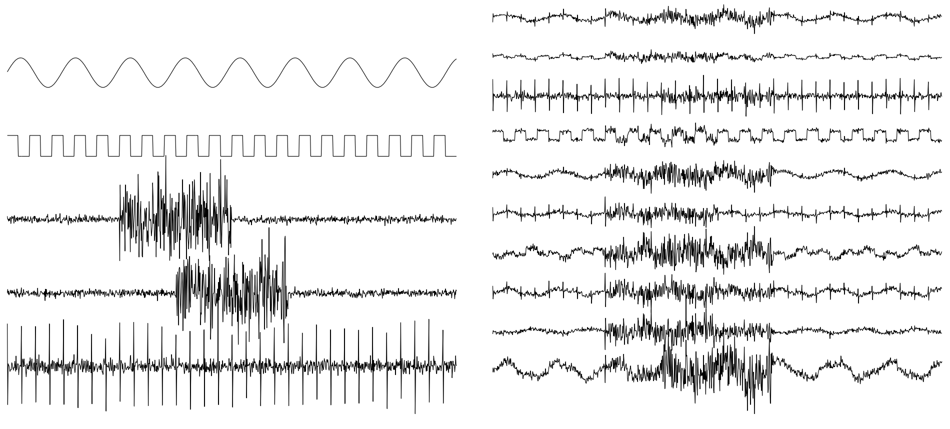


Figure 6: *(a) Five artificial signals with simple frequency content (signals 1 and 2), simple on/off non-stationarity in time domain (signals 3 and 4) or quasi-periodicity (signal 5). (b) Ten mixtures of the signals in (a).*

5.1.2 NONLINEAR EXPLORATORY DENOISING

In this section, we describe an exploratory source separation of the artificial signals. One author of this paper gave the mixtures to the other author whose task was to separate the original signals. The testing author did not receive any additional information, so he was forced to apply a blind approach. He chose to use the masking procedure based on the instantaneous variance estimate, described in Sec. 4.2. To enable the separation of both sub- and super-Gaussian sources in the MAP-based signal-variance-estimate denoising, he used the spectral shift (47). To ensure convergence, he used the 179-rule to control the step size γ (28). Finally, he did not smooth $s^2(t)$ but used it directly as the estimate of the total instantaneous variance $\sigma^2_{\mathrm{tot}}(t)$.

Based on the separation results of the variance-based DSS, he further devised specific masks for each of the sources. He chose to denoise the first source in frequency domain with a strict band-pass filter around the main frequency. The testing author decided to denoise the second source by a simple denoising function $\mathbf{f}(\mathbf{s}) = \mathrm{sign}(\mathbf{s})$. This makes quite an accurate signal model though it neglects the behaviour of the source in time. The third and fourth signal seemed to have periods of activity and non-activity. He found an estimate for the active periods by inspecting the instantaneous variance estimates $\mathbf{s}^2$, and devised simple binary masks. The last signal seemed to consist of alternating positive and negative peaks with a fixed inter-peak-interval as well as some additive Gaussian noise. The signal model was tuned to model the peaks only.

5.1.3 SEPARATION RESULTS

In this section, we compare the separation results of the linear denoising (Sec. 5.1.1), variance-based denoising and adapted denoising (Sec 5.1.2) to other DSS algorithms. In particular, we compare to the popular denoising schemes $\mathbf{f}(\mathbf{s}) = \mathbf{s}^3$ and $\mathbf{f}(\mathbf{s}) = \tanh(\mathbf{s})$, suggested for use with FastICA (1998).

We compare to JADE (Cardoso, 1999) as well. During sphering in JADE, the number of dimensions was either reduced ($n = 5$) or all the ten dimensions were kept ($n = 10$).

We restrained from using deflation in all the different DSS schemes to avoid suffering from cumulative errors in the separation of the first sources. Instead one source was extracted with each of the masks several times using different initial vector $\mathbf{w}$ until five sufficiently different source estimates were reached (see Himberg and Hyvärinen, 2003; Meinecke et al., 2002, for further possibilities along these lines). Deflation was only used if no estimate could be found for all the 5 sources. This was often the case for poor SNR under 0dB.

To get some idea of statistical significance of the results, each algorithm was used to separate the sources ten times with the same mixtures, but with different measurement noises. The average SNRs of the sources are depicted in Fig. 7. The straight line above all the DSS schemes represents the optimal separation. It is achieved by calculating the unmixing matrix explicitly using the true sources.

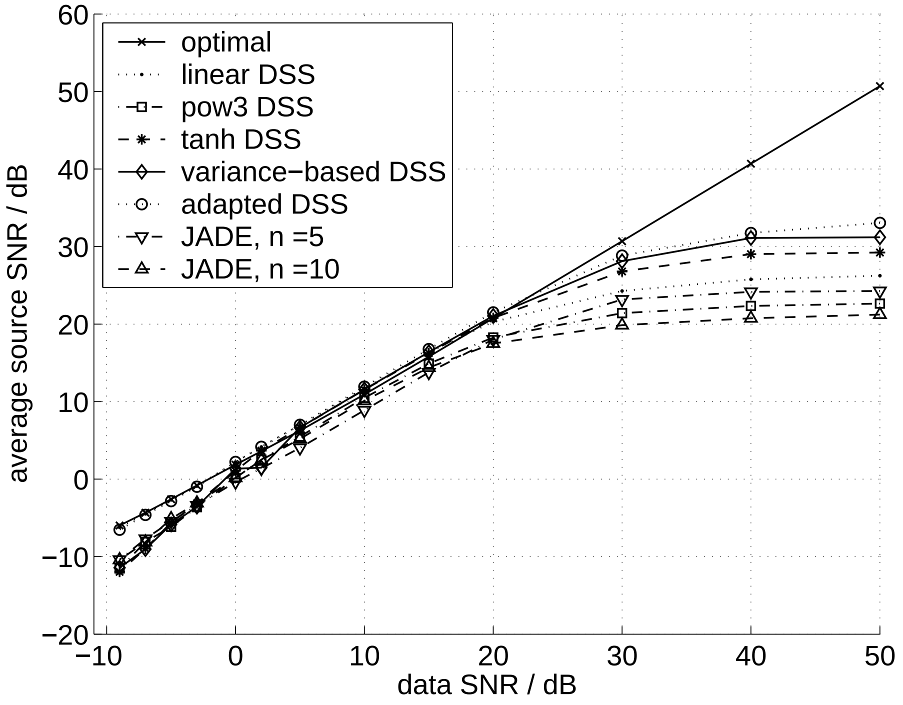


Figure 7: *Average SNRs for the estimated sources averaged over 10 runs.*

With outstanding SNR ($>$ 20 dB), linear DSS together with JADE and kurtosis-based DSS perform the worst, while the other, nonlinear DSS approaches: tanh-based, sophisticated variance estimate and the adapted one perform better. The gap between these groups is more than two standard deviations of the 10 runs, making the difference statistically significant.

With moderate SNRs (between 0 and 20 dB), all algorithms perform quite alike. With poor SNR ($<$ 0 dB), the upper group consist of the linear and adapted DSS as well as the optimal one and the lower group consists of the blind approaches. This seems reasonable, since it makes sense to rely more on prior knowledge when the data are very noisy.

5.2 Exploratory Source Separation in Rhythmic MEG Data

In biomedical research it is usual to design detailed experimental frameworks to examine interesting phenomena. Hence it offers a nice field of application for both blind and specialised DSS schemes. In the following, we test the developed algorithms in signal analysis of magnetoencephalograms (MEG, Hämäläinen et al., 1993). MEG is a completely non-invasive brain imaging technique measuring the magnetic fields on scalp caused by synchronous activity in the cortex.

Since the early EEG and MEG recordings, cortical electromagnetic rhythms have played an important role in clinical research, *e.g.*, in detection of various brain disorders, and in studies of development and aging. It is believed that the spontaneous rhythms, in different parts of the brain, form a kind of resting state that allows for quicker responses to stimuli by those specific areas. For example deprivation of visual stimuli by closing one's eyes induces so-called α-rhythm on the visual cortex, characterised by a strong 8–13 Hz frequency component. For a more comprehensive discussion regarding EEG and MEG, and their spontaneous rhythms, see the works by Niedermeyer and Lopes da Silva (1993) and Hämäläinen et al. (1993).

In this paper, we examine an MEG experiment where the subject is asked to relax by closing her eyes (producing α-rhythm). There is also a control state where the subject has her eyes open. The data has been sampled with $f_s = 200$ Hz, and there are $T = 65536$ time samples giving total of more than 300 seconds of measurement. The magnetic fields are measured using a 122-channel MEG device. Some source separation results of this data have been reported by Särelä et al. (2001). Prior to any analysis, the data are high-pass filtered with cut-off frequency of 1 Hz, to get rid of the dominating very low frequencies.

5.2.1 DENOISING IN RHYTHMIC MEG

Examination of the average spectrogram in Fig. 8a reveals clear structures indicating the existence of several, presumably distinct, phenomena. The burst-like activity around 10 Hz and the steady activity at 50 Hz dominate the data, but there seem to be some weaker phenomena as well, *e.g.*, on frequencies higher than 50 Hz. To amplify these, we not only sphere the data spatially but temporally as well. This temporal decorrelation actually makes the separation harder but finding the weaker phenomena easier. The normalised and filtered spectrogram is shown in Fig. 8b.

The spectrogram data seems well suited for demonstrating the exploratory-data-analysis use of DSS. As some of the sources seem to have quite steady frequency content in time, along with others changing in time, we used two different time-frequency analyses as described in Sec. 4.1.3 with lengths of the spectra $T_f = 1$ and $T_f = 256$. The first spectrogram is then actually the original frequency-normalised and filtered data with time information only.

We apply the several noise-reduction principles based on the estimated variance of the signal and the noise discussed in Sec. 4.2. Specifically, the power spectrogram of the source estimate is smoothed over time and frequency using 2-D convolution with Gaussian windows. The standard deviations of the Gaussian windows were $\sigma_t = 8/\pi$ and $\sigma_f = 8/\pi$. After this, the instantaneous estimate of the source variance is found using Eq. (44). Then we get the denoised source estimate using Eq. (45) together with the spectral shift (47). Initially we have set $\mu = 1.3$. This is then decreased by 0.1 every time DSS has converged, until $\mu < 1$ is reached. Finally, the new projection vector is calculated using the stabilised version (28), (29) with the 179-rule in order to ensure convergence.

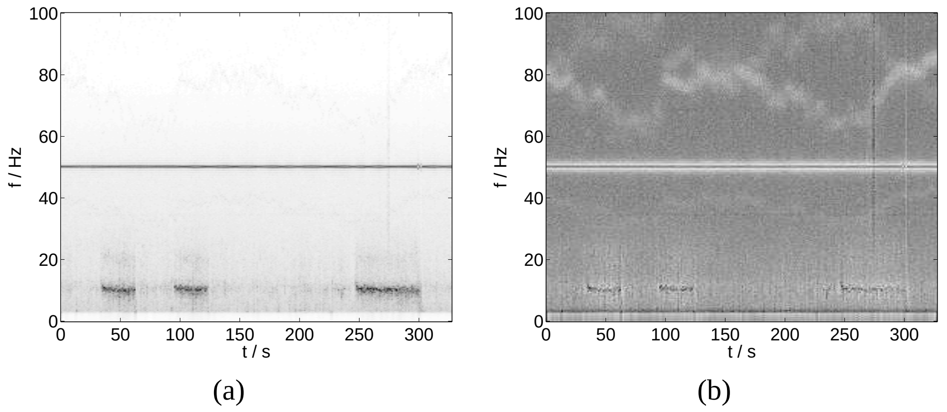


Figure 8: *(a) Averaged spectrogram of all 122 MEG channels. (b) Frequency normalised spectrogram.*

5.2.2 SEPARATION RESULTS

The separated signals, depicted in Fig. 9, include several interesting sources. Due to poor contrast in Fig. 9, we show enhanced and smoothed spectrograms of selected interesting, but low contrast, components (1a, 1b, 1c and 4c) in Fig. 10. There exist several sources with α-activity (1a, 1d and 2b for example). The second and fifth source are clearly related to the power-line. The third source depicts an interesting signal caused probably by some anomaly in either the measuring device itself or its physical surroundings. In source 4c, there is another, presumably artefactual source, composed of at least two steady frequencies around 70 Hz.

The DSS approach described above seems to be reliable and fast: the temporal decorrelation of the data enabled the finding of very weak sources and yet we found several clear α-sources as well. Valpola and Särelä (2004) have further studied the convergence speed, reliability and stability of DSS with various speedup methods, such as the spectral shift used in FastICA. Convergence speed exceeding standard FastICA by 50 % was reported.

Though quite a clear separation of the sources was achieved, some cross-talk between the signals remains. Better SNR and less talk would probably be achieved by tuning the denoising to the characteristics of each different signal group. In the next section, we show that with specific knowledge it is possible to find even very weak phenomena in MEG data using DSS.

5.3 Adaptive Extraction of the Cardiac Subspace in MEG

Cardiac activity causes magnetic fields as well. Sometimes these are strongly reflected in MEG and can pose a serious problem for the signal analysis of the neural phenomena of interest. In this data, however, the cardiac signals are not visible to the naked eye. Thus, we want to demonstrate the capability of DSS to extract some very weak cardiac signals, using detailed prior information in an adaptive manner.

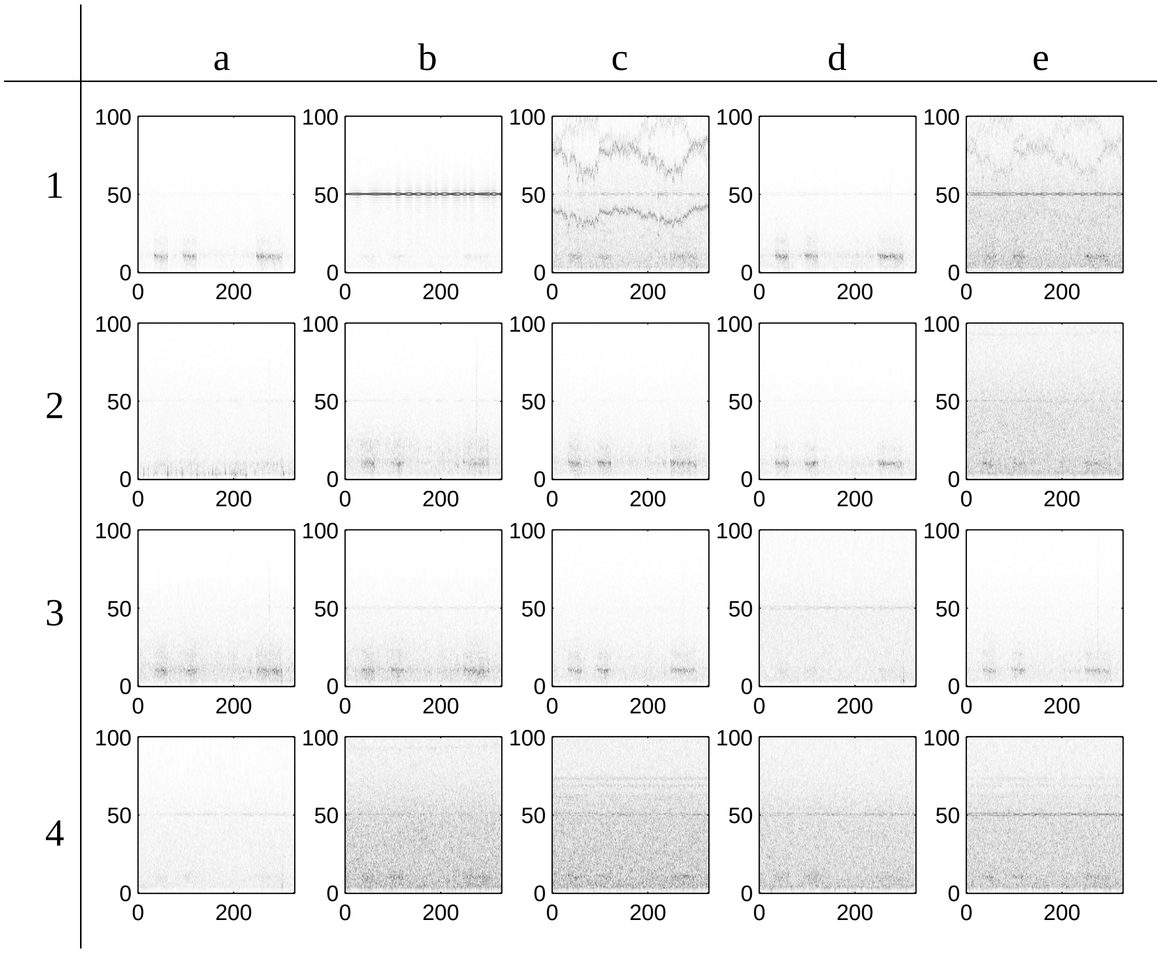


Figure 9: *Spectrograms of the extracted components (comps. 1a–1e on the topmost row). Time and frequency axes as in Fig. 8.*

5.3.1 DENOISING OF THE CARDIAC SUBSPACE

A clear QRS complex, which is the main electromagnetic pulse in the cardiac cycle, can be extracted from the MEG data using standard BSS methods, such as kurtosis- or tanh-based denoising. Due to its sparse nature, this QRS signal can be used to estimate the places of the heart beats. With the places known, we can guide further search using the averaging DSS, as described in Sec. 4.1. Every now and then, we re-estimate the QRS onsets needed for the averaging DSS.

When the estimation of the QRS locations has been stabilised, a subspace that is composed of signals having activity phase-locked to the QRS complexes can be extracted.

5.3.2 SEPARATION RESULTS

Figure 11 depicts five signals averaged around the QRS complexes, found using the procedure above.[6] The first signal presents a very clear QRS complex, whereas the second one contains the

6. For clarity, two identical cycles of averaged heart beats are always shown.

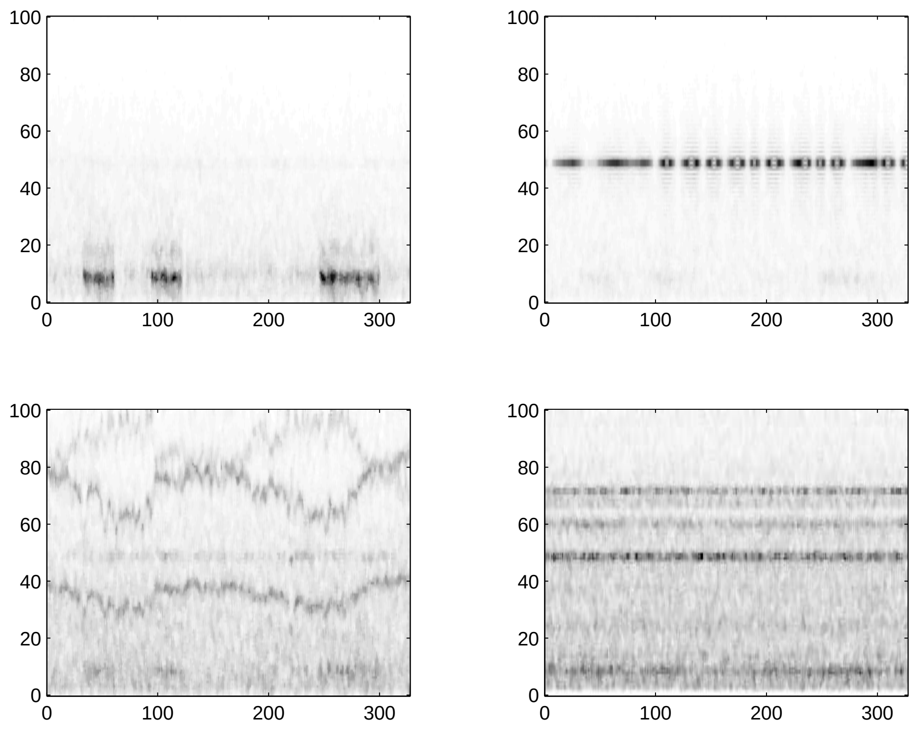


Figure 10: *Enhanced and smoothed spectrograms of the selected components (correspond to sources 1a, 1b, 1c and 4c in Fig. 9). Time and frequency axes as in Fig. 8.*

small P and the T waves. An interesting phenomenon is found in the third signal: there is a clear peak at the QRS onset, which is followed by a slow attenuation phase. We presume that it originates from some kind of relaxing state.

Two other heart-related signals were also extracted. They both show a clear deflection during the QRS complex, but have as well significant activity elsewhere. These two signals might present a case of overfitting, which was contemplated in Sec. 3.4. To test this hypothesis, we performed DSS using the same procedure and the same denoising function, but for time-reversed data. As the estimated QRS onsets will then be misaligned, the resulting signals should be pure overfits. The results are shown in Fig. 12. The eigenvalues corresponding to the QRS complex and the second signal having the P and T waves are approximately 10 times higher than the principal eigenvalue of the reversed data. Thus they clearly exhibit some real structure in the data, as already expected. The eigenvalues corresponding to the last three signals are comparable to the principal eigenvalue of the reversed data, the two largest being somewhat greater. It is reasonable to expect that all three carry some real structure as there is a nonzero correlation between the first two signals having the main cardiac responses and the overfitted component corresponding to the largest eigenvalue from the reversed data. In the three other signals, there probably occurs some overfitting as well, since the signals have similar structures to the last two signals of the actual subspace experiment shown in Fig. 11.

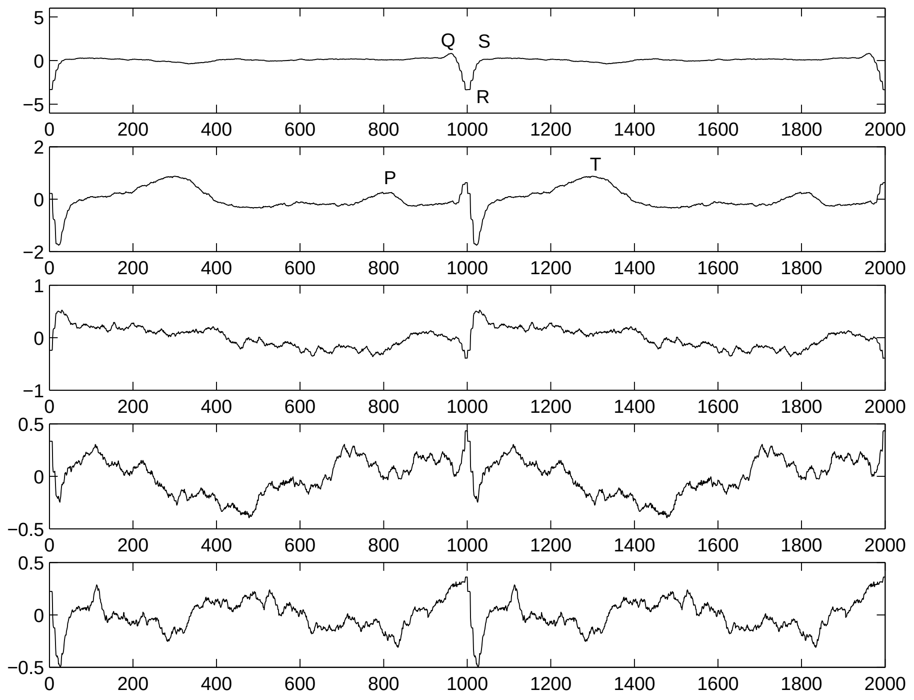


Figure 11: *Averages of three heart-related signals and presumably two overfitting results.*

It is worth noticing that even the strongest component of the cardiac subspace is rather weakly present in the original data. The other components of the subspace are hardly detectable without advanced methods beyond blind source separation. This clearly demonstrates the power that DSS can provide for an exploring researcher.

5.4 Signal Recovery in CDMA

Mobile systems constitute another important signal processing application area, in addition to biomedical signal processing. There are several ways to allow multiple users to use the same communication channel, one being a modulation scheme called code-division-multiple-access (CDMA, Viterbi, 1995). In this section we consider bit-stream recovery in a simplified simulation of a CDMA network.

In CDMA, each user has a unique signature quasi-orthogonal to the signatures of the other users. The user codes each complex bit[7] which he sends using this signature. This coded bit stream is transmitted through the communication channel, where it is mixed with the signals of the other transmitters. The mixture is corrupted by some noise as well, due to multi-path propagation, Doppler shifts, interfering signals, etc.

To recover the sent bit stream, the receiver decodes the signal with the known signature. Ideally then, the result would be ones and zeros repeated the number of times corresponding to the signa-

7. Here a scheme called QAM is used: two bits are packed into one complex bit by making a 90° phase shift in the other bit.

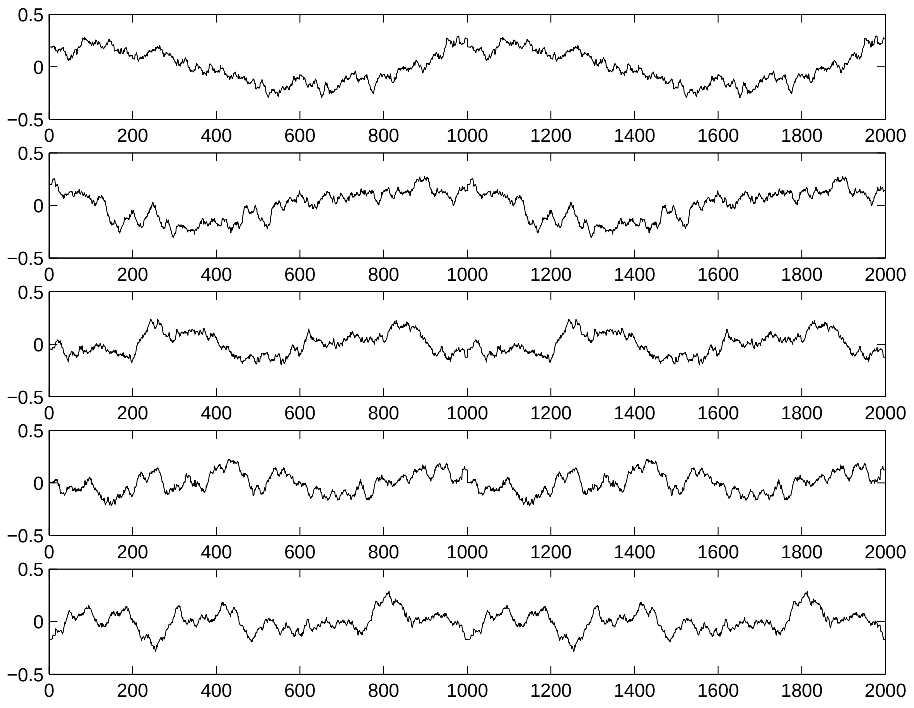


Figure 12: *Averages of five signals from the cardiac control experiment, showing clear overfits.*

ture length. In practice, noise and other interfering signals cause variation and the bits are usually extracted by majority voting.

If there are multiple paths through which a particular bit stream is sent to the receiver or the transmitter and receiver have multiple antennas, the so-called RAKE procedure can be used: The path coefficients are estimated based on the so-called pilot bit streams that are fixed known bit streams and sent frequently by the transmitter. Different bit streams are then summed together before the majority voting. In RAKE-ICA (Raju and Ristaniemi, 2002), ICA is used to blindly separate the desired signal from the interference of other users and noise. This yields better results in the majority voting.

5.4.1 DENOISING OF CDMA SIGNALS

We know that the original bit stream should consist of repeated coding signatures convoluted by the original complex bits. First the bit stream is decoded using a standard detection algorithm. The denoised signal is then the recoding of the decoded bit stream.

This DSS approach is nonlinear. If the original bit-stream estimate is very inaccurate, *e.g.*, due to serious interference of other users or external noise, the nonlinear approach might get stuck in a deficient local minimum. To prevent this, we first initialise by running a simpler, linear DSS. There we only exploit the fact that the signal should consist of repetitions of the signature multiplied by a complex number. The nonlinearity of the denoising is gradually increased in the first iterations.

5.4.2 SEPARATION RESULTS

We sent 100 blocks of 200 complex bits. The sent bits were mixed using the streams of 15 other users. For simplicity we set all the path delays to zero. The signal-to-noise-ratio (SNR) varied from -10 to 15 dB. The length of the spreading signature was 31. The mixtures were measured using three antennas. We did not consider multi-path propagation.

Figure 13 sums up the results of the CDMA experiments. The comparison to the RAKE algorithm shows that DSS performs better in all situations except in the highest SNR, where RAKE is slightly better. Note that RAKE needs the pilot bits to estimate the mixing while our implementation of DSS was able to do without them. The better performance of DSS for low SNR is explained by the fact that DSS actively cancels disturbing signals while RAKE ignores them.

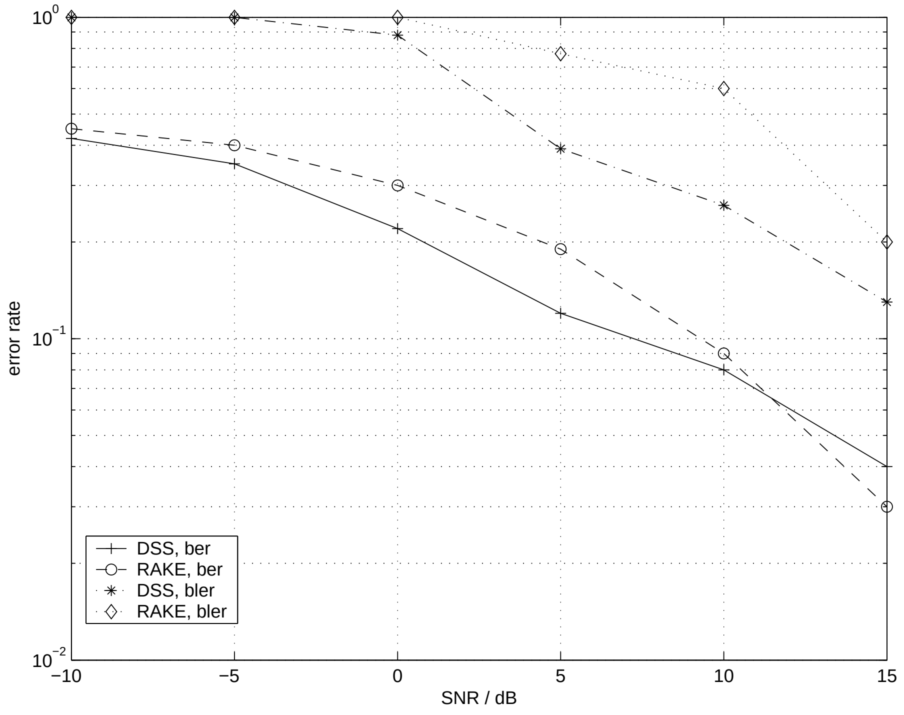


Figure 13: *Bit- and block-error rates for different SNRs for DSS and RAKE.*

CDMA bit streams consist of known headers that are necessary for standard CDMA techniques to estimate several properties of the transmission channel. The DSS framework is able to use the redundancy of the payload signal, and therefore less pilot sequences are needed. In addition, bits defined by the actual data such as error-correcting or check bits allow an even better denoising of the desired stream. Furthermore, it is possible to take multi-path propagation into account using several delayed versions of the received signal. This should then result in a kind of averaging denoising when a proper delay is used analogous to the multi-resolution spectrogram DSS described in Sec. 4.1.3. In the case of moving transmitters and receivers, DSS may exploit the Doppler effect.

6. Discussion

In this paper, we developed several DSS algorithms. Moreover, DSS offers a promising framework for developing additional extensions. In this section, we first summarise the extensions that have already been mentioned in previous sections and then discuss some auxiliary extensions.

We discussed an online learning strategy in Sec. 4.3, where we noted that asymmetric online denoising may fail to converge within a 2-D subspace. However, symmetric denoising procedures performing similar functions may easily be generated.

We also noted that the masking based on the instantaneous variance in Sec. 4.2 may have problems in separating the actual sources, though it effectively separates the noise subspace from the signal subspace. We proposed a simple modification to magnify small differences between the variance estimates of different sources. Furthermore, we noted that a better founded alternative is to consider explicitly the leakage of variance between the signals. Then the variances of the signals can be decorrelated using similar techniques to those suggested by Schwartz and Simoncelli (2001). This idea has been pursued further in the DSS framework (Valpola and Särelä, 2004), making the variance-based masking a very powerful approach to source separation. Furthermore, the variance-based mask saturates on large values. This reduces the tendency to suffer from outliers. However, data values that differ utterly from other data points probably carry no interesting information at all. Even more robustness could then be achieved if the mask would start to decrease on large enough values.

In this paper, we usually considered the sources to have a one-dimensional structure, which is used to implement the denoising. We already applied successfully two-dimensional denoising techniques for the spectrograms. Furthermore, it was mentioned in Sec. 2 that the index t of different samples $\mathbf{s}(t)$ might refer as well to space as to time. In space it becomes natural to apply filtering in 2D or even in 3D. For example, the astrophysical ICA (Funaro et al., 2003) would clearly benefit from multi-dimensional filtering.

Source separation is not the only application of ICA-like algorithms. Another, important field of application is feature extraction. ICA has been used for example in the extraction of features from natural images, similar to those that are found in the primary visual cortex (Olshausen and Field, 1996). It is reasonable to consider DSS extensions that have been suggested in the field of feature extraction as well. For instance, until now we have only considered the extraction of multiple components by forcing the projections to be orthogonal. However, nonorthogonal projections resulting from over-complete representations provide some clear advantages, especially in sparse codes (Földiák, 1990), and may be found useful in the DSS framework as well.

Throughout this paper, we have considered linear mapping from the sources to the observations but nonlinear mappings can be used, too. One such approach is slow feature analysis (SFA, Wiskott and Sejnowski, 2002) where the observations are first expanded nonlinearly and sphered. The expanded data are then high-pass filtered and projections minimising the variance are estimated. Due to the nonlinear expansion, it is possible to stack several layers of SFA on top of each others to extract higher-level slowly changing features, resulting in hierarchical SFA.

Interestingly, SFA is directly related to DSS. Instead of minimising the variance after high-pass filtering as in SFA, the same result may be obtained by maximising the variance after low-pass filtering. SFA is thus equivalent to DSS with nonlinear data expansion and low-pass filtering as denoising. This is similar to earlier proposals, *e.g.*, by Földiák (1991).

There are several possibilities for the nonlinear feature expansion in hierarchical DSS. For instance kernel PCA (Schölkopf et al., 1998), sparse coding or liquid state machines (Maass et al., 2002) can be used.

The hierarchical DSS can be used in a fully supervised setting by fixing the activations of the topmost layer to target outputs. Supervised learning often suffers from slow learning in deep hierarchies because the way information is represented gradually changes in the hierarchy. It is therefore difficult to use the information about the target output for learning the layers close to the inputs. The benefit of hierarchical DSS is that learning on lower levels is not dependent only on the information propagated from the target output because the context includes lateral or delayed information from the inputs. In this approach, the mode of learning shifts smoothly from mostly unsupervised learning to mostly supervised learning from the input layer towards the output layer. A similar mixture of supervised and unsupervised learning has been suggested by Körding and König (2001).

7. Conclusion

The work in linear source separation has concentrated on blind approaches to fix the rotational ambiguity left by the factor analysis model. Usually, however, there is additional information available to find the rotation either more efficiently or more accurately. In this paper we developed an algorithmic framework called denoising source separation (DSS). We showed that denoising can be used for source separation and that the results are often better than with blind approaches. The better the denoising is, the better the results are. Furthermore, many blind source separation techniques can be interpreted as DSS algorithms using very general denoising principles. In particular, we showed that FastICA is a special case of DSS which also implies that DSS can be computationally very efficient.

The main benefit of the DSS framework is that it allows for easy development of new source separation algorithms which are optimised for the specific problem at hand. There is a wide literature on signal denoising to choose from and in some cases denoising would be used for post-processing in any case. All the tools needed for DSS are then readily available.

We have launched an open-source MATLAB package for implementing DSS algorithms (DSS, 2004). It contains the denoising functions and speedup method presented here. But more importantly, the modular coding style makes it easy to tune the denoising functions to better suit the separation problems at hand and even to build in completely new denoising functions to achieve better performance.

In the experimental section, we demonstrated DSS in various source separation tasks. We showed how denoising can be adapted to the observed characteristics of signals extracted with denoising based on vague knowledge. From MEG signals, we were able to extract very accurately subspaces such as the α-subspace or the very weak components of the cardiac subspace. DSS also proved to be able to recover CDMA signals better than the standard RAKE technique under poor SNR.

Finally, we discussed potential extensions of DSS. It appears that DSS offers a sound basis for developing hierarchical, nonlinear feature extraction methods and the connections to cortical models of attention and perception suggest a promising starting point for future work.

Acknowledgments

This work is funded by the Academy of Finland, under the project New information processing principles, and by European Commission, under the project ADAPT (IST-2001-37137).

We would like to show gratitude to Dr. Ricardo Vigário for the fruitful discussions concerning the method in general as well as the MEG experiments in detail and Dr. Aapo Hyvärinen for the method itself and its connections to ICA. We would like to thank as well Mr. Karthikesh Raju for his suggestions and help concerning the CDMA experiments and Mr. Kosti Rytkönen who is the main author of the DSS MATLAB package. Our sincere thanks are also to the editor and the anonymous referees for their thorough inspection of the article. Finally, we would like to thank prof. Erkki Oja for his comments on the draft version of this manuscript.

References

B. D. Anderson and J. B. Moore. *Optimal filtering*. Prentice-Hall, 1979.

H. Attias. Independent factor analysis. *Neural Computation*, 11(4):803–851, 1999.

A. Belouchrani, K. Abed Meraim, J.-F. Cardoso, and E. Moulines. A blind source separation technique based on second order statistics. *IEEE Transactions on Signal Processing*, 45(2):434–44, 1997.

O. Bermond and J.-F. Cardoso. Approximate likelihood for noisy mixtures. In *Proceedings of the First International Workshop on Independent Component Analysis and Signal Separation (ICA'99)*, pages 325–330, Aussois, France, Jan. 11-15, 1999.

J.-F. Cardoso. High-order contrasts for independent component analysis. *Neural Computation*, 11 (1):157 – 192, 1999.

K.-L. Chan, T.-W. Lee, and T. J. Sejnowski. Variational Bayesian learning of ICA with missing data. *Neural Computation*, 15 (8):1991–2011, 2003.

R. A. Choudrey and S. J. Roberts. Flexible Bayesian independent component analysis for blind source separation. In *Proceedings of the Third International Conference on Independent Component Analysis and Signal Separation (ICA2001)*, pages 90–95, San Diego, USA, 2001.

P. A. d. F. R. Højen-Sørensen, O. Winther, and L. K. Hansen. Mean-field approaches to independent component analysis. *Neural Computation*, 14(4):889–918, 2002.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B (Methodological)*, 39(1):1–38, 1977.

D. L. Donoho, I. M. Johnstone, G. Kerkyacharian, and D. Picard. Wavelet shrinkage: asymptopia? *Journal of the Royal Statistical Society, Series B (Methodological)*, 57:301–337, 1995.

S. C. Douglas and A. Cichocki. Neural networks for blind decorrelation of signals. *IEEE Transactions on Signal Processing*, 45(11):2829 – 2842, 1997.

DSS. The DSS MATLAB package. 2004. Available at `http://www.cis.hut.fi/projects/dss/`.

FastICA. The FastICA MATLAB package. 1998. Available at `http://www.cis.hut.fi/projects/ica/fastica/`.

P. Földiák. Forming sparse representations by local anti-hebbian learning. *Biological Cybernetics*, 64:165 – 170, 1990.

P. Földiák. Learning invariance from transformation sequences. *Neural Computation*, 3:194–200, 1991.

M. Funaro, E. Oja, and H. Valpola. Independent component analysis for artefact separation in astrophysical images. *Neural Networks*, 16(3 – 4):469 – 478, 2003.

M. S. Gazzaniga, editor. *The New Cognitive Neurosciences*. A Bradford book/MIT Press, 2nd edition, 2000.

X. Giannakopoulos, J. Karhunen, and E. Oja. Experimental comparison of neural algorithms for independent component analysis and blind separation. *International Journal of Neural Systems*, 9(2):651–656, 1999.

M. Hämäläinen, R. Hari, R. Ilmoniemi, J. Knuutila, and O. V. Lounasmaa. Magnetoencephalography—theory, instrumentation, and applications to noninvasive studies of the working human brain. *Reviews of Modern Physics*, 65:413–497, 1993.

J. Himberg and A. Hyvärinen. Icasso: software for investigating the reliability of ica estimates by clustering and visualization. In *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing (NNSP'2003)*, pages 259–268, Toulouse, France, 2003.

A. Hyvärinen. New approximations of differential entropy for independent component analysis and projection pursuit. In *Advances in Neural Information Processing 10 (NIPS'98)*, pages 273–279. MIT Press, 1998.

A. Hyvärinen. Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks*, 10(3):626–634, 1999.

A. Hyvärinen, P. Hoyer, and M. Inki. Topographic independent component analysis. *Neural Computation*, 13(7):1525–1558, 2001a.

A. Hyvärinen, J. Karhunen, and E. Oja. *Independent component analysis*. Wiley, 2001b.

JADE. The JADE MATLAB package. 1999. Available at `http://www.tsi.enst.fr/icacentral/Algos/cardoso/`.

K. H. Knuth. Bayesian source separation and localization. In A. Mohammad-Djafari, editor, *SPIE'98 Proceedings: Bayesian Inference for Inverse Problems*, pages 147–158, San Diego, USA, 1998.

P. Kuosmanen and J. T. Astola. *Fundamentals of nonlinear digital filtering*. CRC press, 1997.

K. P. Körding and P. König. Neurons with two sites of synaptic integration learn invariant representations. *Neural Computation*, 13:2823 – 2849, 2001.

H. Lappalainen. Ensemble learning for independent component analysis. In *Proceedings of the First International Workshop on Independent Component Analysis and Signal Separation, (ICA'99)*, pages 7–12, Aussois, France, 1999.

D. G. Luenberger. *Optimization by Vector Space Methods*. John Wiley & Sons, 1969.

W. Maass, T. Natschläger, and H. Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531 – 2560, 2002.

F. Meinecke, A. Ziehe, M. Kawanabe, and K.-R. Müller. A resampling approach to estimate the stability of one- and multidimensional independent components. *IEEE Transactions on Biomedical Engineering*, 49(12):1514 – 1525, 2002.

J. Miskin and David J. C. MacKay. Ensemble learning for blind source separation. In S. Roberts and R. Everson, editors, *Independent Component Analysis: Principles and Practice*, pages 209–233. Cambridge University Press, 2001.

J. Molgedey and H. G. Schuster. Separation of a mixture of independent signals using time delayed correlations. *Physical Review Letters*, 72:541–557, 1994.

E. Niedermeyer and F. Lopes da Silva, editors. *Electroencephalography. Basic principles, clinical applications, and related fields*. Baltimore: Williams & Wilkins, 1993.

E. Oja. Principal components, minor components, and linear neural networks. *Neural Networks*, 5: 927–935, 1992.

B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, 1996.

D. B. Percival and W. T. Walden. *Spectral Analysis for Physical Applications: Multitaper and Conventional Univariate Techniques*. Cambridge University Press, Cambridge, UK, 1993.

D.-T. Pham and J.-F. Cardoso. Blind separation of instantaneous mixtures of non stationary sources. *IEEE Transactions on Signal Processing*, 49:1837–1848, 2001.

K. Raju and T. Ristaniemi. ICA-RAKE switching for jammer cancellation in DS-CDMA array systems. In *Proceedings of the IEEE International Symposium on Spread Spectrum Techniques and Applications (ISSSTA)*, pages 638 – 642, Prague, September 2002.

R. M. Rangayyan. *Biomedical signal analysis: A case-study approach*. IEEE Press Series in Biomedical Engineering, 2002.

J. Särelä, H. Valpola, R. Vigário, and E. Oja. Dynamical factor analysis of rhythmic magnetoencephalographic activity. In *Proceedings of the Third International Conference on Independent Component Analysis and Signal Separation (ICA2001)*, pages 451–456, San Diego, USA, 2001.

J. Särelä and R. Vigário. Overlearning in marginal distribution-based ICA: analysis and solutions. *Journal of Machine Learning Research*, 4 (Dec):1447–1469, 2003.

B. Schölkopf, S. Mika, A. Smola, Gunnar Rätsch, and K.-R. Müller. Kernel PCA pattern reconstruction via approximate pre-images. In *Proceedings of the 8th International Conference on Artificial Neural Networks (ICANN'98)*, pages 147 – 152, Skövde, 1998.

O. Schwartz and E. P. Simoncelli. Natural signal statistics and sensory gain control. *Nature Neuroscience*, 4(8):819 – 825, 2001.

L. Tong, V. Soo, R. Liu, and Y. Huang. Indeterminacy and identifiability of blind identification. *IEEE Transactions on Circuits and Systems*, 38:499–509, 1991.

H. Valpola and P. Pajunen. Fast algorithms for Bayesian independent component analysis. In *Proceedings of the Second International Workshop on Independent Component Analysis and Signal Separation (ICA2000)*, pages 233–237, Helsinki, Finland, 2000.

H. Valpola, T. Raiko, and J. Karhunen. Building blocks for hierarchical latent variable models. In *Proceedings of the Third International Conference on Independent Component Analysis and Signal Separation (ICA2001)*, pages 710–715, San Diego, USA, 2001.

H. Valpola and J. Särelä. Accurate, fast and stable denoising source separation algorithms. In *Proceedings of the Fifth International Conference on Independent Component Analysis and Signal Separation (ICA2004)*, pages 64 – 71, Granada, Spain, 2004.

M. Vetterli and J. Kovacevic. *Wavelets and subband coding*. Prentice-Hall, 1995.

R. Vigário, J. Särelä, V. Jousmäki, M. Hämäläinen, and E. Oja. Independent component approach to the analysis of EEG and MEG recordings. *IEEE Transactions on Biomedical Engineering*, 47 (5):589–593, 2000.

V. Vigneron, A. Paraschiv-Ionescu, A. Azancot, O. Sibony, and C. Jutten. Fetal electrocardiogram extraction based on non-stationary ICA and wavelet denoising. In *Proceedings of the Seventh International Symposium on Signal Processing and its Applications (ISSPA2003)*, Paris, France, July 2003.

A. J. Viterbi. *CDMA : Principles of Spread Spectrum Communication*. Wireless Info Networks Series. Addison-Wesley, 1995.

J. H. Wilkinson. *The algebraic eigenvalue problem*. Monographs on numerical analysis. Clarendon press, London, 1965.

L. Wiskott and T. Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural Computation*, 14:715 – 770, 2002.

A. Ziehe and K.-R. Müller. TDSEP — an effective algorithm for blind separation using time structure. In *Proceedings of the 8th International Conference on Artificial Neural Networks (ICANN'98)*, pages 675–680, Skövde, Sweden, 1998.

Journal of Machine Learning Research 6 (2005) 273–306 Submitted 10/02; Revised 11/04; Published 3/05

Tutorial on Practical Prediction Theory for Classification

John Langford JL@HUNCH.NET
Toyota Technological Institute at Chicago
1427 East 60th Street
Chicago, IL 60637, USA

Editor: Robert Schapire

Abstract

We discuss basic prediction theory and its impact on classification success evaluation, implications for learning algorithm design, and uses in learning algorithm execution. This tutorial is meant to be a comprehensive compilation of results which are both theoretically rigorous and quantitatively useful.

There are two important implications of the results presented here. The first is that common practices for reporting results in classification should change to use the test set bound. The second is that train set bounds can sometimes be used to directly motivate learning algorithms.

Keywords: sample complexity bounds, classification, quantitative bounds

1. Introduction

Classifiers are functions which partition a set into two classes (for example, the set of rainy days and the set of sunny days). Classifiers appear to be the most simple nontrivial decision making element so their study often has implications for other learning systems. Classifiers are sufficiently complex that many phenomena observed in machine learning (theoretically or experimentally) can be observed in the classification setting. Yet, classifiers are simple enough to make their analysis easy to understand. This combination of sufficient yet minimal complexity for capturing phenomena makes the study of classifiers especially fruitful.

The goal of this paper is an introduction to the theory of prediction for classification. Here "prediction theory" means statements about the future error rate of learned classifiers. A typical statement has the form, "With probability $1-\delta$ over an i.i.d. draw of some sample, the expected future error rate of a classifier is bounded by $f(\delta, \text{error rate on sample})$". These statements are confidence intervals on the error rate of a learned classifier. Many of these results have been presented elsewhere, although the style, tightness, and generality of the presentation are often new here (and particularly oriented towards practical use). The focus of this tutorial is on those results which are both theoretically sound and practically useful.

There are several important aspects of learning which the theory here casts light on. Perhaps the most important of these is the problem of performance reporting for classifiers. Many people use some form of empirical variance to estimate upper and lower bounds. This is an error-prone practice, and the test set bound in Section 3 implies a better method by nearly any metric. Hopefully, this will become common practice.

After discussing the test set bound we cover the Occam's Razor bound, the simplest train set bound, which explains (and quantifies) the common phenomenon of overfitting. We also prove that the Occam's Razor bound cannot be improved without incorporating extra information and apply the bound to decision trees.

Next, we discuss two train set bounds, the PAC-Bayes bound and the sample compression bound, which have proved to give practical results for more general classifiers, such as support vector machines and neural networks. All of the results here should be easily approachable and understandable. The proofs are simple, and examples are given. Pointers to related work are also given.

There are some caveats about the scope of this document.

1. All of the results presented here fall in the realm of classical statistics. In particular, all randomizations are over draws of the data, and our results have the form of confidence intervals.

2. This tutorial is *not* comprehensive for prediction theory in general (which would be extremely difficult due to the scope of the subject). We only focus on those results yielding quantifiably interesting performance.

3. In particular, other nonquantitative uses of bounds (such as providing indirect motivations for learning algorithms via constant fitting) do exist. We do not focus on those uses here.

The layout of this document is as follows.

- Section 2 presents the formal model.
- Section 3 presents the test set bound.
- Section 4 presents the Occam's Razor bound.
- Section 5 presents the PAC-Bayes bound.
- Section 6 presents the sample compression bound.

The formal model and test set bound must be understood in order to appreciate all later results. There is no particular dependency between the various train set bounds we present.

2. Formal Model

There are many somewhat arbitrary choices of learning model. The one we use can (at best) be motivated by its simplicity. Other models such as the online learning model (Kivinen and Warmuth, 1997), PAC learning (Valiant, 1984), and the uniform convergence model (Vapnik and Chervonenkis, 1971) differ in formulation, generality, and in the scope of addressable questions. The strongest motivation for studying the prediction theory model here is simplicity and corresponding generality of results. The appendix discusses the connections between various models.

2.1 Basic Quantities

We are concerned with a learning model in which examples of (input, output) pairs come independently from some unknown distribution (similar to Shawe-Taylor et al., 1998, and many other papers). The goal is to find a function capable of predicting the output given the input. There are several mathematical objects we work with.

Object	Description
X	The (arbitrary) space of the input to a classifier
$Y = \{-1, 1\}$	The output of a classification.
D	An (unknown) distribution over $X \times Y$
S	A sequence of examples drawn independently from D.
m	$= \lvert S \rvert$ the number of examples
c	A function mapping X to Y

Table 1: Mathematical objects in the considered model.

There are several distinctions between this model and other (perhaps more familiar) models. There is no mention of a classifier space, because the results do not depend upon a classifier space. Also, the notion of a distribution on $X \times Y$ is strictly more general than the "target concept" model which assumes that there exists some function $f : X \to Y$ used to generate the label (Valiant, 1984). In particular we can model noisy learning problems which do not have a particular Y value for each X value. This generalization is essentially "free" in the sense that it does not add to the complexity of presenting the results.

It is worth noting that the *only* unverifiable assumption we make is that examples are drawn independently from D. The strength of all the results which follow rests upon the correctness of this assumption.

Sometimes, we decorate these objects with labels like S_{train} (a train set[1]) or S_{test} (a test set). These decorations should always be clear.

Example 1 *Weather prediction: Will it rain today or not? In this case X = barometric pressure, observations of cloud cover or other sensory input and $Y = 0$ if the prediction is "no rain" and 1 otherwise. The distribution D is over sensory inputs and outcomes. The sample set S, might consist of $m = 100$ (observation, outcome) pairs such as (pressure low, cloudy, rain), (pressure high, cloudy, not rain), etc. A classifier, c, is any function which predicts "rain" or "not rain" based upon the observation.*

Note that the independence assumption here is not perfectly satisfied although it seems to be a reasonable approximation for well-separated days. In any application of this theory, it must be carefully judged whether the independence assumption holds or not.

2.2 Derived Quantities

There are several derived quantities which the results are stated in terms of.

Definition 2.1 *(True Error) The true error c_D of a classifier c is defined as the probability that the classifier errs:*

$$c_D \equiv \Pr_{(x,y)\sim D}(c(x) \neq y)$$

under draws from the distribution D.

1. Throughout this tutorial we use the word 'set' when 'sequence' is what is actually meant. This usage pattern is historical.

The true error is sometimes called the "generalization error". Unfortunately, the true error is not an observable quantity in our model because the distribution D is unknown. However, there is a related quantity which is observable.

Definition 2.2 *(Empirical Error) Given a sample set S, the* empirical error*,* $\hat{c}_S$ *is the observed number of errors:*

$$\hat{c}_S \equiv m \Pr_{(x,y)\sim S}(c(x) \neq y) = \sum_{i=1}^{m} I(c(x_i) \neq y_i)$$

where $I()$ *is a function which maps "true" to* 1 *and "false" to* 0*. Also,* $\Pr_{(x,y)\sim S}(\ldots)$ *is a probability taken with respect to the uniform distribution over the set of examples, S.*

The empirical error is sometimes called the "training error", "test error", or "observed error" depending on whether it is the error on a training set, test set, or a more general set.

Example 2 *(continued) The classifier c which always predicts "not rain" might have an empirical error of* 38 *out of* 100 *examples and an unknown true error rate (which might in fact be* 0.5*).*

2.3 Addressable Questions

Given the true error c_D of a classifier c we can precisely describe the distribution of success and failure on future examples drawn according to D. This quantity is derived from the unknown distribution D, so our effort is directed toward upper and lower bounding the value of c_D for a classifier c.

The variations in all of the bounds that we present are related to the method of choosing a classifier c. We cover two types of bounds:

1. Test: Use examples in a test set which were not used in picking c.

2. Train: Use examples for both choosing c and evaluating c.

These methods are addressed in the next two sections.

It is worth noting that one question that *cannot* be addressed in this model is "Can learning occur for my problem?" Extra assumptions (Valiant, 1984; Vapnik and Chervonenkis, 1971) are inherently necessary.

3. The Test Set Method

The simplest bound arises for the classical technique of using m fresh examples to evaluate a classifier. In a statistical setting, this can be viewed as computing a confidence interval for the binomial distribution as in (Clopper and Pearson, 1934). This section is organized into two subsections:

- Subsection 3.1 presents the basic upper bound on the true error rate, handy approximations, and a lower bound.

- Subsection 3.2 discusses the implications of the test set bound on error reporting practice. A better method for error reporting is applied to several datasets and the results are shown.

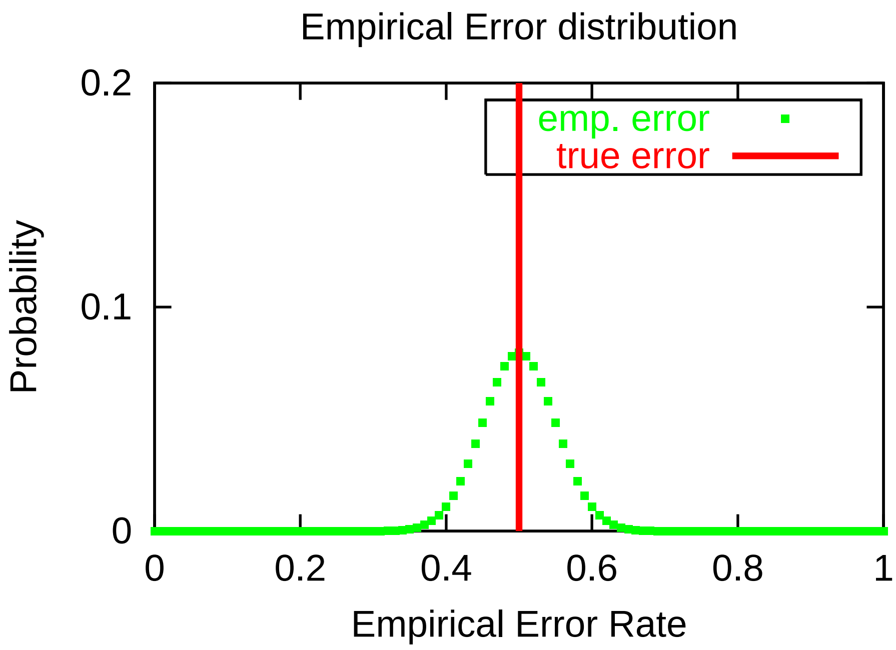


Figure 1: A depiction of the binomial distribution. The cumulative of the binomial is the area under the curve up to some point on the horizontal axis.

3.1 The Bound

Before stating the bound, we note a few basic observations which make the results less surprising. The principal observable quantity is the empirical error $\hat{c}_S$ of a classifier. What is the distribution of the empirical error for a fixed classifier? For each example, our independence assumption implies the probability that the classifier makes an error is given by the true error, c_D. This can be modeled by a biased coin flip: heads if you are right and tails if you are wrong.

What is the probability of observing k errors (heads) out of m examples (coin flips)? This is a very familiar distribution in statistics called the binomial and so it should not be surprising that the bounds presented here are fundamentally dependent upon the cumulative distribution of a binomial. For the following definition $\mathcal{B}(p)$ is the distribution of a Bernoulli coin flip.

Definition 3.1 *(Binomial Tail Distribution)*

$$Bin(m,k,c_D) \equiv \Pr_{Z_1,\ldots Z_m \sim \mathcal{B}(c_D)^m}\left(\sum_{i=1}^{m} Z_i \le k\right) = \sum_{j=0}^{k} \binom{m}{j} c_D^j (1-c_D)^{m-j}$$

equals the probability that m examples (coins) with error rate (bias) c_D produce k or fewer errors (heads).

A depiction of the binomial distribution is given in Figure 1.

For the learning problem, we always choose a bias of c_D and X_i =error or not on the ith example. With these definitions, we can interpret the binomial tail as the probability of an empirical error less than or equal to k.

Since we are interested in calculating a bound on the true error given a confidence δ, and an empirical error $\hat{c}_S$, it is handy to define the inversion of a binomial tail.

Definition 3.2 *(Binomial Tail Inversion)*

$$\overline{Bin}(m,k,\delta) \equiv \max_p \{p : \ Bin(m,k,p) \ge \delta\}$$

equals the largest true error such that the probability of observing k or more "heads" is at least δ.

For intuition's sake, the quantity $\overline{\mathrm{Bin}}(m,k,\delta)$ obeys the following inequalities (some of which we prove later).

1. $\overline{\mathrm{Bin}}(m,k,\delta) \le \frac{k}{m} + \sqrt{\frac{\ln\frac{1}{\delta}}{2m}}$
2. $\overline{\mathrm{Bin}}(m,k,\delta) \le \frac{k}{m} + \sqrt{\frac{2\frac{k}{m}\ln\frac{1}{\delta}}{m}} + \frac{2\ln\frac{1}{\delta}}{m}$
3. $\overline{\mathrm{Bin}}(m,0,\delta) \le \frac{\ln\frac{1}{\delta}}{m}$

With these definitions finished, the results are all very simple statements.

Theorem 3.3 *(Test Set Bound) For all D, for all classifiers c, for all $\delta \in (0,1]$*

$$\Pr_{S\sim D^m}\left(c_D \le \overline{Bin}(m,\hat{c}_S,\delta)\right) \ge 1-\delta.$$

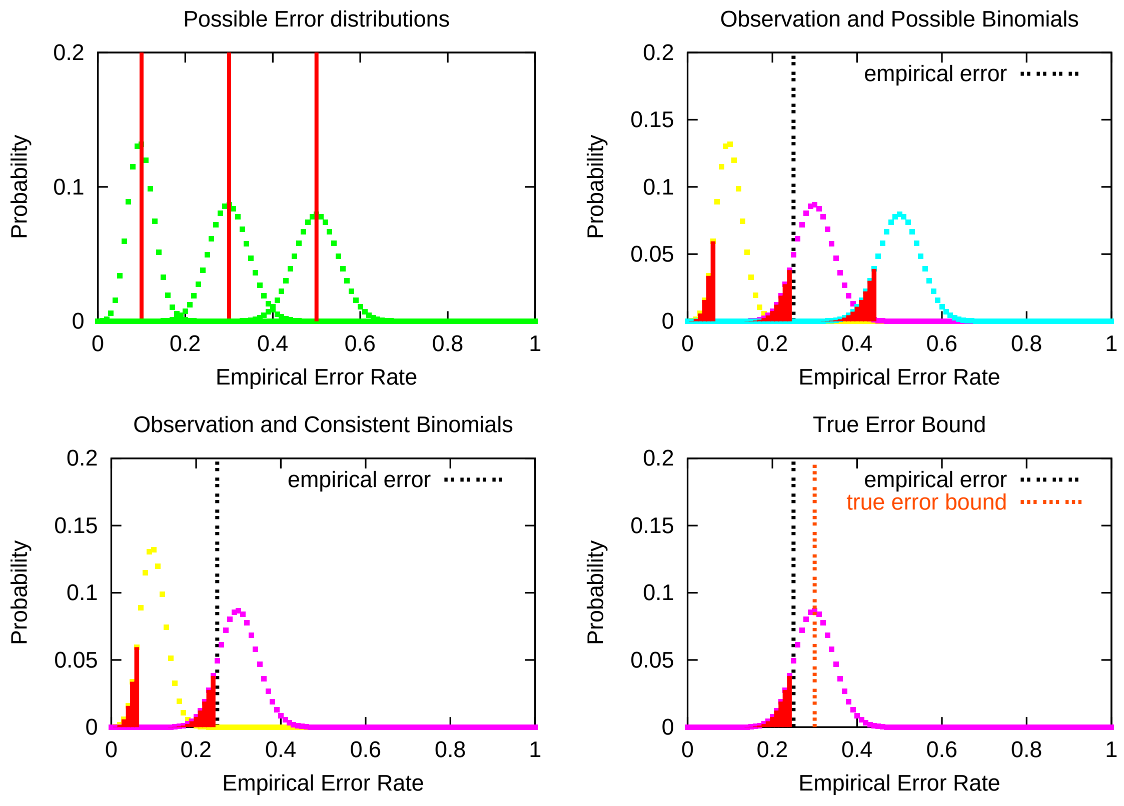


Figure 2: A graphical depiction of the test set bound. The first graph depicts several possible binomials given their true error rates. The second depicts several binomials, each with a tail cut. The third figure shows the binomials consistent with the tail cut and observed test error. The worst case over all true error rates is the consistent binomial with the largest bias.

Note that m in this equation is $m_{\text{test}} = |S_{\text{test}}|$, the size of the test set.

Proof (pictorially in 2) The proof is just a simple identification with the binomial. For any distribution over (x,y) pairs and any classifier c, there exists some probability c_D that the classifier predicts incorrectly. We can regard this event as a coin flip with bias c_D. Since each example is picked independently, the distribution of the empirical error is a binomial distribution.

Whatever our true error c_D is, with probability $1-\delta$ the observation $\hat{c}_S$ will not fall into a tail of size δ. Assuming (correctly with probability $1-\delta$) that the empirical error is not in the binomial tail, we can constrain (and therefore bound) the value of the true error c_D. ■

The test set bound is, essentially, perfectly tight. For any classifier with a sufficiently large true error, the bound is violated exactly a δ portion of the time.

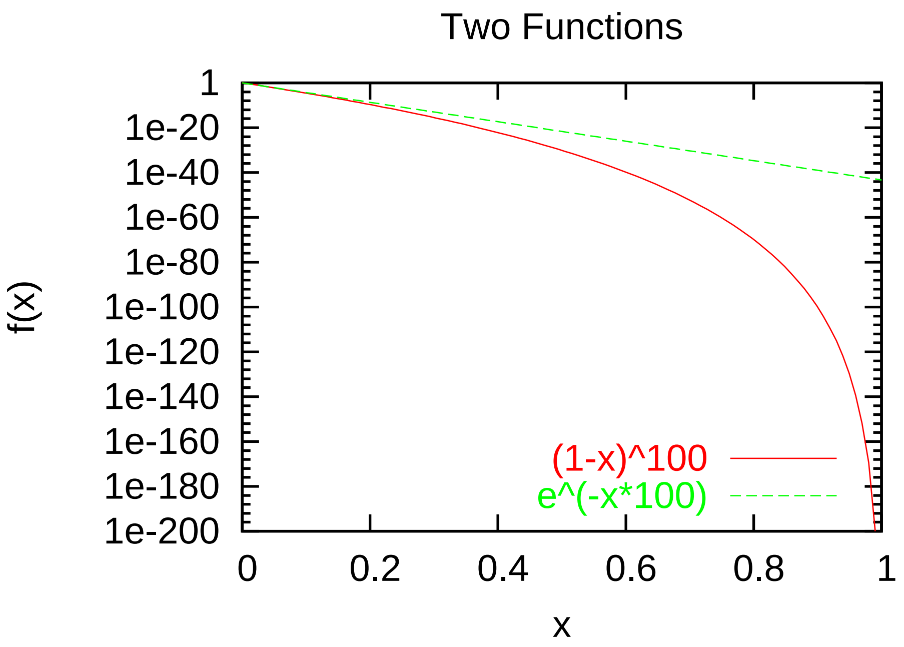


Figure 3: A graph suggesting $e^{-\epsilon m} \geq (1-\epsilon)^m$.

3.1.1 APPROXIMATIONS

There are several immediate corollaries of the test set bound (3.3) which are more convenient when a computer is not handy. The first corollary applies to the limited "realizable" setting where you happen to observe 0 test errors.

Corollary 3.4 *(Realizable Test Set Bound) For all D, For all classifiers c, for all* $\delta \in (0,1]$

$$\Pr_{S \sim D^m}\left(\hat{c}_S = 0 \Rightarrow c_D \leq \frac{\ln \frac{1}{\delta}}{m}\right) \geq 1-\delta.$$

Proof Specializing the test set bound (Theorem 3.3) to the zero empirical error case, we get

$$\mathrm{Bin}(m,0,\epsilon) = (1-\epsilon)^m \leq e^{-\epsilon m}.$$

Setting this equal to δ and solving for ϵ gives us the result. The last inequality can be most simply motivated by comparing graphs as in figure 3.

■

Approximations which hold for arbitrary (nonzero) error rates rely upon the Chernoff bound which we state next, for completeness. For this bound (and it's later applications) we overload the definition of KL-divergence so it applies to two $p,q \in [0,1]$ variables.

Definition 3.5 *(KL-divergence overload)* $KL_+(q||p) = q\log\frac{q}{p} + (1-q)\log\frac{1-q}{1-p}$ *for* $p > q$ *and* 0 *otherwise.*

Lemma 3.6 *(Relative Entropy Chernoff Bound)*[2] *For $\frac{k}{m} < p$:*

$$Bin\,(m,k,p) \le e^{-mKL_{+}\left(\frac{k}{m}||p\right)}.$$

Proof (Originally from (Chernoff, 1952). The proof here is based on (Seung).) For all $\lambda > 0$, we have

$$\text{Bin}\,(m,k,p) = \Pr_{X^m \sim p^m}\left(\sum_{i=1}^{m} X_i \le k\right) = \Pr_{X^m \sim p^m}\left(e^{-m\lambda\frac{1}{m}\sum_{i=1}^m X_i} \ge e^{-m\lambda\frac{k}{m}}\right).$$

Using Markov's inequality ($X \ge 0$, $EX = \mu$, $\Rightarrow \Pr(X \ge \delta) \le \frac{\mu}{\delta}$), this must be less than or equal to

$$\frac{E_{X^m \sim p^m} e^{-\lambda \sum_{i=1}^m X_i}}{e^{-\lambda k}}.$$

Using independence, this expression is equal to

$$e^{\lambda k}\left(pe^{-\lambda} + (1-p)\right)^m,$$

and rewriting, we get

$$e^{mf(\lambda)},$$

where $f(\lambda) = \lambda\frac{k}{m} + \ln\left(pe^{-\lambda} + 1 - p\right)$.

λ is a free parameter which can be optimized to find the tightest possible bound. To find the optimal value, find λ^* so that $f'(\lambda^*) = 0$.

$$0 = f'(\lambda^*) = \frac{k}{m} - \frac{pe^{-\lambda^*}}{pe^{-\lambda^*} + 1 - p}$$

$$\Rightarrow \frac{\frac{k}{m}}{p}\left(pe^{-\lambda^*} + 1 - p\right) = e^{-\lambda^*}$$

$$\Rightarrow \frac{\frac{k}{m}}{p}(1-p) = \left(1 - \frac{k}{m}\right)e^{-\lambda^*}$$

$$\Rightarrow e^{\lambda^*} = \frac{p\left(1-\frac{k}{m}\right)}{\frac{k}{m}(1-p)},$$

which is valid for $p > \frac{k}{m}$. Using this, we get

$$f(\lambda^*) = \frac{k}{m}\ln\frac{p\left(1-\frac{k}{m}\right)}{\frac{k}{m}(1-p)} + \ln\left(\frac{1-p}{1-\frac{k}{m}}\right)$$

$$= \frac{k}{m}\ln\frac{p}{\frac{k}{m}} + \left(1-\frac{k}{m}\right)\ln\left(\frac{1-p}{1-\frac{k}{m}}\right) = -\text{KL}\left(\frac{k}{m}||p\right).$$

■

Using the Chernoff bound, we can loosen the test set bound to achieve a more analytic form.

2. The closely related Hoeffding bound (Hoeffding, 1963) makes the same statement for sums of $[0,1]$ random variables.

Corollary 3.7 *(Agnostic Test Set Bound) For all D, for all classifiers c, for all* $\delta \in (0,1]$

$$\Pr_{S\sim D^m}\left(KL\left(\frac{\hat{c}_S}{m}||c_D\right) \leq \frac{\ln\frac{1}{\delta}}{m}\right) \geq 1-\delta.$$

Proof Loosening the test set bound (theorem 3.3) with the Chernoff approximation for $\frac{k}{m} < c_D$ we get

$$\text{Bin}\left(m,k,c_D\right) \leq e^{-m\text{KL}\left(\frac{k}{m}||c_D\right)}.$$

Setting this equal to δ, and solving for ε gives the result. ■

The agnostic test set bound can be further loosened by bounding the value of $\text{KL}(q||p)$.

Corollary 3.8 *(Agnostic Test Set Bound II) For all classifiers c, for all* $\delta \in (0,1]$

$$\Pr_{S\sim D^m}\left(c_D \leq \frac{\hat{c}_S}{m} + \sqrt{\frac{\ln\frac{1}{\delta}}{2m}}\right) \geq 1-\delta.$$

Proof Use the approximation

$$\text{KL}\left(\frac{k}{m}||c_D\right) \geq 2(c_D - \frac{k}{m})^2$$

with the Chernoff bound and test set bounds to get the result. ■

The differences between the agnostic and realizable case are fundamentally related to the decrease in the variance of a binomial as the bias (i.e. true error) approaches 0. Note that this implies using the exact binomial tail calculation can result in *functional* (rather than merely constant) improvements on the above corollary.

3.1.2 A Test Set Lower Bound

The true error can be lower bounded using a symmetric application of the same techniques.

Theorem 3.9 *(Test Set Lower Bound) For all classifiers, c, for all* $\delta \in (0,1]$

$$\Pr_{S\sim D^m}\left(c_D \geq \min_p\{p:\ 1 - Bin\left(m,\hat{c}_S,p\right) \geq \delta\}\right) \geq 1-\delta.$$

The proof is completely symmetric. Note that both bounds hold with probability $1-2\delta$ since $\Pr(A \text{ or } B) \leq \Pr(A)+\Pr(B)$. This is particularly convenient when the square-root version of the Chernoff approximation is used in both directions to get

$$\forall c\ \Pr_{S\sim D^m}\left(\left|c_D - \frac{\hat{c}_S}{m}\right| \leq \sqrt{\frac{\ln\frac{2}{\delta}}{2m}}\right) \geq 1-\delta.$$

Example 3 *(continued) let* $\delta = 0.1$. *Using the square root Chernoff bound with* $\hat{c}_S = 38$ *out of* 100 *examples, we get the confidence interval* $c_D \in [0.26, 0.50]$. *Using an exact calculation for the binomial tail, we get* $c_D \in [0.30, 0.47]$. *In general, as the observed error moves toward* 0, *the exact calculation provides a tighter confidence interval than the agnostic approximation.*

3.1.3 THE STATE OF THE ART

Although the test set bound is very well understood, the same cannot be said of other testing methods. Only weak general results in this model are known for some variants of cross validation (see Blum et al., 1999). For specific learning algorithms (such as nearest neighbor), stronger results are known (see Devroye et al., 1996). There are a wide range of essentially unanalyzed methods and a successful analysis seems particularly tricky although very worthwhile if completed.

3.2 Test Set Bound Implications

There are some common practices in machine learning which can be improved by application of the test set bound. When attempting to calculate a confidence interval on the true error rate given the test set, many people follow a standard statistical prescription:

1. Calculate the empirical mean $\hat{\mu} = \frac{\hat{c}_{S_{\text{test}}}}{m} = \frac{1}{m}\sum_{i=1}^{m} I(h(x_i) \neq y_i)$.
2. Calculate the empirical variance $\hat{\sigma}^2 = \frac{1}{m-1}\sum_{i=1}^{m}(I(c(x_i) = y_i) - \hat{\mu})^2$.
3. Pretend that the distribution is Gaussian with the above variance and construct a confidence interval by cutting the tails of the Gaussian cumulative distribution at the $2\hat{\sigma}$ (or some other) point.

This approach is motivated by the fact that for any *fixed* true error rate, the distribution of the observed accuracy behaves like a Gaussian *asymptotically.* Here, asymptotically means “in the limit as the number of test examples goes to infinity”.

The problem with this approach is that it leads to fundamentally misleading results as shown in Figure 4. To construct this figure, a collection of discrete (aka “nominal”) feature datasets from the UCI machine learning database were split into training and test sets. A decision tree classifier was learned on each training set and then evaluated on the held-out test set.

This “misleading” is both pessimistic and (much worse) optimistic. The pessimism can be seen by intervals with boundaries less than 0 or greater than 1 and the optimism by observing what happens when the test error is 0. When we observe perfect classification, our confidence interval should *not* have size 0 for any finite m.

The basic problem with this approach is that the binomial distribution is not similar to a Gaussian when the error rate is near 0. Since our goal is finding a classifier with a small true error, it is essential that the means we use to evaluate classifiers work in this regime. The test set bound can satisfy this requirement (and, in fact, operates well for all true error regimes).

1. The test set bound approach is *never* optimistic.
2. The test set bound based confidence interval always returns an upper and lower bound in $[0,1]$.

The $2\hat{\sigma}$ method is a relic of times when computational effort was expensive. It is now simple and easy to calculate a bound based upon the cumulative distribution of the binomial (see Langford).

The test set bound can be thought of as a game where a “Learner” attempts to convince a reasonable “Verifier” of the amount of learning which has occurred. Pictorially we can represent this as in Figure 5.

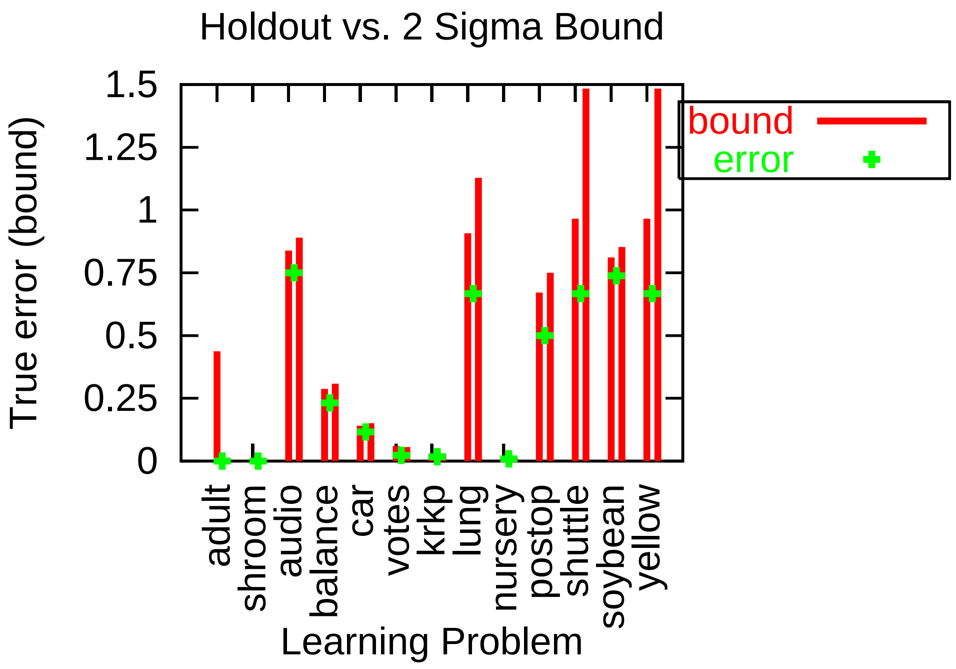


Figure 4: This is a graph of the confidence intervals implied by the test set bound (theorem 3.3) on the left, and the approximate confidence intervals implied using the common two sigma rule motivated by asymptotic normality on the right. The upper bounds of the test set bound have $\delta = 0.025$ failure rate, so as to be comparable with the 2-sigma approach. The test set bound is better behaved as the confidence interval is confined to the interval $[0,1]$ and is never over-optimistic.

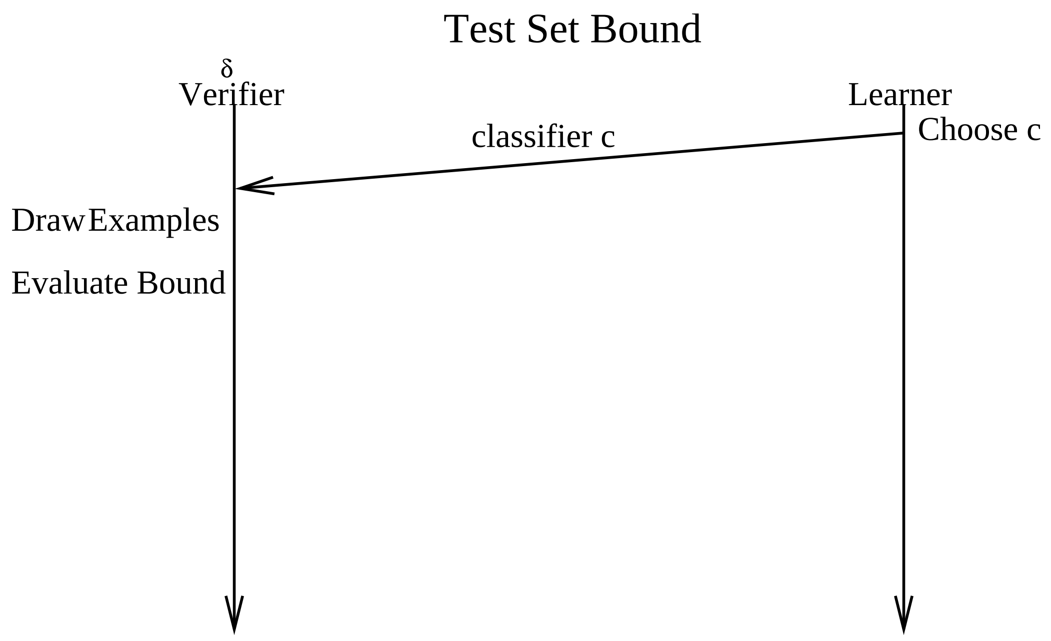


Figure 5: For this diagram "increasing time" is pointing downwards. The only requirement for applying this bound is that the learner must commit to a classifier without knowledge of the test examples. A similar diagram for train set bounds is presented later (and is somewhat more complicated). We can think of the bound as a technique by which the "Learner" can convince the "Verifier" that learning has occurred (and the degree to which it has occurred). Each of the proofs can be thought of as a communication protocol for an interactive proof of learning by the Learner.

4. The Occam's Razor Bound

Given that the simple test set bound works well, why do we need to engage in further work? There is one serious drawback to the test set technique—it requires m_{test} otherwise unused examples. An extra m_{test} examples for the training set decreases the true error of the learned hypothesis to 0 from 0.5 for some natural learning algorithm/learning problem pairs. This loss of performance due to holding out examples is very severe.

There is another reason why training set based bounds are important. Many learning algorithms implicitly assume that the train set accuracy "behaves like" the true error in choosing the hypothesis. With an inadequate number of training examples, there may be very little relationship between the behavior of the train set accuracy and the true error. Training set based bounds can be used *in* the training algorithm and can provide insight into the learning problem itself.

This section is organized into three subsections.

1. Subsection 4.1 states and proves the Occam's Razor bound.

2. Subsection 4.2 proves that the Occam's Razor bound cannot be improved in general.

3. Subsection 4.3 discusses implications of the Occam's Razor bound and shows results for its application.

4.1 The Occam's Razor Bound

This Occam's Razor bound (Blumer et al., 1987; McAllester, 1999) in more approximate forms has appeared elsewhere. We use "prior" (with quotes) here because it is an arbitrary probability distribution over classifiers and not necessarily a Bayesian prior. The distinction is important, because the theory holds regardless of whether or not a Bayesian prior is used.

Theorem 4.1 *(Occam's Razor Bound) For all D, for all "priors" $P(c)$ over the classifiers c, for all $\delta \in (0,1]$:*

$$\Pr_{S\sim D^m}\left(\forall c:\ c_D \leq \overline{Bin}\left(m, \hat{c}_S, \delta P(c)\right)\right) \geq 1-\delta$$

The application of the Occam's Razor bound is somewhat more complicated than the application of the test set bound. Pictorially, the protocol for bound application is given in Figure 6. It is very important to notice that the "prior" $P(c)$ must be selected *before* seeing the training examples.

Proof (pictorially in Figure 7) First, note that if $P(c) = 0$, then $\overline{\text{Bin}}\left(m, \hat{c}_S, 0\right) = 1$ and the bound is always valid. The remainder of this proof applies to the countable set of c satisfying $P(c) > 0$.

The proof starts with the test set bound:

$$\forall c \ \Pr_{S\sim D^m}\left(c_D \leq \overline{\text{Bin}}\left(m, \hat{c}_S, \delta P(c)\right)\right) \geq 1-\delta P(c)$$

Negating this statement, we get

$$\forall c \ \Pr_{S\sim D^m}\left(c_D > \overline{\text{Bin}}\left(m, \hat{c}_S, \delta P(c)\right)\right) < \delta P(c)$$

then, we apply the union bound in a nonuniform manner. The union bound says that $\Pr(A \text{ or } B) \leq \Pr(A) + \Pr(B)$. Applying the union bound to every classifier with a positive measure gives a total probability of failure of

$$\sum_{c:P(c)>0} \delta P(c) = \delta \sum_{c:P(c)>0} P(c) = \delta$$

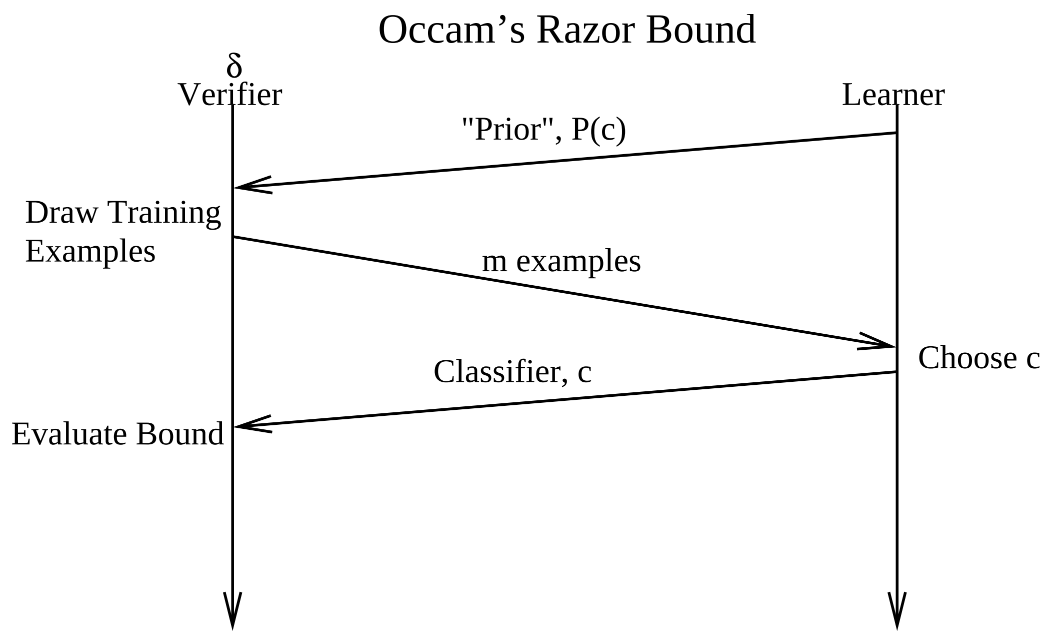


Figure 6: In order to apply the Occam's Razor bound it is necessary that the choice of "prior" be made before seeing any training examples. Then, the bound is calculated based upon the chosen classifier. Note that it *is* "legal" to chose the classifier based upon the prior $P(c)$ as well as the empirical error $\hat{c}_S$.

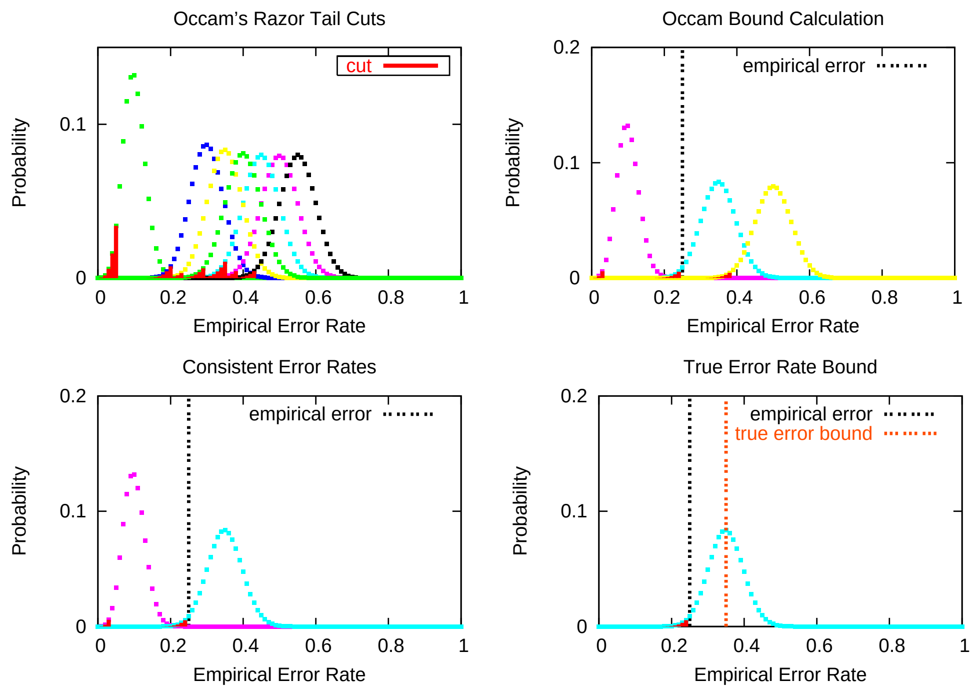


Figure 7: The sequence of pictures is the pictorial representation of the proof of the Occam's Razor Bound. The first figure shows a set of classifiers, each with a tail cut of some varying depth. The second picture shows an observed training error and the possible binomial distributions for a chosen classifier. The third picture shows the true errors which are consistent with the observation and the tail cuts. The fourth picture shows the true error bound.

which implies

$$\Pr_{S\sim D^m}\left(\exists c:\ c_D > \overline{\text{Bin}}\left(m,\hat{c}_S,\delta P(c)\right)\right) < \delta.$$

Negating this again completes the proof. ■

4.1.1 OCCAM'S RAZOR COROLLARIES

Just as with the test set bound, we can relax the Occam's Razor bound (Theorem 4.1) with the Chernoff approximations to get a somewhat more tractable expression.

Corollary 4.2 *(Chernoff Occam's Razor Bound) For all D, for all "priors" P(c) over classifiers, for all* $\delta \in (0,1]$*:*

$$\Pr_{S\sim D^m}\left(\forall c:\ c_D \le \frac{\hat{c}_S}{m} + \sqrt{\frac{\ln\frac{1}{P(c)} + \ln\frac{1}{\delta}}{2m}}\right) \ge 1-\delta$$

Proof Approximate the binomial tail with the Chernoff Bound (lemma 3.6). ■

Many people are more familiar with a degenerate form of this bound where $P(c) = \frac{1}{|H|}$ and H is some set of classifiers. In that case, simply replace $\ln \frac{1}{P(c)}$ with $\ln |H|$. The form presented here is both more general and necessary if the bound is to be used in practice.

Other corollaries as in Section 3.1.1 exist for the Occam's Razor bound. In general, just substitute $\delta \rightarrow \delta P(c)$.

4.1.2 OCCAM'S RAZOR LOWER BOUND

Just as for the test set bound, a lower bound of the same form applies.

Theorem 4.3 *(Occam's Razor Lower Bound) For all D, for all "priors" $P(c)$ over the classifiers, c, for all $\delta \in (0,1]$:*

$$\Pr_{S \sim D^m}\left(\forall c:\ c_D \geq \min_p\{p:\ 1 - Bin(m, \hat{c}_S, p) \geq \delta P(c)\}\right) \geq 1 - \delta.$$

Example 4 *(continued) Suppose that instead of having* 100 *test examples, we had* 100 *train examples. Also suppose that before seeing the train examples, we committed to $P(c) = 0.1$ for c the constant classifier which predicts "no rain". Then, the Chernoff approximations of the upper and lower bound give the interval, $c_D \in [0.22, 0.54]$. With an exact calculation, we get $c_D \in [0.26, 0.51]$.*

4.1.3 THE STATE OF THE ART

A very large amount of work has been done on train set bounds. In addition to those included here, there are:

1. Reinterpretations of uniform convergence (Vapnik and Chervonenkis, 1971) results for continuously parameterized classifiers.
2. Reinterpretations of PAC convergence (Valiant, 1984) results.
3. Shell bounds (Langford and McAllester, 2000) which take advantage of the distribution of true error rates on classifiers.
4. Train and test bounds (Langford, 2002) which combine train set and test set bounds.
5. (Local) Rademacher complexity (Bartlett et al., 2004) results which take advantage of the error geometry of nearby classifiers.

... and many other results.

Of this large amount of work only a small fraction has been shown to be useful on real-world learning algorithm/learning problem pairs. The looseness of train set based bounds often precludes analytical use.

4.2 The Occam's Razor Bound is Sometimes Tight

The question of tightness for train set bounds is important to address, as many of them have been extremely loose. The simplest method to address this tightness is constructive: exhibit a learning problem/algorithm pair for which the bound is almost achieved. For the test set bound, this is trivial as any classifier with a large enough true error will achieve the bound. For the train set bound, this is not so trivial.

How tight is the Occam's Razor bound (4.1)? The answer is *sometimes* tight. In particular, we can exhibit a set of learning problems where the Occam's Razor bound can not be made significantly tighter as a function of the observables, m, δ, $P(c)$, and $\hat{c}_S$. After fixing the value of these quantities we construct a learning problem exhibiting this near equivalence to the Occam's Razor bound.

Theorem 4.4 *(Occam's Razor tightness) For all $P(c)$, m, k, δ there exists a learning problem D and algorithm such that:*

$$\Pr_{S\sim D^m}\left(\exists c:\ \hat{c}_S \le k \text{ and } c_D \ge \overline{Bin}\left(m, \hat{c}_S, \delta P(c)\right)\right) \ge \delta - \delta^2.$$

Furthermore, if c^* is the classifier with minimal training error, then:

$$\Pr_{S\sim D^m}\left(c^*_D \ge \overline{Bin}\left(m, \hat{c}^*_S, \delta P(c)\right)\right) \ge \delta - \delta^2.$$

Intuitively, this theorem implies that we can not improve significantly on the Occam's Razor bound (Theorem 4.1) without using extra information about our learning problem.

Proof The proof is constructive: we create a learning problem on which large deviations are likely. We start with a prior $P(c)$, probability of error δ, m, and a targeted empirical error number, k. For succinctness we assume that $P(c)$ has support on a finite set of size n.

To define the learning problem, let: $X = \{0,1\}^n$ and $Y = \{0,1\}$.

The distribution D can be drawn by first selecting Y with a single unbiased coin flip, and then choosing the ith component of the vector X independently, $\Pr((X_1,\ldots,X_n)|Y) = \Pi_{i=1}^n \Pr(X_i|Y)$. The individual components are chosen so $\Pr(X_i = Y|Y) = \overline{\mathrm{Bin}}\left(m,k,\delta P(c)\right)$.

The classifiers we consider just use one feature to make their classification: $c_i(x) = x_i$. The true error of these classifiers is given by: $c_D = \overline{\mathrm{Bin}}\left(m,k,\delta P(c)\right)$.

This particular choice of true errors implies that if any classifier has a too-small train error, then the classifier with minimal train error must have a too-small train error.

Using this learning problem, we know that:

$$\forall c, \forall\delta \in (0,1]:\ \Pr_{S\sim D^m}\left(c_D \ge \overline{\mathrm{Bin}}\left(m,\hat{c}_S,\delta P(c)\right)\right) = \delta P(c)$$

(negation)

$$\Rightarrow \forall c, \forall\delta \in (0,1]:\ \Pr_{S\sim D^m}\left(c_D < \overline{\mathrm{Bin}}\left(m,\hat{c}_S,\delta P(c)\right)\right) = 1 - \delta P(c)$$

(independence)

$$\Rightarrow \forall\delta \in (0,1]:\ \Pr_{S\sim D^m}\left(\forall c\ c_D < \overline{\mathrm{Bin}}\left(m,\hat{c}_S,\delta P(c)\right)\right) < \prod_c (1-\delta P(c))$$

(negation)

$$\Rightarrow \forall\delta \in (0,1]:\ \Pr_{S\sim D^m}\left(\exists c\ c_D \ge \overline{\mathrm{Bin}}\left(m,\hat{c}_S,\delta P(c)\right)\right) \ge 1 - \prod_c (1-\delta P(c))$$

$$= \sum_{i=1}^n \delta P(c_i) \prod_{j<i} (1-\delta P(c_j)) \ge \sum_{i=1}^n \delta P(c_i)(1-\delta) = \delta - \delta^2$$

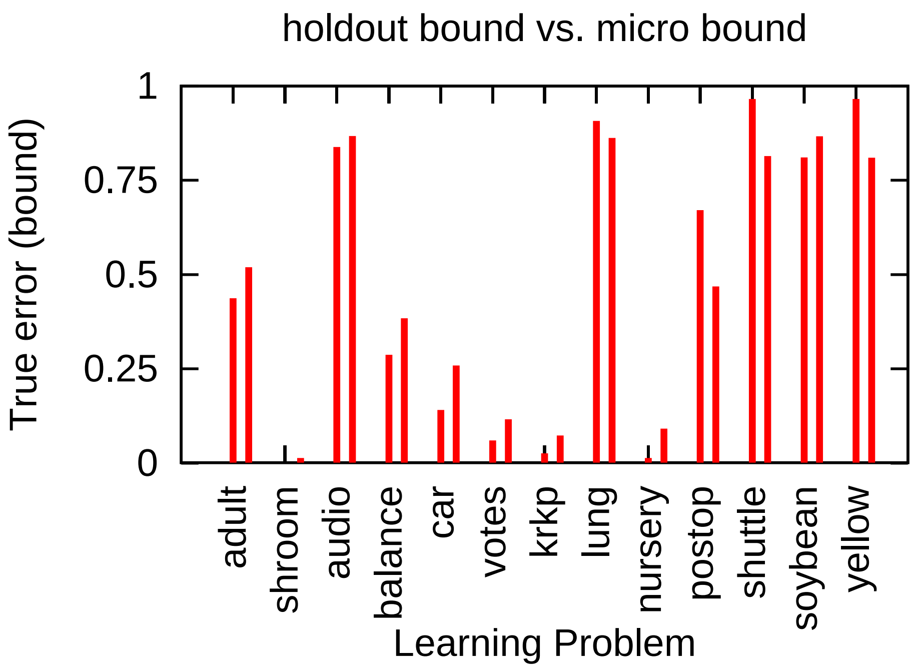


Figure 8: This is a plot comparing confidence intervals built based upon the test set bound (Theorem 3.3) with an 80%/20% train/test split on the left and the Occam's Razor bound (Theorem 4.1) with all data in the training set on the right. The Occam's razor bound is sometimes superior on the smaller data sets and always nonvacuous (in contrast to many other train set bounds).

where the last inequality follows from $(1-a)(1-b) \geq 1-a-b$ for $a,b \in [0,1]$. ■

The lower bound theorem implies that we can not improve an Occam's Razor like statement. However, it is important to note that large improvements are possible if we use other sources of information. To see this, just note the case where every single classifier happens to be the same. In this case the "right" bound would the be the *test* set bound, rather than the Occam's Razor bound. The PAC-Bayes bound and the sample compression bound presented in the next sections use other sources of information. Another common source of information is specialization to classifiers of some specific sort.

4.3 Occam's Razor Bound Implications

The Occam's Razor bound is strongly related to compression. In particular, for any self-terminating description language, $d(c)$, we can associate a "prior" $P(c) = 2^{-|d(c)|}$ with the property that $\sum_c P(c) \leq 1$. Consequently, short description length classifiers tend to have a tighter convergence and the penalty term, $\ln \frac{1}{P(c)}$ is the number of "nats" (bits base e). For any language fixed before seeing the training sequence, classifiers with shorter description lengths have tighter bounds on the true error rate.

One particularly useful description language to consider is the execution trace of a learning algorithm. If we carefully note the sequence of data-dependent choices which a learning algorithm makes, then the output classifier can be specified by a sequence such as "second choice, third choice, first choice, etc...." This is the idea behind microchoice bounds (Langford and Blum, 1999). Results for this approach are reported in Figure 8 and are strong enough to act as an empirical existence proof that Occam's Razor bounds can be made tight enough for useful application.

5. PAC-Bayes Bound

The PAC-Bayes bound (McAllester, 1999) is particularly exciting because it can provide quantitatively useful results for classifiers with *real valued* parameters. This includes such commonly used classifiers as support vector machines and neural networks.[3] This section is divided into three parts:

1. Subsection 5.1 states and proves the PAC-Bayes Bound.
2. Subsection 5.2 shows that the PAC-Bayes Bound is nearly as tight as possible given the observations.
3. Subsection 5.3 discusses results from the application of the PAC-Bayes bound to support vector machines.

5.1 The PAC-Bayes Bound

The PAC-Bayes bound has been improved by tightening (Langford and Seeger, 2001) and then with a much simpler proof (Seeger, 2002) since it was originally stated. The statement and proof presented here incorporate these improvements and improve on them slightly.

The PAC-Bayes bound is dependent upon two derived quantities, an average true error:

$$Q_D \equiv E_{c\sim Q} c_D$$

and an average train error rate:

$$\hat{Q}_S \equiv E_{c\sim Q} \frac{\hat{c}_S}{m}.$$

These quantities can be interpreted as the train error rate and true error of the meta-classifier which chooses a classifier according to Q every time a classification is made. If we refer to this meta-classifier as Q, the notation for error rates is consistent with our earlier notation.

The "interactive proof of learning" viewpoint of the PAC-Bayes bound is shown in Figure 9. It is essentially the same as for the Occam's Razor bound except for the commitment to the metaclassifier Q rather than the classifier c.

Theorem 5.1 *(PAC-Bayes Bound) For all D, for all "priors" $P(c)$ over the classifiers c, for all $\delta \in (0,1]$:*

$$\Pr_{S\sim D^m}\left(\forall Q(c):\ KL_+\left(\hat{Q}_S || Q_D\right) \le \frac{KL(Q||P) + \ln\frac{m+1}{\delta}}{m}\right) \ge 1-\delta$$

where $KL(Q||P) = E_{c\sim Q} \ln \frac{Q(c)}{P(c)}$ is the KL-divergence between Q and P.

3. There is a caveat here—the bound only applies to stochastic versions of the classifiers. However, the probability that the stochastic classifier differs from the classifier can be made very small.

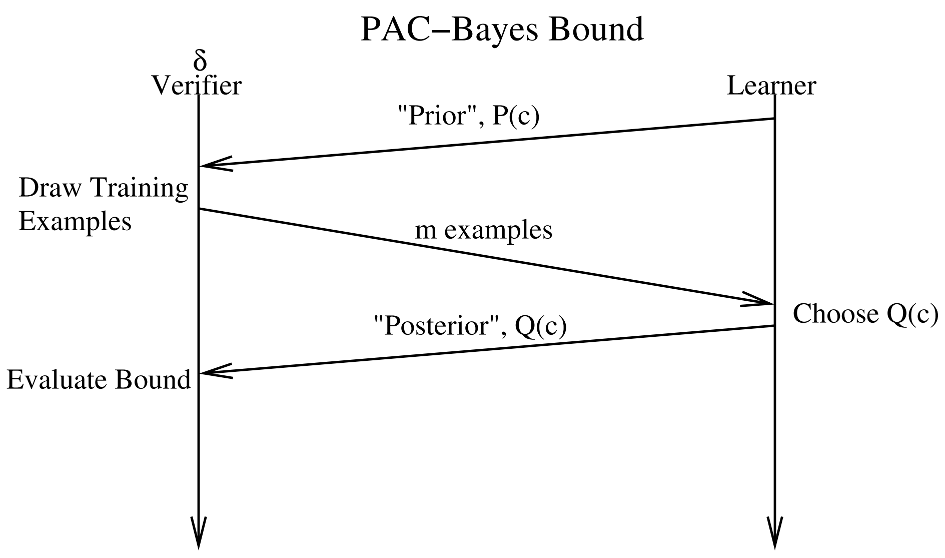


Figure 9: The “interactive proof of learning” associated with the PAC-Bayes bound. The figure is the same as for the Occam’s razor bound, except that instead of committing to a single classifier, the PAC-Bayes bound applies to any distribution over classifiers.

Note that the PAC-Bayes bound applies to any *distribution* over classifiers. When Q is concentrated on one classifier, we have $\mathrm{KL}(Q||P) = \ln \frac{1}{P(c)}$, just as in the Occam’s razor bound,[4] with the only distinction being the additive $\frac{\ln(m+1)}{m}$ term. It is somewhat surprising that the bound holds for *every* distribution Q with only the slight worsening by $\frac{\ln(m+1)}{m}$.

Since the KL-divergence applies to distributions over continuous valued parameters, the PAC-Bayes bound can be nontrivially tight in this setting as well. This fact is used in the application section.

We first state a couple simple lemmas that are handy in the proof. The intuition behind this lemma is that the expected probability of an event is not too small.

Lemma 5.2 *For all D, for all $P(c)$, for all $\delta \in (0,1]$:*

$$\Pr_{S\sim D^m}\left(E_{c\sim P}\frac{1}{\Pr_{S'\sim D^m}(\hat{c}_S = \hat{c}_{S'})} \le \frac{m+1}{\delta}\right) \ge 1-\delta.$$

Proof Note that:

$$\forall c \; E_{S\sim D^m}\frac{1}{\Pr_{S'\sim D^m}(\hat{c}_S = \hat{c}_{S'})} = \sum_{\frac{k}{m}} \Pr_{S\sim D^m}(\hat{c}_S = k)\frac{1}{\Pr_{S'\sim D^m}(\hat{c}_{S'} = k)} = m+1.$$

Taking the expectation over classifiers according to P and switching the order of expectation, we get

4. As weakened with the relative entropy Chernoff bound (Lemma 3.6) on the binomial.

$$E_{S\sim D^m} E_{c\sim P} \frac{1}{\Pr_{S'\sim D^m}(\hat{c}_S = \hat{c}_{S'})} = m+1$$

and using the Markov inequality ($X \geq 0$, $EX = \mu$, $\Rightarrow \Pr(X > \frac{\mu}{\delta}) < \delta$), we get

$$\forall P \Pr_{S\sim D^m}\left(E_{c\sim P} \frac{1}{\Pr_{S'\sim D^m}(\hat{c}_S = \hat{c}_{S'})} > \frac{m+1}{\delta} \right) < \delta.$$

■

The next lemma shows that a certain expectation is bounded by the Kullback-Leibler distance between two coin flips, just as for the relative entropy Chernoff bound (Lemma 3.6).

Lemma 5.3 *Fix all example sequences S. For all $Q(c)$:*

$$\frac{E_{c\sim Q} \ln \frac{1}{\Pr_{S'\sim D^m}(\hat{c}_S = \hat{c}_{s'})}}{m} \geq KL(\hat{Q}_S || Q_D).$$

Proof

$$\frac{E_{c\sim Q} \ln \frac{1}{\Pr_{S'\sim D^m}(\hat{c}_S = \hat{c}_{S'})}}{m} = \frac{1}{m} E_{c\sim Q} \ln \frac{1}{\binom{m}{\hat{c}_S} c_D^{\hat{c}_S} (1-c_D)^{m-\hat{c}_S}}$$

$$\geq \frac{1}{m} E_{c\sim Q} \ln \frac{1}{\sum_{k=0}^{\hat{c}_S} \binom{m}{k} c_D^k (1-c_D)^{m-k}} \geq E_{c\sim Q} \mathrm{KL}\left(\frac{\hat{c}_S}{m} || c_D\right)$$

where the last inequality follows from the relative entropy Chernoff bound. Since $\frac{\partial^2}{\partial p \partial q}\mathrm{KL}(q||p) = -\frac{1}{p} - \frac{1}{1-p} < 0$ the function is concave in both arguments. Jensen's inequality ($f(x,y)$ concave $\Rightarrow E f(x,y) \geq f(Ex, Ey)$) gives us

$$\geq \mathrm{KL}(E_{c\sim Q} \hat{c}_S || E_{c\sim Q} c_D),$$

which completes the proof. ■

With these two lemmas, the PAC-Bayes theorem is easy to prove.
Proof (Of the PAC-Bayes theorem) Fix a training set S. Let

$$P_G(c) = \frac{1}{\Pr_{S'\sim D^m}(\hat{c}_{S'} = \hat{c}_S) E_{d\sim P} \frac{1}{\Pr_{S'\sim D^m}(\hat{d}_S = \hat{d}_{S'})}} P(c).$$

$P_G(c)$ is a normalized distribution because it has the form $\frac{a_c}{E a_c} P(c)$ where $P(c)$ is a distribution.

$$\Rightarrow 0 \leq \mathrm{KL}(Q||P_G) = E_{c\sim Q} \ln \left[\frac{Q(c)}{P(c)} \Pr_{S'\sim D^m}(\hat{c}_{S'} = \hat{c}_S) E_{d\sim P} \frac{1}{\Pr_{S'\sim D^m}(\hat{d}_S = \hat{d}_{S'})} \right]$$

$$= \mathrm{KL}(Q||P) - E_{c\sim Q} \ln \frac{1}{\Pr_{S'\sim D^m}(\hat{c}_{S'} = \hat{c}_S)} + \ln E_{d\sim P} \frac{1}{\Pr_{S'\sim D^m}(\hat{d}_S = \hat{d}_{S'})}$$

$$\Rightarrow E_{c\sim Q} \ln \frac{1}{\Pr_{S'\sim D^m}\left(\hat{c}_{S'} = \hat{c}_S\right)} \leq \mathrm{KL}(Q||P) + \ln E_{d\sim P} \frac{1}{\Pr_{S'\sim D^m}\left(\hat{d}_S = \hat{d}_{S'}\right)}.$$

Applying lemma 5.3 on the left hand term we get

$$m\mathrm{KL}(\hat{Q}_S||Q_D) \leq \mathrm{KL}(Q||P) + \ln E_{d\sim P} \frac{1}{\Pr_{S'\sim D^m}\left(\hat{d}_S = \hat{d}_{S'}\right)}.$$

This holds for all S. Applying Lemma 5.2 which randomizes over S, we get the theorem. ■

5.2 The PAC-Bayes Bound is Sometimes Tight

Since the PAC-Bayes bound is (almost) a generalization of the Occam's Razor bound, the tightness result for Occam's Razor also applies to PAC-Bayes bounds.

5.3 Application of the PAC-Bayes Bound

Applying the PAC-Bayes bound requires specialization (Langford and Shawe-Taylor, 2002). Here, we specialize to classifiers of the form

$$c(x) = \mathrm{sign}\left(\vec{w} \cdot \vec{x}\right).$$

Note that via the kernel trick, support vector machines also have this form.

The specialization is naturally expressed in terms of a few derived quantities:

1. The cumulative distribution of a Gaussian. Let $\bar{F}(x) = \int_x^\infty \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$. Here we use $\bar{F}$ rather than F to denote the fact that we integrate from x to ∞ rather than $-\infty$ to x.

2. A "posterior" distribution $Q(\vec{w},\mu)$ which is $N(\mu,1)$ for some $\mu > 0$ in the direction of $\vec{w}$ and $N(0,1)$ in all perpendicular directions.

3. The normalized margin of the examples

$$\gamma(\vec{x},y) = \frac{y\vec{w}\cdot\vec{x}}{||\vec{w}||||\vec{x}||}.$$

4. A stochastic error rate, $\hat{Q}(\vec{w},\mu)_S = E_{\vec{x},y\sim S}\bar{F}\left(\mu\gamma(\vec{x},y)\right)$.

This last quantity in particular is very important to understand. Consider the case as μ approaches infinity. When the margin is negative (indicating an incorrect classification), $\bar{F}\left(\mu\gamma(\vec{x},y)\right)$ approaches 1. When the margin is positive $\bar{F}\left(\mu\gamma(\vec{x},y)\right)$ approaches 0. Thus, $\hat{Q}(\vec{w},\mu)_S$ is a softened form of the empirical error $\hat{c}_S$ which takes into account the margin.

Corollary 5.4 *(PAC-Bayes Margin Bound) For all distributions D, for all* $\delta \in (0,1]$*, we have*

$$\Pr_{S\sim D^m}\left(\forall \vec{w},\mu:\ KL\left(\hat{Q}(\vec{w},\mu)_S||Q(\vec{w},\mu)_D\right) \leq \frac{\frac{\mu^2}{2} + \ln\frac{m+1}{\delta}}{m}\right) \geq 1-\delta.$$

Proof The proof is very simple. We just chose the prior $P = N(0,1)^n$ and work out the implications.

Since the Gaussian distribution is the same in every direction, we can reorient the coordinate system of the prior to have one dimension parallel to w. Since the draws in the parallel and perpendicular directions are independent, we have

$$\begin{aligned}\mathrm{KL}(Q||P) &= \mathrm{KL}(Q_\perp||P_\perp) + \mathrm{KL}(N(\mu,1)||N(0,1)) \\ &= \frac{\mu^2}{2}\end{aligned}$$

as required.

All that remains is calculating the stochastic error rate $\hat{Q}(\vec{w},\mu)_S$. Fix a particular example $(\vec{x},y)$. This example has a natural decomposition $\vec{x} = \vec{x}_{||} + \vec{x}_\perp$ into a component $\vec{x}_{||}$ parallel to the weight vector $\vec{w}$ and a component $\vec{x}_\perp$ perpendicular to the weight vector.

To classify, we draw weight vector $\vec{w}^{'}$ from $\hat{Q}(\vec{w},\mu)$. This $\vec{w}^{'}$ consists of three components, $\vec{w}^{'} = \vec{w}^{'}_{||} + \vec{w}^{'}_\perp + \vec{w}^{'}_{\perp\perp}$. Here $\vec{w}^{'}_{||} \sim N(\mu,1)$ is parallel to the original weight vector, $\vec{w}^{'}_\perp \sim N(0,1)$ which is parallel to $\vec{x}_\perp$ and $\vec{w}^{'}_{\perp\perp}$ is perpendicular to both $\vec{w}$ and $\vec{x}$. We have

$$\begin{aligned}\hat{Q}(\vec{w},\mu)_S &= E_{\vec{x},y\sim S,\vec{w}^{'}\sim Q(\vec{w},\mu)} I\left(y \neq \mathrm{sign}\left(\vec{w}^{'}\cdot\vec{x}\right)\right) \\ &= E_{\vec{x},y\sim S,\vec{w}^{'}\sim Q(\vec{w},\mu)} I\left(y\vec{w}\cdot\vec{x} \leq 0\right).\end{aligned}$$

If we let $w^{'}_{||} = ||\vec{w}^{'}_{||}||$, $w^{'}_\perp = ||\vec{w}^{'}_\perp||$, $x_{||} = ||\vec{x}_{||}||$, and $x_\perp = ||\vec{x}_\perp||$, and assume (without loss of generality) that $y = 1$ we get

$$\begin{aligned} &= E_{\vec{x},y\sim S,w^{'}_{||}\sim N(\mu,1),w^{'}_\perp\sim N(0,1)} I\left(y(w^{'}_{||}x_{||} + w^{'}_\perp x_\perp) \leq 0\right) \\ &= E_{\vec{x},y\sim S} E_{w^{'}_{||}\sim N(\mu,1)} E_{w^{'}_\perp\sim N(0,1)} I\left(y(w^{'}_{||}x_{||} + w^{'}_\perp x_\perp) \leq 0\right) \\ &= E_{\vec{x},y\sim S} E_{z^{'}\sim N(0,1)} E_{w^{'}_\perp\sim N(0,1)} I\left(y\mu \leq -yz^{'} - yw^{'}_\perp \frac{x_\perp}{x_{||}}\right).\end{aligned}$$

Using the symmetry of the Gaussian, this is:

$$= E_{\vec{x},y\sim S} E_{z^{'}\sim N(0,1)} E_{w^{'}_\perp\sim N(0,1)} I\left(y\mu \leq yz^{'} + yw^{'}_\perp \frac{x_\perp}{x_{||}}\right)$$

Using the fact that the sum of two Gaussians is a Gaussian:

$$\begin{aligned} &= E_{\vec{x},y\sim S} E_{v\sim N\left(0,1+\frac{x_\perp^2}{x_{||}^2}\right)} I\left(y\mu \leq yv\right) \\ &= E_{\vec{x},y\sim S} E_{v\sim N\left(0,\frac{1}{\gamma(\vec{x},y)^2}\right)} I\left(y\mu \leq yv\right) \\ &= E_{\vec{x},y\sim S} \bar{F}\left(\mu\gamma(\vec{x},y)\right)\end{aligned}$$

finishing the proof. ■

Using the corollary, the true error bound $\bar{Q}(\vec{w},\mu)_D$ satisfies the equation:

$$\mathrm{KL}\left(\hat{Q}(\vec{w},\mu)_S || \bar{Q}(\vec{w},\mu)_D\right) = \frac{\frac{\mu^2}{2} + \ln\frac{m+1}{\delta}}{m}.$$

This is an implicit equation for $\bar{Q}$ which can be easily solved numerically.

The bound is stated in terms of dot products here, so naturally it is possible to kernelize the result using methods from (Herbrich and Graepel, 2001). In kernelized form, the bound applies to classifiers (as output by SVM learning algorithms) of the form:

$$c(x) = \text{sign}\left(\sum_{i=1}^{m} \alpha_i k(x_i,x)\right). \tag{1}$$

Since, by assumption, k is a kernel, we know that $k(x_i,x) = \vec{\Phi}(x_i)\cdot\vec{\Phi}(x)$ where $\vec{\Phi}(x)$ is some projection into another space. In kernelized form, we get $\vec{w}\cdot\vec{x} = \sum_{i=1}^{m} \alpha_i k(x_i,x)$, $\vec{x}\cdot\vec{x} = k(x,x)$, $\vec{w}\cdot\vec{w} = \sum_{i,j}\alpha_i\alpha_j k(x_i,x_j)$, defining all of the necessary quantities to calculate the normalized margin,

$$\gamma(x,y) = \frac{\sum_{i=1}^{m}\alpha_i k(x_i,x)}{\sqrt{k(x,x)\sum_{i,j=1,1}^{m,m}\alpha_i\alpha_j k(x_i,x_j)}}.$$

One element remains, which is the value of μ. Unfortunately the bound can be nonmonotonic in the value of μ, but it turns out that for classifiers learned by support vector machines on reasonable datasets, there is only one value of μ which is (locally, and thus globally) minimal. A binary search over some reasonable range of μ (say from 1 to 100) can find the minima quickly, given the precomputation of the margins. It is worth noting again here that we are not "cheating"—the bound holds for all values of μ simultaneously.

The computational time of the bound calculation is dominated by the calculation of the margins which is $O\left(m^2\right)$ where m is the number of support vectors with a nonzero associated α. This computational time is typically dominated by the time of the SVM learning algorithm.

5.3.1 RESULTS

Application of this bound to support vector machines is of significant importance because SVMs are reasonably effective and adaptable classifiers in common and widespread use. An SVM learns a kernelized classifier as per equation (1).[5]

We apply the support vector machine to 8 UCI database problems chosen to fit the criteria "two classes" and "real valued input features". The problems vary in size over an order of magnitude from 145 to 1428 examples. In Figure 10 we use a 70/30 train/test split of the data.

In all experiments, we use SVMlight (Joachims) with a Gaussian kernel and the default bandwidth. Results for other reasonable choices of the "C", bandwidth,[6] and kernel appear to be qualitatively similar (although of course they differ quantitatively).

It is important to note that the PAC-Bayes margin bound is *not* precisely a bound (or confidence interval) on the true error rate of the learned classifier. Instead, it is a true error rate bound on an

5. Some SVM learning algorithms actually learn a classifier of the form: $c(x) = \text{sign}\left(b + \sum_{i=1}^{m}\alpha_i k(x_i,x)\right)$. We do not handle this form here.
6. Note that the bandwidth of a Gaussian kernel used by an SVM is not directly related to the optimized value of μ we find.

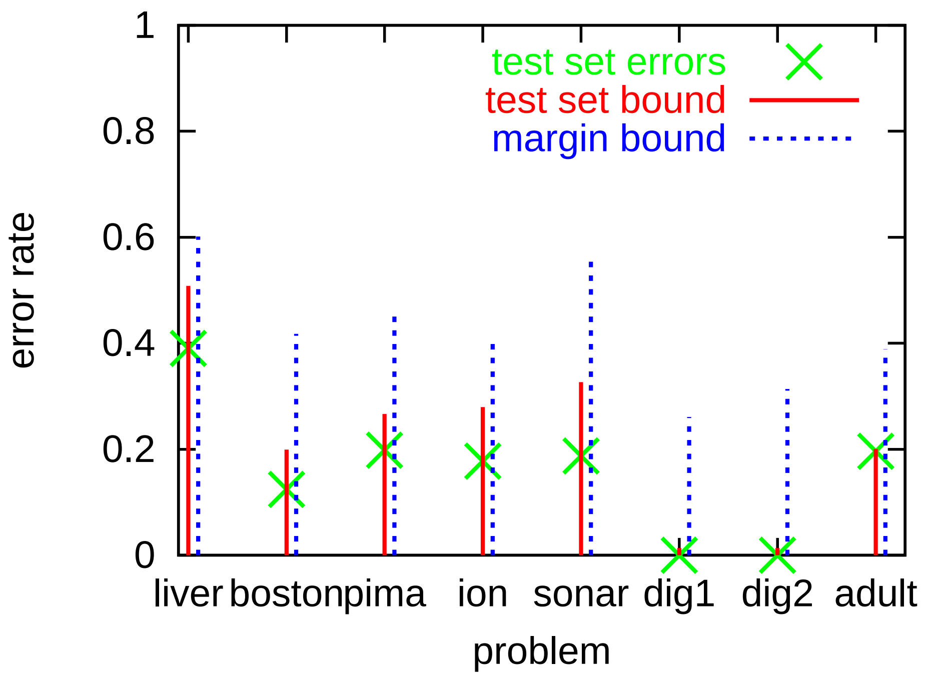


Figure 10: This figure shows the results of applying SVMlight to 8 datasets with a Gaussian kernel and a 70/30 train/test split. The observed test error rate is graphed as an X. On the test set, we calculate a binomial confidence interval (probability of bound failure equals 0.01) which upper bounds the true error rate. On the training set we calculate the PAC-Bayes margin bound for an optimized choice of μ.

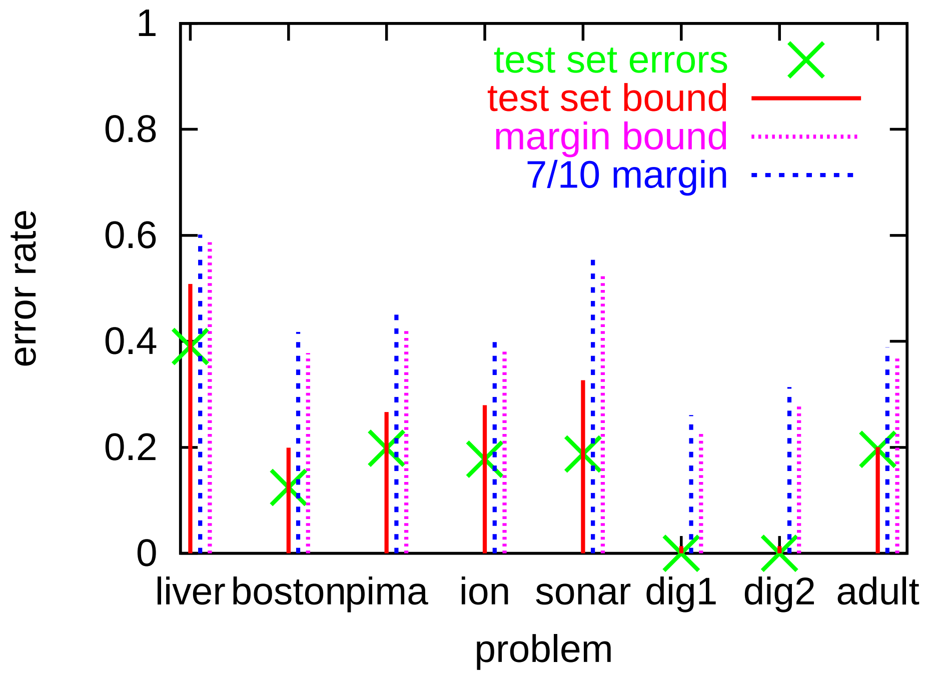


Figure 11: In addition to comparing with everything in Figure 10, we graph the margin bound when *all* of the data is used for the train set. Note that it improves somewhat on the margin bound calculated using the 70% train set (7/10 margin bound), but not enough to compete with the test set bound.

associated stochastic classifier chosen so as to have a similar test error rate. These bounds can be regarded as bounds for the original classifier only under an additional assumption: that picking a classifier according to the majority vote of this stochastic distribution does not worsen the true error rate. This is not true in general, but may be true in practice.

It is of course unfair to compare the train set bound with the test set bound on a 70/30 train/test split because a very tight train set bound would imply that it is unnecessary to even have a test set. In Figure 11 we compare the true error bounds on all of the data to the true error bounds generated from the 70/30 train/test split.

The results show that the PAC-Bayes margin bound is tight enough to give useful information, but still not competitive with the test set bounds. This is in strong contrast with a tradition of quantitatively impractical margin bounds. There are several uses available for bounds which provide some information but which are not fully tight.

1. They might be combined with a train/test bound (Langford, 2002).
2. The train set bound might easily become tighter for smaller sample sizes. This was observed in (Langford, 2002).
3. The train set bound might still have the right "shape" for choosing an optimal parameter setting, such as "C" in a support vector machine.

6. Sample Compression Bound

The sample compression bound (Littlestone and Warmuth), (Floyd and Warmuth, 1995) is like the PAC-Bayes bound in that it applies to arbitrary precision continuous valued classifiers. Unlike the PAC-Bayes bound, it applies meaningfully to nonstochastic classifiers. Mainstream learning algorithms do not optimize the sample compression metric, so the bound application is somewhat rarer. Nonetheless, there do exist some reasonably competitive learning algorithms for which the sample compression bound produces significant results.

The section is organized as follows:

1. Subsection 6.1 states and proves the sample compression bound.
2. Subsection 6.2 shows that the sample compression bound is nearly as tight as possible given the observations.
3. Subsection 6.3 discusses results from the application of the sample compression bound to support vector machines.

6.1 The Sample Compression Bound

The sample compression bound (Littlestone and Warmuth) (Floyd and Warmuth, 1995) stated here differs from older results by generalization and simplification but the bound behavior is qualitatively identical.

Suppose we have a learning algorithm $A(S)$ whose training is "sparse"[7] in the sense that the output classifier is dependent upon only a subset of the data, $A(S) = A(S')$ for $S' \subseteq S$. The sample compression bound is dependent on the errors, $\hat{c}_{S-S'}$ on the subset $S - S'$. The motivation here is that the examples which the learning algorithm does *not* depend upon are "almost" independent and so we can "almost" get a test set bound. In general, the bound becomes tighter as the dependent subset S' becomes smaller and as the error number on the nondependent subset $S - S'$ becomes smaller.

Viewed as an interactive proof of learning (in Figure 12), the sample compression bound is unique amongst training set bounds because it does not require *any* initial commitment to a measure over the classifiers.[8]

Theorem 6.1 *(Sample Compression Bound) For all* $\delta \in (0,1]$*, D, A:*

$$\Pr_{S \sim D^m}\left(\forall S' \subseteq S \text{ with } c = A(S') :\ c_D \le \overline{Bin}\left(m, \hat{c}_{S-S'}, \frac{\delta}{m\binom{m}{|S-S'|}}\right)\right) \ge 1 - \delta.$$

Proof Suppose we knew in advance that the learning algorithm will not depend upon some subset of the examples. Then, the "undependent" subset acts like a test set and gives us a test set bound:

$$\forall S' \subseteq S,\ c = A(S') : \Pr_{S \sim D^m}\left(c_D \le \overline{\text{Bin}}\left(m, \hat{c}_{S-S'}, \frac{\delta}{m\binom{m}{|S-S'|}}\right)\right) \ge 1 - \frac{\delta}{m\binom{m}{|S-S'|}}.$$

7. This is satisfied, for example, by the support vector machine algorithm which only depends upon the set of support vectors.
8. However, we can regard the commitment to a learning algorithm as an implicit commitment to a measure over classifiers which is dependent on the learning algorithm and the distribution generating the data. Viewed from this perspective, the sample compression bound is the Occam's Razor bound, except for the minor detail that the set of evaluating examples varies.

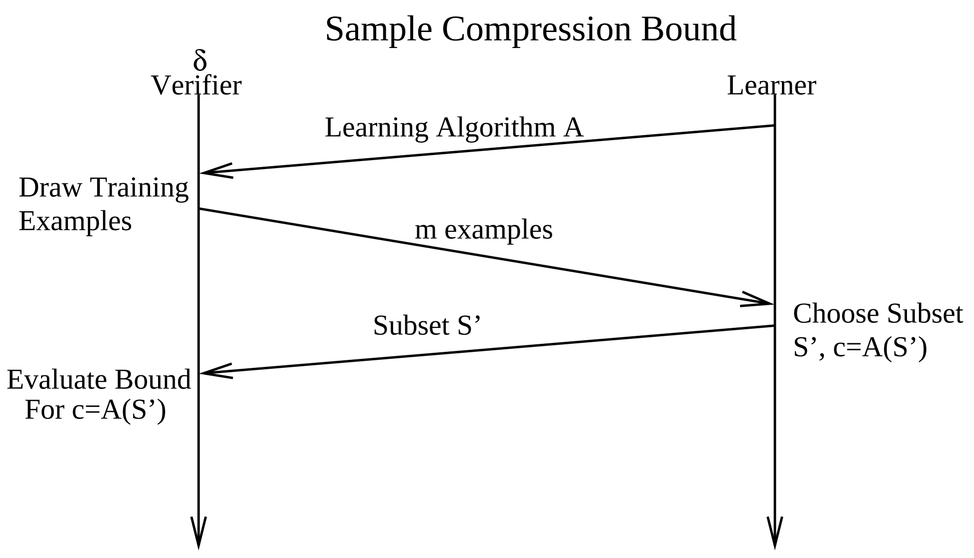


Figure 12: The interactive proof of learning for the sample compression bound. Note that the learning algorithm is arbitrary here, similar to the test set bound.

(Note that, technically, it is possible to refer to S' unambiguously before randomizing over S by specifying the indexes of S contained in S'.) Negating this, we get

$$\forall S' \subseteq S,\ c = A(S') : \Pr_{S\sim D^m}\left(c_D > \overline{\mathrm{Bin}}\left(m, \hat{c}_{S-S'}, \frac{\delta}{m\binom{m}{|S-S'|}} \right) \right) < \frac{\delta}{m\binom{m}{|S-S'|}}$$

and using the union bound ($\Pr(A \text{ or } B) \leq \Pr(A) + \Pr(B)$) over each possible subset, S', we get

$$\Pr_{S\sim D^m}\left(\exists S' \subseteq S \text{ with } c = A(S') :\ c_D > \overline{\mathrm{Bin}}\left(m, \hat{c}_{S-S'}, \frac{\delta}{m\binom{m}{|S-S'|}} \right) \right) < \delta.$$

Negating this again gives us the proof. ■

6.2 The Sample Compression Bound is Sometimes Tight

We can construct a learning algorithm/learning problem pair such that the sample compression bound is provably near optimal, as a function of its observables.

Theorem 6.2 *(Sample Compression Tightness) For all* $\delta \in (0,1]$, *m, k, there exists a distribution D and learning algorithm A s.t.*

$$\Pr_{S\sim D^m}\left(\exists S' \subseteq S \text{ with } c = A(S') :\ c_D > \overline{Bin}\left(m, \hat{c}_{S-S'}, \frac{\delta}{m\binom{m}{|S-S'|}} \right) \right) > \delta - \delta^2.$$

furthermore, if S^* *minimizes* $\overline{Bin}\left(\hat{c}_{S-S'}, \frac{\delta}{m\binom{m}{|S-S'|}}\right)$, *then*

$$\Pr_{S\sim D^m}\left(c^* = A(S^*):\ c^*_D > \overline{Bin}\left(m, \hat{c}^*_{S-S^*}, \frac{\delta}{m\binom{m}{|S-S^*|}}\right)\right) > \delta - \delta^2.$$

Proof The proof is constructive and similar to the Occam's Razor tightness result. In particular, we show how to construct a learning algorithm which outputs classifiers that err independently depending on the subset S' used.

Consider an input space $X = \{0,1\}^{2^m}$. Each variable in the input space $x_{S'}$ can be thought of as indexing a unique subset $S' \subseteq S$ of the examples. In the rest of the proof, we index variables by the subset they correspond to.

Draws from the distribution D can be made by first flipping an unbiased coin to get $y = 1$ with probability 0.5 and $y = -1$ with probability 0.5. The distribution on X consists of a set of independent values after conditioning on y. Choose

$$\Pr(x_{S'} \neq y) = \overline{\mathrm{Bin}}\left(m, k, \frac{\delta}{m\binom{m}{|S-S'|}}\right).$$

Now, the learning algorithm $A(S')$ is very simple—it just outputs the classifier $c(x) = x_{S'}$. On the set $S - S'$, we have

$$\forall S' \Pr_{S\sim D^m}\left(\hat{c}_{S-S'} \geq \frac{k}{m}\right) = 1 - \frac{\delta}{m\binom{m}{|S-S'|}}.$$

Using independence, we get

$$\Pr_{S\sim D^m}\left(\forall S'\ \hat{c}_{S-S'} \geq \frac{k}{m}\right) = \prod_{S'}\left(1 - \frac{\delta}{m\binom{m}{|S-S'|}}\right).$$

Negating, we get

$$\Pr_{S\sim D^m}\left(\forall S'\ \hat{c}_{S-S'} < \frac{k}{m}\right) = 1 - \prod_{S'}\left(1 - \frac{\delta}{m\binom{m}{|S-S'|}}\right)$$

and doing some algebra, we get the result. ■

6.3 Application of the Sample Compression Bound

One obvious application of the sample compression bound is to support vector machines, since the learned classifier is only dependent on the set of support vectors. If S' is the set of support vectors then $S - S'$ is the set of nonsupport vectors. Unfortunately, it turns out that this does not work so well, as observed in Figure 13.

There are other less common learning algorithms for which the sample compression bound works well. The set covering machine (Marchand and Shawe-Taylor, 2001) has an associated bound which is a variant of the sample compression bound.

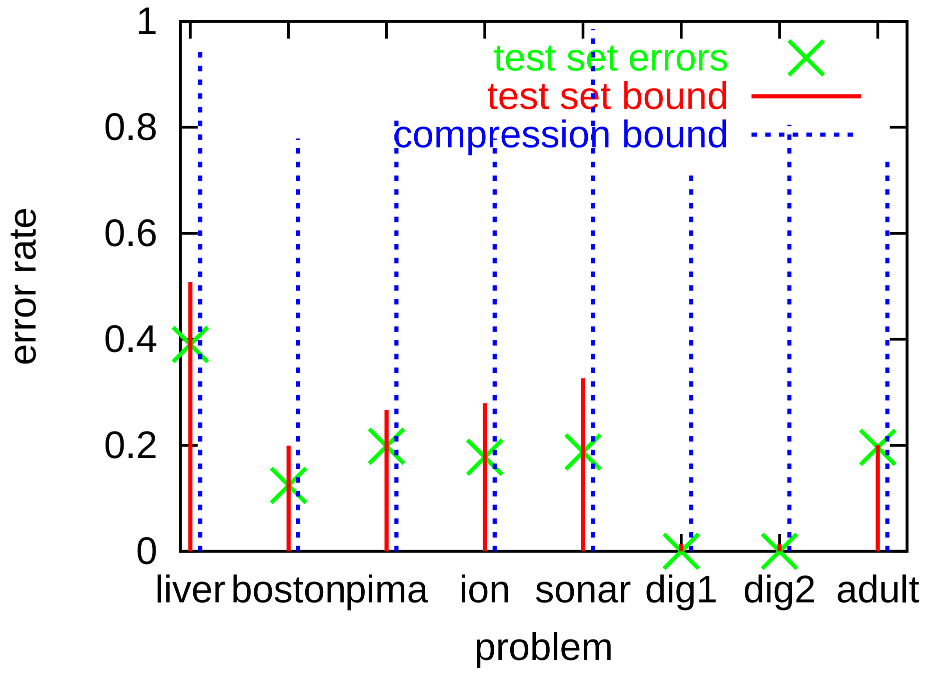


Figure 13: The sample compression bound applied to the output of a support vector machine with a Gaussian kernel. Here we use $\delta = 0.01$

7. Discussion

Here, we discuss several aspects and implications of the presented bounds.

7.1 Learning Algorithm Design

Every train set bound implies a learning algorithm: choose the classifier which minimizes the true error bound. This sounds like a rich source of learning algorithms, but there are some severe caveats to that statement.

1. It is important to note that the form of a train set bound does *not* imply that this minimization is a good idea. Choosing between two classifiers based upon their true error bound implies a better worst-case bound on the true error. It does not imply an improved true error. In many situations, there is some other metric of comparison (such as train error) which in fact creates better behavior.

2. Another strong caveat is that, historically, train set bounds have simply not been tight enough on real datasets for a nonvacuous application. This is changing with new results, but more progress is necessary.

3. Often the optimization problem is simply not very tractable. In addition to sample complexity, learning algorithms must be concerned with run time and space usage.

7.2 Philosophy

Train set bounds teach us about ways in which verifiable learning is possible, a subject which borders on philosophy. The train set bound presented here essentially shows that a reasonable person will be convinced of learning success when a short-description classifier does well on train set data. The results here do *not* imply that this is the only way to convincingly learn. In fact, the (sometimes large) looseness of the Occam's Razor bound suggests that other methods for convincing learning processes exist. This observation is partially shown by the other train set bounds which are presented.

7.3 Conclusion

This introduction to prediction theory covered two styles of bound: the test set bound and the train set bound. There are two important lessons here:

1. Test set bounds provide a better way to report error rates and confidence intervals on future error rates than some current methods.

2. Train set bounds can provide useful information.

It is important to note that the train set bound and test set bound techniques are not mutually exclusive. It is possible to use both simultaneously (Langford, 2002), and doing so is often desirable. Test set bounds are improved by the "free" information about the training error and train set bounds become applicable, even when not always tight.

Acknowledgments

Many people were critical to this tutorial. This includes Sam Roweis who started this, anonymous reviewers who were patient and capable, several coauthors on previous papers, Imre Kondor, Arindam Banerjee, Alina Beygelzimer, Varsha Dani, Tom Hayes, Timothy Ross, and Robert Schapire. I would also like to thank IBM which supported me during much of the preparation of this tutorial.

Appendix A.

For those interested in comparing models, uniform convergence (Vapnik and Chervonenkis, 1971) additionally requires the axiom of choice (to achieve empirical risk minimization) and a hypothesis space of bounded complexity. Typical theorems are of the form "after m examples, all training errors are near to true errors".

The PAC learning model (Valiant, 1984) requires a polynomial time complexity learning algorithm and the assumption that the learning problem comes from some class. Theorems are of the form "after m examples learning will be achieved".

Both of these models can support stronger statements than the basic prediction theory model presented here. Results from both of these models can apply here after appropriate massaging.

The online learning model (Kivinen and Warmuth, 1997) makes *no* assumptions. Typical theorems have the form "This learning algorithm's performance will be nearly as good as anyone of a set of classifiers." The online learning model has very general results and no[9] ability to answer questions about future performance as we address here.

The prediction theory model can most simply be understood as a slight refinement of the information theory model.

References

P. L. Bartlett, O. Bousquet, and S. Mendelson. Local Rademacher complexities. *Annals of Statistics*, 2004.

A. Blum, A. Kalai, and J. Langford. Beating the holdout: Bounds for k-fold and progressive cross-validation. In *Computational Learning Theory (COLT)*, 1999.

A. Blumer, A. Ehrenfueucht, D. Haussler, and M. Warmuth. Occam's razor. *Information Processing Letters*, 24:377–380, 1987.

H. Chernoff. A measure of asymptotic efficiency of tests of a hypothesis based upon the sum of the observations. *Annals of Mathematical Statistics*, 24:493–507, 1952.

C. J. Clopper and E. S. Pearson. The use of confidence intervals for fiducial limits illustrated in the case of the binomial. *Biometrika*, 26:404–413, 1934.

L. Devroye, L. Gyorfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer-Verlag, New York, 1996.

S. Floyd and M. Warmuth. Sample compression, learnability, and the vapnik-chervonenkis dimension. *Machine Learning*, 21:269–304, 1995.

R. Herbrich and T. Graepel. Large scale bayes point machines. In *Advances in Neural Information System Processing 13 (NIPS)*, pages 528–534, 2001.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

T. Joachims. program SVMlight.

J. Kivinen and M. Warmuth. Additive versus exponentiated gradient updates for linear prediction. *Information and Computation*, 132(1):1–64, 1997.

J. Langford. Program **bound**.

J. Langford. Combining train set and test set bounds. In *International Conference on Machine Learning*, 2002.

J. Langford and A. Blum. Microchoice bounds and self bounding learning algorithms. *Machine Learning*, 1999.

9. Note that there do exist online to batch conversions, but these conversions always occur under an assumption of i.i.d. samples, essentially changing the setting to the one described here.

J. Langford and D. McAllester. Computable shell decomposition bounds. In *Computational Learning Theory (COLT)*, 2000.

J. Langford and M. Seeger. Bounds for averaging classifiers. Technical report, Carnegie Mellon, Department of Computer Science, 2001.

J. Langford and J. Shawe-Taylor. PAC-Bayes & margins. In *Neural Information Processing Systems (NIPS)*, 2002.

N. Littlestone and M. Warmuth. Relating data compression and learnability.

M. Marchand and J. Shawe-Taylor. The set covering machine. In *International Conference on Machine Learning (ICML)*, 2001.

D. McAllester. PAC-Bayesian model averaging. In *Computational Learning Theory (COLT)*, 1999.

M. Seeger. PAC-Bayesian generalization error bounds for gaussian process classification. *Journal of Machine Learning Research*, 3:233–269, 2002.

S. Seung. Unpublished notes.

J. Shawe-Taylor, P. Bartlett, R. Williamson, and M. Anthony. Structural risk minimization over data-dependent hierarchies. *IEEE Transactions on Information Theory*, 44(5):1926–1940, 1998.

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

V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.

Journal of Machine Learning Research 6 (2005) 307–340 Submitted 6/04; Revised 12/04; Published 3/05

Generalization Bounds and Complexities Based on Sparsity and Clustering for Convex Combinations of Functions from Random Classes

Savina Andonova Jaeger JAEGER@HCP.MED.HARVARD.EDU
Harvard Medical School
Harvard University
Boston, MA 02115, USA

Editor: John Shawe-Taylor

Abstract

A unified approach is taken for deriving new generalization data dependent bounds for several classes of algorithms explored in the existing literature by different approaches. This unified approach is based on an extension of Vapnik's inequality for VC classes of sets to random classes of sets - that is, classes depending on the random data, invariant under permutation of the data and possessing the increasing property. Generalization bounds are derived for convex combinations of functions from random classes with certain properties. Algorithms, such as SVMs (support vector machines), boosting with decision stumps, radial basis function networks, some hierarchies of kernel machines or convex combinations of indicator functions over sets with finite VC dimension, generate classifier functions that fall into the above category. We also explore the individual complexities of the classifiers, such as sparsity of weights and weighted variance over clusters from the convex combination introduced by Koltchinskii and Panchenko (2004), and show sparsity-type and cluster-variance-type generalization bounds for random classes.

Keywords: complexities of classifiers, generalization bounds, SVM, voting classifiers, random classes

1. Introduction

Statistical learning theory explores ways of estimating functional dependency from a given collection of data. It, also referred to as the theory of finite samples, does not rely on a priori knowledge about a problem to be solved. Note that "to control the generalization in the framework of this paradigm, one has to take into account two factors, namely, the quality of approximation of given data by the chosen function and the capacity of the subset of functions from which the approximating function was chosen" (Vapnik, 1998). Typical measures of the capacity of sets of functions are entropy measures, VC-dimensions and V-γ dimensions. Generalization inequalities such as Vapnik's inequalities for VC-classes, which assert the generalization performance of learners from *fixed* class of functions and take into account the quality of approximation of given data by the chosen function and the capacity of the class of functions, were proven to be useful in building successful learning algorithms such as SVMs (Vapnik, 1998).

An extension of Vapnik's inequality, for VC classes of sets (Vapnik, 1998; Anthony and Shawe-Taylor, 1993) and VC-major classes of functions to classes of functions satisfying Dudley's uniform entropy conditions, was shown by Panchenko (2002). A class of functions $\mathcal{F} = \{f : \mathcal{X} \to [-1,1]\}$

satisfies Dudley's uniform entropy condition if

$$\int_0^\infty \log^{1/2} D(\mathcal{F},u)du < \infty,$$

where $D(\mathcal{F},u)$ denotes Koltchinksii packing numbers defined for example by Dudley (1999) or Panchenko (2002). Applications of the inequality were shown in several papers (Koltchinskii and Panchenko, 2002; Koltchinskii et al., 2003a; Koltchinskii and Panchenko, 2004) which explored the generalization ability of ensemble classification methods, that is, learning algorithms that combine several classifiers into new voting classifiers with better performance. "The study of the convex hull, $\text{conv}(\mathcal{H})$, of a given base function class $\mathcal{H}$ has become an important object of study in machine learning literature" (Koltchinskii and Panchenko, 2004). New measures of individual complexities of voting classifiers derived in related work (Koltchinskii et al., 2003a; Koltchinskii and Panchenko, 2004; Koltchinskii et al., 2003b) were shown theoretically and experimentally to play an important role in the generalization performance of the classifiers from $\text{conv}(\mathcal{H})$ of a given base function class $\mathcal{H}$. In order to do so, the base class $\mathcal{H}$ is assumed to have Koltchinskii packing numbers satisfying the following condition

$$D(\mathcal{H},u) \le K(V)u^{-V},$$

for some $V > 0$, and where K depends only on V. "New margin type bounds that are based to a greater extent on complexity measures of individual classifier functions from the convex hull, are more adaptive and more flexible than previously shown bounds" (Koltchinskii and Panchenko, 2004).

Here, we are interested in studying the generalization performance of functions from a convex hull of *random* class of functions (random convex hull), that is, the class of learners is no longer fixed and depends on the data. This is done by deriving a new version of Vapnik's inequality applied to random classes, that is, a bound for relative deviations of frequencies from probabilities for random classes of events. The proof of the inequality mirrors the proofs of Vapnik's inequality for non-random classes of sets (see Vapnik et al., 1974; Vapnik, 1998; Anthony and Shawe-Taylor, 1993) but with the observation that the symmetrization step of the proof can be carried out for random classes of sets. The new version of Vapnik's inequality is then applied to derive flexible and adaptive bounds on the generalization errors of learners from random convex hulls. We exploit techniques previously used in deriving generalization bounds for convex combinations of functions from non-random classes in (Koltchinskii and Panchenko, 2004), and several measures of individual classifier complexities, such as effective dimension, pointwise variance and weighted variance over clusters, similar to the measures introduced by Koltchinskii and Panchenko (2004).

Surprisingly, the idea of studying random convex hulls allows one simultaneously to prove generalization results, and incorporate measures of classifier complexities in the bounds, for several existing algorithms such as SVMs, boosting with decision stumps, radial basis function networks and combinations of indicator functions over sets with finite VC dimension. It is also noteworthy that an extension on the VC theory of statistical learning to data dependent spaces of classifiers was recently found by Cannon et al., 2002, who defined a measure of complexity for data dependent hypothesis classes and provide data dependent versions of bounds on error deviance and estimation error.

2. Definition of Random Classes

First, an inequality that concerns the uniform relative deviation over a random class of events of relative frequencies from probabilities is exhibited. This inequality is an extension of the following Vapnik's inequality for a fixed VC-class $\mathcal{C}$ (with finite VC-dimension V) of sets (see Vapnik et al. (1974); Vapnik (1998); Anthony and Shawe-Taylor (1993)):

$$\mathbb{P}^n\Big(\sup_{C\in\mathcal{C}}\Big[\frac{\mathbb{P}(C)-\frac{1}{n}\sum_{i=1}^n I(x_i\in C)}{\sqrt{(\mathbb{P}(C))}}\Big]\geq t\Big)\leq 4\Big(\frac{2en}{V}\Big)^V e^{-\frac{nt^2}{4}}. \tag{2.1}$$

Inequality (2.1) allows one to prove stronger generalization results for several problems discussed in (Vapnik, 1998). In order to extend the above inequality to random classes of sets, we introduce the following definitions. Let $(\mathcal{Z},\mathcal{S},\mathbb{P})$ be a probability space. For a sample $\{z_1,\dots,z_n\}$, $z_i\in\mathcal{Z}$, $i=1,\dots,n$, define $z^n=(z_1,\dots,z_n)$ and let $I(z^n)=\{z_i:1\leq i\leq n\}$. Let $\mathcal{C}(z^n)\in\mathcal{S}$ be a class of sets, possibly dependent on the sample $z^n=(z_1,\dots,z_n)\in\mathcal{Z}^n$.
The integer $\Delta_{\mathcal{C}(z^n)}(z^n)$ is defined to be the number of distinct sets of the form $A\bigcap I(z^n)$, where A runs through $\mathcal{C}(z^n)$, that is, $\Delta_{\mathcal{C}(z^n)}(z^n)=\text{card}\{A\bigcap\{z_1,\dots,z_n\},A\in\mathcal{C}(z^n)\}$. The random collection of level sets $\mathcal{C}(z^n)=\Big\{A=\{z\in\mathcal{Z}:h(z)\leq 0\},h\in\mathcal{H}(z_1,\dots,z_n)\Big\}$, where $\mathcal{H}(z^n)$ is a random class of functions possibly depending on z^n serves as a useful example. We call $A\bigcap I(z^n)$ a representation of the sample z^n by the set A. $\Delta_{\mathcal{C}(z^n)}(z^n)$ is the number of different representation of $\{z_1,\dots,z_n\}$ by functions from $\mathcal{H}(z^n)$.

Now consider the random collection $\mathcal{C}(z^n)$ of $\mathcal{S}$-measurable subsets of $\mathcal{Z}$,

$$\mathcal{C}(z^n)=\{A:A\in\mathcal{S}\},$$

having the following properties:

$$1.)\ \mathcal{C}(z^n)\subseteq\mathcal{C}\Big(z^n\bigcup y\Big),\ z^n\in\mathcal{Z}^n, y\in\mathcal{Z} \tag{2.2}$$

(*the incremental property*)

$$2.)\ \mathcal{C}(z_{\pi(1)},\dots,z_{\pi(n)})\equiv\mathcal{C}(z_1,\dots,z_n), \tag{2.3}$$

for any permutation π of $\{z_1,\dots,z_n\}$ (*the permutation property*).

The relative frequency of $A\in\mathcal{C}(z^n)$ on $z^n=(z_1,\dots,z_n)\in\mathcal{Z}^n$ is defined to be

$$\mathbb{P}_{z^n}(A)=\frac{1}{n}|\{i:z_i\in A\}|=\frac{1}{n}|I(z^n)\cap A|,$$

where $|A|$ denotes the cardinality of a set A.

Let $\mathbb{P}^n$ be the product measure on n copies of $(\mathcal{Z},\mathcal{S},\mathbb{P})$, and $\mathbb{E}_n$ the expectation with respect to $\mathbb{P}^n$. Define

$$G(n)=\mathbb{E}_n\Delta_{\mathcal{C}(z^n)}(z^n).$$

3. Main Results

Given the above definitions, the following theorem holds.

Theorem 1 *For any $t > 0$,*

$$\mathbb{P}^n \left\{ z^n \in \mathcal{Z}^n : \sup_{A \in \mathcal{C}(z^n)} \frac{\mathbb{P}(A) - \mathbb{P}_{z^n}(A)}{\sqrt{\mathbb{P}(A)}} \geq t \right\} \leq 4G(2n)e^{-\frac{nt^2}{4}}. \tag{3.4}$$

The proof of this theorem is given in the following Section 4. Observe that if the random collection $\mathcal{C}$ of sets is a VC-class (Vapnik, 1998), then the inequality (3.4) is the same as Vapnik's inequality (2.1) for VC-classes. Based on this theorem and the above definitions, several results on the generalization performance and the complexity of classifiers from random classes are exhibited below.

The following notation and definitions will be used from here on. Let $(\mathcal{X}, \mathcal{A})$ be a measurable space (space of instances) and take $\mathcal{Y} = \{-1, 1\}$ to be the set of labels. Let $\mathbb{P}$ be the probability measure on $\left(\mathcal{X} \times \mathcal{Y}, \mathcal{A} \times 2^{\{-1,1\}}\right)$ and let $(X_i, Y_i), i = 1, \ldots, n$ be i.i.d random pairs in $\mathcal{X} \times \mathcal{Y}$, randomly sampled with respect to the distribution $\mathbb{P}$ of a random variable (X, Y). The probability measure on the main sample space on which all of the random variables are defined will be denoted by P. Let $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, $Z_i = (X_i, Y_i), i = 1, \ldots, n$ and $Z^n = (Z_1, \ldots, Z_n)$. We will also define several random classes of functions and show how several learning algorithms generate functions from the convex hulls of random classes.

Consider the following four problems for which bounds on the generalization errors will be shown using inequality (3.4).

Problem 1. *Support vector machine (SVM) classifiers with uniformly bounded kernels.*

Consider any solution of an SVM algorithm $f(x) = \sum_{i=1}^{n} \lambda_i Y_i K(X_i, x)$, where $K(.,.) : \mathcal{X} \times \mathcal{X} \to [0, 1]$ is the kernel and $\lambda_i \geq 0$. $\text{sign}(f(x))$ is used to classify $x \in \mathcal{X}$ in class $+1$ or -1. Take the random function class

$$\mathcal{H}(Z^n) = \{Y_i K(X_i, x) : i = 1, \ldots, n\},$$

which depends on the random sample $Z^n \in \mathcal{Z}^n$. The classifier function

$$f'(x) = \sum_{i=1}^{n} \lambda'_i Y_i K(X_i, x),\ \lambda'_i = \frac{\lambda_i}{\sum_{j=1}^{n} \lambda_j},\ i = 1, \ldots, n$$

belongs to $\text{conv}(\mathcal{H}(Z^n))$ and the probability of error $\mathbb{P}(Y f(X) \leq 0) = \mathbb{P}(Y f'(X) \leq 0)$.

Problem 2. *Classifiers, built by some two-level component based hierarchies of SVMs (Heisele et al. (2001);Andonova (2004)) or kernel-based classifiers (like the one produced by radial basis function (RBF) networks).*

We explore component based hierarchies, such that the first level of the hierarchy is formed by SVM classifiers (with kernel K) built on each component (formed for example by projecting of the input space $\mathcal{X} \subseteq \mathbb{R}^m$ of instances onto subspace of $\mathbb{R}^l$, $l < m$) and the second level of the hierarchy is a linear combination of the real-valued outputs on each component of the classifier functions from the first level (for example, applying SVM with linear kernel or boosting methods on the output from the first level). In our formulation, the components of the hierarchy can depend on the training data (for example, found through dimensionality reduction algorithms, such as self-organizing maps (SOM, Kohonen (1990))). The type of the hierarchical classifier functions are of this form $\mathrm{sign}(f(x))$, where

$$f(x,\alpha,Q,w_2)=\sum_{j=1}^{d}w_2^j\sum_{i=1}^{n}\alpha_i^j Y_i K(Q^j X_i,Q^j x), Y_i=\pm 1,$$

where Q^j are the projections of the instances (determining the “components”), $w_2^j\in\mathbb{R},\alpha_i^j\geq 0$. One can consider Q^j being nonlinear transformation of the instance space, for example applying filter functions. Let $|K(x,t)|\leq 1,\forall x,t\in\mathcal{X}$. Consider the random function class

$$\mathcal{H}(X_1,\dots,X_n)=\{\pm K(Q^j X_i,Q^j x): i\leq n, j=1,\dots,d\},$$

where n is the number of training points (X_i,Y_i) and d is the number of the components.

In the case of RBF networks with one hidden layer and a linear threshold, the classifier function is of the form

$$f(x)=\sum_{j=1}^{d}\sum_{i=1}^{\hat{n}}\alpha_j^i K_{\sigma_j}(c_i,x),$$

where $c_i, i=1,\dots,\hat{n}$ are centers of clusters, formed by clustering the training points $\{X_1,\dots,X_n\}$ and σ_j (they can depend on the training data $(X_i,Y_i), i=1,\dots,n$) are different widths for the Gaussian kernel, $K_{\sigma_j}(c_i,x)=e^{-\frac{\|c_i-x\|^2}{\sigma_j^2}}$. Consider the following random function class

$$\mathcal{H}(Z^n)=\{\pm K_{\sigma_j}(c_i,x): i\leq\hat{n}, j\leq d\},$$

where $\hat{n}$ is the number of clusters, which is bounded by the number n of training points, and the cluster centers $\{c_i\}_{i=1}^{\hat{n}}$ depend on the training instances $\{X_i\}_{i=1}^{n}$.

Without loss of generality, we can consider $f\in\mathrm{conv}(\mathcal{H}(Z^n))$ in both of the above described algorithms, after normalizing the classifier function with the sum of the absolute values of the coefficients in front of the random functions.

Problem 3. *Boosting over decision stumps.*

Given a finite set of d functions $\{h_i:\mathcal{X}\times\mathcal{X}\to[-1,1]\}$ for $i\leq d$, define the random class of as $\mathcal{H}(X_1,\dots,X_n)=\{h_i(X_j,x): j\leq n, i\leq d\}$, where n is the number of training points (X_i,Y_i). This type of random class is used for example in aggregating combined classifier by boosting over decision stumps. Indeed, decision stumps are simple classifiers, h, of the types $2I(x^i\leq a)-1$ or $2I(x^i\geq a)-1$, where $i\in\{1,\dots,m\}$ is the direction of splitting ($\mathcal{X}\subset\mathbb{R}^m$) and $a\in\mathbb{R}$ is the threshold. It is clear that the threshold a can be chosen among $X_1^i,\dots,X_n^i$ (the performance of the stump will remain the same on the training data). In this case, take $h_i(X_j,x)=2I(x^i\leq X_j^i)-1$ or $\widetilde{h}_i(X_j,x)=2I(x^i\geq X_j^i)-1$ and $\mathcal{H}(X_1,\dots,X_n)=\{h_i(X_j,x),\widetilde{h}_i(X_j,x): j\leq n, i\leq m\}$ and take $d=2m$.

Without loss of generality, we can consider $f\in\mathrm{conv}(\mathcal{H}(X_1,\dots,X_n))$, after normalizing with the sum of the absolute values of the coefficients.

Problem 4. *VC-classes of sets.*

Let the random class of functions $\mathcal{H}(Z^n)$ has the property that for all $h\in\mathcal{H}(Z^n)$, $h\in\{-1,1\}$ the VC dimension V of the class of sets $\{\{x\in\mathcal{X}: h(x)=1\}, h\in\mathcal{H}(Z^n)\}$ is finite.

A classifier is formed by taking convex combinations of functions from the class $\mathcal{H}(Z^n)$. Problem 4, in the case when the class $\mathcal{H}$ is not depending on the random sample Z^n, was approached before with the previously existing VC-inequalities for indicator functions (Vapnik, 1998; Vapnik et al., 1968). The results shown here for Problem 4 in the case when $\mathcal{H}$ is a random class, are comparable to those derived before for indicators over class of sets with finite VC dimension.

In *general*, all of the above four types of problems, consider the convex combinations of functions from the random convex hull

$$\mathcal{F}(Z^n) = \operatorname{conv}(\mathcal{H}(Z^n)) = \bigcup_{T \in \mathbb{N}} \mathcal{F}_T(Z^n),$$
$$\mathcal{F}_T(Z^n) = \left\{ \Sigma_{i=1}^T \lambda_i h_i, \lambda_i \geq 0, \Sigma_{i=1}^T \lambda_i = 1, h_i \in \mathcal{H}(Z^n) \right\}, \tag{3.5}$$

where $\mathcal{H}(Z^n)$ is for example one of the random classes defined above, such that either $|\mathcal{H}(Z^n)| = \operatorname{card}(\mathcal{H}(Z^n))$ is finite, or $\mathcal{H}(Z^n)$ is a collection of indicators over random class of sets with finite VC-dimension.

General problem:

We are interested is the following general problem:

Let $\mathcal{H}$ be a general-base class of uniformly bounded functions with values in $[-1,1]$. Let $Z_1,\ldots,Z_n, Z_i = (X_i,Y_i) \in \mathcal{X} \times \mathcal{Y}$ be i.i.d. (training data) sampled with respect to the distribution $\mathbb{P}$. Assume that based on the data $Z_1,\ldots,Z_n$ one selects a class of functions $\mathcal{H}(Z^n) \subseteq \mathcal{H}$ that is either

i) with *finite cardinality* depending on the data, such that
$\frac{\ln(\sup_{Z^n} |\mathcal{H}(Z^n)|)\ln n}{n} \to 0$ for $n \to \infty$, or

ii) $\mathcal{H}(Z^n) \subseteq \mathcal{H}$ is a collection of indicator functions $\{2I_C - 1 : C \in \mathcal{C}_{Z^n}\}$ over a class of sets $\mathcal{C}_{Z^n}$ with *finite VC-dimension* V.

We will call $\mathcal{H}(Z^n)$ a *random-base class* of functions. We are interested in studying the generalization errors for classifier functions $f \in \operatorname{conv}(\mathcal{H}(Z^n))$ that are produced by broad classes of algorithms. Let us take

$$G^*(n,\mathcal{H}) = \sup_{Z^n \in (\mathcal{X}\times\mathcal{Y})^n} |\mathcal{H}(Z^n)|,$$

when $\mathcal{H}$ is the general-base class and the random-base classes $\mathcal{H}(Z^n)$ are with finite cardinality H(Z^n), and take

$$G^*(n,\mathcal{H}) = \left(\frac{ne}{V}\right)^V,$$

when $\mathcal{H}$ is the general-base and the random-base classes $\mathcal{H}(Z^n)$ are collections of indicators over class of sets with finite VC-dimension V (Problem 4).

From the definitions and Problems 1, 2 and 3, it is clear that $G^*(n,\mathcal{H}) \leq 2n$ for Problem 1 and $G^*(n,\mathcal{H}) \leq 2nd$ for Problems 2 and 3. For completeness of the results in case of $\mathcal{H}(Z^n)$ being a collection of indicators over class of sets with finite VC-dimension V, we will assume that $n \geq \frac{V}{2e}$.

Following the notation by Koltchinskii and Panchenko (2004), let $\mathcal{P}(\mathcal{H}(Z^n))$ be the collection of all discrete distributions over the random-base class $\mathcal{H}(Z^n)$. Let $\lambda \in \mathcal{P}(\mathcal{H}(Z^n))$ and $f(x) = \int h(x)\lambda(dh)$, which is equivalent to $f \in \operatorname{conv}(\mathcal{H}(Z^n))$. The generalization error of any classifier function is defined as

$$\mathbb{P}(\operatorname{sign}(f(x)) \neq y) = \mathbb{P}(yf(x) \leq 0) = \mathbb{E}(I(yf(x) \leq 0)).$$

Given an i.i.d sample $(X_i,Y_i), i=1,\dots n$ from the distribution $\mathbb{P}$, let $\mathbb{P}_n$ denote the empirical distribution and for any measurable function g on $\mathcal{X}\times\mathcal{Y}$, let

$$\mathbb{P}g=\int g(x,y)d\mathbb{P}(x,y),\ \ \mathbb{P}_n g=\frac{1}{n}\sum_{i=1}^{n} g(X_i,Y_i).$$

The first bound we show for the generalization errors of functions from random classes of functions is the following:

Theorem 2 *Let $\mathcal{H}$ be a general-base class of functions. For any $t>0$, with probability at least $1-e^{-t}$, for any n i.i.d. random pairs $(X_1,Y_1),\dots,(X_n,Y_n)$ randomly drawn with respect to the distribution $\mathbb{P}$, for all $\lambda\in\mathcal{P}(\mathcal{H}(Z^n))$ and $f(x)=\int h(x)\lambda(dh)$,*

$$\mathbb{P}(yf(x)\le 0)\le \inf_{0<\delta\le 1}\left(U^{\frac{1}{2}}+\left(\mathbb{P}_n(yf(x)\le 2\delta)+\frac{1}{n}+U\right)^{\frac{1}{2}}\right)^2+\frac{1}{n}, \tag{3.6}$$

where

$$U=\frac{1}{n}\left(t+\ln\frac{4}{\delta}+\frac{8\ln n}{\delta^2}\ln G^*(2n,\mathcal{H})+\ln(8n+4)\right).$$

The proof of this theorem is given in Section 4. It is based on random approximation of a function and Hoeffding-Černoff inequality as in (Koltchinskii and Panchenko, 2004), exploring the properties of random class of the level sets of the margins of the approximating functions, defined in the proof and Inequality (3.4).

The first result for the generalization error of classifiers from $\operatorname{conv}(\mathcal{H})$, where $\mathcal{H}$ is a fixed VC-class, was achieved by Schapire et al. (1998). They explained the generalization ability of classifiers from $\operatorname{conv}(\mathcal{H})$ in terms of the empirical distribution $\mathbb{P}_n(yf(x)\le\delta)$, $f\in\operatorname{conv}(\mathcal{H})$ of the quantity $yf(x)$, known in the literature as *margin* ("confidence" of prediction of the example x) and proposed several boosting algorithms that are built on the idea of maximizing the margin. Important properties, development, improvements and optimality of the generalization results of this type for broader fixed classes of functions $\mathcal{H}$ were shown by Koltchinskii and Panchenko (2004). The bound on the generalization error shown here is valid for random classes of functions and is not optimized for convergence with respect to n. Here, we have a different goal: to prove generalization results for random classes of functions that relate to broader classes of algorithms. Exploring the optimality of this result remains an open question.

In the rest of the paper, we will explore the individual complexity of classifier $f\in\operatorname{conv}(\mathcal{H})$, following the line of investigation begun by Koltchinskii and Panchenko (2004). We will explore the structure of the *random convex hull* and show bounds similar to the ones by Koltchinskii and Panchenko (2004) that reflect some measures of complexity of convex combinations.

First, we explore how the sparsity of the weights of a function from a random convex hull influences the generalization performance of the convex combination. Here, recall from Koltchinskii and Panchenko (2004), by sparsity of the weights in convex combination, we mean rapidity of decrease. For $\delta>0$ and $f(x)=\sum_{i=1}^{T}\lambda_i h_i(x), \sum_i\lambda_i=1, \lambda_i\ge 0$, let us define the *dimension* function by

$$e_n(f,\delta)=\left(T-\sum_{k=1}^{T}(1-\lambda_k)^{\frac{8\ln n}{\delta^2}}\right). \tag{3.7}$$

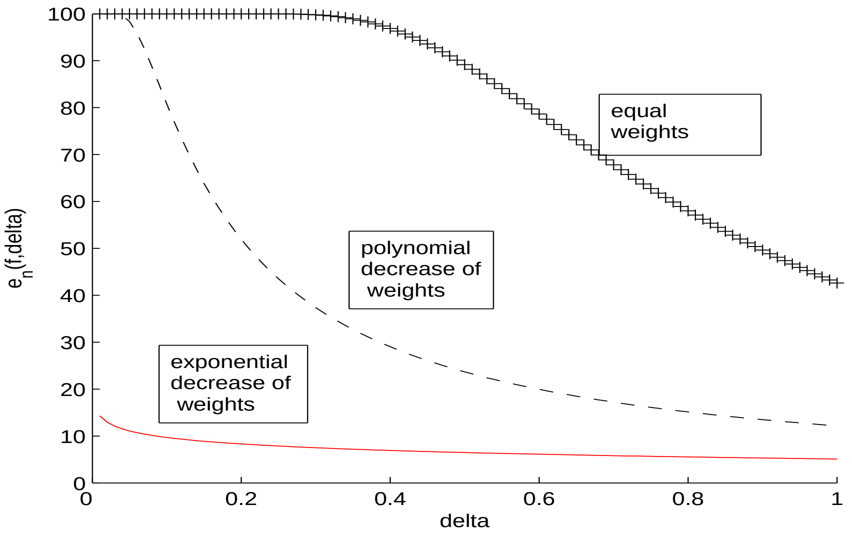


Figure 1: Dimension Function; From top to bottom: equal,polynomial, exponential decay; the x-axis is δ, the y-axis is dimension function value.

The name of this function is motivated by the fact that it can be interpreted as a dimension of a subset of the random convex hull $\mathrm{conv}(\mathcal{H})$ containing a function approximating f "well enough" (Koltchinskii and Panchenko, 2004). In a way, the dimension function measures the sparsity of the weights in the convex combination f. We plot the dimension function (see Fig. 1) in the cases when $T = 100, n = 1000$ and the weights $\{\lambda_i\}_{i=1}^T$ are equal, polynomially decreasing ($\lambda_i = i^{-2}$) and exponentially decreasing ($\lambda_i = e^{(-i+1)}$). One can see in Fig. 1 that when the weights decrease faster, the dimension function values are uniformly smaller with respect to $\delta \in (0,1]$. (For different sparsity measures of voting classifiers from convex hulls of VC-major classes see (Koltchinskii et al. (2003a); Koltchinskii and Panchenko (2004); Andonova (2004).)

Theorem 3 *(Sparsity bound) For any $t > 0$, with probability at least $1-e^{-t}$, for any n i.i.d. random pairs $(X_1,Y_1),\ldots,(X_n,Y_n)$ randomly drawn with respect to the distribution $\mathbb{P}$, for all $\lambda \in \mathcal{P}(\mathcal{H}(Z^n))$ and $f(x) = \sum_{i=1}^T \lambda_i h_i(x)$,*

$$\mathbb{P}(yf(x) \le 0) \le \inf_{0<\delta\le 1}\Big(U^{1/2} + (\mathbb{P}_n(yf(x) \le 2\delta) + U + \frac{1}{n})^{1/2}\Big)^2 + \frac{1}{n}, \tag{3.8}$$

where

$$U = \frac{1}{n}\Big(t + \ln\frac{4}{\delta} + e_n(f,\delta)\ln G^*(2n,\mathcal{H}) + e_n(f,\delta)\ln(\frac{8}{\delta^2}\ln n) + \ln(8n+4)\Big).$$

The proof of this theorem is also shown in Section 4. It is based on random approximation of functions similarly to the proof of Theorem 2, Hoeffding-Černoff inequality, properties of conditional expectation, exploring the capacity of random class of the level sets of the margins of the approximating functions and Inequality (3.4). The constants are explicit. For many experimental results, showing the behavior of the above bound in the case of convex combinations of decision stumps, such as those produced by various boosting algorithms (see Andonova, 2004). There, it is shown experimentally that the sparsity bound is indicative of the generalization performance of a classifier from the convex hull of stumps. For further results and new algorithms, utilizing various notions of sparsity of the weight of convex combinations (see Koltchinskii et al., 2003a). In the case of hard margin SVM classifiers $f(x) = \sum_{i=1}^{T} \lambda_i K(X_i, x)$ with uniformly bounded kernels, the bound with margin δ becomes of order

$$\mathbb{P}(yf(x) \leq 0) \preceq \frac{\min(T, 8\delta^{-2} \ln n)}{n} \ln \frac{n}{\delta},$$

because

$$e_n(f, \delta) \leq \min(T, 8\delta^{-2} \ln n).$$

The inequality for $e_n(f, \delta)$ follows from the inequality $(1-\lambda)^p \geq 1 - p\lambda$ for $\lambda \in [0,1]$ and $p \geq 1$, and the fact that $\sum_{k=1}^{T} \lambda_k \leq 1$.

This zero-error bound is comparable to the compression bound (Littlestone and Warmuth, 1986) of order $\frac{T}{n-T} \ln \frac{n}{T}$, and the bounds of Bartlett and Shawe-Taylor (1999), where $U \sim \frac{R^2}{n\delta^2} \ln^2 n$ and $R \leq 1$ in case of $K(x,y) \leq 1$. When $T \ll n$ the bound in (3.8) is an improvement of the last bound. For example, $T \ll n$ when SVMs produce very "sparse" solutions (small number of support vectors), that is, the vector of weights $(\lambda_1, \ldots, \lambda_T)$ is sparse. The sparseness (in the sense of there being a small number of support vectors) of the solutions of SVMs was recently explored by Steinwart (2003), where lower bounds (of order $O(n)$) on the number T of support vectors for specific types of kernels were shown; in those cases, the bound in (3.8), relaxed to the upper bound of $e_n(f, \delta) \leq \min(T, 8\delta^{-2} \ln n)$, is probably not a significant improvement of the result of Bartlett and Shawe-Taylor (1999). The sparsity of weights of the solutions of SVMs, understood as rapidity of decrease of weights, is in need of further exploration, as it would provide better insight into the bound (3.8) of the generalizations error.

We now notice also that, because $e_n(f, \delta) \leq \min(T, 8\delta^{-2} \ln n)$ and $G^*(n, \mathcal{H}) \leq 2n$ for Problem 1 and $G^*(n, \mathcal{H}) \leq 2nd$ for Problems 2, 3 and $G^*(n, \mathcal{H}) = \left(\frac{ne}{V}\right)^V$, the bound (3.8) is extension of the results of Breiman (1999) for zero-error case, and is similar in nature to the result of Koltchinskii and Panchenko (2004) and Koltchinskii et al. (2003b), but now holding for *random* classes of functions.

Motivations for considering different bounds on the generalization error of classifiers that take into account measures of closeness of random functions in convex combinations and their clustering properties were given by Koltchinskii and Panchenko (2004). We now review those complexities and show bounds on the generalization error, that are similar to the ones proven by Koltchinskii and Panchenko (2004), but applied for different classes of functions. The proofs of the results are similar to those exhibited by Koltchinskii and Panchenko (2004).

Recall that a pointwise variance of h with respect to the distribution $\lambda \in \mathcal{P}(\mathcal{H}(Z^n))$ is defined by

$$\sigma_\lambda^2(x) = \int \left(h(x) - \int h(x) \lambda(dh) \right)^2 \lambda(dh), \tag{3.9}$$

where, $\sigma_\lambda^2(x) = 0$ if and only if $h_1(x) = h_2(x)$ for all $h_1, h_2 \in \mathcal{H}(Z^n)$ (Koltchinskii and Panchenko, 2004). The following theorem holds:

Theorem 4 *For any $t > 0$ with probability at least $1 - e^{-t}$, for any n i.i.d. random pairs $(X_1, Y_1), \ldots, (X_n, Y_n)$ randomly drawn with respect to the distribution $\mathbb{P}$, for all $\lambda \in \mathcal{P}(\mathcal{H}(Z^n))$ and $f(x) = \int h(x) \lambda(dh)$,*

$$\begin{aligned}\mathbb{P}(yf(x)\le 0) \quad &\le \quad \inf_{0<\delta\le\gamma\le 1}\Big(2\mathbb{P}_n(yf(x)\le 2\delta)+4\mathbb{P}_n(\sigma^2_\lambda(x)\ge\frac{\gamma}{3})+ \\ &+ \quad \frac{8}{n}\left(\frac{56\gamma}{\delta^2}(\ln n)\ln G^*(2n,\mathcal{H})+\ln(8n+4)+t+\ln\frac{2\gamma}{\delta}\right)+\frac{6}{n}\Big). \qquad (3.10)\end{aligned}$$

The proof is given in Section 4. This time, the proof incorporates random approximations of the classifier function and its variance, Bernstein's inequality as in (Koltchinskii and Panchenko, 2004), exploring the capacity of random class of the level sets of the margins of the approximating functions and Inequality (3.4).

This result is an improvement of the above margin-bound in the case that the total pointwise variance is small, that is, the classifier functions h_i in the convex combination f are close to each other. The constants in the bound are explicit. From the Remark of Theorem 3 in (Koltchinskii and Panchenko, 2004) and the above inequality (3.10), one can see that the quantity $\mathbb{P}_n\sigma^2_\lambda$ might provide a complexity penalty in the *general class* of problems defined above.

A result that improves inequality (3.10) by exploring the clustering properties of the convex combination from a *random* convex hull will now be shown.

Given $\lambda\in\mathcal{P}(\mathcal{H}(Z^n))$ and $f(x)=\int h(x)\lambda(dh)$, represent f as

$$f=\sum_{j=1}^{p}\alpha_j\sum_{k=1}^{T}\lambda_k^{(j)}h_k^{(j)}$$

with $\sum_{j\le p}\alpha_j=1, T\le\infty, h_k^{(j)}\in\mathcal{H}(Z^n)$.

Consider an element $c\in\mathcal{C}^p(\lambda)$, that is, $c=(\alpha_1,\ldots,\alpha_p,\lambda^1,\ldots,\lambda^p)$, such that $\alpha_i\in\Delta(m)=\Big\{t_k m^{-k}, k\in\mathbb{N}, t_k\in\{1,2,3,\ldots,m^k\}\Big\}$, $m\in\mathbb{N}$, $\lambda=\sum_{i=1}^p\alpha_i\lambda^i$, and $\lambda^i\in\mathcal{P}(\mathcal{H}(Z^n)), i=1,\ldots,p$. Denote by $\alpha_c^*=\min_{i\in\{1,\ldots,p\}}\alpha_i$, where $\{\alpha_i\}_{i=1}^p$ are called the cluster weights. c is interpreted as a decomposition of λ into p clusters as in (Koltchinskii and Panchenko, 2004). For an element $c\in\mathcal{C}^p(\lambda)$, a weighted variance over clusters is defined by

$$\sigma^2(c;x)=\sum_{k=1}^{p}\alpha_k^2\sigma^2_{\lambda^k}(x), \qquad (3.11)$$

where $\sigma^2_{\lambda^k}(x)$ are defined in (3.9). One can see in Fig. 2 that when the number of the clusters increases, the weighted variance over clusters uniformly decreases (shifts to the left). If there are p small clusters among functions $h_1,\ldots,h_T$, then one should be able to choose element $c\in\mathcal{C}^p(\lambda)$ so that $\sigma^2(c;x)$ will be small on the majority of data points $X_1,\ldots,X_n$ (Koltchinskii and Panchenko, 2004). The following theorem holds:

Theorem 5 *For any* $m\in\mathbb{N}$, *for any* $t>0$ *with probability at least* $1-e^{-t}$, *the following is true for any* n *i.i.d. random pairs* $(X_1,Y_1),\ldots,(X_n,Y_n)$ *randomly drawn with respect to the distribution* $\mathbb{P}$, *for any* $p\ge 1$, $c\in\mathcal{C}^p(\lambda)$, $\lambda=\sum_{i=1}^p\alpha_i\lambda^i\in\mathcal{P}(\mathcal{H}(Z^n))$, *such that* $\alpha_1,\ldots,\alpha_p\in\Delta(m)$ *with* $\sum_i\alpha_i\le 1$ *and* $f(x)=\int h(x)\lambda(dh)$

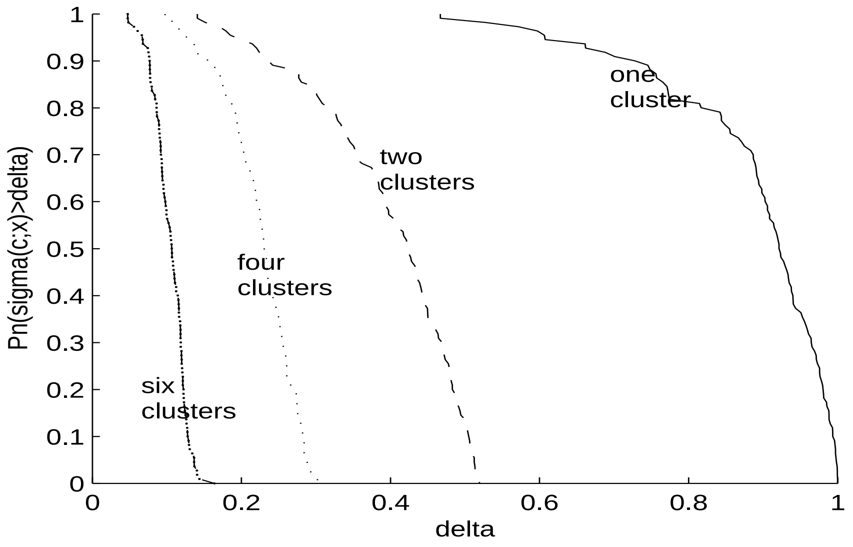


Figure 2: Empirical distribution of weighted variance over clusters; From right to left: one, tow, four, six clusters; the x-axis is δ.

$$\begin{aligned}\mathbb{P}(yf(x)\le 0) &\le \inf_{0<\delta\le\gamma\le 1}\Big(2\mathbb{P}_n(yf(x)\le 2\delta)+4\mathbb{P}_n(\sigma^2(c;x)\ge\gamma/3)+\\ &+\frac{8}{n}\Big(56p\frac{\gamma}{\delta^2}(\ln n)\ln G^*(2n,\mathcal{H})+\ln(8n+4)+2\sum_{j=1}^{p}\ln\left(\log_m\frac{\alpha_j}{\alpha_c^*}+1\right)+\\ &+2\ln\left(\log_m\frac{1}{\alpha_c^*}+1\right)+2p\ln 2+t+\ln\frac{p^2\pi^4\gamma}{18\delta}\Big)+\frac{6}{n}\Big).\end{aligned}\tag{3.12}$$

The proof is given in the following Section 4. Here, the proof incorporates more sophisticated random approximations of the classifier function and its weighted variance over clusters, Bernstein's inequality as in (Koltchinskii and Panchenko, 2004), exploring the capacity of random class of the level sets of the margins of the approximating functions and Inequality (3.4). The above bound can be simplified in the following way:

$$\begin{aligned}\mathbb{P}(yf(x)\le 0) &\le \inf_{0<\delta\le\gamma\le 1}\Big(2\mathbb{P}_n(yf(x)\le 2\delta)+4\mathbb{P}_n(\sigma^2(c;x)\ge\gamma/3)+\\ &+\frac{8}{n}\Big(56p\frac{\gamma}{\delta^2}(\ln n)\ln G^*(2n,\mathcal{H})+\ln(8n+4)+\\ &+2(p+1)\ln\left(\log_m\frac{1}{\alpha_c^*}+1\right)+2p\ln 2+t+\ln\frac{p^2\pi^4\gamma}{18\delta}\Big)\Big).\end{aligned}$$

Define the number $\widehat{p}_\lambda(m,n,\gamma,\delta)$ of (γ,δ)-clusters of λ as the smallest p, for which there exists $c\in\mathcal{C}_\lambda^p$ such that (Koltchinskii and Panchenko, 2004)

$$\mathbb{P}_n(\sigma^2(c;x)\ge\gamma)\le 56p\frac{\gamma}{n\delta^2}(\ln n)\ln G^*(2n,\mathcal{H}).$$

Recall that $G^*(n,\mathcal{H}) \leq 2n$ for Problem 1 and $G^*(n,\mathcal{H}) \leq 2nd$ for Problems 2, 3 and $G^*(n,\mathcal{H}) = \left(\frac{ne}{V}\right)^V$. Then the above simplified bound implies that for all $\lambda = \sum_{i=1}^p \alpha_i \lambda^i \in \mathbb{P}(\mathcal{H}(Z^n))$, such that $\alpha_1,\ldots,\alpha_p \in \Delta(m)$ with $\sum_i \alpha_i \leq 1$,

$$\mathbb{P}(yf(x) \leq 0) \leq K \inf_{0<\delta\leq\gamma\leq 1}\Big(\mathbb{P}_n(yf(x) \leq \delta) \quad + \quad \widehat{p}_\lambda(m,n,\gamma,\delta)\frac{\gamma}{n\delta^2}(\ln n)\ln G^*(2n,\mathcal{H}) \\ + \quad \widehat{p}_\lambda(m,n,\gamma,\delta)\frac{\ln\left(\log_m \frac{1}{\alpha_c^*}+1\right)}{n}\Big).$$

Observe that if $\gamma = \delta$, then

$$\mathbb{P}(yf(x) \leq 0) \leq K\Big(\mathbb{P}_n(yf(x) \leq \delta) \quad + \quad \widehat{p}_\lambda(m,n,\delta,\delta)\frac{(\ln n)\ln G^*(2n,\mathcal{H})}{n\delta} \\ + \quad \widehat{p}_\lambda(m,n,\delta,\delta)\frac{\ln\left(\log_m \frac{1}{\alpha_c^*}+1\right)}{n}\Big).$$

The above bound is an improvement of the previous bounds in the case when there is a small number $\widehat{p}_\lambda$ of clusters so that the resulting weighted variance over clusters is small, and provided that the minimum of the cluster weights α_c^* is not too small. The bounds shown above are similar in nature to the bounds by Koltchinskii and Panchenko (2004) for base-classes $\mathcal{H}$ satisfying a general entropy condition. The advantages of the current results are that they are applicable for random classes of functions. The bounds derived here are with explicit constants. For more information regarding the empirical performance of the bounds and the complexities in the case of boosting with stumps and decision trees (see Koltchinskii et al., 2003b; Andonova, 2004). There, it is shown experimentally that generalization bounds based on weighted variance over clusters and margin capture the generalization performance of classifiers produced by several boosting algorithms over decision stumps. Our goal here is to show theoretically the impact of the complexity terms on the generalization performance of functions from random convex hulls, which happen to capture well known algorithms such as SVMs. More experimental evidences are needed to explore the above complexities in the setting of the *general problem* defined here.

4. Proofs

First we will prove the following lemma that will be used in the proof of Theorem 1.

Lemma 6 *For n large enough, if X is a random variable with values in* $\{0,1\}$, $P(X=1)=p,\ p \in \left[\frac{2}{n},1\right]$ *and* $X_1,\ldots,X_n$ *are independent random realizations of X (Bernoulli trials), then*

$$P\left(\frac{1}{n}\sum_{i=1}^n X_i \geq p\right) \geq \frac{1}{4}.$$

Sketch of the Proof of Lemma 6.
We want to prove that

$$P\left(\frac{1}{n}\sum_{i=1}^n X_i \geq p\right) = \sum_{k\geq np}\binom{n}{k} p^k(1-p)^{n-k} \geq \frac{1}{4}.$$

Observe that if $\frac{n-1}{n} < p \leq 1$, then $n \geq np > n-1$ and the inequality becomes $p^n > \left(\frac{n-1}{n}\right)^n \geq \frac{1}{4}$, which is true for $n \geq 2$.

Assume that $p \leq \frac{n-1}{n}$. The proof of the inequality in this case relies on Poisson and Gaussian approximation to binomial distribution. Let $S_n = \sum_{i=1}^n X_i$ and $Z_n = \frac{\sum_{i=1}^n (X_i - p)}{\sqrt{np(1-p)}}$. Notice that

$$P\left(\frac{1}{n}\sum_{i=1}^n X_i \geq p\right) = P(S_n \geq np) = P(Z_n \geq 0).$$

We want to show that there is n_0, such that for any $n \geq n_0$ the following is true for any $p \in \left[\frac{2}{n}, 1 - \frac{1}{n}\right]$

$$P\left(\frac{1}{n}\sum_{i=1}^n X_i \geq p\right) \geq \frac{1}{4}$$

From the Poisson-Verteilung approximation theorem, (see Borowkow, 1976, Theorem 7, chapter 5, page 85) it follows that

$$P(S_n \geq \mu) \geq \sum_{k \geq np} \frac{\mu^k}{k!} e^{-\mu} - \frac{\mu^2}{n},$$

where $\mu = np \geq 2$. From the properties of the Poisson cumulative distribution function $F(x|\mu) = e^{-\mu}\sum_{i=0}^{\lfloor x \rfloor} \frac{\mu^i}{i!}$, one can see that $1 - F(x|\mu) > 1 - F(2|2) > 0.32$ for $x < \mu$ and $\mu \geq 2$. Therefore,

$$P(S_n \geq \mu) \geq 1 - F(x|\mu) - \frac{\mu^2}{n} > 0.32 - \frac{\mu^2}{n} = 0.32 - np^2.$$

Now, from the Berry-Esséen Theorem (see Feller, 1966, chapter XVI, page 515) one can derive that

$$|P(Z_n \geq 0) - 0.5| < \frac{33}{4} \cdot \frac{E(X - EX)^3}{\sqrt{n(E(X-EX)^2)^3}} = \frac{33}{4} \cdot \frac{p^2 + (1-p)^2}{\sqrt{np(1-p)}}.$$

Therefore, $P(Z_n \geq 0) > 0.5 - \frac{33}{4} \cdot \frac{p^2+(1-p)^2}{\sqrt{np(1-p)}}$. The goal is to find n_0 such that for any $n \geq n_0$ and $p \in \left[\frac{2}{n}, 1 - \frac{1}{n}\right]$ the following is true:

$$\max\left\{0.32 - np^2, 0.5 - \frac{33}{4} \cdot \frac{p^2 + (1-p)^2}{\sqrt{np(1-p)}}\right\} \geq \frac{1}{4}.$$

Let $x = np^2$. One can see that the first term $0.32 - np^2 = 0.32 - x$ is decreasing with respect to x and the second term $0.5 - \frac{33}{4} \cdot \frac{p^2+(1-p)^2}{\sqrt{np(1-p)}} = 0.5 - \frac{33}{4} \cdot \frac{p^2+(1-p)^2}{\sqrt{(1-p)}(nx)^{\frac{1}{4}}}$ is increasing with respect to x. The solution $x(n)$ of the equation

$$0.32 - x = 0.5 - \frac{33}{4} \cdot \frac{x/n + (1 - x/n)^2}{\sqrt{\left(1 - \sqrt{x/n}\right)}(nx)^{\frac{1}{4}}}$$

is decreasing with respect to n and therefore one can find n_0, such that for $n > n_0$ the inequality $0.32 - x(n) \geq 0.25$ is true.

Remark: A shorter proof could be achieved if one directly shows that for $p \in \left[\frac{2}{n}, 1\right]$,

$$P\left(\frac{1}{n}\sum_{i=1}^{n} X_i \geq p\right) = \sum_{k \geq np} \left(\begin{array}{c} n \\ k \end{array}\right) p^k (1-p)^{n-k} \geq \frac{1}{4}.$$

A stronger version of the above inequality for any p and n was used in (Vapnik (1998), page 133); however, a reference to a proof of this inequality appears currently to be unavailable.

$\square$

Proof of Theorem 1.

The proof of Inequality (3.4) for random collection of sets of Theorem 1 follows the three main steps - Symmetrization, Randomization and Tail Inequality (see Vapnik (1998); Anthony and Shawe-Taylor (1993)). The difference with other approaches is that the symmetrization step of the proof is carried out for random classes invariant under permutation, after one combines the training set with a ghost sample and uses the incremental property of the random class. Note that symmetrization for a random subset under similar incremental and permutation properties was proved for the "standard" Vapnik's inequality by Gat (1999) (bounding the absolute deviation).

Let $t > 0$ be fixed. Assume that $n \geq 2/t^2$, otherwise if $n < 2/t^2$, then $4\exp^{-nt^2/4} > 1$; nothing more need be proved.

Denote the set

$$A = \left\{ x = (x_1, \ldots, x_n) \in \mathcal{Z}^n : \sup_{C \in \mathcal{C}(x)} \frac{\mathbb{P}(C) - \frac{1}{n}\sum I(x_i \in C)}{\sqrt{\mathbb{P}(C)}} \geq t \right\}.$$

Assume there exist a set C_x, such that

$$\frac{\mathbb{P}(C_x) - \frac{1}{n}\sum I(x_i \in C_x)}{\sqrt{\mathbb{P}(C_x)}} \geq t. \tag{4.13}$$

Then $\mathbb{P}(C_x) \geq t^2$. We have assumed that $t^2 \geq \frac{2}{n}$, therefore $\mathbb{P}(C_x) \geq \frac{2}{n}$.

Let $x' = (x'_1, \ldots, x'_n)$ be independent copy of $x = (x_1, \ldots, x_n)$. It can be observed (see Lemma 6 and Anthony and Shawe-Taylor (1993), Theorem 2.1) that since $\mathbb{P}(C_x) = \mathbb{E}(I(y \in C_x)) \geq \frac{2}{n}$, then with probability at least $1/4$

$$\mathbb{P}(C_x) \leq \frac{1}{n}\sum I(x'_i \in C_x). \tag{4.14}$$

From the assumption (4.13) and (4.14), then since $\frac{x-a}{\sqrt{x+a}}$ is a monotone and increasing function in $x > 0$ ($a > 0$), we have that

$$\begin{aligned} 0 < t \quad & \leq \frac{\mathbb{P}(C_x) - \frac{1}{n}\sum I(x_i \in C_x)}{\sqrt{\mathbb{P}(C_x)}} \\ & \leq \frac{\mathbb{P}(C_x) - \frac{1}{n}\sum I(x_i \in C_x)}{\sqrt{\frac{1}{2}(\mathbb{P}(C_x) + \frac{1}{n}\sum I(x_i \in C_x))}} \\ & \leq \frac{\frac{1}{n}\sum I(x'_i \in C_x) - \frac{1}{n}\sum I(x_i \in C_x)}{\sqrt{\frac{1}{2}(\frac{1}{n}\sum I(x'_i \in C_x) + \frac{1}{n}\sum I(x_i \in C_x))}}. \end{aligned}$$

From (4.14) and the above inequality,

$$\begin{aligned}
\frac{1}{4}I(x\in A) &\le \mathbb{P}_{x'}\left(\mathbb{P}(C_x)\le \frac{1}{n}\sum I(x_i'\in C_x)\right)I(x\in A)\\
&\le \mathbb{P}_{x'}\left(\frac{\frac{1}{n}\sum I(x_i'\in C_x)-\frac{1}{n}\sum I(x_i\in C_x)}{\sqrt{\frac{1}{2}(\frac{1}{n}\sum I(x_i'\in C_x)+\frac{1}{n}\sum I(x_i\in C_x))}}\ge t\right)\\
&\le \mathbb{P}_{x'}\left(\sup_{C\in\mathcal{C}(x)}\frac{\frac{1}{n}\sum I(x_i'\in C)-\frac{1}{n}\sum I(x_i\in C)}{\sqrt{\frac{1}{2}(\frac{1}{n}\sum I(x_i'\in C)+\frac{1}{n}\sum I(x_i\in C))}}\ge t\right).
\end{aligned}$$

Taking the expectation $\mathbb{E}_x$ of both sides,

$$\begin{aligned}
&\mathbb{P}_x\left(\sup_{C\in\mathcal{C}(x)}\frac{\mathbb{P}(C)-\frac{1}{n}\sum_i I(x_i\in C)}{\sqrt{\mathbb{P}(C)}}\ge t\right)\le\\
&\le 4\mathbb{P}_{x,x'}\left(\sup_{C\in\mathcal{C}(x)}\frac{\frac{1}{n}\sum_i I(x_i'\in C)-\frac{1}{n}\sum_i I(x_i\in C)}{\sqrt{\frac{1}{2}(\frac{1}{n}\sum_i I(x_i'\in C)+\frac{1}{n}\sum_i I(x_i\in C))}}\ge t\right)
\end{aligned}$$

(using increasing property)

$$\le 4\mathbb{P}_{x,x'}\left(\sup_{C\in\mathcal{C}(x,x')}\frac{\frac{1}{n}\sum_i I(x_i'\in C)-\frac{1}{n}\sum_i I(x_i\in C)}{\sqrt{\frac{1}{2}(\frac{1}{n}\sum_i I(x_i'\in C)+\frac{1}{n}\sum_i I(x_i\in C))}}\ge t\right)$$

(using permutation property)

$$= 4\mathbb{P}_{x,x',\varepsilon}\left(\sup_{C\in\mathcal{C}(x,x')}\frac{\frac{1}{n}\sum_i \varepsilon_i(I(x_i'\in C)-\sum_i I(x_i\in C))}{\sqrt{\frac{1}{2}(\frac{1}{n}\sum_i I(x_i'\in C)+\frac{1}{n}\sum_i I(x_i\in C))}}\ge t\right)$$

(using Hoeffding-Azuma's inequality)

$$\begin{aligned}
&\le 4\mathbb{E}\left(\Delta_{\mathcal{C}(x,x')}(x_1,\dots,x_n,x_1',\dots,x_n')\dot{\exp}\left(-\frac{nt^2}{4\frac{\sum_i(I_i-I_i')^2}{\sum_i(I_i+I_i')}}\right)\right)\\
&\le 4\mathbb{E}_{x,x'}\left(\Delta_{\mathcal{C}(x,x')}(x_1,\dots,x_n,x_1',\dots,x_n')\dot{\exp}\left(-\frac{nt^2}{4}\right)\right)=\\
&= 4G(2n)\exp\left(-\frac{nt^2}{4}\right).
\end{aligned}$$

Here the increasing (2.2) and permutation (2.3) properties of the random collection of sets have been used .

□

The following lemma will be useful in the proofs of Theorems 2, 3, 4 and 5.

Lemma 7 *Let $Z_1,\dots,Z_n$ be n i.i.d. random variables randomly drawn with respect to the distribution $\mathbb{P}$, $Z_i=(X_i,Y_i)\in\mathcal{X}\times\mathcal{Y}$. Let*

$$\mathcal{C}_{N,k}(Z^n)=\{C:C=\{(x,y)\in\mathcal{X}\times\mathcal{Y}:yg(x)\le\delta\},\ g\in\mathcal{G}_{N,k}(Z^n),\delta\in[0,1]\},$$

where

$$\mathcal{G}_{N,k}(Z^n)=\left\{g:g(z)=\frac{1}{N}\sum_{i=1}^{N}k_ih_i(z),h_i\in\mathcal{H}(Z^n),1\le k_i\le N-k+1,k_i\in\mathbb{N}\right\},\ N,k\in\mathbb{N}$$

and $\mathcal{H}(Z^n)$ is a random-base class from the general problem. Then

$$G(n)=\mathbb{E}_n\Delta_{\mathcal{C}_{N,k}(Z^n)}(Z^n)\le\min\left((n+1)(N-k+1)^k(G^*(n,\mathcal{H}))^k,2^n\right).$$

If $k=N$, then $k_i=1$ and $\mathcal{G}_{N,N}(Z^n)=\left\{g:g(z)=\frac{1}{N}\sum_{i=1}^N h_i(z),h_i\in\mathcal{H}(Z^n)\right\}$, where $N\in\mathbb{N}$. In this case, it is clear that $G(n)=\mathbb{E}_n\Delta_{\mathcal{C}_{N,N}(Z^n)}(Z^n)\le\min\left((n+1)(G^*(n,\mathcal{H}))^N,2^n\right)$.

Proof.

Following the notation we have to prove that if $\mathcal{H}(Z^n)$ is with finite cardinality $H(Z^n)$, then

$$G(n)=\mathbb{E}_n\Delta_{\mathcal{C}_{N,k}(Z^n)}(Z^n)\le\min\left((n+1)(N-k+1)^k\mathbb{E}_n\left(H(Z^n)^k\right),2^n\right)$$

and if $\mathcal{H}(Z^n)$ is a collection of indicators from the general problem, then

$$G(n)=\mathbb{E}_n\Delta_{\mathcal{C}_{N,k}(Z^n)}(Z^n)\le\min\left((n+1)(N-k+1)^k\left(\frac{ne}{V}\right)^{Vk},2^n\right).$$

First, let $\mathcal{H}(Z^n)$ be with finite cardinality $H(Z^n)$. Then

$$\operatorname{card}\mathcal{G}_{N,k}(Z^n)\le(N-k+1)^kH(Z^n)^k,$$

because for each $g\in\mathcal{G}_{N,k}(Z^n)$ there are k different functions $h_i\in\mathcal{H}(Z^n)$ participating in the convex combination and the integer coefficients $k_i\in\{1,\dots,N-k+1\}$. Also, for fixed $g\in\mathcal{G}_{N,k}(Z^n)$, it follows that

$$\operatorname{card}\left\{\{yg(x)\le\delta\}\bigcap\{z_1,\dots,z_n\},\delta\in[-1,1]\right\}\le(n+1).$$

(This is clear after re-ordering $Y_1g(X_1),\dots,Y_ng(X_n)\to Y_{i_1}g(X_{i_1})\le\dots\le Y_{i_n}g(X_{i_n})$ and taking for values of $\delta\in\{Y_{i_1}g(X_{i_1}),\dots,Y_{i_n}g(X_{i_n}),1\}$.) Therefore,

$$G(n)=\mathbb{E}_n\Delta_{\mathcal{C}_{N,k}(Z^n)}(Z^n)\le\min\left((n+1)(N-k+1)^k\mathbb{E}_nH(Z^n)^k,2^n\right)\le$$

$$\le\min\left((n+1)(N-k+1)^k(G^*(n,\mathcal{H}))^k,2^n\right).$$

Next, let $\mathcal{H}(Z^n)$ be a collection of indicators over class of sets with finite VC-dimension V. Then, for fixed $\delta\in[0,1]$, the number of possible representations of $(Z_1,\dots,Z_n)$ by the class $\mathcal{C}_{N,k}(Z^n,\delta)=$

$\{C : C = \{(x,y) \in \mathcal{X} \times \mathcal{Y} : yg(x) \leq \delta\}, \, g \in \mathcal{G}_{N,k}(Z^n)\}$ is bounded by $(N-k+1)^k \left(\frac{ne}{V}\right)^{Vk}$. Similarly to the previous case, for fixed $g \in \mathcal{G}_{N,k}(Z^n)$,

$$\operatorname{card}\Big\{\{yg(x) \leq \delta\} \bigcap \{z_1, \ldots, z_n\}, \delta \in [0,1]\Big\} \leq (n+1),$$

and therefore

$$G(n) = \mathbb{E}_n \Delta_{\mathcal{C}_{N,k}(Z^n)}(Z^n) \leq \min\left((n+1)(N-k+1)^k \left(\frac{ne}{V}\right)^{Vk}, 2^n\right) =$$
$$= \min\Big((n+1)(N-k+1)^k (G^*(n,\mathcal{H}))^k, 2^n\Big).$$

□

Next, the proofs of Theorem 2,3, 4 and 5 are shown. They follow closely the proofs given by Koltchinskii and Panchenko (2004) and Koltchinskii et al. (2003b) for non random classes of functions. We adjust the proofs to hold for random classes of functions by using Inequality 3.4 from Theorem 1.

Define the function

$$\phi(a,b) = \frac{(a-b)^2}{a} I(a \geq b),$$

that is convex for $a > 0$ and increasing with respect to a, decreasing with respect to b.

Proof of Theorem 2.

Let $Z_1 = (X_1,Y_1), \ldots, Z_n = (X_n,Y_n)$ be i.i.d samples randomly drawn with respect to the distribution $\mathbb{P}$. Let us first fix $\delta \in (0,1]$ and let $f = \sum_{k=1}^T \lambda_k h_k \in \operatorname{conv}(\mathcal{H}(Z^n))$ be any function from the convex hull of $\mathcal{H}(Z^n)$, where $\mathcal{H}(Z^n)$ is the random-base class defined in the general problem.
Given $N \geq 1$, generate i.i.d sequence of functions $\xi_1, \ldots, \xi_N$ according to the distribution $\lambda = (\lambda_1, \ldots, \lambda_T), \mathbb{P}_\xi(\xi_i = h_k) = \lambda_k$ for $k = 1, \ldots, T$ and ξ_i are independent of $\{(X_k,Y_k)\}_{k=1}^n$. Then $\mathbb{E}_\xi \xi_i(x) = \sum_{k=1}^T \lambda_k h_k(x)$.

Consider a function

$$g(x) = \frac{1}{N} \sum_{k=1}^N \xi_k(x),$$

which plays the role of a random approximation of f in the following sense:

$$\begin{aligned}
\mathbb{P}(yf(x) \leq 0) \quad &= \mathbb{P}\Big(yf(x) \leq 0, yg(x) \leq \delta\Big) + \mathbb{P}\Big(yf(x) \leq 0, yg(x) > \delta\Big) \\
&\leq \mathbb{P}\Big(yg(x) \leq \delta\Big) + \mathbb{E}_{x,y}\mathbb{P}_\xi\Big(\mathbb{E}_\xi yg(x) \leq 0, yg(x) \geq \delta\Big) \\
&\leq \mathbb{P}\Big(yg(x) \leq \delta\Big) + \mathbb{E}_{x,y}\mathbb{P}_\xi\Big(yg(x) - \mathbb{E}_\xi yg(x) \geq \delta\Big) \\
&= \mathbb{P}\Big(yg(x) \leq \delta\Big) + \mathbb{E}_{x,y}\mathbb{P}_\xi\left(\sum_{k=1}^N (y\xi_i(x) - y\mathbb{E}_\xi \xi_i(x)) \geq N\delta\right) \\
&\leq \mathbb{P}\Big(yg(x) \leq \delta\Big) + \exp\Big(\frac{-N\delta^2}{2}\Big), \qquad (4.15)
\end{aligned}$$

where in the last step is applied Hoeffding-Černoff inequality. Then,

$$\mathbb{P}\Big(yf(x) \leq 0\Big) \leq \mathbb{P}\Big(yg(x) \leq \delta\Big) + \exp(-N\delta^2/2). \qquad (4.16)$$

Similarly to the above inequality, one can derive that,

$$\mathbb{E}_\xi \mathbb{P}_n\Big(yg(x) \le \delta\Big) \le \mathbb{P}_n\Big(yf(x) \le 2\delta\Big) + \exp(-N\delta^2/2). \tag{4.17}$$

For any random realization of the sequence $\xi_1, \dots, \xi_N$, the random function g belongs to the class $\mathcal{G}_N(Z^n) = \Big\{\frac{1}{N}\sum_{i=1}^N h_i(x) : h_i \in \mathcal{H}(Z^n)\Big\}$.
Consider the random collection of level sets for fixed $N \in \mathbb{N}$,

$$\mathcal{C}(Z^n) = \Big\{C = \{(x,y) \in \mathcal{X} \times \mathcal{Y} : yg(x) \le \delta\}, g \in \mathcal{G}_N(Z^n), \delta \in (0,1]\Big\}.$$

Clearly $\mathcal{C}(Z^n)$ satisfies conditions (2.2) and (2.3). In order to apply the inequality for the random collection of sets (3.4), one has to estimate $G(n) = \mathbb{E}^n \Delta_{\mathcal{C}(Z^n)}(Z^n)$. By Lemma 7 it follows that $G(n) \le (G^*(n, \mathcal{H}))^N (n+1)$.
From this and Theorem 1, we have

$$\begin{aligned} \mathbb{P}^n\left(\sup_{C \in \mathcal{C}(Z^n)} \frac{\mathbb{P}(C) - \frac{1}{n}\sum_{i=1}^n I(X_i \in C)}{\sqrt{\mathbb{P}(C)}} \ge t\right) &\le 4G(2n)e^{-\frac{nt^2}{4}} \le \\ &\le 4\left(G^*(2n, \mathcal{H})\right)^N (2n+1)e^{-\frac{nt^2}{4}} = e^{-u}, \end{aligned}$$

where a change of variables $t = \sqrt{\frac{4}{n}(u + N\ln(G^*(2n,\mathcal{H})) + \ln(8n+4))}$ is made. So, for a fixed $\delta \in (0,1]$, for any $u > 0$ with probability at least $1 - e^{-u}$, it follows that

$$\frac{\mathbb{P}(yg(x) \le \delta) - \frac{1}{n}\sum_{i=1}^n I(Y_i g(X_i) \le \delta)}{\sqrt{\mathbb{P}(yg(x) \le \delta)}} \le \sqrt{\frac{4}{n}(u + N\ln(G^*(2n,\mathcal{H})) + \ln(8n+4))}. \tag{4.18}$$

The function $\phi(a,b), a > 0$ is convex. Therefore,

$$\mathbb{E}_\xi \phi\Big(\mathbb{P}(yg(x) \le \delta), \mathbb{P}_n(yg(x) \le \delta)\Big) \ge \phi\Big(\mathbb{E}_\xi \mathbb{P}(yg(x) \le \delta), \mathbb{E}_\xi \mathbb{P}_n(yg(x) \le \delta)\Big).$$

Based on the monotonic properties of $\phi(a,b)$ and inequalities (4.16) and (4.17), it is obtained that for any $\delta \in (0,1]$, for any $u > 0$ with probability at least $1 - e^{-u}$,

$$\begin{aligned} &\phi\Big(\mathbb{P}(yf(x) \le 0) - \exp(-N\delta^2/2), \mathbb{P}_n(yf(x) \le 2\delta + \exp(-N\delta^2/2))\Big) \le \\ &\le \frac{4}{n}(u + N\ln(G^*(2n,\mathcal{H})) + \ln(8n+4)). \end{aligned} \tag{4.19}$$

Choose $N = \frac{2\ln n}{\delta^2}$, such that $\exp(-N\delta^2/2) = \frac{1}{n}$. Take

$$U = \frac{1}{n}\left(u + \frac{2\ln n}{\delta^2}\ln(G^*(2n,\mathcal{H})) + \ln(8n+4)\right).$$

Solving the above inequality with respect to $\mathbb{P}(yf(x) \le 0)$, it follows that

$$\mathbb{P}(yf(x) \le 0) \le \left(\sqrt{U} + \sqrt{\mathbb{P}_n(yf(x) \le 2\delta) + \frac{1}{n} + U}\right)^2 + \frac{1}{n}.$$

In order to make the bound uniform with respect to $\delta \in (0,1]$, we apply standard union bound techniques (Koltchinskii and Panchenko, 2004). First, we prove the uniformity for $\delta \in \Delta = \{2^{-k}, k = 0,1,\ldots.\}$. Apply the above inequality for fixed $\delta \in \Delta$ by replacing u by $u + \ln\frac{2}{\delta}$ and hence e^{-u} replaced by $\frac{\delta}{2}e^{-u}$. Denote

$$U' = \frac{1}{n}\left(u + \ln\frac{2}{\delta} + \frac{2\ln n}{\delta^2}\ln(G^*(2n,\mathcal{H})) + \ln(8n+4)\right).$$

Then

$$\mathrm{P}\left[\bigcap_{\delta\in\Delta}\left\{\mathbb{P}(yf(x)\le 0) \le \left(\sqrt{U'} + \sqrt{\mathbb{P}_n(yf(x)\le 2\delta) + \frac{1}{n} + U'}\right)^2 + \frac{1}{n}\right\}\right] \ge$$
$$\ge 1 - e^{-u}\sum_{k=1}^{\infty} 2^{-k} \ge 1 - e^{-u}.$$

Now, in order to make the bound for any $\delta \in (0,1]$, observe that if $\delta_0 \in (0,1]$ then there is $k \in \mathbb{Z}_+$, $2^{-k-1} \le \delta_0 < 2^{-k}$.

Therefore, if the above bound holds for fixed $\delta_0 \in (0,1]$, then

$$\mathbb{P}_n(yf(x) \le \delta_0) \le \mathbb{P}_n\left(yf(x) \le 2^{-k}\right)$$

and

$$1/\delta_0^2 \le 2^{2k+2},\ \ln\frac{2}{\delta_0} \le \ln 2^{k+2}.$$

So, changing the constants in the bound, denote

$$U = \frac{1}{n}\left(t + \ln\frac{4}{\delta} + \frac{8\ln n}{\delta^2}\ln(G^*(2n,\mathcal{H})) + \ln(8n+4)\right).$$

It follows that, for any $t > 0$ with probability at least $1 - e^{-t}$ for any $\delta \in (0,1]$, the following holds:

$$\mathbb{P}(yf(x)\le 0) \le \left(\sqrt{U} + \sqrt{\mathbb{P}_n(yf(x)\le 2\delta) + \frac{1}{n} + U}\right)^2 + \frac{1}{n}$$

Thus, the Theorem 2 and inequality (3.6) hold.

□

Now, the **proof of Sparsity bound of Theorem 3** will be shown.

Denote $\Delta = \{2^{-k} : k \ge 1\}$ and $z = (x,y)$, $Z^n = \Big((X_1,Y_1),\ldots,(X_n,Y_n)\Big)$.

Let us fix $f(x) = \sum_{k=1}^{T}\lambda_k h_k(x) \in \operatorname{conv}(\mathcal{H}(Z^n))$. Given $N \ge 1$, generate an i.i.d. sequence of functions $\xi_1,\ldots,\xi_N$ according to the distribution $\mathbb{P}_\xi(\xi_i(x) = h_k(x)) = \lambda_k$ for $k = 1,\ldots,T$ and independent of $\{(X_i,Y_i)\}_{i=1}^n$. Clearly, $\mathbb{E}_\xi\xi_i(x) = \sum_{k=1}^{T}\lambda_k h_k(x)$. Consider the function

$$g(x) = \frac{1}{N}\sum_{k=1}^{N}\xi_k(x),$$

which plays the role of a random approximation of f and $\mathbb{E}_\xi g(x) = f(x)$. One can write,

$$\begin{aligned}\mathbb{P}(yf(x) \le 0) &= \mathbb{E}_\xi \mathbb{P}\Big(yf(x) \le 0, yg(x) < \delta\Big) + \mathbb{E}_\xi \mathbb{P}\Big(yf(x) \le 0, yg(x) \ge \delta\Big) \le \\ &\le \mathbb{E}_\xi \mathbb{P}\Big(yg(x) \le \delta\Big) + \mathbb{E}\mathbb{P}_\xi\Big(yg(x) \ge \delta, \mathbb{E}_\xi yg(x) \le 0\Big).\end{aligned}$$

In the last term for a fixed $(x,y) \in \mathcal{X} \times \mathcal{Y}$,

$$\begin{aligned}\mathbb{P}_\xi\Big(yg(x) \ge \delta, \mathbb{E}_\xi yg(x) \le 0\Big) &\le \mathbb{P}_\xi\Big(yg(x) - \mathbb{E}_\xi yg(x) \ge \delta\Big) = \\ &= \mathbb{P}_\xi\left(\sum_{i=1}^{N}(y\xi_i(x) - y\mathbb{E}_\xi \xi_i(x)) \ge N\delta\right) \le \exp\big(-N\delta^2/2\big).\end{aligned}$$

where in the last step Hoeffding-Černoff inequality has been applied. Hence,

$$\mathbb{P}(yf(x) \le 0) - e^{-N\delta^2/2} \le \mathbb{E}_\xi \mathbb{P}(yg(x) \le \delta). \tag{4.20}$$

Similarly,

$$\mathbb{E}_\xi \mathbb{P}_n(yg(x) \le \delta) \le \mathbb{P}_n(yf(x) \le 2\delta) + e^{-N\delta^2/2}. \tag{4.21}$$

Clearly, for any random realization of the sequence $\xi_1, \ldots, \xi_N$, the function $g(x)$ belongs to the class

$$F_{N,k}(Z^n) = \left\{\frac{1}{N}\sum_{i=1}^{k} k_i h_i(x) : \sum_{i=1}^{k} k_i = N, 1 \le k_i \le N, h_i \in \mathcal{H}(Z^n)\right\},$$

for some $k \in \mathbb{N}$, which is the number of different indices i and $k_i \in \mathbb{N}$ is the number of repeating function h_i in the representation of g. Recall, $\mathcal{H}(Z^n)$ is the random-base class from the general problem. Then, $1 \le k \le \min(T,N)$. Let $p_{k,N} = \mathbb{P}_\xi(g \in F_{N,k}(Z^n))$.

Then the expectation $\mathbb{E}_\xi$ can be represented as

$$\mathbb{E}_\xi(L(g)) = \sum_{k \ge 1} p_{k,N} \mathbb{E}_\xi\left(L(g) | g \in F_{N,k}(Z^n)\right),$$

where L is a real valued measurable function and g is the random function

$$g(x) = \frac{1}{N}\sum_{k=1}^{N} \xi_k(x).$$

Now consider the random collection of sets

$$\mathcal{C}_{N,k}(Z^n) = \Big\{C : C = \{(x,y) : yg(x) \le \delta\}, g \in F_{N,k}(Z^n), \delta \in (0,1]\Big\},$$

where $N,k \in \mathbb{N}$. Clearly $\mathcal{C}_{N,k}(Z^n)$ satisfies conditions (2.2) and (2.3). In order to apply the inequality for random collection of sets (3.4), one has to estimate $G'(n) = \mathbb{E}_n \Delta_{\mathcal{C}_{N,k}(Z^n)}(Z^n)$.

By Lemma 7, it follows that

$$G'(n) \le (G^*(n,\mathcal{H}))^k (N-k+1)^k (n+1) \le (G^*(n,\mathcal{H}))^k N^k (n+1).$$

Now apply Inequality (3.4) for the random collection of sets $\mathcal{C}_{N,k}(Z^n)$. Then, with probability at least $1-e^{-t}$

$$\frac{(\mathbb{P}_{x,y}(yg(x)\le\delta)-\mathbb{P}_n(yg(x)\le\delta))^2}{\mathbb{P}_{x,y}(yg(x)\le\delta)}\le\frac{4}{n}(t+k\ln G^*(2n,\mathcal{H})+k\ln N+\ln(8n+4)).$$

The function $\phi(a,b),a>0$ is convex, so $\phi(\mathbb{E}_\xi a,\mathbb{E}_\xi b)\le\mathbb{E}_\xi\phi(a,b)$ for $a>0$.

Therefore,

$$\frac{(\mathbb{E}_\xi\mathbb{P}_{x,y}(yg(x)\le\delta)-\mathbb{E}_\xi\mathbb{P}_n(yg(x)\le\delta))^2}{\mathbb{E}_\xi\mathbb{P}_{x,y}(yg(x)\le\delta)}\le\mathbb{E}_\xi\frac{(\mathbb{P}_{x,y}(yg(x)\le\delta)-\mathbb{P}_n(yg(x)\le\delta))^2}{\mathbb{P}_{x,y}(yg(x)\le\delta)}=$$

$$=\sum_{k\ge1}p_{k,N}\mathbb{E}_\xi\Big(\frac{(\mathbb{P}_{x,y}(yg(x)\le\delta)-\mathbb{P}_n(yg(x)\le\delta))^2}{\mathbb{P}_{x,y}(yg(x)\le\delta)}|g\in F_{N,k}(Z^n)\Big)\le$$

$$\le\sum_{k\ge1}p_{k,N}\frac{4}{n}(t+k\ln G^*(2n,\mathcal{H})+k\ln N+\ln(8n+4)).$$

Observe that

$$\sum_{k\ge1}kp_{k,N}=\mathbb{E}\mathrm{card}\{k:k'th\text{ index is picked at least once }\}=$$

$$\sum_{k=1}^{T}\mathbb{E}I(k\text{ is picked at least once })=\sum_{k=1}^{T}(1-(1-\lambda_k)^N).$$

Denote $e_n(f,\delta)=\sum_{k=1}^{T}\Big(1-(1-\lambda_k)^N\Big)$. Let $N=\frac{2}{\delta^2}\ln n$, so that $e^{-N\delta^2/2}=\frac{1}{n}$.
The function $\phi(a,b)$ is increasing in a and decreasing in b. Combine the last result with (4.20) and (4.21):

$$\phi\Big(\mathbb{P}(yf(x)\le0)-n^{-1},\mathbb{P}_n(yf(x)\le2\delta)+n^{-1}\Big)\le$$
$$\le\frac{4}{n}(t+e_n(f,\delta)\ln G^*(2n,\mathcal{H})+e_n(f,\delta)\ln(\frac{2}{\delta^2}\ln n)+\ln(8n+4)).$$

Denote
$$W=\frac{1}{n}(t+e_n(f,\delta)\ln G^*(2n,\mathcal{H})+e_n(f,\delta)\ln(\frac{2}{\delta^2}\ln n)+\ln(8n+4)).$$

After solving the above inequality for $\mathbb{P}(yf(x)\le0)$, one can get that, for a fixed $\delta\in\{2^{-k}:k\ge1\}$, for every $t>0$ with probability at least $1-e^{-t}$ the following holds

$$\mathbb{P}(yf(x)\le0)\le\Big(\sqrt{W}+\sqrt{\mathbb{P}_n(yf(x)\le2\delta)+\frac{1}{n}+W}\Big)^2+\frac{1}{n}.\qquad(4.22)$$

It remains to make the bound uniform over $\delta\in(0,1]$, which is done again by using standard union bound techniques shown in the proof of Theorem 2 and the observation that if $\delta_0\in(0,1]$, then there is $k\in\mathbb{Z}_+,\ 2^{-k-1}<\delta_0\le2^{-k}$ and $e_n(f,\delta_0)\le\sum_{k=1}^{T}(1-(1-\lambda_i)^{8(\ln n)2^{2k}})$.

Redefine $e_n(f,\delta) = \sum_{k=1}^{T}(1-(1-\lambda_i)^{\frac{8\ln n}{\delta^2}})$.

So, by changing the constants in the bound, it follows that for any $t > 0$, with probability at least $1-e^{-t}$ for any $\delta \in (0,1]$ the following holds:

$$\mathbb{P}(yf(x) \le 0) \le \left(\sqrt{U} + \sqrt{\mathbb{P}_n(yf(x) \le 2\delta) + \frac{1}{n} + U}\right)^2 + \frac{1}{n},$$

where

$$U = \frac{1}{n}\left(t + \ln\frac{4}{\delta} + e_n(f,\delta)\ln G^*(2n,\mathcal{H}) + e_n(f,\delta)\ln\left(\frac{8}{\delta^2}\ln n\right) + \ln(8n+4)\right).$$

Thus, the Theorem 3 and inequality (3.8) hold.

$\square$

We now show the **proof** for the bound with the total variance in **Theorem 4**, using Theorem 1.

Given $f(x) = \sum_{k=1}^{T}\lambda_k h_k(x)$, and given $N \ge 1$, first generate an i.i.d. sequence of functions $\xi_1,\dots,\xi_N$ independently of $\{(X_i,Y_i)\}$ and according to the distribution $\mathbb{P}_\xi(\xi_i = h_k) = \lambda_k$, for $k = 1,\dots,T$, and consider a function

$$g(x) = \frac{1}{N}\sum_{i=1}^{N}\xi_i(x),$$

which plays the role of random approximation of f.

The main difference from the proof of the above theorems is that in equation (4.15) the condition on the variance $\sigma_\lambda^2(x)$ is also introduced. Namely, one can write

$$\begin{aligned}\mathbb{P}(yf(x) \le 0) \le \mathbb{E}_\xi\mathbb{P}(yg(x) \le \delta) \quad &+ \quad \mathbb{P}\left(\sigma_\lambda^2(x) \ge \gamma\right) + \\ &+ \quad \mathbb{E}\mathbb{P}_\xi\left(yg(x) \ge \delta, yf(x) \le 0, \sigma_\lambda^2(x) \le \gamma\right).\end{aligned}$$

The variance of ξ_i's, for a fixed $x \in \mathcal{X}$, is

$$\mathrm{Var}_\xi(\xi_i(x)) = \sigma_\lambda^2(x).$$

$-1 \le \xi_i(x) \le 1$, as well. Bernstein's inequality,

$$\begin{aligned}&\mathbb{P}_\xi\left(yg(x) \ge \delta, yf(x) \le 0, \sigma_\lambda^2(x) \le \gamma\right) \le \\ &\le \mathbb{P}_\xi\left(\sum_{i=1}^{N}(y\xi_i(x) - y\mathbb{E}_\xi\xi_i(x)) \ge N\delta | \mathrm{Var}_\xi(\xi_1(x)) \le \gamma\right) \le \\ &\le 2\exp\Big(-\frac{1}{4}\min\Big(\frac{N\delta^2}{\gamma}, N\delta\Big)\Big) = 2\exp\Big(-\frac{1}{4}\frac{N\delta^2}{\gamma}\Big),\end{aligned}$$

is used, since it is assumed that $\gamma \ge \delta$. Making this term negligible by taking $N = 4(\frac{\gamma}{\delta^2})\ln n$,

$$\mathbb{P}(yf(x) \le 0) \le \mathbb{E}_\xi\mathbb{P}(yg(x) \le \delta) + \mathbb{P}\left(\sigma_\lambda^2(x) \ge \gamma\right) + n^{-1}. \tag{4.23}$$

Similarly,

$$\begin{aligned}\mathbb{E}_\xi \mathbb{P}_n(yg(x) \le \delta) \le \mathbb{P}_n(yf(x) \le 2\delta) \quad &+ \quad \mathbb{P}_n\left(\sigma^2_\lambda(x) \ge \gamma\right) + \\ &+ \quad \mathbb{P}_n \mathbb{P}_\xi\left(yg(x) \le \delta, yf(x) \ge 2\delta, \sigma^2_\lambda(x) \le \gamma\right).\end{aligned}$$

Applying Bernstein's inequality to the last term with the same choice of $N = 4(\frac{\gamma}{\delta^2})\ln n$, one has

$$\mathbb{E}_\xi \mathbb{P}_n(yg(x) \le \delta) \le \mathbb{P}_n(yf(x) \le 2\delta) + \mathbb{P}_n\left(\sigma^2_\lambda(x) \ge \gamma\right) + \frac{1}{n}. \tag{4.24}$$

Now, similarly to the proof of Theorem 2, we derive inequality (4.18). For any $\gamma \ge \delta \in (0,1], N = 4(\frac{\gamma}{\delta^2})\ln n,$ for any $t > 0$ with probability at least $1 - e^{-t}$, the following holds:

$$\begin{aligned}&\phi\Big(\mathbb{E}_\xi \mathbb{P}(yg(x) \le \delta), \mathbb{E}_\xi \mathbb{P}_n(yg(x) \le \delta)\Big) \le \mathbb{E}_\xi \phi\Big(\mathbb{P}(yg(x) \le \delta), \mathbb{P}_n(yg(x) \le \delta)\Big) \le \\ &\le \frac{4}{n}\Big(4\frac{\gamma}{\delta^2}(\ln n)\ln G^*(2n,\mathcal{H}) + \ln(8n+4) + t\Big),\end{aligned} \tag{4.25}$$

where the fact that the function $\phi(a,b) = \frac{(a-b)^2}{a} I(a \ge b), a > 0$ is convex has been used; so, $\phi(\mathbb{E}_\xi a, \mathbb{E}_\xi b) \le \mathbb{E}_\xi \phi(a,b)$. The function $\phi(a,b)$ is increasing in a and decreasing in b; combining the last result with (4.23) and (4.24), one has

$$\begin{aligned}&\phi\Big(\mathbb{P}(yf(x) \le 0) - \mathbb{P}(\sigma^2_\lambda(x) \ge \gamma) - n^{-1}, \mathbb{P}_n(yf(x) \le 2\delta) + \mathbb{P}_n(\sigma^2_\lambda(x) \ge \gamma) + n^{-1}\Big) \\ &\le \frac{4}{n}(t + 4\frac{\gamma}{\delta^2}(\ln n)\ln G^*(2n,\mathcal{H}) + \ln(8n+4)).\end{aligned}$$

After solving this inequality for $\mathbb{P}(yf(x) \le 0)$, one has that, for any $\delta \in (0,1]$, any $1 \ge \gamma \ge \delta$, for any $t > 0$ with probability at least $1 - e^{-t}$, the following inequality holds

$$\begin{aligned}\mathbb{P}(yf(x) \le 0) \quad &\le \quad \mathbb{P}(\sigma^2(x) \ge \gamma) + \frac{1}{n} + \\ &+ \quad \left(\left(\mathbb{P}_n(yf(x) \le 2\delta) + \mathbb{P}_n(\sigma^2(x) \ge \gamma) + \frac{1}{n} + U\right)^{\frac{1}{2}} + U^{\frac{1}{2}}\right)^2,\end{aligned} \tag{4.26}$$

where

$$U = \frac{1}{n}\Big(t + 4\frac{\gamma}{\delta^2}(\ln n)\ln G^*(2n,\mathcal{H}) + \ln(8n+4)\Big).$$

Next, in (4.23),(4.24) and (4.26), the term $\mathbb{P}(\sigma^2_\lambda(x) \ge \gamma)$ is related to the term $\mathbb{P}_n(\sigma^2_\lambda(x) \ge \gamma)$ that appears. In order to be able to do this, generate two independent sequences ξ^1_k and ξ^2_k as above and consider

$$\sigma^2_N(x) = \frac{1}{2N}\Sigma^N_{k=1}(\xi^2_k(x) - \xi^1_k(x))^2 = \frac{1}{N}\Sigma^N_{k=1}\xi_k(x),$$

where

$$\xi_k(x) = \frac{1}{2}\Big(\xi^1_k(x) - \xi^2_k(x)\Big)^2.$$

Notice that $\xi_k(x)$ are i.i.d. random variables and $\mathbb{E}_\xi \xi_k(x) = \sigma_\lambda^2(x)$. Since $\xi_k^1, \xi_k^2 \in \mathcal{H}(Z^n)$, then $|\xi_k^1(x) - \xi_k^2(x)| \leq 2$. The variance

$$\mathrm{Var}_\xi(\xi_1(x)) \leq \mathbb{E}_\xi \xi_1^2(x) \leq 2\mathbb{E}_\xi \xi_1(x) = 2\sigma_\lambda^2(x).$$

Bernstein's inequality implies that for any $c > 0$,

$$\mathbb{P}_\xi\left(\sigma_N^2(x) - \sigma^2(x) \leq 2\sqrt{\frac{\sigma_\lambda^2(x)\gamma}{c}} + 8\frac{\gamma}{3c}\right) \geq 1 - e^{(-\frac{N\gamma}{c})}$$

and

$$\mathbb{P}_\xi\left(\sigma_\lambda^2(x) - \sigma_N^2(x) \leq 2\sqrt{\frac{\sigma_\lambda^2(x)\gamma}{c}} + 8\frac{\gamma}{3c}\right) \geq 1 - e^{(-\frac{N\gamma}{c})}.$$

Let choose $c = 18$. If $\sigma_\lambda^2(x) \leq \gamma$, then with probability at least $1 - e^{-N\gamma/18}$, it follows from the first inequality that $\sigma_N^2(x) \leq 2\gamma$. On the other hand, if $\sigma_N^2(x) \leq 2\gamma$, then with probability at least $1 - e^{-N\gamma/18}$, it follows from the second inequality that $\sigma_\lambda^2(x) \leq 3\gamma$. Based on this,

$$\mathbb{P}_\xi\left(\sigma_N^2(x) \geq 2\gamma, \sigma_\lambda^2(x) \leq \gamma\right) \leq e^{(-\frac{N\gamma}{18})},$$

and

$$\mathbb{P}_\xi\left(\sigma_N^2(x) \leq 2\gamma, \sigma_\lambda^2(x) \geq 3\gamma\right) \leq e^{(-\frac{N\gamma}{18})}.$$

One can write

$$\begin{aligned}
\mathbb{P}(\sigma_\lambda^2(x) \geq 3\gamma) &= \mathbb{E}_\xi \mathbb{P}\left(\sigma_\lambda^2(x) \geq 3\gamma, \sigma_N^2(x) \geq 2\gamma\right) + \mathbb{E}_\xi \mathbb{P}\left(\sigma_\lambda^2(x) \geq 3\gamma, \sigma_N^2(x) \leq 2\gamma\right) \\
&\leq \mathbb{E}_\xi \mathbb{P}\left(\sigma_N^2(x) \geq 2\gamma\right) + \mathbb{E}\mathbb{P}_\xi\left(\sigma_N^2(x) \leq 2\gamma, \sigma_\lambda^2(x) \geq 3\gamma\right)
\end{aligned}$$

and

$$\mathbb{E}_\xi \mathbb{P}_n\left(\sigma_N^2(x) \geq 2\gamma\right) \leq \mathbb{P}_n\left(\sigma_\lambda^2(x) \geq \gamma\right) + \mathbb{E}_\xi \mathbb{P}_n\left(\sigma_N^2(x) \geq 2\gamma, \sigma_\lambda^2(x) \leq \gamma\right).$$

Setting $N = c\gamma^{-1}\ln n$, then

$$\mathbb{P}\left(\sigma_\lambda^2(x) \geq 3\gamma\right) \leq \mathbb{E}_\xi \mathbb{P}\left(\sigma_N^2(x) \geq 2\gamma\right) + \frac{1}{n}, \tag{4.27}$$

and

$$\mathbb{E}_\xi \mathbb{P}_n\left(\sigma_N^2(x) \geq 2\gamma\right) \leq \mathbb{P}_n\left(\sigma_\lambda^2(x) \geq \gamma\right) + \frac{1}{n}. \tag{4.28}$$

For any realization of $\xi_k^{j,1}, \xi_k^{j,2}$, the functions σ_N^2 belong to the class

$$\mathcal{F}_N(Z^n) = \left\{\frac{1}{2N}\sum_{k=1}^{N}(h_k^{j,1} - h_k^{j,2})^2 : h_k^{j,1}, h_k^{j,2} \in \mathcal{H}(Z^n)\right\},$$

where $\mathcal{H}(Z^n)$ is defined as the random-base function class in the general problem.

Now, consider the random collection of sets

$$\mathcal{C}(Z^n) = \left\{C : C = \left\{x \in \mathcal{X} : \{\sigma_N^2(x) \geq \gamma\}\right\}, \sigma_N^2 \in \mathcal{F}_N(Z^n), \gamma \in (0,1]\right\}.$$

In order to bound $G'(n) = \mathbb{E}^n \Delta_{\mathcal{C}(Z^n)}(Z^n)$ take into account that if $\mathcal{H}(Z^n)$ is a random-base class of finite cardinality, then card$\mathcal{F}_N(Z^n) \leq G^*(n,\mathcal{H})^{2N}$. In the case of the base-random class $\mathcal{H}(Z^n)$ being a collection of indicators, similarly to the proof of Lemma 7, one can count the maximum number of different representations of $\{X_1,\ldots,X_n\}$ by

$$\mathcal{C}(Z^n,\gamma) = \left\{C : C = \left\{x \in \mathcal{X} : \{\sigma_N^2(x) \geq \gamma\}\right\}, \sigma_N^2 \in \mathcal{F}_N(Z^n)\right\}$$

for a fixed $\gamma \in (0,1]$. It is bounded by $\left(\frac{ne}{V}\right)^{2N}$. Then varying γ over the ordered discrete set $\{1, \sigma_N^2(X_{i_1}), \sigma_N^2(X_{i_2}), \ldots, \sigma_N^2(X_{i_n})\}$ for a fixed $\sigma_N^2 \in \mathcal{F}_N(Z^n)$, one can see that $G'(n) \leq (n+1)G^*(n,\mathcal{H})^{2N}$. Now, we apply Theorem 1 for the random collection of sets $\mathcal{C}$, for $N = 18\gamma^{-1}\ln n$. Then for any $t > 0$ with probability at least $1-e^{-t}$ for any sample Z^n, the following holds

$$\begin{aligned}\phi\Big(\mathbb{E}_\xi \mathbb{P}(\sigma_N^2(x) \geq \gamma), \mathbb{E}_\xi \mathbb{P}_n(\sigma_N^2(x) \geq \gamma)\Big) &\leq \mathbb{E}_\xi \phi\Big(\mathbb{P}(\sigma_N^2(x) \geq \gamma), \mathbb{P}_n(\sigma_N^2(x) \geq \gamma)\Big) \leq \\ &\leq \frac{4}{n}\Big(2N \ln G^*(2n,\mathcal{H}) + \ln(8n+4) + t\Big).\end{aligned}$$

Here, the monotonic property of $\phi(a,b)$ is used together with (4.27) and (4.28), in order to obtain the following bound under the above conditions:

$$\begin{aligned}&\phi\Big(\mathbb{P}(\sigma_\lambda^2(x) \geq 3\gamma) - \frac{1}{n}, \mathbb{P}_n(\sigma_\lambda^2(x) \geq \gamma) + \frac{1}{n}\Big) \leq \\ &\leq \frac{4}{n}\Big(36\gamma^{-1}(\ln n)\ln G^*(2n,\mathcal{H}) + \ln(8n+4) + t\Big),\end{aligned}$$

Solving the above inequality for $\mathbb{P}(\sigma_\lambda^2(x) \geq \gamma)$, we obtain

$$\mathbb{P}(\sigma_\lambda^2(x) \geq \gamma) \leq \frac{1}{n} + \left(W^{\frac{1}{2}} + \left(W + \mathbb{P}_n(\sigma_\lambda^2(x) \geq \gamma/3) + \frac{1}{n}\right)^{\frac{1}{2}}\right)^2,$$

where

$$W = \frac{1}{n}\left(t + \frac{108}{\gamma}(\ln n)\ln G^*(2n,\mathcal{H}) + \ln(8n+4)\right).$$

Combining the above inequality with the inequality (4.26) and using the inequalities $(a+b)^2 \leq 2a^2 + 2b^2$ and $\frac{1}{\gamma} \leq \frac{\gamma}{\delta^2}$ for $\gamma \geq \delta$, one has that, for any $\delta \in (0,1]$ and any $\gamma \in (0,1], \gamma \geq \delta$, for all $t > 0$ with probability at least $1-e^{-t}$, for any random sample Z^n, for any $\lambda \in \mathcal{P}(\mathcal{H}(Z^n))$ and $f(x) = \int h(x)\lambda(dh)$,

$$\begin{aligned}\mathbb{P}(yf(x) \leq 0) \leq 2\mathbb{P}_n(yf(x) \leq 2\delta) + 2\mathbb{P}_n(\sigma_\lambda^2(x) \geq \gamma) + 2\mathbb{P}_n(\sigma_\lambda^2(x) \geq \gamma/3) + \\ + \frac{8t}{n} + \frac{8\ln(8n+4)}{n} + \frac{6}{n} + \frac{448\gamma(\ln n)\ln G^*(2n,\mathcal{H})}{n\delta^2}.\end{aligned}$$

Observe that $\mathbb{P}_n(\sigma^2_\lambda(x) \geq \gamma) \leq \mathbb{P}_n(\sigma^2_\lambda(x) \geq \gamma/3)$. Rewrite

$$\begin{aligned}\mathbb{P}(yf(x) \leq 0) \quad &\leq \quad 2\mathbb{P}_n(yf(x) \leq 2\delta) + 4\mathbb{P}_n\left(\sigma^2_\lambda(x) \geq \frac{\gamma}{3}\right) + \\ &+ \quad \frac{448\gamma(\ln n)\ln G^*(2n,\mathcal{H})}{n\delta^2} + \frac{8t}{n} + \frac{8\ln(8n+4)}{n} + \frac{6}{n}.\end{aligned}$$

Next, the bound is made uniform with respect to $\gamma \in (0,1]$ and $\delta \in (0,1]$. First, one makes the bound uniform when $\gamma \in \Delta = \{2^{-k}, k \in \mathbb{Z}_+\}$, and $\delta \in \Delta$. Apply the above inequality for fixed $\delta \leq \gamma \in \Delta$ by replacing t by $t' + \ln\frac{2\gamma}{\delta}$ and, hence, e^{-t} replaced by $e^{-t'} = e^{-t}\frac{\delta}{2\gamma}$, where δ and $\gamma \in \Delta = \{2^{-k} : k \geq 0\}$.

$$\begin{aligned}\mathrm{P}\Big[\bigcap_{\delta,\gamma}\Big\{ \quad &\mathbb{P}(yf(x) \leq 0) \leq \Big(2\mathbb{P}_n(yf(x) \leq 2\delta) + 4\mathbb{P}_n\left(\sigma^2_\lambda(x) \geq \frac{\gamma}{3}\right) + \frac{6}{n} + \\ + \quad &\frac{8}{n}\left(t + \ln\frac{2\gamma}{\delta} + \ln(8n+4) + \frac{56\gamma}{\delta^2}(\ln n)\ln G^*(2n,\mathcal{H})\right)\Big)\Big\}\Big] \geq \\ \geq \quad &1 - \sum_{l \in \mathbb{Z}_+}\frac{2^{-l}}{2}.e^{-t} \geq 1 - e^{-t},\end{aligned}$$

where is used $\sum_{l\in\mathbb{Z}_+} 2^{-l} < 2$. Then the union bound should be applied in the whole range of $\delta, \gamma \in (0,1]$.

For any $t > 0$ with probability at least $1 - e^{-t}$, for any $\lambda \in \mathcal{P}(\mathcal{H})$ and $f(x) = \int h(x)\lambda(dh)$,

$$\begin{aligned}\mathbb{P}(yf(x) \leq 0) \leq \inf_{0<\delta\leq\gamma\leq 1}\Big(&2\mathbb{P}_n(yf(x) \leq 2\delta) + 4\mathbb{P}_n(\sigma^2_\lambda(x) \geq \frac{\gamma}{3}) + \\ &+ \frac{8}{n}\left(t + \ln\frac{2\gamma}{\delta} + \ln(8n+4) + \frac{56\gamma}{\delta^2}(\ln n)\ln G^*(2n,\mathcal{H})\right) + \frac{6}{n}\Big).\end{aligned}$$

□

Now the **proof of Theorem 5** regarding cluster-variance bound is given. Let us fix

$$\alpha_1, \ldots, \alpha_p, \sum_{i=1}^{p}\alpha_i \leq 1, \alpha_i > 0$$

used for the weights of the clusters in

$$c = (\alpha_1, \ldots, \alpha_p, \lambda^1, \ldots, \lambda^p),\ \lambda = \sum_{i=1}^{p}\alpha_i\lambda^i,\ \lambda^i \in \mathcal{P}(\mathcal{H}(Z^n)).$$

Generate functions from each cluster independently from each other and independently of the data and take their sum to approximate $f(x) = \int h(x)\lambda(dh) = \sum_{i=1}^{T}\lambda_i h_i(x)$. Given $N \geq 1$, generate independent $\xi^j_k(x), k \leq N, j \leq p$, where for each j, $\xi^j_k(x)$'s are i.i.d. and have the distribution $\mathbb{P}_\xi(\xi^j_k(x) = h_i(x)) = \lambda^j_i, i \leq T$. Consider a function that plays role of a random approximation of f

$$g(x) = \frac{1}{N}\sum_{j=1}^{p}\alpha_j\sum_{k=1}^{N}\xi^j_k(x) = \frac{1}{N}\sum_{k=1}^{N}g_k(x),$$

where $g_k(x) = \sum_{j=1}^{p} \alpha_j \xi_k^j(x)$.

For a fixed $x \in \mathcal{X}$ and $k \leq N$, the expectation of g_k with respect to the distribution $\mathbb{P}_\xi = \mathbb{P}_{\xi^1} \times, \ldots, \times \mathbb{P}_{\xi^p}$ is

$$\mathbb{E}_\xi(g_k(x)) = \Sigma_{j=1}^{p} \alpha_j \mathbb{E}_\xi(\xi_k^j(x)) = f(x);$$

its variance is

$$\mathrm{Var}_\xi(g_k(x)) = \sum_{j=1}^{p} \mathrm{Var}_\xi(\xi_k^j(x)) = \sum_{j=1}^{p} \alpha_j^2 \sigma_{\lambda^j}^2(x) = \sigma^2(c;x).$$

Then

$$\begin{aligned} \mathbb{P}(yf(x) \leq 0) &\leq \mathbb{E}_\xi \mathbb{P}(yg(x) \leq \delta) + \mathbb{P}\left(\sigma^2(c;x) \geq \gamma\right) + \\ &+ \mathbb{E}\mathbb{P}_\xi\left(yg(x) \geq \delta, yf(x) \leq 0, \sigma^2(c;x) \leq \gamma\right). \end{aligned}$$

Using Bernstein's inequality, $\gamma \geq \delta > 0$, $|g_k(x)| \leq 1$ and taking $N = \lceil 2 + 4/3 \rceil (\frac{\gamma}{\delta^2}) \ln n = 4\frac{\gamma}{\delta^2} \ln n$ will make the last term negligible. Thus,

$$\mathbb{P}(yf(x) \leq 0) \leq \mathbb{E}_\xi \mathbb{P}(yg(x) \leq \delta) + \mathbb{P}\left(\sigma^2(c;x) \geq \gamma\right) + \frac{1}{n}. \tag{4.29}$$

Also,

$$\begin{aligned} \mathbb{E}_\xi \mathbb{P}_n(yg(x) \leq \delta) &\leq \mathbb{P}_n(yf(x) \leq 2\delta) + \mathbb{P}_n\left(\sigma^2(c;x) \geq \gamma\right) + \\ &+ \mathbb{P}_n \mathbb{P}_\xi\left(yg(x) \geq \delta, yf(x) \leq 2\delta, \sigma^2(c;x) \leq \gamma\right). \end{aligned}$$

Applying Bernstein's inequality to the last term with the same choice of $N = 4\frac{\gamma}{\delta^2} \ln n$, it follows that

$$\mathbb{E}_\xi \mathbb{P}_n(yg(x) \leq \delta) \leq \mathbb{P}_n(yf(x) \leq 2\delta) + \mathbb{P}_n\left(\sigma^2(c;x) \geq \gamma\right) + \frac{1}{n}. \tag{4.30}$$

Now, consider the random collection of level sets

$$\mathcal{C}(Z^n) = \{C : C = \{(x,y) : yg(x) \leq \delta, \, (x,y) \in \mathcal{X} \times \mathcal{Y}\}, g \in F_N(Z^n), \delta \in [-1,1]\},$$

where

$$F_N(Z^n) = \left\{ \frac{1}{N} \Sigma_{i=1}^{N} g_i, g_i \in G(\alpha_1, \ldots, \alpha_p)_{[Z^n]} \right\}$$

and

$$G(\alpha_1, \ldots, \alpha_p)_{[Z^n]} = \left\{ g_k(x) = \sum_{j=1}^{p} \alpha_j \xi_k^j(x), \xi_k^j \in \mathcal{H}(Z^n) \right\},$$

where $\mathcal{H}(Z^n)$ is the random-base class of functions, defined in the general problem.

Similarly to the proof of Theorem 2, for fixed $g \in F_N(Z^n)$, we have

$$\mathrm{card}\left\{ C \bigcap Z_1, \ldots, Z_n \right\} \leq (n+1)$$

and

$$\mathrm{card} F_N(Z^n) \leq G^*(n, \mathcal{H})^{Np}.$$

Therefore, $G'(n) = \mathbb{E}^n \Delta_{\mathcal{C}}(Z^n) \le (n+1)G^*(n,\mathcal{H})^{Np}$. Apply Inequality (3.4) from Theorem 1 for random the collection of sets $\mathcal{C}$. Then, with probability at least $1-e^{-t}$

$$\frac{(\mathbb{P}_{x,y}(yg(x)\le\delta)-\mathbb{P}_n(yg(x)\le\delta))^2}{\mathbb{P}_{x,y}(yg(x)\le\delta)} \le \frac{4}{n}(t+Np\ln G^*(2n,\mathcal{H})+\ln(8n+4)).$$

The function $\phi(a,b)=\frac{(a-b)^2}{a}I(a\ge b), a>0$ is convex, so $\phi(\mathbb{E}_\xi a,\mathbb{E}_\xi b)\le\mathbb{E}_\xi\phi(a,b)$

$$\frac{(\mathbb{E}_\xi\mathbb{P}_{x,y}(yg(x)\le\delta)-\mathbb{E}_\xi\mathbb{P}_n(yg(x)\le\delta))^2}{\mathbb{E}_\xi\mathbb{P}_{x,y}(yg(x)\le\delta)} \le \frac{4}{n}(t+Np\ln G^*(2n,\mathcal{H})+\ln(8n+4)).$$

The function $\phi(a,b)$ is increasing in a and decreasing in b and combined with the last result with (4.29) and (4.30) (recall that $N=4(\frac{\gamma}{\delta^2})\ln n$)

$$\phi\Big(\mathbb{P}(yf(x)\le 0)-\mathbb{P}(\sigma^2(c;x)\ge\gamma)-\frac{1}{n},\mathbb{P}_n(yf(x)\le 2\delta)+\mathbb{P}_n(\sigma^2(c;x)\ge\gamma)+\frac{1}{n}\Big)\le$$
$$\le\frac{4}{n}\left(t+4p\frac{\gamma}{\delta^2}(\ln n)\ln G^*(2n,\mathcal{H})+\ln(8n+4)\right).$$

After solving this inequality for $\mathbb{P}(yf(x)\le 0)$, one can get that, for any $\gamma,\delta\in(0,1]$, $\gamma\ge\delta$, and $\alpha^1,\dots,\alpha^p,\sum\alpha_i\le 1,\alpha_i>0$ for any $t>0$ with probability at least $1-e^{-t}$,

$$\mathbb{P}(yf(x)\le 0)\le\mathbb{P}\left(\sigma^2(c;x)\ge\gamma\right)+$$
$$+\left(\left(\mathbb{P}_n(yf(x)\le 2\delta)+\mathbb{P}_n(\sigma^2(c;x)\ge\gamma)+\frac{1}{n}+U\right)^{\frac{1}{2}}+U^{\frac{1}{2}}+\frac{1}{n}\right)^2, \tag{4.31}$$

where

$$U=\frac{1}{n}\left(t+4\frac{p\gamma}{\delta^2}(\ln n)\ln G^*(2n,\mathcal{H})+\ln(8n+4)\right),$$

$c\in\mathcal{C}^p(\lambda)$, $\lambda=\sum_{j=1}^p\alpha_j\lambda^j$, $\lambda_j\in\mathcal{P}(\mathcal{H})$.

Now, $\mathbb{P}(\sigma^2(c;x)\ge\gamma)$ has to be estimated. Generate two independent random sequences of functions $\xi_k^{j,1}(x)$ and $\xi_k^{j,2}(x)$, $j=1,\dots p, k=1,\dots,N$ as before ($\mathbb{P}_\xi(\xi_k^{j,1}(x)=h_i(x))=\lambda_i^j$, $\mathbb{P}_\xi(\xi_k^{j,2}(x)=h_i(x))=\lambda_i^j$) and consider

$$\sigma_N^2(c;x)=\frac{1}{2N}\Sigma_{k=1}^N\left(\Sigma_{j=1}^p\alpha^j(\xi_k^{j,2}(x)-\xi_k^{j,1}(x))\right)^2=\frac{1}{N}\Sigma_{k=1}^N\xi_k(x),$$

where

$$\xi_k(x)=\frac{1}{2}\Big(\Sigma_{j=1}^p\alpha^j(\xi_k^{j,1}(x)-\xi_k^{j,2}(x))\Big)^2. \tag{4.32}$$

Then $\xi_k(x)$ are i.i.d. random variables and $\mathbb{E}_\xi\xi_k(x)=\sigma^2(c;x)$. Since $\xi_k^{j,1},\xi_k^{j,2}\in\mathcal{H}$, then $|\xi_k^{j,1}(x)-\xi_k^{j,2}(x)|\le 2$. The variance satisfies the following inequality

$$\mathrm{Var}_\xi(\xi_1(x))\le\mathbb{E}_\xi\xi_1^2(x)\le 2\mathbb{E}_\xi\xi_1(x)=2\sigma^2(c;x).$$

Bernstein's inequality implies that

$$\mathbb{P}_\xi\left(\sigma_N^2(c;x)-\sigma^2(c;x)\le 2\sqrt{\frac{\sigma^2(c;x)}{K}}+8\frac{\gamma}{3K}\right)\ge 1-e^{(-\frac{N\gamma}{K})}$$

and

$$\mathbb{P}_\xi\left(\sigma^2(c;x)-\sigma_N^2(c;x)\le 2\sqrt{\frac{\sigma^2(c;x)}{K}}+8\frac{\gamma}{3K}\right)\ge 1-e^{(-\frac{N\gamma}{K})}.$$

Based on this, for large enough $K>0$ ($K=18$ is sufficient),

$$\mathbb{P}_\xi\left(\sigma_N^2(c;x)\ge 2\gamma,\sigma^2(c;x)\le\gamma\right)\le e^{(-\frac{N\gamma}{K})},$$

and

$$\mathbb{P}_\xi\left(\sigma_N^2(c;x)\le 2\gamma,\sigma^2(c;x)\ge 3\gamma\right)\le e^{(-\frac{N\gamma}{K})}.$$

One can write

$$\mathbb{P}\left(\sigma^2(c;x)\ge 3\gamma\right)\le\mathbb{E}_\xi\mathbb{P}\left(\sigma_N^2(c;x)\ge 2\gamma\right)+\mathbb{E}\mathbb{P}_\xi\left(\sigma_N^2(c;x)\le 2\gamma,\sigma^2(c;x)\ge 3\gamma\right),$$

and

$$\mathbb{E}_\xi\mathbb{P}_n\left(\sigma_N^2(c;x)\ge 2\gamma\right)\le\mathbb{P}_n\left(\sigma_N^2(c;x)\ge\gamma\right)+\mathbb{P}_n\mathbb{P}_\xi\left(\sigma_N^2(c;x)\ge 2\gamma,\sigma^2(c;x)\le\gamma\right).$$

Choose $N=K\gamma^{-1}\ln n$; then

$$\mathbb{P}\left(\sigma^2(c;x)\ge 3\gamma\right)\le\mathbb{E}_\xi\mathbb{P}\left(\sigma_N^2(c;x)\ge 2\gamma\right)+\frac{1}{n},\tag{4.33}$$

and

$$\mathbb{E}_\xi\mathbb{P}_n\left(\sigma_N^2(c;x)\ge 2\gamma\right)\le\mathbb{P}_n\left(\sigma_N^2(c;x)\ge\gamma\right)+\frac{1}{n}.\tag{4.34}$$

Now, consider the random collection of sets

$$\mathcal{C}_{Z^n}=\left\{C:C=\left\{x:\sigma_N^2(c;x)\ge 2\gamma\right\},\sigma_N^2(c;x)\in\mathcal{F}_N(Z^n),\gamma\in(0,1]\right\},$$

where

$$\mathcal{F}_N(Z^n)=\left\{\frac{1}{2N}\Sigma_{k=1}^N\left(\Sigma_{j=1}^p\alpha^j(h_k^{j,1}-h_k^{j,2})\right)^2,h_k^{j,1},h_k^{j,2}\in\mathcal{H}(Z^n)\right\}.$$

For any $\{x_1,\dots,x_n\}$ and a fixed $\sigma_N^2(c;.)\in\mathcal{F}_N(Z^n)$, it follows that

card$\{C\bigcap\{X_1,\dots,X_n\}\}\le(n+1)$ and card$\mathcal{F}_N(Z^n)\le G^*(n,\mathcal{H})^{2Np}$

if the base-random class $\mathcal{H}(Z^n)$ is of finite cardinality. Therefore, $G'_C(n)=\mathbb{E}^n\Delta_{\mathcal{C}_{Z^n}}(Z^n)\le(n+1)G^*(n,\mathcal{H})^{2Np}$. The case of $\mathcal{H}(Z^n)$ being a collection of indicators as in the general problem is similar and dealt with in the previous proofs of the theorems.

The rest of the arguments are similar to the proof of the Theorem 4. Apply the inequality (3.4) from Theorem 1 for random collection of sets $\mathcal{C}_{Z^n}$, and based on convexity of $\phi(a,b)$, one has that for $\gamma\in(0,1]$, $\alpha_1,\dots,\alpha_p,\sum_{j=1}^p\alpha_j\le 1,\alpha_j>0$ and for any $t>0$ with probability at least $1-e^{-t}$

$$\frac{(\mathbb{E}_\xi \mathbb{P}\left(\sigma_N^2(c;x)\geq\gamma\right)-\mathbb{E}_\xi \mathbb{P}_n\left(\sigma_N^2(c;x)\geq\gamma\right))^2}{\mathbb{E}_\xi \mathbb{P}\left(\sigma_N^2(c;x)\geq\gamma\right)}\leq\frac{2}{n}(t+2Np\ln G^*(2n,\mathcal{H})+\ln(8n+4)),$$

for any $\lambda^j\in\mathcal{P}(\mathcal{H}(Z^n)), j=1,\ldots,p$. Combining the last result ($\phi(a,b)$ is increasing in a and decreasing in b) with (4.33) and (4.34) (recall that $N=18\gamma^{-1}\ln n$),

$$\begin{aligned}&\phi\Big(\mathbb{P}\left(\sigma^2(c;x)\geq3\gamma\right)-n^{-1},\mathbb{P}_n\left(\sigma^2(c;x)\geq\gamma\right)+n^{-1}\Big)\leq\\&\leq\frac{4}{n}\left(t+36p\frac{\gamma}{\delta^2}(\ln n)\ln G^*(2n,\mathcal{H})+\ln(8n+4)\right).\end{aligned}$$

Solving the above inequality for $\mathbb{P}(\sigma_\lambda^2(x)\geq\gamma)$, then with probability at least $1-e^{-t}$

$$\mathbb{P}\left(\sigma_\lambda^2(x)\geq\gamma\right)\leq\frac{1}{n}+\left(W^{\frac{1}{2}}+\left(W+\mathbb{P}_n\left(\sigma_\lambda^2(x)\geq\gamma/3\right)+\frac{1}{n}\right)^{\frac{1}{2}}\right)^2,$$

where

$$W=\frac{1}{n}\left(t+\frac{108p}{\gamma}(\ln n)\ln G^*(2n,\mathcal{H})+\ln(8n+4)\right).$$

Finally, combining this with (4.31) and using the inequalities $(a+b)^2\leq2a^2+2b^2$ and $\frac{1}{\gamma}\leq\frac{\gamma}{\delta^2}$ for $\gamma\geq\delta$, one obtains: for any $\delta\in(0,1]$ and any $\gamma\in(0,1],\gamma\geq\delta$, for all $t>0$ with probability at least $1-e^{-t}$, for any random sample Z^n, for any $\lambda\in\mathcal{P}(\mathcal{H}(Z^n))$ and $f(x)=\int h(x)\lambda(dh)$,

$$\begin{aligned}&\mathbb{P}(yf(x)\leq0)\leq2\mathbb{P}_n(yf(x)\leq2\delta)+2\mathbb{P}_n\left(\sigma_\lambda^2(x)\geq\gamma\right)+2\mathbb{P}_n\left(\sigma_\lambda^2(x)\geq\gamma/3\right)+\\&+\frac{8t}{n}+\frac{8\ln(8n+4)}{n}+\frac{6}{n}+\frac{448p\gamma(\ln n)\ln G^*(2n,\mathcal{H})}{n\delta^2}.\end{aligned}$$

Observe that $\mathbb{P}_n\left(\sigma_\lambda^2(x)\geq\gamma\right)\leq\mathbb{P}_n\left(\sigma_\lambda^2(x)\geq\gamma/3\right)$. Then, rewrite

$$\begin{aligned}\mathbb{P}(yf(x)\leq0)&\leq&&2\mathbb{P}_n(yf(x)\leq2\delta)+4\mathbb{P}_n\left(\sigma_\lambda^2(x)\geq\frac{\gamma}{3}\right)+\\&+&&\frac{448p\gamma(\ln n)\ln G^*(2n,\mathcal{H})}{n\delta^2}+\frac{8t}{n}+\frac{8\ln(8n+4)}{n}+\frac{6}{n}.\end{aligned}$$

The next step is to make this bound uniform with respect to $\alpha_j>0, j=1,\ldots,p,\sum_{j=1}^p\alpha_j\leq1$.

First, consider simply $\alpha_i\in\Delta=\{2^{-j},j=1,2,\ldots\}$. The case of $\alpha_i=1$ is proven in the previous Theorem 4 for total variance. Let $\alpha_j=2^{-l_j}$. Redefine cluster $c(l_1,\ldots,l_p):=c(\alpha_1,\ldots,\alpha_p,\lambda^1,\ldots,\lambda^p)$. Then consider the event

$$\begin{aligned}&A_{c(l_1,\ldots,l_p)}=\Big\{\forall f\in\mathcal{F}_d:\mathbb{P}(yf(x)\leq0)\leq2\mathbb{P}_n(yf(x)\leq2\delta)+4\mathbb{P}_n\left(\sigma_\lambda^2(x)\geq\frac{\gamma}{3}\right)+\\&+\frac{448p\gamma(\ln n)\ln G^*(2n,\mathcal{H})}{n\delta^2}+\frac{8t}{n}+\frac{8\ln(8n+4)}{n}+\frac{6}{n}\Big\},\end{aligned}$$

that holds with probability $1-e^{-t}$. Make change of variables t by $t+2\sum_{j=1}^{p}\ln l_j + p\ln 4$ in the last bound. With this choice, the event $A_{c(l_1,\ldots,l_p)}$ can be rewritten as

$$A_{c(l_1,\ldots,l_p)} = \Big\{\forall f\in\mathcal{F}_d : \mathbb{P}(yf(x)\le 0)\le 2\mathbb{P}_n(yf(x)\le 2\delta)+4\mathbb{P}_n\left(\sigma_\lambda^2(x)\ge\frac{\gamma}{3}\right)+ \\ +\frac{448p\gamma(\ln n)\ln G^*(2n,\mathcal{H})}{n\delta^2}+\frac{8t}{n}+\frac{16\sum_{j=1}^{p}\ln l_j}{n}+\frac{8p\ln 4}{n}+\frac{8\ln(8n+4)}{n}+\frac{6}{n}\Big\},$$

which holds with probability at least

$$\mathrm{P}(A_{c(l_1,\ldots,l_p)})\ge 1-\prod\frac{1}{l_j^2}e^{-t}4^{-p}.$$

This implies the probability of the intersection

$$\mathrm{P}\left(\bigcap_{l_1,\ldots,l_p}A_{c(l_1,\ldots,l_p)}\right)\ge 1-\sum_{l_1,\ldots,l_p\in\mathbb{N}}\prod\frac{1}{l_j^2}e^{-t}4^{-p}=$$

$$=1-4^{-p}e^{-t}\Big(\sum_{l_i\in\mathbb{N}}\frac{1}{l_i^2}\Big)^p=1-4^{-p}e^{-t}(1+\pi^2/6)^p\ge 1-e^{-t}\ge 1-e^{-t}$$

and $\sum\ln l_j=\sum\ln(|\log_2\alpha_j|)$. For fixed $p\ge 1$ and $1\ge\gamma\ge\delta>0$ and $\forall t>0$ with probability at least $1-e^{-t}$, the following is true for any $\alpha_1,\ldots,\alpha_p\in\Delta$, $\sum_{i=1}^{p}\alpha_i\le 1$, $\Delta=\{2^{-j}, j=0,1,\ldots\}$:

$$\mathbb{P}(yf(x)\le 0)\le\Big(2\mathbb{P}_n(yf(x)\le 2\delta)+4\mathbb{P}_n\left(\sigma^2(c;x)\ge\gamma/3\right)+ \\ +\frac{448p\gamma(\ln n)\ln G^*(2n,\mathcal{H})}{n\delta^2}+\frac{8t}{n}+\frac{16\sum_{j=1}^{p}\ln(|\log_2\alpha_j|)}{n}+\frac{8p\ln 4}{n}+\frac{8\ln(8n+4)}{n}+\frac{6}{n}\Big).$$

Next, for a fixed $m\in\mathbb{N}$, consider the following discretization of $\alpha_j=t_jm^{-s}$, for a fixed a priori $s\in\mathbb{Z}_+$, and $t_j\in\{1,2,3,\ldots,m^s\}$. Therefore $s+\log_m\alpha_j\ge 0$.

For any $\alpha_j=t_jm^{-s}$ there is $l_j\in\mathbb{Z}_+$, such that

$$m^{-l_j-1}<\alpha_j=t_j.m^{-s}\le m^{-l_j}.$$

That is $s-l_j-1<\log_m t_j\le s-l_j$, $l_j\le s$.

This time we make the change of variables $t'=t+\sum_{j=1}^{p}2\ln(s+\log_m\alpha_j+1)+2p\ln 2$ and apply the bound for that t'.

Then $e^{-t'}=e^{-t}\sum_{j=1}^{p}\frac{1}{(\log_m t_j+1)^2}4^{-p}\le e^{-t}\sum_{j=1}^{p}\frac{1}{(s-l_j+1)^2}4^{-p}$. Applying union bound trick as before, shows that for any $t>0$, with probability at least

$$1-e^{-t}4^{-p}\Big(\sum_{j=1}^{s}\frac{1}{(s-l_j+1)^2}\Big)^p>1-e^{-t},$$

for any $\alpha_j=t_jm^{-s}$, $t_j\in\{1,2,3,\ldots,m^s\}$, $j=1,\ldots,p$, the following bound holds:

$$\begin{aligned}\mathbb{P}(yf(x)\le 0)\quad&\le\quad\Big(2\mathbb{P}_n(yf(x)\le 2\delta)+4\mathbb{P}_n\left(\sigma^2(c;x)\ge\gamma/3\right)+\\ &+\quad\frac{448p\gamma(\ln n)\ln G^*(2n,\mathcal{H})}{n\delta^2}+\frac{8t}{n}+\frac{16\sum_{j=1}^{p}\ln(s+\log_m\alpha_j+1)}{n}+\\ &+\quad\frac{16p\ln 2}{n}+\frac{8\ln(8n+4)}{n}+\frac{6}{n}\Big).\end{aligned}$$

In order to make the bound uniform for all $s, p \geq 1$ and $1 \geq \gamma \geq \delta > 0$, apply the above inequality for fixed $p \in \mathbb{N}, \delta \leq \gamma \in \Delta = \{2^{-k} : k \geq 1\}$ by replacing t by $t' = t + \ln \frac{s^2 p^2 \pi^4 \gamma}{\delta 18}$ and hence replacing e^{-t} by $e^{-t'} = e^{-t} \frac{\delta 18}{s^2 p^2 \pi^4 \gamma}$, where δ and $\gamma \in \Delta = \{2^{-k} : k \geq 1\}$.

$$
\begin{aligned}
P\Big[\bigcap_{\delta,\gamma,p}\Big\{ \quad & \mathbb{P}(yf(x) \leq 0) \leq \Big(2\mathbb{P}_n(yf(x) \leq 2\delta) + 4\mathbb{P}_n\left(\sigma^2(c;x) \geq \gamma/3\right) + \\
+ \quad & \frac{448 p\gamma(\ln n)\ln G^*(2n,\mathcal{H})}{n\delta^2} + \frac{8t}{n} + \frac{8\ln\frac{s^2p^2\pi^4\gamma}{18\delta}}{n} + \frac{16\sum_{j=1}^{p}\ln(s+\log_m \alpha_j + 1)}{n} + \\
+ \quad & \frac{16p\ln 2}{n} + \frac{8\ln(8n+4)}{n} + \frac{6}{n}\Big)\Big\}\Big] \\
& \geq 1 - \sum_{l\in\mathbb{Z}_+} \frac{2^{-l}36}{2\pi^4}.(\sum_k \frac{1}{k^2})^2 e^{-t} \geq 1 - e^{-t},
\end{aligned}
$$

where we have applied $\sum_{k\in\mathbb{Z}_+}\frac{1}{k^2} \leq \frac{\pi^2}{6}$ and $\sum_{l\in\mathbb{Z}_+} 2^{-l} \leq 2$.

Finally, $\forall t > 0$ with probability at least $1 - e^{-t}$, the following is true for all $s \in \mathbb{N}, \alpha_1, \ldots, \alpha_p \in \Delta = \left\{t_j m^{-s}, 0 < t_j \leq m^s, t_j \in \mathbb{N}\right\}$, $p \in \mathbb{N}$ and $1 \geq \gamma \geq \delta > 0$

$$
\begin{aligned}
\mathbb{P}(yf(x) \leq 0) \quad \leq \quad & \Big(2\mathbb{P}_n(yf(x) \leq 2\delta) + 4\mathbb{P}_n\left(\sigma^2(c;x) \geq \gamma/3\right) + \\
+ \quad & \frac{1}{n}\Big(\frac{448p\gamma(\ln n)\ln G^*(2n,\mathcal{H})}{\delta^2} + 8t + 8\ln\frac{s^2p^2\pi^4\gamma}{18\delta} + \\
+ \quad & 16\sum_{j=1}^{p}\ln(s+\log_m\alpha_j+1) + 16p\ln 2 + 8\ln(8n+4) + 6\Big)\Big).
\end{aligned}
$$

From here, by replacing s with $\lceil \log_m(\frac{1}{\alpha_c^*})\rceil$ in the above inequality, the result (3.12) follows.

□

5. Conclusions

Here, we showed unified data-dependent generalization bounds for classifiers from *random* convex hulls in the setting of the *general problem* defined above. Such classifiers are generated, for example, by broad classes of algorithms such as SVMs, RBF networks and boosting. The bounds involve the individual complexities of the classifiers introduced by Koltchinskii and Panchenko (2004), such as sparsity of weights and weighted variance over clusters. This was achieved by proving a version of Vapnik's inequality applied to random classes, that is, a bound for relative deviations of frequencies from probabilities for random classes of events (Theorem 1). The results show how various algorithms fit in a single, *general class*. Also, it was indicated that algorithms controlling the individual complexities of the classifiers can produce classifiers with good generalization ability (see Koltchinskii et al. (2003a); Koltchinskii et al. (2003b); Andonova (2004) for some experimental results in the setting of various boosting algorithms). Experimental investigations of the above complexities in the setting of the *general problem* are desirable.

Acknowledgments

The author expresses special thanks to Dmitry Panchenko for introducing me to the area of statistical learning and for his valuable advice and input during the exploration of the above problems. The author also expresses thanks to Vladimir Koltchinskii for valuable communication and input. Valuable remarks and suggestions about the paper and references were given by Sayan Mukherjee, the reviewers and the editor.

References

Andonova, S. *Theoretical and experimental analysis of the generalization ability of some statistical learning algorithms.* PhD thesis, Department of Mathematics and Statistics, Boston University, Boston, MA, 2004.

Anthony, M., Shawe-Taylor, J. A result of Vapnik with applications. *Discrete Applied Mathematics,* 47(3): 207–217, 1993.

Bartlett, P., Shawe-Taylor, J. Generalization performance of support vector machines and other pattern classifiers. *Advances in Kernel Methods. Support Vector Learning. Schölkopf, Burges and Smola (Eds.),* 1, The MIT Press, Cambridge, 1999.

Borowkow, A. A. *Wahrscheinlichkeits–theorie.* Birkhäuser Verlag, 1976.

Cannon, A., Ettinger, M., Hush, D., Scovel, C. Machine Learning with Data Dependent Hypothesis Classes. *Journal of Machine Learning Research,* 2: 335–358, 2002.

Breiman, L. Prediction games and arcing algorithms. *Neural Computation,* 11(7): 1493–1517, 1999.

Dudley, R. M. *Uniform Central Limit Theorems.* Cambridge University Press, 1999.

Feller, W. *An Introduction to probability theory and its applications, volume II.* John Wiley, 1966.

Gat, Y. A bound concerning the generalization ability of a certain class of learning algorithms. Technical Report 548, University of California, Berkeley, CA, 1999.

Heisele, B., Serre, T., Mukherjee, S., Poggio, T. Feature Reduction and Hierarchy of Classifiers for Fast Object Detection in Video Images. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* 2: 18–24, 2001.

Kohonen, T. The self-organizing map. In *Proceedings of IEEE,* 78: 1464–1479, 1990.

Koltchinskii, V., Panchenko, D. Empirical margin distributions and bounding the generalization error of combined classifiers. *The Annals of Statistics,* 30(1), 2002.

Koltchinskii, V., Panchenko, D. Complexities of convex combinations and bounding the generalization error in classification. *submitted,* 2004.

Koltchinskii, V., Panchenko, D., Lozano, F. Bounding the generalization error of convex combinations of classifiers: balancing the dimensionality and the margins. *The Annals of Applied Probability,* 13(1), 2003a.

Koltchinskii, V., Panchenko, D., Andonova, S. Generalization bounds for voting classifiers based on sparsity and clustering. In *Proceedings of the Annual Conference on Computational Learning Theory, Lecture Notes in Artificial Intelligence*, M. Warmuth and B. Schoelkopf (eds.). Springer, New York, 2003b.

Littlestone, N., Warmuth, M. Relating data compression and learnability. Technical Report, University of California, Santa Cruz, CA, 1986.

Panchenko, D. Some extensions of an inequality of Vapnik and Červonenkis. *Electronic Communications in Probability,* 7: 55–65, 2002.

Schapire, R., Freund, Y., Bartlett, P., Lee, W. S. Boosting the margin: A new explanation of effectiveness of voting methods. *The Annals of Statistics,* 26: 1651–1687, 1998.

Schapire, R., Singer, Y. Improved Boosting Algorithms using Confidence-Rated Predictions. *Machine Learning,* 37: 297–336, 1999.

Steinwart, I. Sparseness of Support Vector Machines. *Journal of Machine Learning Research,* 2: 1071–1105, 2003.

Vapnik, V. N., Červonenkis, A. Ya. On the uniform convergence of relative frequencies of event to their probabilities. *Soviet Math. Dokl.,* 9: 915 – 918, 1968.

Vapnik, V. N., Červonenkis A. Ya. *Theory of Pattern Recognition.* Nauka, Moscow (in Russian), 1974.

Vapnik, V. *Statistical Learning Theory.* John Wiley & Sons, New York, 1998.

Vapnik, V. *Estimation of Dependencies Based on Empirical Data.* SpringerVerlag, New York, 1982.

Journal of Machine Learning Research 6 (2005) 341-361 Submitted 11/04; Published 03/05

A Modified Finite Newton Method for Fast Solution of Large Scale Linear SVMs

S. Sathiya Keerthi SATHIYA.KEERTHI@OVERTURE.COM
Dennis DeCoste DENNIS.DECOSTE@OVERTURE.COM
Yahoo! Research Labs
210 South Delacey Avenue
Pasadena, CA 91105, USA

Editor: Thorsten Joachims

Abstract

This paper develops a fast method for solving linear SVMs with L_2 loss function that is suited for large scale data mining tasks such as text classification. This is done by modifying the finite Newton method of Mangasarian in several ways. Experiments indicate that the method is much faster than decomposition methods such as SVM$^{\text{light}}$, SMO and BSVM (e.g., 4-100 fold), especially when the number of examples is large. The paper also suggests ways of extending the method to other loss functions such as the modified Huber's loss function and the L_1 loss function, and also for solving ordinal regression.

Keywords: linear SVMs, classification, conjugate gradient

1. Introduction

Linear SVMs (SVMs whose feature space is the same as the input space of the problem) are powerful tools for solving large-scale data-mining tasks such as those arising in the textual domain. Quite often, these large-scale problems have a large number of examples as well as a large number of features and the data matrix is very sparse (e.g. $>$99.9% sparse "bag of words" in text classification). In spite of their excellent accuracy, SVMs are sometimes not preferred because of the huge training times involved (Chakrabarti et al., 2003). Thus, it is important to have fast algorithms for solving them. Traditionally, linear SVMs have been trained using decomposition techniques such as SVM$^{\text{light}}$ (Joachims, 1999), SMO (Platt, 1999), and BSVM (Hsu and Lin, 2002), which solve the dual problem by optimizing a small subset of the variables in each iteration. Each iteration costs $O(n_{nz})$ time, where n_{nz} is the number of non-zeros in the data matrix ($n_{nz} = mn$ if the data matrix is full where m is the number of examples and n is the number of features). The number of iterations, which is a function of m and the number of support vectors, tends to grow super-linearly with m and thus these algorithms can be inefficient when m is large.

With SVMs two particular loss functions for imposing penalties on slacks (violations on the wrong side of the margin, usually denoted by ξ) have been popularly used. L_1-SVMs penalize slacks linearly (penalty=ξ) while L_2-SVMs penalize slacks quadratically (penalty=$\xi^2/2$). Though both SVMs give good generalization performance, L_1-SVMs are more popularly used because they usually yield classifiers with a much less number of support vectors, thus leading to better classification speed. For linear SVMs the number of support vectors is not a matter of much concern since the final classifier is directly implemented using the weight vector in feature space. In this paper we focus on L_2-SVMs.

Since linear SVMs are directly formulated in the input space, it is possible and worthwhile to think of direct methods of solving the primal problem without using the kernel trick. Primal approaches are attractive because they assure a continuous decrease in the primal objective function.

Recently some promising primal algorithms have been given for training linear classifiers. Fung and Mangasarian (Fung and Mangasarian, 2001) have given a primal version of the least squares formulation of SVMs given by Suykens and Vandewalle (1999). Komarek (2004) has effectively applied conjugate gradient schemes to logistic regression. Zhang et al. (2003) have given an indirect algorithm for linear L_1-SVMs that works by approximating the L_1 loss function by a sequence of smooth modified logistic regression loss functions and then sequentially solving the resulting smooth primal modified logistic regression problems by nonlinear conjugate gradient methods. A particular drawback of that method is its inability to exploit the sparsity property of SVMs: that only the support vectors determine the final solution.

A direct primal algorithm for L_2-SVMs that exploits the sparsity property, called the finite Newton method, was given by Mangasarian (2002). It was mainly presented for problems in which the number of features is small. The main aim of this paper is to introduce appropriate tools that transform this method into a powerfully fast technique for solving large scale data mining problems (with a large number of examples and/or a large number of features). Our main contributions are: (1) we modify the finite Newton method by keeping the least squares nature of the problem intact in each iteration and using exact line search; (2) we bring in special, numerically robust conjugate gradient techniques to implement the Newton iterations; and (3) we introduce heuristics that speed-up the baseline implementation considerably. The result is an algorithm that is numerically robust and very fast. An attractive feature of the algorithm is that it is also very easy to implement; Appendix A gives a pseudocode that can be easily transcribed into a working code.

We show that the method that we develop for L_2-SVMs can be extended in a straight-forward way to the modified Huber's loss function (Zhang, 2004) and, in a slightly more complicated way to the L_1 loss function. We also show how the algorithm can be modifed to solve ordinal regression.

The paper is organized as follows. Section 2 formulates the problem and gives basic results. The modified finite Newton algorithm is developed in Section 3. Section 4 gives full details associated with a practical implementation of this algorithm. Section 5 gives computational experiments demonstrating the efficiency of the method in comparison with standard methods such as SVM$^{\text{light}}$ (Joachims, 1999) and BSVM (Hsu and Lin, 2002). Section 6 suggests ways for extending the method to other loss functions and Section 7 explains how ordinal regression can be solved. Section 8 contains some concluding remarks.

2. Problem Formulation and Some Basic Results

Consider a binary classification problem with training examples, $\{x_i, t_i\}_{i=1}^m$ where $x_i \in R^n$ and $t_i \in \{+1, -1\}$. To obtain a linear classifier $y = w \cdot x + b$, L_2-SVM solves the following primal problem:

$$\min_{(w,b)} \frac{1}{2}(\|w\|^2 + b^2) + \frac{C}{2}\sum_i^m \xi_i^2 \quad \text{s.t.} \;\; t_i(w \cdot x_i + b) \geq 1 - \xi_i \quad \forall\, i \tag{1}$$

where C is the regularization parameter. We have included the $b^2/2$ term so that standard regularized least squares algorithms can be used directly. Our experience shows that generalization performance is not affected by this inclusion. In a particular problem if there is reason to believe that the added term does affect performance, one can proceed as follows: take $\gamma^2 b^2/2$ as the term to be included; then define $\tilde{b} = \gamma b$ as the new bias variable and take the classifier to be $y = w \cdot x + (1/\gamma)\tilde{b}$. The parameter γ can either be chosen to be a small positive value or be tuned by cross validation.

For applying least squares ideas neatly, it is convenient to transform (1) to an equivalent formulation by eliminating the ξ_i's and dividing the objective function by the factor C. This gives[1]

$$\min_{\beta} f(\beta) = \frac{\lambda}{2}\|\beta\|^2 + \frac{1}{2}\sum_{i \in I(\beta)} d_i^2(\beta) \tag{2}$$

where $\beta = (w, b)$, $\lambda = 1/C$, $d_i(\beta) = y_i(\beta) - t_i$, $y_i(\beta) = w \cdot x_i + b$, and $I(\beta) = \{i : t_i y_i(\beta) < 1\}$.

Least Squares SVM (LS-SVM) (Suykens and Vandewalle, 1999) corresponds to (1) with the inequality constraints replaced by the equality constraints, $t_i(w \cdot x_i + b) = 1 - \xi_i$ for all i. When transformed to the form (2), this is equivalent to setting $I(\beta)$=$\{1, \ldots, m\}$; thus, LS-SVM is solved via a single regularized least squares solution. In contrast, the dependance of $I(\beta)$ on β complicates the L_2-SVM solution. In spite of this complexity, it can be advantageous to opt for using L_2-SVMs because they do not allow well-classified examples to disturb the classifier design. This is especially true in problems where the support vectors are a small subset of all examples.[2]

Let us now review several basic results concerning f, some of which are given in Mangasarian (2002). First note that f is a piecewise quadratic function. The presence of the $\lambda\|\beta\|^2/2$ term makes f strictly convex. Thus it has a unique minimizer. f is continuously differentiable in spite of the jumps in $I(\beta)$, the reason for this being that when an index i causes a jump in $I(\beta)$ at some β, its d_i is 0. The gradient of f is given by

$$\nabla f(\beta) = \lambda\beta + \sum_{i \in I(\beta)} d_i(\beta) \begin{pmatrix} x_i \\ 1 \end{pmatrix}. \tag{3}$$

Given an index set $I \subset \{1, \ldots, m\}$, let us define the function f_I as

$$f_I(\beta) = \frac{\lambda}{2}\|\beta\|^2 + \frac{1}{2}\sum_{i \in I} d_i^2(\beta). \tag{4}$$

Clearly f_I is a strictly convex quadratic function and so it has a unique minimizer. It follows directly from (3) that, for any $\bar{\beta}$, $\nabla f(\beta)|_{\beta=\bar{\beta}} = \nabla f_{\bar{I}}(\beta)|_{\beta=\bar{\beta}}$ where $\bar{I} = I(\bar{\beta})$. In fact, there exists an open set around $\bar{\beta}$ in which f and $f_{\bar{I}}$ are identical. It follows that $\bar{\beta}$ minimizes f iff it minimizes $f_{\bar{I}}$.

3. The Modified Finite Newton Algorithm

Mangasarian's finite newton method (Mangasarian, 2002) does iterations of the form

$$\beta_{k+1} = \beta_k + \delta_k p_k,$$

1. To help see the equivalence of (1) and (2), note that at a given (w, b), the minimization of ξ_i^2 in (1) will automatically choose $\xi_i = 0$ for all $i \notin I(\beta)$.
2. We find that L_2-SVMs usually achieve better generalization performance over LS-SVMs. Interestingly, we also find L_2-SVMs often train faster than LS-SVMs, due to sparseness arising from support vectors.

where the search direction p_k is based on a second order approximation of the objective function at β_k:

$$p_k = -H(\beta_k)^{-1}\nabla f(\beta_k).$$

Since f is not twice differentiable at β where at least one of the d_i is zero, $H(\beta)$ is taken to be the generalized Hessian defined by $H(\beta) = \lambda J + C^T DC$ where J is the $n \times n$ identity matrix, C is a matrix whose rows are $(x_i^T, 1)$ and D is a diagonal matrix whose diagonal elements are given by:

$$D_{ii} = \begin{cases} 1 & \text{if } t_i y_i(\beta) < 1 \\ \text{some specific element of } [0,1] & \text{if } t_i y_i(\beta) = 1 \\ 0 & \text{if } t_i y_i(\beta) > 1 \end{cases} \tag{5}$$

Note that examples with indices satisfying $t_i y_i(\beta) > 1$ do not affect $H(\beta)$ and p_k. (This property contributes greatly to the overall efficiency of the method.) The step size δ_k is chosen to satisfy an Armijo condition that ensures convergence, and it is found by applying a halving method of line search in the $[0,1]$ interval. (If C in (1) is sufficiently small then it is shown in Mangasarian (2002) that the fixed step size, $\delta_k = 1$ suffices for convergence.)

We modify the algorithm slightly in two ways. First, we avoid doing anything special for cases where $t_i y_i(\beta) = 1$ occurs. (Essentially, we set $D_{ii} = 0$ in (5) for such cases.) This lets us keep the least squares nature of the problem intact. More precisely, instead of computing the Newton direction p_k, we compute the Newton point, $\beta_k + p_k$, which is the solution of a regularized least squares problem. As we will see in the next section, this has useful implications on stable algorithmic implementation. Second, we do an exact line search to determine δ_k. This feature allows us to directly apply convergence results from nonlinear optimization theory. (In the next section we give a fast method for exact line search.) Thus, at one iteration, given a point β we set $I = I(\beta)$ and minimize f_I to obtain the Newton point, $\bar{\beta}$. Then an exact line search on the ray from β to $\bar{\beta}$ yields the next point of the method. These iterations are repeated till the algorithm converges. The overall algorithm is given below. Implementation details associated with each step are discussed in Section 4.

Algorithm L_2-SVM-MFN.

1. Choose a suitable starting β_0. Set $k = 0$ and go to step 2.

2. Check if β_k is the optimal solution of (2). If so, stop with β_k as the solution. Else go to step 3.

3. Let $I_k = I(\beta_k)$. Solve
$$\min_{\beta} f_{I_k}(\beta). \tag{6}$$
Let $\bar{\beta}$ denote the solution obtained.

4. Do a line search to decrease the "full" objective function, f:
$$\min_{\beta \in L} f(\beta), \tag{7}$$
where $L = \{\beta = \beta_k + \delta(\bar{\beta} - \beta_k) : \delta \geq 0\}$. Let $\delta^\star$ denote the solution of this line search. Set $\beta_{k+1} = \beta_k + \delta^\star(\bar{\beta} - \beta_k)$, $k := k+1$ and go back to step 2 for another iteration.

Theorem 1. Algorithm L_2-SVM-MFN converges to the solution of (1) in a finite number of iterations.

Proof. Let $p_k = (\bar{\beta} - \beta_k)$. Note that $p_k = -H_{I_k}^{-1}\nabla f(\beta_k)$ and $\lambda I \leq H_{I_k} \leq H_{\text{all}}$ where H_{I_k} is the Hessian of f_{I_k} and H_{all} is the Hessian of $f_{\{1,\ldots,m\}}$. By Proposition 1.2.1 of Bertsekas (1999) it follows[3] that $\{\beta_k\}$ converges to the minimizer of f. Proof of finite convergence is exactly as in Mangasarian's proof of finite convergence of his algorithm, and goes as follows. Let $\beta^\star$ denote the minimizer of f and $I^\star = I(\beta^\star)$. Let $O = \{\beta : I(\beta) = I^\star\}$. Clearly O is an open set that contains $\beta^\star$. Since $\{\beta_k\}$ converges to $\beta^\star$, there exists a k such that $\beta_k \in O$. When this happens in step 2 of the algorithm, we get $\bar{\beta} = \beta^\star$ in step 3 and so $\beta_{k+1} = \beta^\star$. ■

4. Practical Implementation

In this section we discuss details associated with the implementation of the various steps of the modified finite Newton algorithm and also introduce some useful heuristics for speeding up the algorithm. The discussion leads to a fast and robust implementation of L_2-SVM-MFN. The final algorithm is also very easy to implement; Appendix A gives a pseudocode that can be easily transcribed into a working code. A number of data sets are used in this section to illustrate the effectiveness of various implementation features. These data sets are described in Appendix B. All our computations were done on a 2.4 GHz machine with Intel Xeon processor and having four Gb RAM.

4.1 Step 1: Initialization

If no guess of β is available, then the simplest starting point is $\beta_0 = 0$. For this point we have $y_i = 0$ for all i and so $I_0 = \{1,\ldots,m\}$. Therefore, with such a zero initialization, the $\bar{\beta}$ obtained in step 3 is exactly the LS-SVM solution.

3. To apply Proposition 1.2.1 of Bertsekas (1999) note the following: (a) Because $\lambda > 0$ and H_{all} is positive definite, condition (1.12) of Bertsekas (1999) holds; (b) Bertsekas (1999) shows that (1.12) implies (1.13) given there; (c) in Bertsekas (1999) exact line search is referred as the minimization rule.

	Adult-9	Web-8	News20	Financial	Yahoo
No β-seeding	60.26	105.16	1321.87	1130.35	11650.56
β-seeding	36.80	24.43	944.27	692.47	4419.89

Table 1: Effectiveness of β-seeding on five data sets. The following 21 C values were used: $\sqrt{2}^k$, $k = -10, -9, \ldots, 9, 10$. All computational times are in seconds.

Suppose we have a guess $\tilde{w}$ for the weight vector w. It is possible that $\tilde{w}$ comes from an inexpensive classification method, such as the Naive-Bayes classifier. In that case it is necessary to rescale $\tilde{w}$ and also choose b_0 so as to form a β_0 that is good for starting the SVM solution. So we set $\beta_0 = (\gamma\tilde{w}, b_0)$ and choose suitable values for γ and b_0. Suppose we also assume that I, a guess of the optimal set of active indices, is available. (If no guess is available, we can simply let I be the set of all training indices.) Then choose γ and b_0 to minimize the cost

$$\frac{\lambda}{2}[\gamma^2\|\tilde{w}\|^2 + b_0^2] + \frac{1}{2}\sum_{i\in I}[\gamma\tilde{w}\cdot x_i + b_0 - t_i]^2. \tag{8}$$

It is easy to check that the resulting γ and b_0 are given by

$$\gamma = (p_{22}q_1 - p_{12}q_2)/d \quad \text{and} \quad b_0 = (p_{11}q_2 - p_{12}q_1)/d. \tag{9}$$

where $p_{11} = \lambda\|\tilde{w}\|^2 + \sum_{i\in I}(\tilde{w}\cdot x_i)^2$, $p_{22} = \lambda + |I|$, $p_{12} = \sum_{i\in I}\tilde{w}\cdot x_i$, $q_1 = \sum_{i\in I} t_i\tilde{w}\cdot x_i$, $q_2 = \sum_{i\in I} t_i$ and $d = p_{11}p_{22} - (p_{12})^2$. Once γ and b_0 are thus obtained, we can set $\beta_0 = (\gamma\tilde{w}, b_0)$ and start the algorithm.[4] Note that the set of initial indices I_0 chosen by the algorithm in step 3 is the set of active indices at β_0 and so it could be different from I.

There is another situation where the above initialization comes in handy. Suppose we solve (2) for one C and want to re-solve (2) for a slightly changed C. Then we can use the $\tilde{w}$ and I that are obtained from the optimal solution of the first value of C to do the above mentioned initialization process for the second value of C. For this situation we have also tried simply choosing $\gamma = 1$ and b_0 equal to the b that is optimal for the first value of C. This simple initialization also works quite well. We will refer to this initialization for the 'slightly changed C' situation as β-*seeding*; seeding is crucial to efficiency when (2) is to be solved for many C values (such as when tuning C via cross-validation). The β-seeding idea used here is very much similar to the idea of α-seeding popularly employed in the solution of SVM duals(DeCoste and Wagstaff, 2000).

Eventhough full details of the final version of our implementation of L_2-SVM-MFN is only developed further below, here let us compare the final implementation[5] with and without β-seeding. Table 1 gives computational times for several data sets. Clearly, β-seeding gives useful speed-ups.

4. To get a better β_0 one can also reset $I = I(\beta_0)$ and repeat (9) and get revised values for γ and b_0. This computation is cheap since $\tilde{w}\cdot x_i$ need not be recomputed.
5. For no β-seeding, the implementation corresponds to the use of 'both heuristics' mentioned in Table 3. For β-seeding, the two heuristics are not used since they do not contribute much when β-seeding is done.

4.2 Step 2: Checking Convergence

Checking of the optimality of β_k is done by first calculating $y_i(\beta_k)$ and $d_i(\beta_k)$ for all i, determining the active index set I_k and then checking if

$$\|\nabla f_{I_k}(\beta_k)\| = 0. \tag{10}$$

For practical reasons it is necessary to employ a tolerance parameter when checking (10). We deal with this important issue below after the discussion of the implementation of step 3.

4.3 Step 3: Regularized Least Squares Solution

The solution of (6) can be approached in one of the following ways: using factorization methods such as QR or SVD; or, using an iterative method such as the conjugate gradient (CG) method. We prefer the latter method due to the following reasons: (a) it can make effective use of knowledge of good starting vectors; and (b) it is much better suited for large-scale problems having sparse data sets. To setup the details of the CG method, let: X be the matrix whose rows are $(x_i^T, 1)$, $i \in I_k$; and t be a vector whose elements are t_i, $i \in I_k$. Then (6) is the same as the regularized least squares problem

$$\min_\beta f_{I_k}(\beta) = \frac{\lambda}{2}\|\beta\|^2 + \frac{1}{2}\|X\beta - t\|^2. \tag{11}$$

This corresponds to the solution of the normal system,

$$(\lambda I + X^T X)\beta = X^T t. \tag{12}$$

With CG methods there are several ways of approaching the solution of (11/12). A simple approach is to solve (12) using the CG method meant for solving positive definite systems. However, such an approach will not be numerically well-conditioned, especially when λ is small. As pointed out by Paige and Saunders (Paige and Saunders, 1982) this is mainly due to the explicit use of vectors of the form $X^T X p$. An algorithm with better numerical properties can be easily derived by an algorithmic rearrangement that is special to the regularized least squares solution, which makes use of the intermediate vector Xp. LSQR (Paige and Saunders, 1982) and CGLS (Björck, 1996) are two such special CG algorithms. Another very important reason for using one of these algorithms (as opposed to using a general purpose CG solver) is that, for the special methods it is easy to derive a good stopping criterion to approximately terminate the CG iterations using the intermediate residual vector. We discuss this issue in more detail below.

For our work we have used the version of the CGLS algorithm given in Algorithm 3 of Frommer and Maaß (Frommer and Maaß, 1999) which uses initial seed solutions neatly.[6] Following is the CGLS algorithm for solving (11).

Algorithm CGLS. Set $\beta^0 = \beta_k$ (where β_k is as in steps 2 and 3 of Algorithm L_2-SVM-MFN). Compute $z^0 = t - X\beta^0$, $r^0 = X^T z^0 - \lambda\beta^0$,[7] set $p^0 = r^0$ and do the following steps for $j = 0, 1, \ldots$

6. Frommer and Maaß have also given interesting variations of the CGLS method for efficiently solving (12) for several values of λ. But we have not tried those methods in this work.
7. It is useful to note that, at any point of the CGLS algorithm $-z^j$ is the vector containing the classifier residuals, d_i, $i \in I_k$ and r^j is the negative of the gradient of $f_{I_k}(\beta)$ at β^j. At the beginning ($j = 0$), z^0 and r^0 are already available in view of the computations in step 2 of Algorithm L_2-SVM-MFN. This fact can be used to gain some efficiency.

```
$q^j = X p^j$
$\gamma^j = \|r^j\|^2/(\|q^j\|^2 + \lambda\|p^j\|^2)$
$\beta^{j+1} = \beta^j + \gamma^j p^j$
$z^{j+1} = z^j - \gamma^j q^j$
$r^{j+1} = X^T z^{j+1} - \lambda\beta^{j+1}$
If $r^{j+1} = 0$ stop with $\beta^{j+1}$ as the solution.
$\omega^j = \|r^{j+1}\|^2/\|r^j\|^2$
$p^{j+1} = r^{j+1} + \omega^j p^j$
```

There are exactly two matrix-vector operations in each iteration; sparsity of the data matrix can be effectively used to do these operations in $O(n_z)$ time where n_z is the number of non-zero elements in the data matrix.

Let us now discuss the convergence properties of CGLS. It is known that the algorithm will take at most l iterations where l is the rank of X. Note that $l \leq \min\{m,n\}$ where m is the number of examples and n is the number of features. The actual number of iterations required to achieve good practical convergence is usually much smaller than $\min\{m,n\}$ and it depends on the number of singular values of X that are really significant.

Stopping the CG iterations with the right accuracy is very important because, an excessively accurate solution would lead to too much work while an inaccurate solution will not give good descent. Using a simple absolute tolerance on the size of the gradient of f_{I_k} in order to stop is a bad idea, even for a given data set since the typical size of the gradient varies a lot as λ is varied over a range of values. For a method such as CGLS it is easy to find effective practical stopping criteria. We can decide to stop when the negative gradient vector r^{j+1} has come near zero up to some relative precision. To do this we can use the bound, $\|r^{j+1}\| \leq \|X\|\|z^{j+1}\| + \|\lambda\beta^{j+1}\|$. Thus a good stopping criterion is

$$\|r^{j+1}\| \leq \varepsilon(\rho\|z^{j+1}\| + \lambda\|\beta^{j+1}\|),$$

where $\rho = \|X\|$. Since X varies at different major iterations of the L_2-SVM-MFN algorithm, we can simply take $\rho = \|\hat{X}\|$ where $\hat{X}$ is the entire $m \times n$ data matrix.

In most datamining tasks the data is normalized so that all values in the data matrix are in the unity range. For such data sets we have $\|X\| \leq \sqrt{n}$. One can take a conservative approach and simply use the stopping criterion

$$\|r^{j+1}\| \leq \varepsilon\|z^{j+1}\|. \tag{13}$$

We have found this criterion to be very effective and have used it for all the computational experiments reported in this paper. The parameter ε is a relative tolerance parameter. A value of $\varepsilon = 10^{-6}$, which roughly yields solutions accurate up to six decimal digits, is a good choice.

Since $r = -\nabla f_{I_k}$ we can apply exactly the same criteria as in (13) for approximately checking (10) also. Almost always, termination of L_2-SVM-MFN occurs when, after the least squares solution at step 3, exact line search in step 4 gives $\beta^{k+1} = \bar{\beta}$ (i.e., $\delta^\star = 1$), and the active set remains unchanged, i.e., $I(\beta^k) = I(\bar{\beta}) = I(\beta^{k+1})$.

Let us illustrate the effectiveness of the CGLS method using the LS-SVM solution as an example. For LS-SVM we can set $I = \{1,\ldots,m\}$ and solve (12) using the CGLS method mentioned above; let us refer to such an implementation as LS-SVM-CG. In their proximal SVM implementation of LS-SVM, Fung and Mangasarian (2001) solve (12) using Matlab routines that employ factorization techniques on X^TX. For large and sparse data sets it is much more efficient to use CG methods and avoid the formation of X^TX and its factorization. Table 2 illustrates this fact using two

Table 2: Ratio of the computational cost (averaged over $C = 2^{-5}, 2^{-4}, \ldots, 2^{5}$) of the proximal SVM algorithm to that of LS-SVM-CG. s is the sparsity factor, $s = n_{nz}/(nm)$.

	Australian ($n = 14$, $m = 690$, $s = 1.0$)	Web-7 ($n = 300$, $m = 24692$, $s = 0.04$)
Ratio	0.72	15.72

data sets. The complexity of the original proximal SVM implementation is $O(n_{nz}n + n^3)$ whereas the complexity of the CGLS implementation is $O(n_{nz}l)$, where l is the number of iterations needed by the CGLS algorithm.[8] When the data matrix is dense (e.g., *Australian*) the factorization approach is slightly faster than the CGLS approach. But, when the data matrix is sparse (e.g., *Web-7*) the CGLS approach is considerably faster, even with n being small.

Similar observations hold for Mangasarian's finite Newton method as well as our L_2-SVM-MFN. When the data matrix is sparse, factorization techniques are much more expensive compared to CG methods, even when n is not too big. Of course, factorization methods get completely ruled out when both m and n are large.

4.4 Step 4: Exact Line Search

Let $\beta(\delta) = \beta_k + \delta(\bar{\beta} - \beta_k)$. The one dimensional function, $\phi(\delta) = f(\beta(\delta))$ is a continuously differentiable, strictly convex, piecewise quadratic function. To determine the minimizer of this function analytically, we compute the points at which the second derivative jumps. For any given i, let us define: $\delta_i = (t_i - y_i^k)/(\bar{y}_i - y_i^k)$, where $\bar{y}_i = y_i(\bar{\beta})$ and $y_i^k = y_i(\beta_k)$. The jump points mentioned above are given by

$$\Delta = \Delta_1 \cup \Delta_2, \tag{14}$$

where

$$\Delta_1 = \{\delta_i : i \in I_k, t_i(\bar{y}_i - y_i^k) > 0\} \quad \text{and} \quad \Delta_2 = \{\delta_i : i \notin I_k, t_i(\bar{y}_i - y_i^k) < 0\}. \tag{15}$$

For Δ_1 we are not using i with $t_i(\bar{y}_i - y_i^k) \leq 0$ because they do not cause switching at a positive δ; similarly, for Δ_2 we are not using i with $t_i(\bar{y}_i - y_i^k) \geq 0$.

Take one $\delta_i \in \Delta_1$. When we increase δ across δ_i, the index i leaves $I(\beta(\delta))$. Thus, the term $d_i^2/2$ has to be left out of the objective function for all $\delta > \delta_i$. Similarly, for $\delta_i \in \Delta_2$, when we increase δ across δ_i, the index i enters $I(\beta(\delta))$. Thus, the term $d_i^2/2$ has to be included into the objective function for all $\delta > \delta_i$.

The slope, $\phi'(\delta)$ is a continuous piecewise linear function that changes its slope only at one of the δ_i's. The optimal point $\delta^\star$ is the point at which $\phi'(\delta)$ crosses 0.[9] To determine this point we first sort all δ_i's in Δ in non-decreasing order. To simplify the notations, let us assume that δ_i, $i = 1, 2, \ldots$ denotes that ordering. Between δ_i and δ_{i+1} we know that $\phi'(\delta)$ is a linear function. Just for doing calculations extend this line both sides (left and right) to meet the $\delta = 0$ and $\delta = 1$ vertical lines.

8. It is useful to note here that the proximal SVM implementation solves (12) exactly while LS-SVM-CG uses the practical stopping condition (13) that contributes further to its efficiency.
9. Note that this point may not necessarily be at one of the δ_i's.

Let us call the ordinate values at these two meeting points as l_i and r_i respectively. It is very easy to keep track of the changes in l_i and r_i as indices get dropped and added to the active set of indices.

We move from left to right to find the zero crossing of $\phi'(\delta)$. At the beginning we are at $\delta_0 = 0$. Between δ_0 and δ_1 we have I_k as the active set. It is easy to get, from the definition of $\phi(\delta)$ that

$$l_0 = \lambda\beta_k \cdot (\bar{\beta} - \beta_k) + \sum_{i \in I_k} (y_i^k - t_i)(\bar{y}_i - y_i^k) \tag{16}$$

and

$$r_0 = \lambda\bar{\beta} \cdot (\bar{\beta} - \beta_k) + \sum_{i \in I_k} (\bar{y}_i - t_i)(\bar{y}_i - y_i^k). \tag{17}$$

(If, at step 3 of the L_2-SVM-MFN algorithm, we solve (6) exactly, then it is easy to check that $r_0 = 0$. However, in view of the use of the approximate termination mentioned in (13) it is better to compute r_0 using (17).) Find the point where the line joining $(0, l_0)$ and $(1, r_0)$ points on the (δ, ϕ') plane crosses zero. If the zero crossing point of this line is between 0 and δ_1 then that point is $\delta^\star$. If not, we move over to searching between δ_1 and δ_2. Here l_1 and r_1 need to be computed. This can be done by a simple updating over l_0 and r_0 since only the term $d_i^2/2$ enters or leaves. Thus, for a general situation where we already have l_i, r_i computed for the interval δ_i to δ_{i+1} and we need to get l_{i+1}, r_{i+1} for the interval δ_{i+1} to δ_{i+2}, we use the update formula

$$l_{i+1} = l_i + s(y_i^k - t_i)(\bar{y}_i - y_i^k) \quad \text{and} \quad r_{i+1} = r_i + s(\bar{y}_i^k - t_i)(\bar{y}_i - y_i^k), \tag{18}$$

where $s = -1$ if $\delta_i \in \Delta_1$ and $s = 1$ if $\delta_i \in \Delta_2$. Thus we keep moving to the right until we get a zero satisfying the condition that the root determined by interpolating $(0, l_i)$ and $(1, r_i)$ lies between δ_i and δ_{i+1}. The process is bound to converge since we know the existence of the minimizer (we are dealing with a strictly convex function). In a typical application of the above line search algorithm, many δ_i's are crossed before $\delta^\star$ is reached, especially in the early stages of the algorithm, causing $|I(\beta_{k+1})|$ to be much different from $|I(\beta_k)|$. This is the crucial step where the support vectors of the problem get identified.

The complexity of the above exact line search algorithm is $O(m \log m)$. Since the least squares solution (step 3) is much more expensive, the cost of exact line search is negligible.

4.5 Complexity Analysis

The bulk of the cost of the algorithm is associated with step 3, which only deals with examples that are active at the current point. (The full set of examples is involved only in step 4.) This crucial factor greatly contributes to the overall efficiency of the algorithm. The number of iterations, i.e., loops of steps 2-4 is usually small, say 5-20. Thus, the empirical complexity of the algorithm is $O(n_{nz} l_{av})$ where l_{av}, the average number of CG iterations in step 3, is bounded by the rank of the data matrix and so $l_{av} \leq \min\{m, n\}$. As already mentioned, l_{av} usually turns out to be much smaller than both, m and n. For example, when applied to the *financial* data set that has 198788 examples and 252472 features, for $C = 1$ and $\beta = 0$ initialization, L_2-SVM-MFN took 11 iterations, with $l_{av} = 102$.

4.6 Speed-up Heuristics

Suppose we are solving a problem for which the number of support vectors, i.e., $|I(\beta^\star)|$ is a small fraction of m, and we use the initialization, $\beta_0 = 0$. Since $I(\beta_0) = \{1, \ldots, m\}$, step 3 corresponds to

	Adult-9	Web-8	News20	Financial	Yahoo
No heuristics	7.28	11.79	98.06	456.85	1443.23
Heuristic 1	5.12	9.24	67.85	70.86	904.99
Heuristic 2	3.57	4.17	90.54	202.62	848.99
Both heuristics	3.00	3.90	52.73	62.18	524.16
SV fraction	0.605	0.219	0.650	0.068	0.710

Table 3: Effectiveness of the two speed-up heuristics on five data sets. The value $C = 1$ was used. All computational times are in seconds. SV fraction is the ratio of the number of support vectors to the number of training examples.

solving an unnecessarily large least squares problem; it is wasteful to solve it accurately. One (or both) of the following two heuristics can be employed to avoid this.

Heuristic 1. Whenever β_0 is a crude approximation (say, $\beta_0 = 0$), terminate the least squares solution of (6) after a fixed, small number (say, 10) of CGLS iterations at the first call to step 3. Even with the crude β thus generated, the following step 4 usually leads to a point β_1 with $|I(\beta_1)|$ much smaller than m, and a good bulk of the non-support vectors get identified correctly.

Heuristic 2. First run the L_2-SVM-MFN algorithm using a crude tolerance, say $\varepsilon = 10^{-2}$. Use the β thus generated as the starting vector and make another run with $\varepsilon = 10^{-6}$, the final desired accuracy.

Table 3 gives the effectiveness of the above heuristics on a few data sets. Clearly both heuristics are useful. It is not easy to say which one is more effective and so using both of them is the appropriate thing to do. This gives at least a 2-fold speed-up. As expected, the amount of speed-up is big if the fraction of examples that are support vectors is small. The pseudocode of Appendix B uses both heuristics.

A third heuristic may also be used when working with a very large number of examples. First choose a small random subset of the examples and run the algorithm. Then use the β thus generated to seed a second run, this time using all the examples for training.

We end this section on implementation by explaining how a solution of the SVM dual can be obtained after L_2-SVM-MFN solves the primal.

4.7 Obtaining a Dual Solution

Note that the SVM dual variables, α_i, $i = 1, \ldots, m$ are not involved anywhere in the algorithm. But it is easy to recover them once we solve (2) using L_2-SVM-MFN. From the structure of (3) it is easy to see that $\alpha_i = -t_i d_i/\lambda$, if $i \in I(\beta)$ and $\alpha_i = 0$ otherwise. In a practical solution we do not get the true solution due to the use of (13). In such a situation it is useful to ask as to how well the α defined above satisfies the KKT optimality conditions of the dual. This can be easily done. After computing α as mentioned above, set $\hat{\beta} = \sum_i \alpha_i t_i (x_i^T, 1)^T$, $\xi_i = \lambda \alpha_i \ \forall i$, $g_i = t_i y_i(\hat{\beta}) + \xi_i - 1 \ \forall i$, and obtain the maximum dual KKT violation as $\max\{\max_{i:\alpha_i>0} |g_i|, \max_{i:\alpha_i=0} \max\{0, -g_i\}\}$. If, keeping the maximum dual KKT violation within some specified tolerance (say, $\tau = 0.001$) is important for some reason, then one can proceed as follows. First solve L_2-SVM-MFN using $\varepsilon = 10^{-3}$ and then check the maximum dual KKT violation as described above. If it does not satisfy the required tolerance then continue the L_2-SVM-MFN solution with a suitably chosen smaller value of ε.

5. Comparison with SVM$^{\text{light}}$ and BSVM

The experiments of this section compare L_2-SVM-MFN against two dual-based methods: the popular SVM$^{\text{light}}$ (Joachims, 1999)[10] and the more modern BSVM (Hsu and Lin, 2002). In order to make a proper comparison L_2-SVM-MFN was forced to satisfy the same dual KKT tolerance of $\tau = 0.001$ as the other methods. (The procedure given at the end of the last section was used to do this.) We used default `-q` (subproblem size) for SVM$^{\text{light}}$ & BSVM; other values tried were not faster.[11] We also tried SMO (Platt, 1999), but found it slower than the others for these linear problems. An explanation for this is given by Kao et al (Kao et al., 2004) in Section 4 of their paper via the fact that, for the linear SVM implementation, the cost of updating the gradient of the dual is independent of the number of dual variables that are optimized in each basic iteration.

Tables 4-6 report training times and 10-fold cross-validation (CV) error rates for *Adult-9*, *Web-8* and *News20* data sets.

We show training times for various C's, with optimal (lowest) mean CV error rate for each method shown in bold. Due to the different loss functions used, a direct comparison of these methods is challenging and necessarily approximate on non-separable data. Therefore, in the following tables we show results for a range of C values around values of C yielding minimum cross-validation errors for each of the three methods. A reasonable conservative speedup for our method can then be determined by selecting the slowest training time for a near-optimal C value for our method versus the fastest training time for a near-optimal C value for an alternative method.

For example, for *Adult-9* one could compare the time for SVM$^{\text{light}}$'s CV-optimal (112.6 secs) versus the time for L_2-SVM-MFN's CV-optimal (1.6 secs), yielding a speedup ratio of 70.4. Alternatively, *nearly*-CV-optimal cases for *Adult-9* (e.g. 15.23%>15.22% for SVM$^{\text{light}}$ with C=2^{-3}) yield other speedups (e.g. 13.2 if both SVM$^{\text{light}}$ and L_2-SVM-MFN use C=2^{-3}). Over all such nearly-optimal cases for all three data sets, speedups are consistently significant (e.g. 4-100 over SVM$^{\text{light}}$ and 4-40 over BSVM). Even for *News20* which has more than a million features[12] L_2-SVM-MFN is more than four times faster than SVM$^{\text{light}}$ and BSVM.

For the *Yahoo* data set having a million examples, SVM$^{\text{light}}$ and BSVM could not complete the solution even after one full day, while L_2-SVM-MFN took only about 10 minutes to obtain a solution.

We also did an experiment to study how the algorithms scale with m, the number of examples. Figure 1 gives log-log plots showing the variation of training times as a function of m, for four of the conventional *Adult* and *Web* subsets. The times plotted for each data subset/method pair are for the corresponding CV-optimal C's. These plots show that not only was L_2-SVM-MFN always faster, but it also scaled much better with m.

10. We report results using version 5.0 of SVM$^{\text{light}}$. We also tried the newer version 6.0, but found for our particular experiments with linear kernels that it was no faster, and sometimes even slower.
11. Specifically, the values used for q were 10 for SVM$^{\text{light}}$ and 30 for BSVM.
12. Dual algorithms such as SVM$^{\text{light}}$ and BSVM are efficient when the number of features is large. Their cost scales linearly with the number of features.

	SVM$^{\text{light}}$		BSVM		L_2-SVM-MFN	
$\log_2 C$	secs	CV%	secs	CV%	secs	CV%
-4.5	16.8	15.29	7.3	15.29	1.2	15.30
-4.0	15.3	15.28	8.1	15.27	1.3	15.26
-3.5	18.6	15.25	8.9	15.27	1.4	15.22
-3.0	21.1	15.23	10.0	15.23	**1.6**	**15.21**
-2.5	25.8	15.24	11.7	15.23	1.8	15.21
-2.0	44.4	15.25	13.3	15.24	1.9	15.23
-1.5	47.0	15.23	15.9	15.24	2.2	15.23
-1.0	58.9	15.26	20.3	15.25	3.1	15.22
-0.5	80.9	15.23	25.8	15.24	2.9	15.22
0.0	**112.6**	**15.22**	**32.4**	**15.22**	2.9	15.23
0.5	189.2	15.23	41.7	15.23	3.1	15.23
1.0	235.9	15.24	54.4	15.22	3.5	15.23

Table 4: Results for *Adult-9*

	SVM$^{\text{light}}$		BSVM		L_2-SVM-MFN	
$\log_2 C$	secs	CV%	secs	CV%	secs	CV%
-0.5	14.3	1.36	11.4	1.35	3.2	1.34
0.0	14.9	1.35	15.8	1.35	4.2	1.34
0.5	19.6	1.34	20.7	1.34	5.0	1.33
1.0	29.1	1.34	28.4	1.34	**5.0**	**1.33**
1.5	46.8	1.33	-	1.33	4.7	1.33
2.0	**60.3**	**1.33**	61.4	1.33	5.7	1.33
2.5	110.5	1.33	**82.4**	**1.33**	7.6	1.33
3.0	131.6	1.33	139.0	1.33	8.9	1.34
3.5	232.6	1.33	191.4	1.33	10.8	1.34
4.0	279.8	1.34	282.9	1.34	10.8	1.34
4.5	347.5	1.35	441.1	1.35	11.6	1.34
5.0	615.1	1.35	647.1	1.34	11.2	1.34

Table 5: Results for *Web-8*

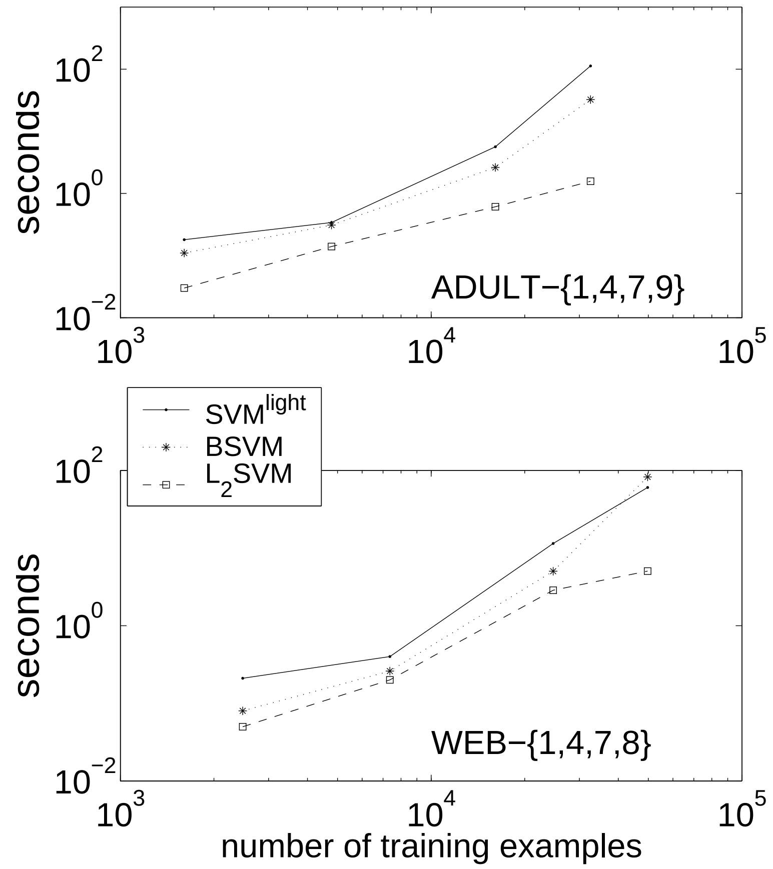


Figure 1: Training time versus *m* for the Adult and Web data sets, on a log-log plot. Note that the vertical axes are only marked at 10^{-2}, 10^0 and 10^2.

	SVM$^{\text{light}}$		BSVM		L_2-SVM-MFN	
$\log_2 C$	secs	CV%	secs	CV%	secs	CV%
0.0	335.9	2.93	330.4	2.93	69.3	3.35
0.5	407.6	2.82	392.6	2.81	55.4	3.16
1.0	437.1	2.78	436.1	2.78	84.5	3.07
1.5	442.2	2.78	434.4	2.78	62.7	2.98
2.0	**466.8**	**2.74**	**437.3**	**2.73**	76.0	2.89
2.5	466.9	2.77	432.3	2.78	75.0	2.88
3.0	455.9	2.85	450.4	2.85	91.6	2.88
3.5	471.3	3.02	439.0	3.02	**98.0**	**2.86**
4.0	513.7	3.07	432.2	3.07	114.0	2.86
4.5	554.7	3.07	437.0	3.07	120.4	2.89
5.0	525.7	7.76	421.6	3.07	159.2	2.90
5.5	535.7	3.07	426.5	3.08	193.9	2.97

Table 6: Results for *News20*

6. Extension to Other Loss Functions

The previous sections addressed the solution of L_2-SVM, i.e., the SVM primal problem that uses the L_2 loss function:

$$\min_\beta f(\beta) = \frac{\lambda}{2}\|\beta\|^2 + \sum_i L(\xi_i) \tag{19}$$

where $\xi_i = 1 - t_i y_i(\beta)$ and $L = L_2$ where

$$L_2(\xi_i) = \begin{cases} 0 & \text{if } \xi_i \le 0 \\ \xi_i^2/2 & \text{if } \xi_i > 0 \end{cases} \tag{20}$$

The modified Newton algorithm can be adapted for other loss functions too. We briefly explain how to do this for the following loss functions: the modifed Huber's loss function (Zhang, 2004) and the L_1 loss function.

Consider, first, the modified Huber's loss function. The solution for this loss function forms the basis of the solution for the L_1 loss function. Recently, Zhang (Zhang, 2004) pointed out that the modified Huber's loss function has some attractive theoretical properties. The loss function is given by

$$L_h(\xi_i) = \begin{cases} 0 & \text{if } \xi_i \le 0 \\ \xi_i^2/2 & \text{if } 0 < \xi_i < 2 \\ 2(\xi_i - 1) & \text{if } \xi_i \ge 2 \end{cases} \tag{21}$$

With $L = L_h$, the primal objective function f in (19) is strictly convex, continuously differentiable and piecewise quadratic, very much as when (20) is used. So the extension of the modified Newton algorithm to L_h is rather easy. A basic iteration proceeds as follows. Given β_k let $I_0 = \{i : \xi_i(\beta_k) \le 0\}$, $I_1 = \{i : 0 < \xi_i(\beta_k) < 2\}$ and $I_2 = \{i : \xi_i(\beta_k) \ge 2\}$. The natural quadratic approximation of f to minimize is the one which keeps these index sets unchanged, i.e.,

$$\min_\beta \tilde{f}(\beta) = \frac{\lambda}{2}\|\beta\|^2 + \frac{1}{2}\sum_{i\in I_1}(y_i(\beta) - t_i)^2 - 2\sum_{i\in I_2} t_i y_i(\beta). \tag{22}$$

Let

$$q = \frac{2}{\lambda} \sum_{i \in I_2} t_i \begin{pmatrix} x_i \\ 1 \end{pmatrix}$$

and $\tilde{\beta} = \beta - q$ so that (22) can be equivalently rewritten as the solution of

$$\min_{\beta} \bar{f}(\beta) = \frac{\lambda}{2} \|\beta - q\|^2 + \frac{1}{2} \sum_{i \in I_1} (y_i(\beta) - t_i)^2. \tag{23}$$

This is nothing but a regularized least squares solution that is shifted in β space; the CG techniques described in Section 4 can be used to solve for $\tilde{\beta} = \beta - q$ and then β can be obtained. The exact line search for minimizing f on a ray is only slightly more complicated than the one in Section 4: with (21) we need to watch for jumps of examples from/to three sets of the type I_0, I_1 and I_2 defined above. The proof of finite convergence of the overall algorithm is very much as for the L_2 loss function.

Let us now consider the L_1 loss function given by

$$L_1(\xi_i) = \begin{cases} 0 & \text{if } \xi_i \le 0 \\ \xi_i & \text{if } \xi_i > 0. \end{cases} \tag{24}$$

Choose τ, a positive tolerance parameter and define $\tilde{\xi} = (\xi_i/\tau) + 1$. The L_1 loss function can be approximated by $L_h(\tilde{\xi})$. Thus, we can solve the primal problem corresponding to the L_1 loss function as follows. Take a sequence of τ values, say $\tau_j = 2^{-j}$, $j = 0, 1, \ldots$. Start by solving the problem for $j = 0$. Use the β thus obtained to seed the solution of the problem for $j = 1$ and so on until a solution that approximates the true solution of the L_1 loss function satisfactorily is obtained. This is only a rough outline of the main scheme. Several details need to be worked out in order to arrive at an overall method that is actually very efficient. Currently we are working on these details; we will report the results in a future paper.

Recently Zhang et al (Zhang et al., 2003) gave a primal algorithm for SVMs with L_1 loss function in which a modified logistic regression function is used to approximate the L_1 loss function and a sequential approximation scheme similar to what we described above is employed. Our method is expected to be more efficient since the approximating loss function (modified Huber) helps keep the sparsity propery, i.e., examples with $|\xi_i| \ge \tau$ are inactive during the solution of the linear least squares problem at each iteration. However, this claim needs to be corraborated by proper implementation of both methods and detailed numerical experiments.

7. Extension to Ordinal Regression

In this section we explain how the L_2-SVM-MFN algorithm can be adapted to solve ordinal regression problems. In ordinal regression the target variable, t_i takes a value from a finite set, say, $\{1, 2, \ldots, p\}$. Thus, p is the total number of possible ordinal values. Let $J_s = \{i : t_i = s\}$. Let w denote the weight vector and $y_i(w) = w \cdot x_i$ denote the 'score' of the SVM for the i-th example. To set up the SVM formulation we follow the approach given in Chu and Keerthi (2005) and use $p-1$ thresholds, b_s, $s = 1, \ldots, p-1$ to divide the scores into p bins so that the interval, (b_{s-1}, b_s) is assigned for examples which have $t_i = s$.[13] Let β denote the vector which contains w together with

13. To make this statement properly, we take $b_0 = -\infty$ and $b_p = \infty$.

b_s, $s = 1, \ldots, p-1$. For a given β and an $s \in \{1, \ldots, p-1\}$ define the following 'margin-violating' index sets:

$$\mathcal{L}_s(\beta) = \{i : i \in J_l \text{ for some } l \leq s \text{ and } y_i(w) - b_s > -1\}$$

$$\mathcal{U}_s(\beta) = \{i : i \in J_l \text{ for some } l > s \text{ and } y_i(w) - b_s < 1\}.$$

Then the primal SVM problem can be written as

$$\min_\beta f(\beta) = \frac{\lambda}{2}\|\beta\|^2 + \sum_{s=1}^{p-1}\left(\frac{1}{2}\sum_{i\in\mathcal{L}_s(w)}(y_i(w) - b_s + 1)^2 + \frac{1}{2}\sum_{i\in\mathcal{U}_s(w)}(y_i(w) - b_s - 1)^2\right). \tag{25}$$

A nice property of the above formulation is that, as shown in Chu and Keerthi (2005), the solution of (25) automatically satisfies the condition, $b_1 \leq b_2 \leq \cdots \leq b_{p-1}$.

Clearly, f is a differentiable, strictly convex, piecewise quadratic function of β, very much like the f in (2). So, the extension of L_2-SVM-MFN to solve (25) is easy. A basic iteration goes as follows. Given β_k, let $\bar{\mathcal{L}}_k = \mathcal{L}_s(\beta_k)$, $\bar{\mathcal{U}}_k = \mathcal{U}_s(\beta_k)$ for $s = 1, \ldots, p-1$ and solve the following quadratic approximation of f corresponding to keeping those index sets unchanged:

$$\min_\beta \tilde{f}(\beta) = \frac{\lambda}{2}\|\beta\|^2 + \sum_{s=1}^{p-1}\left(\frac{1}{2}\sum_{i\in\bar{\mathcal{L}}_k}(y_i(w) - b_s + 1)^2 + \frac{1}{2}\sum_{i\in\bar{\mathcal{U}}_k}(y_i(w) - b_s - 1)^2\right). \tag{26}$$

Let $\bar{\beta}$ denote the solution of (26). Exact line search to minimize f on the ray from β_k to $\bar{\beta}$ is more complicated than the line search we described in Section 4, but it is quite easy to program in code; also, if p is small, the algorithm is not expensive. As in Section 4, we need to identify the points along the ray at which jumps in the second derivative of f take place. Take one example, say the i-th. Let $\bar{\beta} = (\bar{w}, \bar{b}_1, \ldots, \bar{b}_{p-1})$, $\bar{y}_i = y_i(\bar{w})$ and $l = t_i$. For each $s = 1, \ldots, l-1$, calculate δ_{si} such that $y_i(\beta_k) + \delta_{si}(\bar{y}_i - y_i(\beta_k)) = b_s + 1$. Similarly, for each $s = l, \ldots, p-1$, calculate δ_{si} such that $y_i(\beta_k) + \delta_{si}(\bar{y}_i - y_i(\beta_k)) = b_s - 1$. Each positive δ_{si} is a point where the second derivative jumps. By calculating all such points (there are at most pm of them, where m is the number of training examples), sorting them and using the ideas of Section 4 to locate the minimizer where the slope crosses the zero value, exact line search can be performed. The proof of convergence of the overall algorithm is very much similar to the proof of Theorem 1.

The ideas outlined above for the L_2 loss function can be extended to other loss functions such as the modified Huber's loss function and the L_1 loss function using the ideas of Section 6.

8. Conclusion

In this paper we have modified the finite Newton method of Mangasarian in several ways to obtain a very fast method for solving linear SVMs that is easy to implement and trains much faster than existing alternative SVM methods, making it attractive for solving large-scale classification problems.

We have also tried another method for linear L_2-SVMs. This corresponds to the direct application of a nonlinear CG method (such as Polak-Ribierre) to (2); note that f is a differentiable function. This method also works well, but it is not as efficient and numerically robust as L_2-SVM-MFN. One of the main reasons for this is that the bulk of the computations of L_2-SVM-MFN takes place in the CGLS iterations which operate only with potential support vectors. On the other hand, the nonlinear CG method has to necessarily deal with all examples in each iteration, unless clever shrinking strategies are designed.

It is interesting to ask if the modified finite Newton algorithm can be extended to nonlinear kernels. If (6) is solved via its dual (say, by using an algorithm for LS-SVMs) the new algorithm can indeed be extended to nonlinear kernels. That would be an interesting primal algorithm that is implemented using dual variables. But it is not yet clear whether such an algorithm will be more efficient than existing good dual methods (e.g. SMO or SVM$^{\text{light}}$).

Appendix A. A Pseudocode for L_2-SVM-MFN

Below, β represents the current point; y and I denote the output vector and active index set at β. $F = f(\beta)$ and $I_{all} = \{1,\dots,m\}$.

1. *Initialization.*

   - If no initial guess of β is available, set $ini = 0$, $\beta = 0$, $y_i = 0\ \forall i \in I_{all}$ and $I = I_{all}$.
   - If a guess of w is obtained from another method (say, the Naive Bayes method), set $ini = 0$, use (9) to form β and then compute $y_i\ \forall i \in I_{all}$ and the active index set I at β.
   - If continuing the solution from one C value to another nearby C value, set $ini = 1$ and simply start with the β, y_i, $i \in I_{all}$ and I available from the previous C solution.

   If $ini = 0$ set $\varepsilon = 10^{-2}$ and *Need_Second_Round*=1. If $ini = 1$ set $\varepsilon = 10^{-6}$ and *Need_Second_Round*=0. Compute $F = f(\beta)$.[14] Set $F_{previous} = F$.

2. Set $iter = 0$ and $itermax = 50$.

3. Define X to be a restricted data matrix whose rows are $(x_i^T, 1)$, $i \in I_k$ and t to be the corresponding target vector whose elements are t_i, $i \in I_k$. (X and t are defined just for stating the steps easily here. In the actual implementation there is no need to actually form them.[15]) Set: $iter = iter + 1$, $\bar{\beta} = \beta$, $z = t - X\beta$, $r = X^T z - \lambda\beta$, $\phi_1 = \|r\|^2$, $p = r$, $\phi_2 = \phi_1$. If ($ini = 0$ and $iter = 1$) set $cgitermax = 10$; else set $cgitermax = 5000$. Set $cgiter = 0$, $optimality = 0$ and $exit = 0$.

4. Repeat the following steps until $exit = 1$ occurs:

```
        $cgiter = cgiter + 1$
        $q = Xp$, $\phi_3 = \|q\|^2$
        $\gamma = \phi_1/(\phi_3 + \lambda\phi_2)$, $\bar{\beta} = \bar{\beta} + \gamma p$
        $z = z - \gamma q$, $\phi_4 = \|z\|^2$
        $r = -\lambda\bar{\beta} + X^T z$, $\phi_1^{\text{old}} = \phi_1$, $\phi_1 = \|r\|^2$
        If $\phi_1 \le \varepsilon^2\phi_4$ set $optimality = 1$
        If ($optimality = 1$ or $cgiter \ge cgitermax$) set $exit = 1$
        $\omega = \phi_1/\phi_1^{\text{old}}$, $p = r + \omega p$, $\phi_2 = \|p\|^2$
```

14. If $\beta = 0$ then note that $F = m/2$.
15. It is ideal to store the input data, $\{x_i, t_i\}$ in the SVM$^{\text{light}}$ format where each example is specified by the target value together with a bunch of (feature-index,value) pairs corresponding to the non-zero components.

5. Compute[16] $\bar{y}_i = y_i(\bar{\beta})$, $\forall i \in I_{all}$. Check if the following conditions hold: (a) *optimality = 1*; (b) $t_i\bar{y}_i \leq 1 + tol \ \forall i \in I$; and (c) $t_i\bar{y}_i \geq 1 - tol \ \forall i \notin I$.[17] If all three conditions hold, go to step 8.

6. Compute[18] Δ_1, Δ_2 and Δ using (14) and (15). Sort the δ_i in Δ in non-decreasing order. Let $\{i_1, i_2, \ldots, i_q\}$ denote the list of ordered indices obtained. Compute *ls* and *rs* using (16) and (17). Set $exit = 0$, $j = 0$. Repeat the following steps until $exit = 1$ occurs:

```
    $j = j + 1$, $\delta = \delta_{i_j}$
    $delslope = ls + \delta(rs - ls)$
    If $delslope \geq 0$ set $\delta^\star = -\delta \, ls/(delslope - ls)$ and $exit = 1$
    Use (18) to update $ls$ and $rs$ using $i = i_j$.
```

7. Set $\beta := \beta + \delta^\star(\bar{\beta} - \beta)$, $y := y + \delta^\star(\bar{y} - y)$, and compute the new active index set: $I = \{i \in I_{all} : t_i y_i < 1\}$. Compute $F = f(\beta)$. If ($iter \geq itermax$ or $F > F_{previous}$) stop with an error message. Else, set $F_{previous} = F$ and go back to step 3 for another iteration.

8. If *Need_Second_Round*=0 stop with $\beta = \bar{\beta}$, $y = \bar{y}$ and I_k as the optimal active index set. Else, set $\varepsilon = 10^{-6}$, *Need_Second_Round*=0 and go back to step 3.

Appendix B. A Description of Data Sets Used

As in the main paper, let m, n and n_{nz} denote, respectively, the number of examples, the number of features and the number of non-zero elements in the data matrix. Let $s = n_{nz}/(mn)$ denote the sparsity in the data matrix.

Australian is a small dense data set taken from the UCI repository(Blake and Merz, 1998) and it has $m = 690$, $n = 14$ and $s = 1$.

Adult and *Web* are data sets exactly as those used by Platt(Platt, 1999). For *Adult*, n is 120 and s is 0.21, while, for *Web*, n is 300 and $s = 0.04$. With each of these two data sets, 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 m. *Adult-1*, *Adult-4*, *Adult-7* and *Adult-9* have the m values 1605, 4781, 16100 and 32561. *Web-1*, *Web-4*, *Web-7* and *Web-8* have the m values 2477, 7366, 24692 and 49749.

We generated *News20* for easily reproducible results on a text classification task having both n and m large. It is a size-balanced two-class variant of the UCI "20 Newsgroups" data set (Blake and Merz, 1998). The positive class consists of the 10 groups with names of form `sci.*`, `comp.*`, or `misc.forsale`, and the negative class consists of the other 10 groups. We tokenized via McCallum's Rainbow(McCallum, 1996), using: `rainbow -g 3 -h -s -O 2 -i` (i.e. trigrams, skip message headers, no stoplist, drop terms occurring less than two times), giving $m = 19996$, $n = 1355191$

16. The arrays, p, r, q and z are local to step 4 and are not used elsewhere. Hence it is alright to use the same arrays elsewhere in the implementation. This can help save some memory. For example, p can be used to store the $\bar{y}_i$ computed in step 5 and r can be used to store the δ_i computed in step 6.
17. In view of numerical errors it is a good idea to employ the parameter tol in these checks. A value of $tol = 10^{-8}$ is a good choice.
18. This step refers to several equations from the main paper. To match the notations given there, take: y_i in this algorithm to be y_i^k of the main paper; ls in this pseudocode to stand for l_0, l_i, l_{i+1} etc; and rs in this pseudocode to stand for r_0, r_i, r_{i+1} etc.

and $s = 0.000336$. We used binary term frequencies and normalized each example vector to unit length.

Financial is a text classification data set that we created and corresponds to classifying news stories as financial or non-financial. Unigrams occuring in the news texts were taken as the features and a tf-idf representation was used to form the data. The data set has $m = 198788$, $n = 252472$ and $s = 0.00094$.

Yahoo is a large data set obtained from Yahoo! and is a classification problem concerning the prediction of behavior of customers. It has $m =$1 million, $n = 80$ and $s = 0.098$.

References

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

A. Björck. *Numerical Methods for Least Squares Problems*. SIAM, Philadelphia, 1996.

C. L. Blake and C. J. Merz. UCI repository of machine learning databases. Technical report, University of California, Irvine, 1998. www.ics.uci.edu/∼mlearn/MLRepository.html.

S. Chakrabarti, S. Roy, and M. V. Soundalgekar. Fast and accurate text classification via multiple linear discriminant projections. *The VLDB Journal*, 12:170–185, 2003.

W. Chu and S. S. Keerthi. New approaches to support vector ordinal regression. Technical report, Yahoo! Research Labs, Pasadena, California, USA, 2005.

D. DeCoste and K. Wagstaff. Alpha seeding for support vector machines. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 345–359, 2000.

A. Frommer and P. Maaß. Fast CG-based methods for Tikhonov-Phillips regularization. *SIAM Journal of Scientific Computing*, 20(5):1831–1850, 1999.

G. Fung and O. L. Mangasarian. Proximal support vector machine classifiers. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 77–86, 2001.

C. W. Hsu and C. J. Lin. A simple decomposition method for support vector machines. *Machine Learning*, 46:291–314, 2002.

T. Joachims. Making large-scale SVM learning practical. In *Advances in Kernel Methods - Support Vector Learning*. MIT Press, Cambridge, Massachussetts, 1999.

W. C. Kao, K.M. Chung, T. Sun, and C. J. Lin. Decomposition methods for linear support vector machines. *Neural Computation*, 16:1689–1704, 2004.

P. Komarek. Logistic regression for data mining and high-dimensional classification. Ph.d. thesis, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, 2004.

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

A. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. Technical report, University of Massachssetts, Amherst, Massachussetts, USA, 1996. www.cs.cmu.edu/∼mccallum/bow.

C. C. Paige and M. A. Saunders. LSQR: An algorithm for sparse linear equations and sparse least squares,. *ACM Transactions on Mathematical Software*, 8:43–71, 1982.

J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. In *Advances in Kernel Methods - Support Vector Learning*. MIT Press, Cambridge, Massachussetts, 1999.

J. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.

J. Zhang, R. Jin, Y. Yang, and A. Hauptmann. Modified logistic regression: An approximation to SVM and its applications in large-scale text categorization. In *Twentieth International Conference on Machine Learning*, pages 472–479, 2003.

T. Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *The Annals of Statistics*, 32:56–85, 2004.

Core Vector Machines: Fast SVM Training on Very Large Data Sets

Ivor W. Tsang IVOR@CS.UST.HK
James T. Kwok JAMESK@CS.UST.HK
Pak-Ming Cheung PAKMING@CS.UST.HK
Department of Computer Science
The Hong Kong University of Science and Technology
Clear Water Bay
Hong Kong

Editor: Nello Cristianini

Abstract

Standard SVM training has $O(m^3)$ time and $O(m^2)$ space complexities, where m is the training set size. It is thus computationally infeasible on very large data sets. By observing that practical SVM implementations only *approximate* the optimal solution by an iterative strategy, we scale up kernel methods by exploiting such “approximateness” in this paper. We first show that many kernel methods can be equivalently formulated as minimum enclosing ball (MEB) problems in computational geometry. Then, by adopting an efficient approximate MEB algorithm, we obtain provably approximately optimal solutions with the idea of core sets. Our proposed Core Vector Machine (CVM) algorithm can be used with nonlinear kernels and has a time complexity that is *linear* in m and a space complexity that is *independent* of m. Experiments on large toy and real-world data sets demonstrate that the CVM is as accurate as existing SVM implementations, but is much faster and can handle much larger data sets than existing scale-up methods. For example, CVM with the Gaussian kernel produces superior results on the KDDCUP-99 intrusion detection data, which has about five million training patterns, in only 1.4 seconds on a 3.2GHz Pentium–4 PC.

Keywords: kernel methods, approximation algorithm, minimum enclosing ball, core set, scalability

1. Introduction

In recent years, there has been a lot of interest on using kernels in various machine learning problems, with the support vector machines (SVM) being the most prominent example. Many of these kernel methods are formulated as quadratic programming (QP) problems. Denote the number of training patterns by m. The training time complexity of QP is $O(m^3)$ and its space complexity is at least quadratic. Hence, a major stumbling block is in scaling up these QP’s to large data sets, such as those commonly encountered in data mining applications.

To reduce the time and space complexities, a popular technique is to obtain low-rank approximations on the kernel matrix, by using the Nyström method (Williams and Seeger, 2001), greedy approximation (Smola and Schölkopf, 2000), sampling (Achlioptas et al., 2002) or matrix decompositions (Fine and Scheinberg, 2001). However, on very large data sets, the resulting rank of the kernel matrix may still be too high to be handled efficiently.

Another approach to scale up kernel methods is by chunking (Vapnik, 1998) or more sophisticated decomposition methods (Chang and Lin, 2004; Osuna et al., 1997b; Platt, 1999; Vishwanathan et al., 2003). However, chunking needs to optimize the entire set of non-zero Lagrange multipliers that have been identified, and the resultant kernel matrix may still be too large to fit into memory. Osuna et al. (1997b) suggested optimizing only a fixed-size subset (*working set*) of the training data each time, while the variables corresponding to the other patterns are frozen. Going to the extreme, the sequential minimal optimization (SMO) algorithm (Platt, 1999) breaks the original QP into a series of smallest possible QPs, each involving only two variables.

A more radical approach is to avoid the QP altogether. Mangasarian and his colleagues proposed several variations of the Lagrangian SVM (LSVM) (Fung and Mangasarian, 2003; Mangasarian and Musicant, 2001a,b) that obtain the solution with a fast iterative scheme. However, for nonlinear kernels (which is the focus in this paper), it still requires the inversion of an $m \times m$ matrix. Recently, Kao et al. (2004) and Yang et al. (2005) also proposed scale-up methods that are specially designed for the linear and Gaussian kernels, respectively.

Similar in spirit to decomposition algorithms are methods that scale down the training data before inputting to the SVM. For example, Pavlov et al. (2000b) used boosting to combine a large number of SVMs, each is trained on only a small data subsample. Alternatively, Collobert et al. (2002) used a neural-network-based gater to mix these small SVMs. Lee and Mangasarian (2001) proposed the reduced SVM (RSVM), which uses a random rectangular subset of the kernel matrix. Instead of random sampling, one can also use active learning (Schohn and Cohn, 2000; Tong and Koller, 2000), squashing (Pavlov et al., 2000a), editing (Bakir et al., 2005) or even clustering (Boley and Cao, 2004; Yu et al., 2003) to intelligently sample a small number of training data for SVM training. Other scale-up methods include the Kernel Adatron (Friess et al., 1998) and the SimpleSVM (Vishwanathan et al., 2003). For a more complete survey, interested readers may consult (Tresp, 2001) or Chapter 10 of (Schölkopf and Smola, 2002).

In practice, state-of-the-art SVM implementations typically have a training time complexity that scales between $O(m)$ and $O(m^{2.3})$ (Platt, 1999). This can be further driven down to $O(m)$ with the use of a parallel mixture (Collobert et al., 2002). However, these are only empirical observations and not theoretical guarantees. For reliable scaling behavior to very large data sets, our goal is to develop an algorithm that can be proved (using tools in analysis of algorithms) to be asymptotically efficient in both time and space.

A key observation is that practical SVM implementations, as in many numerical routines, only *approximate* the optimal solution by an iterative strategy. Typically, the stopping criterion uses either the precision of the Lagrange multipliers or the duality gap (Smola and Schölkopf, 2004). For example, in SMO, SVM^{light} (Joachims, 1999) and SimpleSVM, training stops when the Karush-Kuhn-Tucker (KKT) conditions are fulfilled within a tolerance parameter ε. Experience with these softwares indicate that near-optimal solutions are often good enough in practical applications. However, such "approximateness" has never been exploited in the design of SVM implementations.

On the other hand, in the field of theoretical computer science, approximation algorithms with provable performance guarantees have been extensively used in tackling computationally difficult problems (Garey and Johnson, 1979; Vazirani, 2001). Let C be the cost of the solution returned by an approximate algorithm, and C^* be the cost of the optimal solution. An approximate algorithm has *approximation ratio* $\rho(n)$ for an input size n if $\max\left(\frac{C}{C^*}, \frac{C^*}{C}\right) \leq \rho(n)$. Intuitively, this ratio measures how bad the approximate solution is compared with the optimal solution. A large (small) approximation ratio means the solution is much worse than (more or less the same as) the optimal solution. Observe that $\rho(n)$ is always ≥ 1. If the ratio does not depend on n, we may just write ρ and call the algorithm an ρ-*approximation algorithm*. Well-known NP-complete problems that can be efficiently addressed using approximation algorithms include the vertex-cover problem and the set-covering problem. This large body of experience suggests that one may also develop approximation algorithms for SVMs, with the hope that training of kernel methods will become more tractable, and thus more scalable, in practice.

In this paper, we will utilize an approximation algorithm for the *minimum enclosing ball* (MEB) problem in computational geometry. The MEB problem computes the ball of minimum radius enclosing a given set of points (or, more generally, balls). Traditional algorithms for finding exact MEBs (e.g., (Megiddo, 1983; Welzl, 1991)) do not scale well with the dimensionality d of the points. Consequently, recent attention has shifted to the development of approximation algorithms (Bădoiu and Clarkson, 2002; Kumar et al., 2003; Nielsen and Nock, 2004). In particular, a breakthrough was obtained by Bădoiu and Clarkson (2002), who showed that an $(1+\varepsilon)$-approximation of the MEB can be efficiently obtained using *core sets*. Generally speaking, in an optimization problem, a core set is a subset of input points such that we can get a good approximation to the original input by solving the optimization problem directly on the core set. A surprising property of (Bădoiu and Clarkson, 2002) is that the size of its core set can be shown to be *independent* of both d and the size of the point set.

In the sequel, we will show that there is a close relationship between SVM training and the MEB problem. Inspired from the core set-based approximate MEB algorithms, we will then develop an approximation algorithm for SVM training that has an approximation ratio of $(1+\varepsilon)^2$. Its time complexity is *linear* in m while its space complexity is *independent* of m. In actual implementation, the time complexity can be further improved with the use of probabilistic speedup methods (Smola and Schölkopf, 2000).

The rest of this paper is organized as follows. Section 2 gives a short introduction on the MEB problem and its approximation algorithm. The connection between kernel methods and the MEB problem is given in Section 3. Section 4 then describes our proposed Core Vector Machine (CVM) algorithm. The core set in CVM plays a similar role as the working set in decomposition algorithms, which will be reviewed briefly in Section 5. Finally, experimental results are presented in Section 6, and the last section gives some concluding remarks. Preliminary results on the CVM have been recently reported in (Tsang et al., 2005).

2. The (Approximate) Minimum Enclosing Ball Problem

Given a set of points $\mathcal{S} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$, where each $\mathbf{x}_i \in \mathbb{R}^d$, the minimum enclosing ball of $\mathcal{S}$ (denoted MEB($\mathcal{S}$)) is the smallest ball that contains all the points in $\mathcal{S}$. The MEB problem can be dated back as early as in 1857, when Sylvester (1857) first investigated the smallest radius disk enclosing m points on the plane. It has found applications in diverse areas such as computer graphics (e.g., for collision detection, visibility culling), machine learning (e.g., similarity search) and facility locations problems (Preparata, 1985). The MEB problem also belongs to the larger family of shape fitting problems, which attempt to find the shape (such as a slab, cylinder, cylindrical shell or spherical shell) that best fits a given point set (Chan, 2000).

Traditional algorithms for finding exact MEBs (such as (Megiddo, 1983; Welzl, 1991)) are not efficient for problems with $d > 30$. Hence, as mentioned in Section 1, it is of practical interest to study faster approximation algorithms that return a solution within a multiplicative factor of $1+\varepsilon$ to the optimal value, where ε is a small positive number. Let $B(\mathbf{c},R)$ be the ball with center $\mathbf{c}$ and radius R. Given an $\varepsilon > 0$, a ball $B(\mathbf{c},(1+\varepsilon)R)$ is an *$(1+\varepsilon)$-approximation* of $\text{MEB}(\mathcal{S})$ if $R \leq r_{\text{MEB}(\mathcal{S})}$ and $\mathcal{S} \subset B(\mathbf{c},(1+\varepsilon)R)$. In many shape fitting problems, it is found that solving the problem on a subset, called the *core set*, $\mathcal{Q}$ of points from $\mathcal{S}$ can often give an accurate and efficient approximation. More formally, a subset $\mathcal{Q} \subseteq \mathcal{S}$ is a core set of $\mathcal{S}$ if an expansion by a factor $(1+\varepsilon)$ of its MEB contains $\mathcal{S}$, i.e., $\mathcal{S} \subset B(\mathbf{c},(1+\varepsilon)r)$, where $B(\mathbf{c},r) = \text{MEB}(\mathcal{Q})$ (Figure 1).

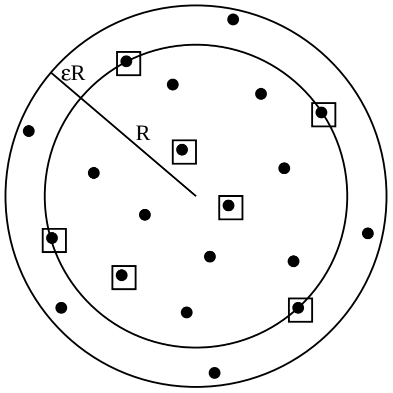


Figure 1: The inner circle is the MEB of the set of squares and its $(1+\varepsilon)$ expansion (the outer circle) covers all the points. The set of squares is thus a core set.

A breakthrough on achieving such an $(1+\varepsilon)$-approximation was recently obtained by Bădoiu and Clarkson (2002). They used a simple iterative scheme: At the tth iteration, the current estimate $B(\mathbf{c}_t,r_t)$ is expanded incrementally by including the furthest point outside the $(1+\varepsilon)$-ball $B(\mathbf{c}_t,(1+\varepsilon)r_t)$. This is repeated until all the points in $\mathcal{S}$ are covered by $B(\mathbf{c}_t,(1+\varepsilon)r_t)$. Despite its simplicity, a surprising property is that the number of iterations, and hence the size of the final core set, depends only on ε but *not* on d or m. The independence of d is important on applying this algorithm to kernel methods (Section 3) as the kernel-induced feature space can be infinite-dimensional. As for the remarkable independence on m, it allows both the time and space complexities of our algorithm to grow slowly (Section 4.3).

3. MEB Problems and Kernel Methods

The MEB can be easily seen to be equivalent to the hard-margin support vector data description (SVDD) (Tax and Duin, 1999), which will be briefly reviewed in Section 3.1. The MEB problem can also be used to find the radius component of the radius-margin bound (Chapelle et al., 2002; Vapnik, 1998). Thus, Kumar et al. (2003) has pointed out that the MEB problem can be used in support vector clustering and SVM parameter tuning. However, as will be shown in Section 3.2, other kernel-related problems, such as the soft-margin one-class and two-class SVMs, can also be viewed as MEB problems. Note that finding the soft-margin one-class SVM is essentially the same as fitting the MEB with outliers, which is also considered in (Har-Peled and Wang, 2004). However, a limitation of their technique is that the number of outliers has to be moderately small in order to be effective. Another heuristic approach for scaling up the soft-margin SVDD using core sets has also been proposed in (Chu et al., 2004).

3.1 Hard-Margin SVDD

Given a kernel k with the associated feature map φ, let the MEB (or hard-margin ball) in the kernel-induced feature space be $B(\mathbf{c}, R)$. The primal problem in the hard-margin SVDD is

$$\min_{R,\mathbf{c}} R^2 \;:\; \|\mathbf{c} - \varphi(\mathbf{x}_i)\|^2 \leq R^2, \; i = 1, \ldots, m. \tag{1}$$

The corresponding dual is

$$\begin{array}{ll} \max_{\alpha_i} & \sum_{i=1}^{m} \alpha_i k(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i,j=1}^{m} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \\ \text{s.t.} & \alpha_i \geq 0, \quad i = 1, \ldots, m \\ & \sum_{i=1}^{m} \alpha_i = 1, \end{array}$$

or, in matrix form,

$$\max_{\alpha} \; \alpha' \text{diag}(\mathbf{K}) - \alpha' \mathbf{K} \alpha \;:\; \alpha \geq \mathbf{0}, \; \alpha' \mathbf{1} = 1, \tag{2}$$

where $\alpha = [\alpha_i, \ldots, \alpha_m]'$ are the Lagrange multipliers, $\mathbf{0} = [0, \ldots, 0]'$, $\mathbf{1} = [1, \ldots, 1]'$ and $\mathbf{K}_{m \times m} = [k(\mathbf{x}_i, \mathbf{x}_j)]$ is the kernel matrix. As is well-known, this is a QP problem. The primal variables can be recovered from the optimal α as

$$\mathbf{c} = \sum_{i=1}^{m} \alpha_i \varphi(\mathbf{x}_i), \quad R = \sqrt{\alpha' \text{diag}(\mathbf{K}) - \alpha' \mathbf{K} \alpha}. \tag{3}$$

3.2 Viewing Kernel Methods as MEB Problems

Consider the situation where

$$k(\mathbf{x}, \mathbf{x}) = \kappa, \tag{4}$$

a constant. All the patterns are then mapped to a sphere in the feature space. (4) will be satisfied when either

1. the isotropic kernel $k(\mathbf{x}, \mathbf{y}) = K(\|\mathbf{x} - \mathbf{y}\|)$ (e.g., Gaussian kernel); or

2. the dot product kernel $k(\mathbf{x}, \mathbf{y}) = K(\mathbf{x}'\mathbf{y})$ (e.g., polynomial kernel) with normalized inputs; or

3. any normalized kernel $k(\mathbf{x}, \mathbf{y}) = \frac{K(\mathbf{x},\mathbf{y})}{\sqrt{K(\mathbf{x},\mathbf{x})}\sqrt{K(\mathbf{y},\mathbf{y})}}$

is used. These three cases cover most kernel functions used in real-world applications. Schölkopf et al. (2001) showed that the hard (soft) margin SVDD then yields identical solution as the hard (soft) margin one-class SVM, and the weight $\mathbf{w}$ in the one-class SVM solution is equal to the center $\mathbf{c}$ in the SVDD solution.

Combining (4) with the condition $\alpha'\mathbf{1} = 1$ in (2), we have $\alpha' \text{diag}(\mathbf{K}) = \kappa$. Dropping this constant term from the dual objective in (2), we obtain a simpler optimization problem:

$$\max_{\alpha} -\alpha' \mathbf{K} \alpha \;:\; \alpha \geq \mathbf{0}, \; \alpha' \mathbf{1} = 1. \tag{5}$$

Conversely, whenever the kernel k satisfies (4), any QP of the form (5) can be regarded as a MEB problem in (1). Note that (2) and (5) yield the same set of optimal α's. Moreover, the optimal (dual) objectives in (2) and (5) (denoted d_1^* and d_2^* respectively) are related by

$$d_1^* = d_2^* + \kappa. \tag{6}$$

In the following, we will show that when (4) is satisfied, the duals in a number of kernel methods can be rewritten in the form of (5). While the 1-norm error has been commonly used for the SVM, our main focus will be on the 2-norm error. In theory, this could be less robust in the presence of outliers. However, experimentally, its generalization performance is often comparable to that of the L1-SVM (Lee and Mangasarian, 2001; Mangasarian and Musicant, 2001a,b). Besides, the 2-norm error is more advantageous here because a soft-margin L2-SVM can be transformed to a hard-margin one. While the 2-norm error has been used in classification (Section 3.2.2), we will also extend its use for novelty detection (Section 3.2.1).

3.2.1 ONE-CLASS L2-SVM

Given a set of unlabeled patterns $\{\mathbf{z}_i\}_{i=1}^m$ where $\mathbf{z}_i$ only has the input part $\mathbf{x}_i$, the one-class L2-SVM separates outliers from the normal data by solving the primal problem:

$$\begin{aligned} \min_{\mathbf{w},\rho,\xi_i} \quad & \|\mathbf{w}\|^2 - 2\rho + C\sum_{i=1}^m \xi_i^2 \\ \text{s.t.} \quad & \mathbf{w}'\varphi(\mathbf{x}_i) \geq \rho - \xi_i, \quad i = 1,\ldots,m, \end{aligned} \tag{7}$$

where $\mathbf{w}'\varphi(\mathbf{x}) = \rho$ is the desired hyperplane and C is a user-defined parameter. Unlike the classification LSVM, the bias is not penalized here. Moreover, note that constraints $\xi_i \geq 0$ are not needed for the L2-SVM (Keerthi et al., 2000). The corresponding dual is

$$\max_{\alpha} \ -\alpha'\left(\mathbf{K} + \frac{1}{C}\mathbf{I}\right)\alpha \ : \ \alpha \geq \mathbf{0}, \ \alpha'\mathbf{1} = 1, \tag{8}$$

where $\mathbf{I}$ is the $m \times m$ identity matrix. From the Karush-Kuhn-Tucker (KKT) conditions, we can recover

$$\mathbf{w} = \sum_{i=1}^m \alpha_i \varphi(\mathbf{x}_i) \tag{9}$$

and $\xi_i = \frac{\alpha_i}{C}$, and then $\rho = \mathbf{w}'\varphi(\mathbf{x}_i) + \frac{\alpha_i}{C}$ from any support vector $\mathbf{x}_i$.

Rewrite (8) in the form of (5) as:

$$\max_{\alpha} \ -\alpha'\tilde{\mathbf{K}}\alpha \ : \ \alpha \geq \mathbf{0}, \ \alpha'\mathbf{1} = 1, \tag{10}$$

where

$$\tilde{\mathbf{K}} = [\tilde{k}(\mathbf{z}_i,\mathbf{z}_j)] = \left[k(\mathbf{x}_i,\mathbf{x}_j) + \frac{\delta_{ij}}{C}\right]. \tag{11}$$

Since $k(\mathbf{x},\mathbf{x}) = \kappa$,

$$\tilde{k}(\mathbf{z},\mathbf{z}) = \kappa + \frac{1}{C} \equiv \tilde{\kappa}$$

is also a constant. This one-class L2-SVM thus corresponds to the MEB problem (1), in which φ is replaced by the nonlinear map $\tilde{\varphi}$ satisfying $\tilde{\varphi}(\mathbf{z}_i)'\tilde{\varphi}(\mathbf{z}_j) = \tilde{k}(\mathbf{z}_i, \mathbf{z}_j)$. It can be easily verified that this $\tilde{\varphi}$ maps the training point $\mathbf{z}_i = \mathbf{x}_i$ to a higher dimensional space, as

$$\tilde{\varphi}(\mathbf{z}_i) = \begin{bmatrix} \varphi(\mathbf{x}_i) \\ \frac{1}{\sqrt{C}}\mathbf{e}_i \end{bmatrix},$$

where $\mathbf{e}_i$ is the m-dimensional vector with all zeroes except that the ith position is equal to one.

3.2.2 TWO-CLASS L2-SVM

In the two-class classification problem, we are given a training set $\{\mathbf{z}_i = (\mathbf{x}_i, y_i)\}_{i=1}^m$ with $y_i \in \{-1, 1\}$. The primal of the two-class L2-SVM is

$$\begin{aligned} \min_{\mathbf{w},b,\rho,\xi_i} \quad & \|\mathbf{w}\|^2 + b^2 - 2\rho + C\sum_{i=1}^m \xi_i^2 \\ \text{s.t.} \quad & y_i(\mathbf{w}'\varphi(\mathbf{x}_i) + b) \geq \rho - \xi_i, \quad i = 1, \ldots, m. \end{aligned} \tag{12}$$

The corresponding dual is

$$\max_{\alpha} \; -\alpha'\left(\mathbf{K} \odot \mathbf{y}\mathbf{y}' + \mathbf{y}\mathbf{y}' + \frac{1}{C}\mathbf{I}\right)\alpha \; : \; \alpha \geq \mathbf{0}, \; \alpha'\mathbf{1} = 1, \tag{13}$$

where $\odot$ denotes the Hadamard product and $\mathbf{y} = [y_1, \ldots, y_m]'$. Again, we can recover

$$\mathbf{w} = \sum_{i=1}^m \alpha_i y_i \varphi(\mathbf{x}_i), \quad b = \sum_{i=1}^m \alpha_i y_i, \quad \xi_i = \frac{\alpha_i}{C}, \tag{14}$$

from the optimal α and then $\rho = y_i(\mathbf{w}'\varphi(\mathbf{x}_i) + b) + \frac{\alpha_i}{C}$ from any support vector $\mathbf{z}_i$. Alternatively, ρ can also be obtained from the fact that QP's have zero duality gap. Equating the primal (12) and dual (13), we have

$$\|\mathbf{w}\|^2 + b^2 - 2\rho + C\sum_{i=1}^m \xi_i^2 = -\sum_{i,j=1}^m \alpha_i\alpha_j\left(y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) + y_i y_j + \frac{\delta_{ij}}{C}\right).$$

Substituting in (14), we then have

$$\rho = \sum_{i,j=1}^m \alpha_i\alpha_j\left(y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) + y_i y_j + \frac{\delta_{ij}}{C}\right). \tag{15}$$

Rewriting (13) in the form of (5), we have

$$\max_{\alpha} \; -\alpha'\tilde{\mathbf{K}}\alpha \;\; : \;\; \alpha \geq \mathbf{0}, \; \alpha'\mathbf{1} = 1, \tag{16}$$

where $\tilde{\mathbf{K}} = [\tilde{k}(\mathbf{z}_i, \mathbf{z}_j)]$ with

$$\tilde{k}(\mathbf{z}_i, \mathbf{z}_j) = y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) + y_i y_j + \frac{\delta_{ij}}{C}, \tag{17}$$

Again, this $\tilde{k}$ satisfies (4), as

$$\tilde{k}(\mathbf{z},\mathbf{z}) = \kappa + 1 + \frac{1}{C} \equiv \tilde{\kappa},$$

a constant. Thus, this two-class L2-SVM can also be viewed as a MEB problem (1) in which φ is replaced by $\tilde{\varphi}$, with

$$\tilde{\varphi}(\mathbf{z}_i) = \begin{bmatrix} y_i \varphi(\mathbf{x}_i) \\ y_i \\ \frac{1}{\sqrt{C}} \mathbf{e}_i \end{bmatrix}$$

for any training point $\mathbf{z}_i$. Note that as a classification (supervised learning) problem is now reformulated as a MEB (unsupervised) problem, the label information gets encoded in the feature map $\tilde{\varphi}$. Moreover, all the support vectors of this L2-SVM, including those defining the margin and those that are misclassified, now reside on the surface of the ball in the feature space induced by $\tilde{k}$. A similar relationship connecting one-class classification and binary classification for the case of Gaussian kernels is also discussed by Schölkopf et al. (2001). In the special case of a hard-margin SVM, $\tilde{k}$ reduces to $\tilde{k}(\mathbf{z}_i, \mathbf{z}_j) = y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) + y_i y_j$ and analogous results apply.

4. Core Vector Machine (CVM)

After formulating the kernel method as a MEB problem, we obtain a transformed kernel $\tilde{k}$, together with the associated feature space $\tilde{\mathcal{F}}$, mapping $\tilde{\varphi}$ and constant $\tilde{\kappa} = \tilde{k}(\mathbf{z},\mathbf{z})$. To solve this kernel-induced MEB problem, we adopt the approximation algorithm described in the proof of Theorem 2.2 in (Bădoiu and Clarkson, 2002). A similar algorithm is also described in (Kumar et al., 2003). As mentioned in Section 2, the idea is to incrementally expand the ball by including the point furthest away from the current center. In the following, we denote the core set, the ball's center and radius at the tth iteration by $\mathcal{S}_t, \mathbf{c}_t$ and R_t respectively. Also, the center and radius of a ball B are denoted by $\mathbf{c}_B$ and r_B. Given an $\varepsilon > 0$, the CVM then works as follows:

1. Initialize $\mathcal{S}_0$, $\mathbf{c}_0$ and R_0.

2. Terminate if there is no training point $\mathbf{z}$ such that $\tilde{\varphi}(\mathbf{z})$ falls outside the $(1+\varepsilon)$-ball $B(\mathbf{c}_t, (1+\varepsilon)R_t)$.

3. Find $\mathbf{z}$ such that $\tilde{\varphi}(\mathbf{z})$ is furthest away from $\mathbf{c}_t$. Set $S_{t+1} = \mathcal{S}_t \cup \{\mathbf{z}\}$.

4. Find the new MEB$(\mathcal{S}_{t+1})$ from (5) and set $\mathbf{c}_{t+1} = \mathbf{c}_{\text{MEB}(\mathcal{S}_{t+1})}$ and $R_{t+1} = r_{\text{MEB}(\mathcal{S}_{t+1})}$ using (3).

5. Increment t by 1 and go back to Step 2.

In the sequel, points that are added to the core set will be called *core vectors*. Details of each of the above steps will be described in Section 4.1. Despite its simplicity, CVM has an approximation guarantee (Section 4.2) and small time and space complexities (Section 4.3).

4.1 Detailed Procedure

4.1.1 INITIALIZATION

Bădoiu and Clarkson (2002) simply used an arbitrary point $\mathbf{z} \in \mathcal{S}$ to initialize $S_0 = \{\mathbf{z}\}$. However, a good initialization may lead to fewer updates and so we follow the scheme in (Kumar et al., 2003).

We start with an arbitrary point $\mathbf{z} \in \mathcal{S}$ and find $\mathbf{z}_a \in \mathcal{S}$ that is furthest away from $\mathbf{z}$ in the feature space $\tilde{\mathcal{F}}$. Then, we find another point $\mathbf{z}_b \in \mathcal{S}$ that is furthest away from $\mathbf{z}_a$ in $\tilde{\mathcal{F}}$. The initial core set is then set to be $\mathcal{S}_0 = \{\mathbf{z}_a, \mathbf{z}_b\}$. Obviously, MEB$(\mathcal{S}_0)$ (in $\tilde{\mathcal{F}}$) has center $\mathbf{c}_0 = \frac{1}{2}(\tilde{\varphi}(\mathbf{z}_a) + \tilde{\varphi}(\mathbf{z}_b))$ On using (3), we thus have $\alpha_a = \alpha_b = \frac{1}{2}$ and all the other α_i's are zero. The initial radius is

$$\begin{aligned} R_0 &= \frac{1}{2}\|\tilde{\varphi}(\mathbf{z}_a) - \tilde{\varphi}(\mathbf{z}_b)\| \\ &= \frac{1}{2}\sqrt{\|\tilde{\varphi}(\mathbf{z}_a)\|^2 + \|\tilde{\varphi}(\mathbf{z}_b)\|^2 - 2\tilde{\varphi}(\mathbf{z}_a)'\tilde{\varphi}(\mathbf{z}_b)} \\ &= \frac{1}{2}\sqrt{2\tilde{\kappa} - 2\tilde{k}(\mathbf{z}_a, \mathbf{z}_b)}. \end{aligned}$$

In a classification problem, one may further require $\mathbf{z}_a$ and $\mathbf{z}_b$ to come from different classes. On using (17), R_0 then becomes $\frac{1}{2}\sqrt{2\left(\kappa + 2 + \frac{1}{C}\right) + 2k(\mathbf{x}_a, \mathbf{x}_b)}$. As κ and C are constants, choosing the pair $(\mathbf{x}_a, \mathbf{x}_b)$ that maximizes R_0 is then equivalent to choosing the closest pair belonging to opposing classes, which is also the heuristic used in initializing the DirectSVM (Roobaert, 2000) and SimpleSVM (Vishwanathan et al., 2003).

4.1.2 DISTANCE COMPUTATIONS

Steps 2 and 3 involve computing $\|\mathbf{c}_t - \tilde{\varphi}(\mathbf{z}_\ell)\|$ for $\mathbf{z}_\ell \in \mathcal{S}$. On using $\mathbf{c} = \sum_{\mathbf{z}_i \in \mathcal{S}_t} \alpha_i \tilde{\varphi}(\mathbf{z}_i)$ in (3), we have

$$\|\mathbf{c}_t - \tilde{\varphi}(\mathbf{z}_\ell)\|^2 = \sum_{\mathbf{z}_i, \mathbf{z}_j \in \mathcal{S}_t} \alpha_i \alpha_j \tilde{k}(\mathbf{z}_i, \mathbf{z}_j) - 2 \sum_{\mathbf{z}_i \in \mathcal{S}_t} \alpha_i \tilde{k}(\mathbf{z}_i, \mathbf{z}_\ell) + \tilde{k}(\mathbf{z}_\ell, \mathbf{z}_\ell). \tag{18}$$

Hence, computations are based on kernel evaluations instead of the explicit $\tilde{\varphi}(\mathbf{z}_i)$'s, which may be infinite-dimensional. Note that, in contrast, existing MEB algorithms only consider finite-dimensional spaces.

However, in the feature space, $\mathbf{c}_t$ cannot be obtained as an explicit point but rather as a convex combination of (at most) $|\mathcal{S}_t|$ $\tilde{\varphi}(\mathbf{z}_i)$'s. Computing (18) for all m training points takes $O(|\mathcal{S}_t|^2 + m|\mathcal{S}_t|) = O(m|\mathcal{S}_t|)$ time at the tth iteration. This becomes very expensive when m is large. Here, we use the probabilistic speedup method in (Smola and Schölkopf, 2000). The idea is to randomly sample a sufficiently large subset $\mathcal{S}'$ from $\mathcal{S}$, and then take the point in $\mathcal{S}'$ that is furthest away from $\mathbf{c}_t$ as the approximate furthest point over $\mathcal{S}$. As shown in (Smola and Schölkopf, 2000), by using a small random sample of, say, size 59, the furthest point obtained from $\mathcal{S}'$ is with probability 0.95 among the furthest 5% of points from the whole $\mathcal{S}$. Instead of taking $O(m|\mathcal{S}_t|)$ time, this randomized method only takes $O(|\mathcal{S}_t|^2 + |\mathcal{S}_t|) = O(|\mathcal{S}_t|^2)$ time, which is much faster as $|\mathcal{S}_t| \ll m$. This trick can also be used in the initialization step.

4.1.3 ADDING THE FURTHEST POINT

Points outside MEB$(\mathcal{S}_t)$ have zero α_i's (Section 4.1.1) and so violate the KKT conditions of the dual problem. As in (Osuna et al., 1997b), one can simply add any such violating point to $\mathcal{S}_t$. Our step 3, however, takes a greedy approach by including the point furthest away from the current center. In the one-class classification case (Section 3.2.1),

$$\begin{aligned}\arg\max_{\mathbf{z}_\ell \notin B(\mathbf{c}_t,(1+\varepsilon)R_t)} \|\mathbf{c}_t - \tilde{\varphi}(\mathbf{z}_\ell)\|^2 &= \arg\min_{\mathbf{z}_\ell \notin B(\mathbf{c}_t,(1+\varepsilon)R_t)} \sum_{\mathbf{z}_i \in \mathcal{S}_t} \alpha_i \tilde{k}(\mathbf{z}_i, \mathbf{z}_\ell) \\ &= \arg\min_{\mathbf{z}_\ell \notin B(\mathbf{c}_t,(1+\varepsilon)R_t)} \sum_{\mathbf{z}_i \in \mathcal{S}_t} \alpha_i k(\mathbf{x}_i, \mathbf{x}_\ell) \\ &= \arg\min_{\mathbf{z}_\ell \notin B(\mathbf{c}_t,(1+\varepsilon)R_t)} \mathbf{w}'\varphi(\mathbf{x}_\ell), \end{aligned} \tag{19}$$

on using (9), (11) and (18). Similarly, in the binary classification case (Section 3.2.2), we have

$$\begin{aligned}\arg\max_{\mathbf{z}_\ell \notin B(\mathbf{c}_t,(1+\varepsilon)R_t)} \|\mathbf{c}_t - \tilde{\varphi}(\mathbf{z}_\ell)\|^2 &= \arg\min_{\mathbf{z}_\ell \notin B(\mathbf{c}_t,(1+\varepsilon)R_t)} \sum_{\mathbf{z}_i \in \mathcal{S}_t} \alpha_i y_i y_\ell (k(\mathbf{x}_i, \mathbf{x}_\ell) + 1) \\ &= \arg\min_{\mathbf{z}_\ell \notin B(\mathbf{c}_t,(1+\varepsilon)R_t)} y_\ell(\mathbf{w}'\varphi(\mathbf{x}_\ell) + b), \end{aligned} \tag{20}$$

on using (14) and (17). Hence, in both cases, step 3 chooses the *worst* violating pattern corresponding to the constraint ((7) and (12) respectively).

Also, as the dual objective in (10) has gradient $-2\tilde{\mathbf{K}}\alpha$, so for a pattern ℓ currently outside the ball

$$(\tilde{\mathbf{K}}\alpha)_\ell = \sum_{i=1}^m \alpha_i \left(k(\mathbf{x}_i, \mathbf{x}_\ell) + \frac{\delta_{i\ell}}{C} \right) = \mathbf{w}'\varphi(\mathbf{x}_\ell),$$

on using (9), (11) and $\alpha_\ell = 0$. Thus, the pattern chosen in (19) also makes the most progress towards maximizing the dual objective. This is also true for the two-class L2-SVM, as

$$(\tilde{\mathbf{K}}\alpha)_\ell = \sum_{i=1}^m \alpha_i \left(y_i y_\ell k(\mathbf{x}_i, \mathbf{x}_\ell) + y_i y_\ell + \frac{\delta_{i\ell}}{C} \right) = y_\ell(\mathbf{w}'\varphi(\mathbf{x}_\ell) + b),$$

on using (14), (17) and $\alpha_\ell = 0$. This subset selection heuristic is also commonly used by decomposition algorithms (Chang and Lin, 2004; Joachims, 1999; Platt, 1999).

4.1.4 FINDING THE MEB

At each iteration of Step 4, we find the MEB by using the QP formulation in Section 3.2. As the size $|\mathcal{S}_t|$ of the core set is much smaller than m in practice (Section 6), the computational complexity of each QP sub-problem is much lower than solving the whole QP. Besides, as only one core vector is added at each iteration, efficient rank-one update procedures (Cauwenberghs and Poggio, 2001; Vishwanathan et al., 2003) can also be used. The cost then becomes quadratic rather than cubic. As will be demonstrated in Section 6, the size of the core set is usually small to medium even for very large data sets. Hence, SMO is chosen in our implementation as it is often very efficient (in terms of both time and space) on data sets of such sizes. Moreover, as only one point is added each time, the new QP is just a slight perturbation of the original. Hence, by using the MEB solution obtained from the previous iteration as starting point (*warm start*), SMO can often converge in a small number of iterations.

4.2 Convergence to (Approximate) Optimality

First, consider $\varepsilon = 0$. The convergence proof in Bădoiu and Clarkson (2002) does not apply as it requires $\varepsilon > 0$. But as the number of core vectors increases in each iteration and the training set size is finite, so CVM must terminate in a finite number (say, τ) of iterations, With $\varepsilon = 0$, MEB($\mathcal{S}_\tau$) is an enclosing ball for all the ($\tilde{\varphi}$-transformed) points on termination. Because $\mathcal{S}_\tau$ is a subset of the whole training set and the MEB of a subset cannot be larger than the MEB of the whole set. Hence, MEB($\mathcal{S}_\tau$) must also be the exact MEB of the whole ($\tilde{\varphi}$-transformed) training set. In other words, when $\varepsilon = 0$, CVM outputs the *exact* solution of the kernel problem.

When $\varepsilon > 0$, we can still obtain an approximately optimal dual objective as follows. Assume that the algorithm terminates at the τth iteration, then

$$R_\tau \leq r_{\text{MEB}(\mathcal{S})} \leq (1+\varepsilon)R_\tau \tag{21}$$

by definition. Recall that the optimal primal objective p^* of the kernel problem in Section 3.2.1 (or 3.2.2) is equal to the optimal dual objective d_2^* in (10) (or (16)), which in turn is related to the optimal dual objective $d_1^* = r^2_{\text{MEB}(\mathcal{S})}$ in (2) by (6). Together with (21), we can then bound p^* as

$$R_\tau^2 \leq p^* + \tilde{\kappa} \leq (1+\varepsilon)^2 R_\tau^2. \tag{22}$$

Hence, $\max\left(\frac{R_\tau^2}{p^*+\tilde{\kappa}}, \frac{p^*+\tilde{\kappa}}{R_\tau^2}\right) \leq (1+\varepsilon)^2$ and thus CVM is an $(1+\varepsilon)^2$-approximation algorithm. This also holds with high probability[1] when probabilistic speedup is used.

As mentioned in Section 1, practical SVM implementations also output approximated solutions only. Typically, a parameter similar to our ε is required at termination. For example, in SMO, SVM^{light} and SimpleSVM, training stops when the KKT conditions are fulfilled within ε. Experience with these softwares indicate that near-optimal solutions are often good enough in practical applications. It can be shown that when CVM terminates, all the training patterns also satisfy similar loose KKT conditions. Here, we focus on the binary classification case. Now, at any iteration t, each training point falls into one of the following three categories:

1. Core vectors: Obviously, they satisfy the loose KKT conditions as they are involved in the QP.

2. Non-core vectors inside/on the ball $B(\mathbf{c}_t, R_t)$: Their α_i's are zero[2] and so the KKT conditions are satisfied.

3. Points lying outside $B(\mathbf{c}_t, R_t)$: Consider one such point ℓ. Its α_ℓ is zero (by initialization) and

$$\begin{aligned}\|\mathbf{c}_t - \tilde{\varphi}(\mathbf{z}_\ell)\|^2 &= \sum_{\mathbf{z}_i,\mathbf{z}_j \in \mathcal{S}_t} \alpha_i\alpha_j \left(y_i y_j k(\mathbf{x}_i,\mathbf{x}_j) + y_i y_j + \frac{\delta_{ij}}{C}\right) \\ &\quad -2\sum_{\mathbf{z}_i \in \mathcal{S}_t} \alpha_i \left(y_i y_\ell k(\mathbf{x}_i,\mathbf{x}_\ell) + y_i y_\ell + \frac{\delta_{i\ell}}{C}\right) + \tilde{k}(\mathbf{z}_\ell,\mathbf{z}_\ell) \\ &= \rho_t + \tilde{\kappa} - 2y_\ell(\mathbf{w}_t'\varphi(\mathbf{x}_\ell) + b_t),\end{aligned} \tag{23}$$

on using (14), (15), (17) and (18). This leads to

$$R_t^2 = \tilde{\kappa} - \rho_t. \tag{24}$$

1. Obviously, the probability increases with the number of points subsampled and is equal to one when all the points are used. Obtaining a precise probability statement will be studied in future research.
2. Recall that all the α_i's (except those of the two initial core vectors) are initialized to zero.

on using (3), (15) and (16). As $\mathbf{z}_\ell$ is inside/on the $(1+\varepsilon)$-ball at the τth iteration, $\|\mathbf{c}_\tau - \tilde{\varphi}(\mathbf{z}_\ell)\|^2 \leq (1+\varepsilon)^2 R_\tau^2$. Hence, from (23) and (24),

$$\begin{aligned}
& (1+\varepsilon)^2(\tilde{\kappa}-\rho_\tau) && \geq \rho_\tau + \tilde{\kappa} - 2y_\ell(\mathbf{w}_\tau'\varphi(\mathbf{x}_\ell)+b_\tau) \\
\Rightarrow \quad & 2y_\ell(\mathbf{w}_\tau'\varphi(\mathbf{x}_\ell)+b_\tau) && \geq \rho_\tau + \tilde{\kappa} - (1+\varepsilon)^2(\tilde{\kappa}-\rho_\tau) \\
& && \geq 2\rho_\tau - (2\varepsilon+\varepsilon^2)(\tilde{\kappa}-\rho_\tau) \\
\Rightarrow \quad & y_\ell(\mathbf{w}_\tau'\varphi(\mathbf{x}_\ell)+b_\tau) - \rho_\tau && \geq -\left(\varepsilon + \frac{\varepsilon^2}{2}\right) R_\tau^2. \qquad (25)
\end{aligned}$$

Obviously, $R_\tau^2 \leq \tilde{k}(\mathbf{z},\mathbf{z}) = \tilde{\kappa}$. Hence, (25) reduces to

$$y_\ell(\mathbf{w}_\tau'\varphi(\mathbf{x}_\ell)+b_\tau) - \rho_\tau \geq -\left(\varepsilon + \frac{\varepsilon^2}{2}\right)\tilde{\kappa} \equiv -\varepsilon_2,$$

which is a loose KKT condition on pattern ℓ (which has $\alpha_\ell = 0$ and consequently $\xi_\ell = 0$ by (14)).

4.3 Time and Space Complexities

Existing decomposition algorithms cannot guarantee the number of iterations and consequently the overall time complexity (Chang and Lin, 2004). In this Section, we show how this can be obtained for CVM. In the following, we assume that a plain QP implementation, which takes $O(m^3)$ time and $O(m^2)$ space for m patterns, is used for the QP sub-problem in step 4. The time and space complexities obtained below can be further improved if more efficient QP solvers were used. Moreover, each kernel evaluation is assumed to take constant time.

Consider first the case where probabilistic speedup is not used in Section 4.1.2. As proved in (Bădoiu and Clarkson, 2002), CVM converges in at most $2/\varepsilon$ iterations. In other words, the total number of iterations, and consequently the size of the final core set, are of $\tau = O(1/\varepsilon)$. In practice, it has often been observed that the size of the core set is much smaller than this worst-case theoretical upper bound[3] (Kumar et al., 2003). As only one core vector is added at each iteration, $|S_t| = t+2$. Initialization takes $O(m)$ time while distance computations in steps 2 and 3 take $O((t+2)^2+tm) = O(t^2+tm)$ time. Finding the MEB in step 4 takes $O((t+2)^3) = O(t^3)$ time, and the other operations take constant time. Hence, the tth iteration takes a total of $O(tm+t^3)$ time. The overall time for $\tau = O(1/\varepsilon)$ iterations is

$$T = \sum_{t=1}^{\tau} O(tm+t^3) = O(\tau^2 m + \tau^4) = O\left(\frac{m}{\varepsilon^2} + \frac{1}{\varepsilon^4}\right),$$

which is *linear* in m for a fixed ε.

Next, we consider its space complexity. As the m training patterns may be stored outside the core memory, the $O(m)$ space required will be ignored in the following. Since only the core vectors are involved in the QP, the space complexity for the tth iteration is $O(|\mathcal{S}_t|^2)$. As $\tau = O(1/\varepsilon)$, the space complexity for the whole procedure is $O(1/\varepsilon^2)$, which is *independent* of m for a fixed ε.

On the other hand, when probabilistic speedup is used, initialization only takes $O(1)$ time while distance computations in steps 2 and 3 take $O((t+2)^2) = O(t^2)$ time. Time for the other operations

3. This will also be corroborated by our experiments in Section 6.

remains the same. Hence, the tth iteration takes $O(t^3)$ time. As probabilistic speeedup may not find the furthest point in each iteration, τ may be larger than $2/\varepsilon$ though it can still be bounded by $O(1/\varepsilon^2)$ (Bădoiu et al., 2002). Hence, the whole procedure takes

$$T = \sum_{t=1}^{\tau} O(t^3) = O(\tau^4) = O\left(\frac{1}{\varepsilon^8}\right).$$

For a fixed ε, it is thus *independent* of m. The space complexity, which depends only on the number of iterations τ, becomes $O(1/\varepsilon^4)$.

When ε decreases, the CVM solution becomes closer to the exact optimal solution, but at the expense of higher time and space complexities. Such a tradeoff between efficiency and approximation quality is typical of all approximation schemes. Moreover, be cautioned that the O-notation is used for studying the asymptotic efficiency of algorithms. As we are interested in handling very large data sets, an algorithm that is asymptotically more efficient (in time and space) will be the best choice. However, on smaller problems, this may be outperformed by algorithms that are not as efficient asymptotically. These will be demonstrated experimentally in Section 6.

5. Related Work

The core set in CVM plays a similar role as the working set in other decomposition algorithms, and so these algorithms will be reviewed briefly in this Section. Following the convention in (Chang and Lin, 2004; Osuna et al., 1997b), the working set will be denoted B while the remaining subset of training patterns denoted N.

Chunking (Vapnik, 1998) is the first decomposition method used in SVM training. It starts with a random subset (*chunk*) of data as B and train an initial SVM. Support vectors in the chunk are retained while non-support vectors are replaced by patterns in N violating the KKT conditions. Then, the SVM is re-trained and the whole procedure repeated. Chunking suffers from the problem that the entire set of support vectors that have been identified will still need to be trained together at the end of the training process.

Osuna et al. (1997a) proposed another decomposition algorithm that fixes the size of the working set B. At each iteration, variables corresponding to patterns in N are frozen, while those in B are optimized in a QP sub-problem. After that, a new point in N violating the KKT conditions will replace some point in B. The SVM^{light} software (Joachims, 1999) follows the same scheme, though with a slightly different subset selection heuristic.

Going to the extreme, the sequential minimal optimization (SMO) algorithm (Platt, 1999) breaks the original, large QP into a series of smallest possible QPs, each involving only two variables. The first variable is chosen among points that violate the KKT conditions, while the second variable is chosen so as to have a large increase in the dual objective. This two-variable joint optimization process is repeated until the loose KKT conditions are fulfilled for all training patterns. By involving only two variables, SMO is advantageous in that each QP sub-problem can be solved analytically in an efficient way, without the use of a numerical QP solver. Moreover, as no matrix operations are involved, extra matrix storage is not required for keeping the kernel matrix. However, as each iteration only involves two variables in the optimization, SMO has slow convergence (Kao et al., 2004). Nevertheless, as each iteration is computationally simple, an overall speedup is often observed in practice.

Recently, Vishwanathan et al. (2003) proposed a related scale-up method called the SimpleSVM. At each iteration, a point violating the KKT conditions is added to the working set by using rank-one update on the kernel matrix. However, as pointed out in (Vishwanathan et al., 2003), storage is still a problem when the SimpleSVM is applied to large dense kernel matrices.

As discussed in Section 4.1, CVM is similar to these decomposition algorithms in many aspects, including initialization, subset selection and termination. However, subset selection in CVM is much simpler in comparison. Moreover, while decomposition algorithms allow training patterns to join and leave the working set multiple times, patterns once recruited as core vectors by the CVM will remain there for the whole training process. These allow the number of iterations, and consequently the time and space complexities, to be easily obtained for the CVM but not for the decomposition algorithms.

6. Experiments

In this Section, we implement the two-class L2-SVM in Section 3.2.2 and illustrate the scaling behavior of CVM (in C++) on several toy and real-world data sets. Table 1 summarizes the characteristics of the data sets used. For comparison, we also run the following SVM implementations:[4]

1. L2-SVM: LIBSVM implementation (in C++);
2. L2-SVM: LSVM implementation (in MATLAB), with low-rank approximation (Fine and Scheinberg, 2001) of the kernel matrix added;
3. L2-SVM: RSVM (Lee and Mangasarian, 2001) implementation (in MATLAB). The RSVM addresses the scale-up issue by solving a smaller optimization problem that involves a random $\bar{m} \times m$ rectangular subset of the kernel matrix. Here, $\bar{m}$ is set to 10% of m;
4. L1-SVM: LIBSVM implementation (in C++);
5. L1-SVM: SimpleSVM (Vishwanathan et al., 2003) implementation (in MATLAB).

Parameters are used in their default settings unless otherwise specified. Since our focus is on nonlinear kernels, we use the Gaussian kernel $k(\mathbf{x},\mathbf{y}) = \exp(-\|\mathbf{x}-\mathbf{y}\|^2/\beta)$ with $\beta = \frac{1}{m^2}\sum_{i,j=1}^{m} \|\mathbf{x}_i - \mathbf{x}_j\|^2$ unless otherwise specified. Experiments are performed on Pentium–4 machines running Windows XP. Detailed machine configurations will be reported in each section.

Our CVM implementation is adapted from the LIBSVM, and uses SMO for solving each QP sub-problem in step 4. As discussed in Section 4.1.4, warm start is used to initialize each QP sub-problem. Besides, as in LIBSVM, our CVM uses caching (with the same cache size as in the other LIBSVM implementations above) and stores all the training patterns in main memory. For simplicity, shrinking (Joachims, 1999) is not used in our current CVM implementation. Besides, we employ the probabilistic speedup method[5] as discussed in Section 4.1.2. On termination, we perform the (probabilistic) test in step 2 a few times so as to ensure that almost all the points have been covered by the $(1+\varepsilon)$-ball. The value of ε is fixed at 10^{-6} in all the experiments. As in other

4. Our CVM implementation can be downloaded from http://www.cs.ust.hk/∼jamesk/cvm.zip. LIBSVM can be downloaded from http://www.csie.ntu.edu.tw/∼cjlin/libsvm/; LSVM from http://www.cs.wisc.edu/dmi/lsvm; and SimpleSVM from http://asi.insa-rouen.fr/∼gloosli/. Moreover, we followed http://www.csie.ntu.edu.tw/∼cjlin/libsvm/faq.html in adapting the LIBSVM package for L2-SVM.
5. Following (Smola and Schölkopf, 2000), a random sample of size 59 is used.

date set	max training set size	# attributes
checkerboard	1,000,000	2
forest cover type	522,911	54
extended USPS digits	266,079	676
extended MIT face	889,986	361
KDDCUP-99 intrusion detection	4,898,431	127
UCI adult	32,561	123

Table 1: Data sets used in the experiments.

decomposition methods, the use of a very stringent stopping criterion is not necessary in practice. Preliminary studies show that $\varepsilon = 10^{-6}$ is acceptable for most tasks. Using an even smaller ε does not show improved generalization performance, but may increase the training time unnecessarily.

6.1 Checkerboard Data

We first experiment on the 4×4 checkerboard data (Figure 2) commonly used for evaluating large-scale SVM implementations (Lee and Mangasarian, 2001; Mangasarian and Musicant, 2001b; Schwaighofer and Tresp, 2001). We use training sets with a maximum of 1 million points and 2,000 independent points for testing. Of course, this problem does not need so many points for training, but it is convenient for illustrating the scaling properties. Preliminary study suggests a value of $C = 1000$. A 3.2GHz Pentium–4 machine with 512MB RAM is used.

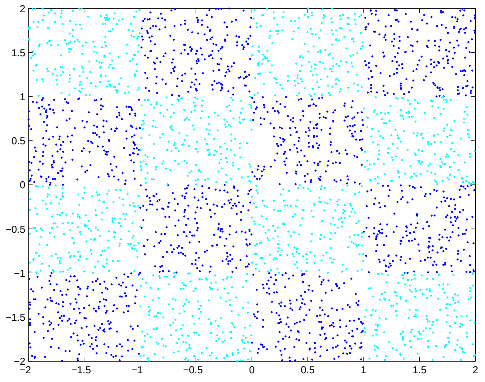


Figure 2: The 4×4 checkerboard data set.

Experimentally, L2-SVM with low rank approximation does not yield satisfactory performance on this data set, and so its result is not reported here. RSVM, on the other hand, has to keep a rectangular kernel matrix of size $\bar{m} \times m$ ($\bar{m}$ being the number of random samples used), and cannot be run on our machine when m exceeds 10K. Similarly, the SimpleSVM has to store the kernel matrix of the active set, and runs into storage problem when m exceeds 30K.

Results are shown in Figure 3. As can be seen, CVM is as accurate as the other implementations. Besides, it is much faster[6] and produces far fewer support vectors (which implies faster testing) on large data sets. In particular, one million patterns can be processed in under 13 seconds. On the other hand, for relatively small training sets, with less than 10K patterns, LIBSVM is faster. This, however, is to be expected as LIBSVM uses more sophisticated heuristics and so will be more efficient on small-to-medium sized data sets. Figure 3(b) also shows the core set size, which can be seen to be small and its curve basically overlaps with that of the CVM. Thus, almost all the core vectors are useful support vectors. Moreover, it also confirms our theoretical findings that both time and space required are constant w.r.t. the training set size, when it becomes large enough.

6.2 Forest Cover Type Data

The forest cover type data set[7] has been used for large scale SVM training (e.g., (Bakir et al., 2005; Collobert et al., 2002)). Following (Collobert et al., 2002), we aim at separating class 2 from the other classes. 1% – 90% of the whole data set (with a maximum of 522,911 patterns) are used for training while the remaining are used for testing. We use the Gaussian kernel with $\beta = 10000$ and $C = 10000$. Experiments are performed on a 3.2GHz Pentium–4 machine with 512MB RAM.

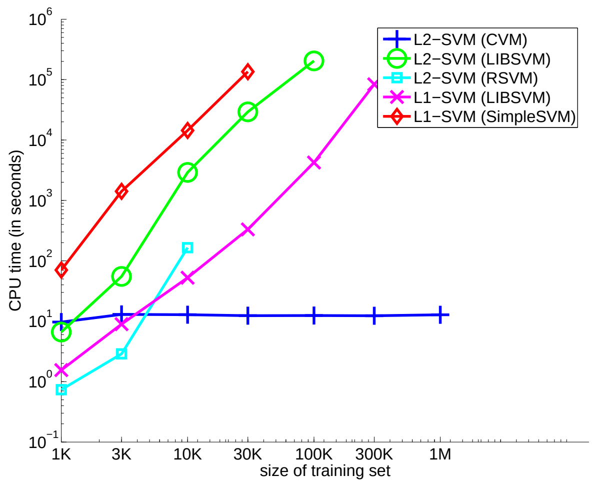
Preliminary studies show that the number of support vectors is over ten thousands. Consequently, RSVM and SimpleSVM cannot be run on our machine. Similarly, for low rank approximation, preliminary studies show that over thousands of basis vectors are required for a good approximation. Therefore, only the two LIBSVM implementations will be compared with the CVM here.

As can be seen from Figure 4, CVM is, again, as accurate as the others. Note that when the training set is small, more training patterns bring in additional information useful for classification and so the number of core vectors increases with training set size. However, after processing around 100K patterns, both the time and space requirements of CVM begin to exhibit a constant scaling with the training set size. With hindsight, one might simply sample 100K training patterns and hope to obtain comparable results.[8] However, for satisfactory classification performance, different problems require samples of different sizes and CVM has the important advantage that the required sample size does not have to be pre-specified. Without such prior knowledge, random sampling gives poor testing results, as demonstrated in (Lee and Mangasarian, 2001).

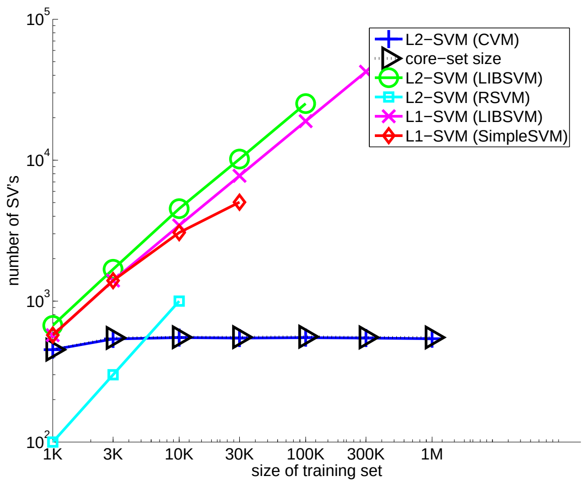
6.3 Extended USPS Digits Data

In this experiment, our task is to classify digits zero from one in an extended version of the USPS data set.[9] The original training set has 1,005 zeros and 1,194 ones, while the test set has 359 zeros and 264 ones. To better study the scaling behavior, we extend this data set by first converting the resolution from 16×16 to 26×26, and then generate new images by shifting the original ones in all directions for up to five pixels. Thus, the resultant training set has a total of $(1005 + 1194) \times 11^2 =$

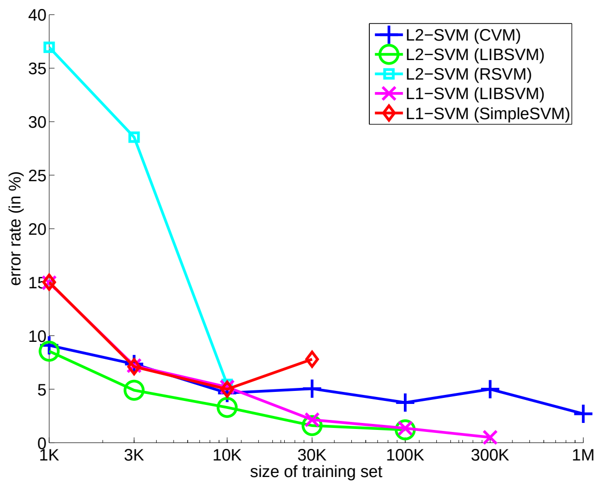
6. The CPU time only measures the time for training the SVM. Time for reading the training patterns into main memory is not included. Moreover, as some implementations are in MATLAB, so not all the CPU time measurements can be directly compared. However, it is still useful to note the constant scaling exhibited by the CVM and its speed advantage over other C++ implementations, when the data set is large.
7. http://kdd.ics.uci.edu/databases/covertype/covertype.html
8. In fact, we tried both LIBSVM implementations on a random sample of 100K training patterns, but their testing accuracies are inferior to that of CVM.
9. http://www.kernel-machines.org/data/usps.mat.gz



(a) CPU time.

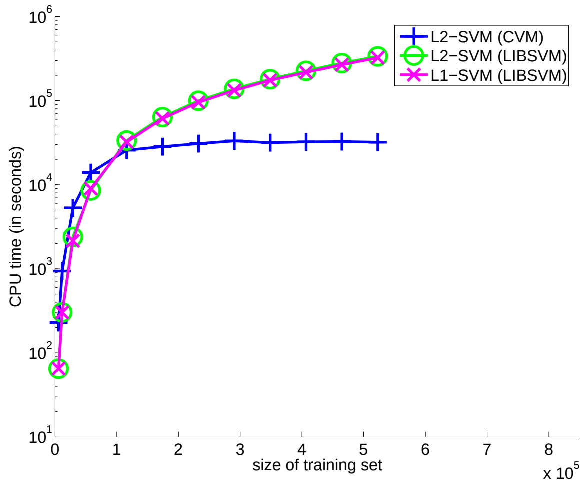


(b) number of support vectors.

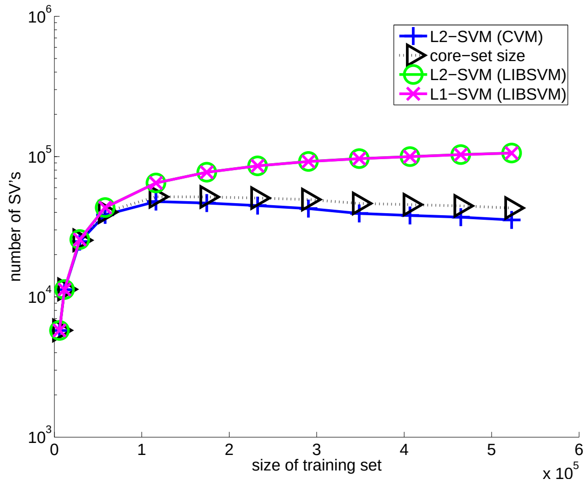


(c) testing error.

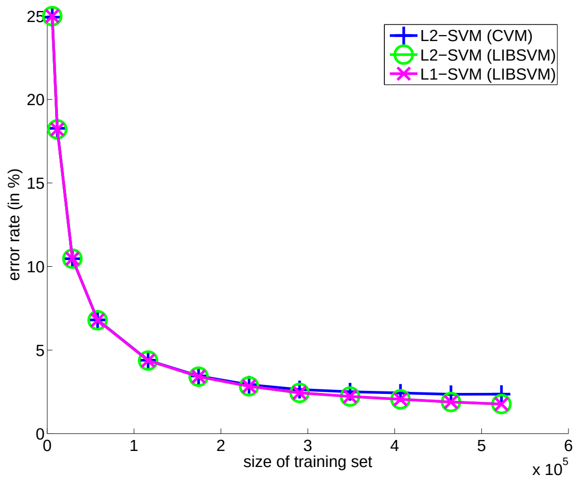
Figure 3: Results on the checkerboard data set (Except for the CVM, the other implementations have to terminate early because of not enough memory and/or the training time is too long). Note that the CPU time, number of support vectors, and size of the training set are in log scale.



(a) CPU time.



(b) number of support vectors.



(c) testing error.

Figure 4: Results on the forest cover type data set. Note that the CPU time and number of support vectors are in log scale.

$266,079$ patterns while the extended test set has $(359+264)\times 11^2 = 753,83$ patterns. In this experiment, $C = 100$ and a 3.2GHz Pentium–4 machine with 512MB RAM is used.

As can be seen from Figure 5, the behavior of CVM is again similar to those in the previous sections. In particular, both the time and space requirements of CVM increase when the training set is small. They then stabilize at around 30K patterns and CVM becomes faster than the other decomposition algorithms.

6.4 Extended MIT Face Data

In this Section, we perform face detection using an extended version of the MIT face database[10] (Heisele et al., 2000; Sung, 1996). The original data set has 6,977 training images (with 2,429 faces and 4,548 nonfaces) and 24,045 test images (472 faces and 23,573 nonfaces). The original 19×19 grayscale images are first enlarged to 21×21. To better study the scaling behavior of various SVM implementations, we again enlarge the training set by generating artificial samples. As in (Heisele et al., 2000; Osuna et al., 1997b), additional nonfaces are extracting from images that do not contain faces (e.g., images of landscapes, trees, buildings, etc.). As for the set of faces, we enlarge it by applying various image transformations (including blurring, flipping and rotating) to the original faces. The following three training sets are thus created (Table 6.4):

1. Set A: This is obtained by adding 477,366 nonfaces to the original training set, with the nonface images extracted from 100 photos randomly collected from the web.

2. Set B: Each training face is blurred by the arithmetic mean filter (with window sizes 2×2, 3×3 and 4×4, respectively) and added to set A. They are then flipped laterally, leading to a total of $2429\times 4\times 2 = 19,432$ faces.

3. Set C: Each face in set B is rotated between -20^o and 20^o, in increments of 2^o. This results in a total of $19432\times 21 = 408,072$ faces.

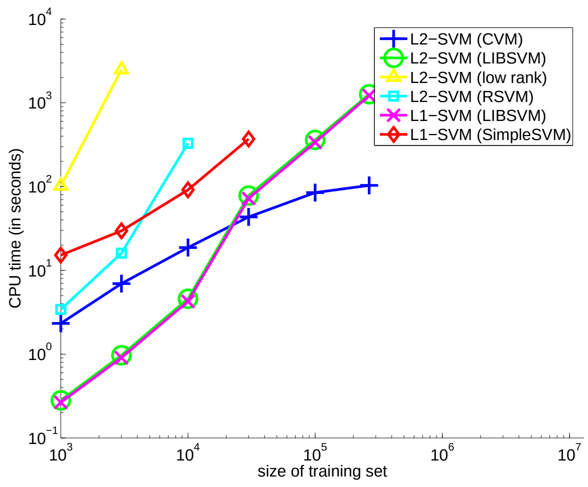
In this experiment, $C = 20$ and a 2.5GHz Pentium–4 machine with 1GB RAM is used. Moreover, a dense data format, which is more appropriate for this data set, is used in all the implementations. Recall that the intent of this experiment is on studying the scaling behavior rather than on obtaining state-of-the-art face detection performance. Nevertheless, the ability of CVM in handling very large data sets could make it a better base classifier in powerful face detection systems such as the boosted cascade (Viola and Jones, 2001).

training set	# faces	# nonfaces	total
original	2,429	4,548	6,977
set A	2,429	481,914	484,343
set B	19,432	481,914	501,346
set C	408,072	481,914	889,986

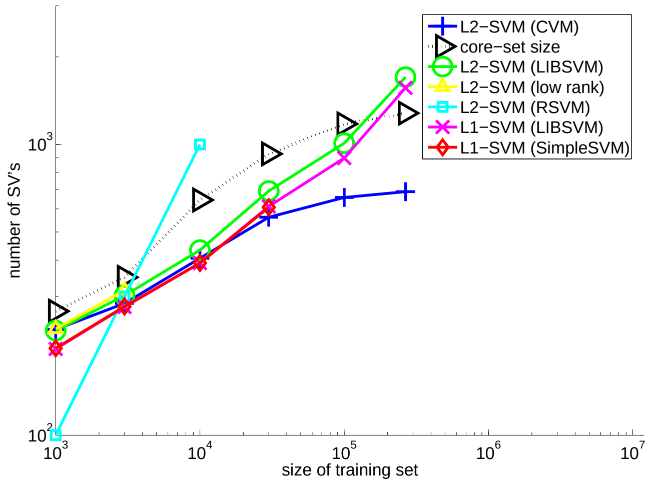
Table 2: Number of faces and nonfaces in the face detection data sets.

Because of the imbalanced nature of this data set, the testing error is inappropriate for performance evaluation here. Instead, we will use the AUC (area under the ROC curve), which has been

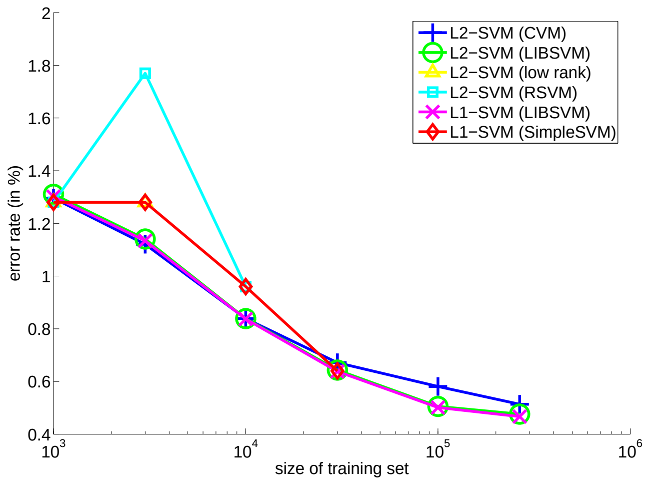
10. http://cbcl.mit.edu/cbcl/software-datasets/FaceData2.html



(a) CPU time.



(b) number of support vectors.



(c) testing error.

Figure 5: Results on the extended USPS digits data set (Except for the CVM, the other implementations have to terminate early because of not enough memory and/or the training time is too long). Note that the CPU time, number of support vectors, and size of the training set are in log scale.

commonly used for face detectors. The ROC (receiver operating characteristic) curve (Bradley, 1997) plots TP on the Y-axis and the false positive rate

$$\text{FP} = \frac{\text{negatives incorrectly classified}}{\text{total negatives}}$$

on the X-axis. Here, faces are treated as positives while non-faces as negatives. The AUC is always between 0 and 1. A perfect face detector will have unit AUC, while random guessing will have an AUC of 0.5. Another performance measure that will be reported is the balanced loss (Weston et al., 2002)

$$\ell_{bal} = 1 - \frac{\text{TP} + \text{TN}}{2},$$

which is also suitable for imbalanced data sets. Here,

$$\text{TP} = \frac{\text{positives correctly classified}}{\text{total positives}}, \quad \text{TN} = \frac{\text{negatives correctly classified}}{\text{total negatives}},$$

are the true positive and true negative rates respectively.

The ROC on using CVM is shown in Figure 6, which demonstrates the usefulness of using extra faces and nonfaces in training. This is also reflected in Figure 7, which shows that the time and space requirements of CVM are increasing with larger training sets. Even in this non-asymptotic case, the CVM still significantly outperforms both LIBSVM implementations in terms of training time and number of support vectors, while the values of AUC and ℓ_{bal} are again very competitive. Note also that the LIBSVM implementations of both L1- and L2-SVMs do not perform well (in terms of ℓ_{bal}) on the highly imbalanced set A. On the other hand, CVM shows a steady improvement and is less affected by the skewed distribution. In general, the performance of SVMs could be improved by assigning different penalty parameters (C's) to the classes. A more detailed study on the use of CVM in an imbalanced setting will be conducted in the future.

6.5 KDDCUP-99 Intrusion Detection Data

This intrusion detection data set[11] has been used for the Third International Knowledge Discovery and Data Mining Tools Competition, which was held in conjunction with KDD-99. The training set contains 4,898,431 connection records, which are processed from about four gigabytes of compressed binary TCP dump data from seven weeks of network traffic. Another two weeks of data produced the test data with 311,029 patterns. The data set includes a wide variety of intrusions simulated in a military network environment. There are a total of 24 training attack types, and an additional 14 types that appear in the test data only.

We follow the same setup in (Yu et al., 2003). The task is to separate normal connections from attacks. Each original pattern has 34 continuous features and 7 symbolic features. We normalize each continuous feature to the range zero and one, and transform each symbolic feature to multiple binary features. Yu et al. (2003) used the clustering-based SVM (CB-SVM), which incorporates the hierarchical micro-clustering algorithm BIRCH (Zhang et al., 1996) to reduce the number of patterns in SVM training. However, CB-SVM is restricted to the use of linear kernels. In our experiments with the CVM, we will continue the use of the Gaussian kernel (with $\beta = 1000$ and

11. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

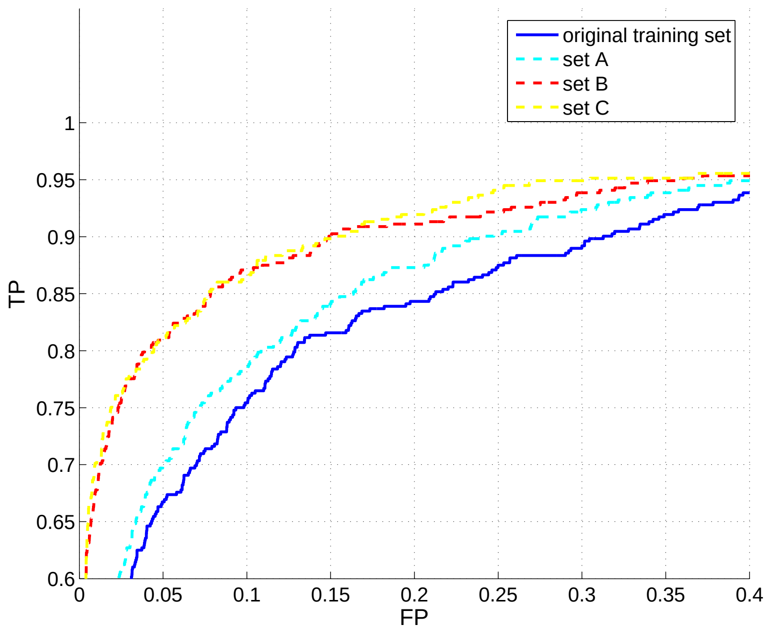


Figure 6: ROC of the extended MIT face data set on using CVM.

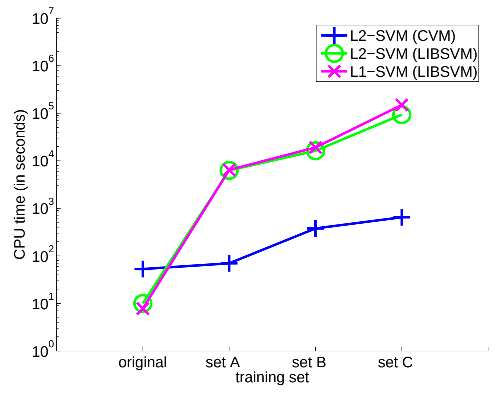
$C = 10^6$) as in the previous sections. Moreover, as the whole data set is stored in the core in our current implementation, we use a 3.2GHz Pentium–4 machine with 2GB RAM.

Table 6.5 compares the results of CVM with those reported in (Yu et al., 2003), which include SVMs using random sampling, active learning (Schohn and Cohn, 2000) and CB-SVM. Surprisingly, our CVM on the whole training set (which has around five million patterns) takes only 1.4 seconds, and yields a lower testing error than all other methods. The performance of CVM on this data set, as evaluated by some more measures and the ROC, are reported in Table 6.5 and Figure 8 respectively.

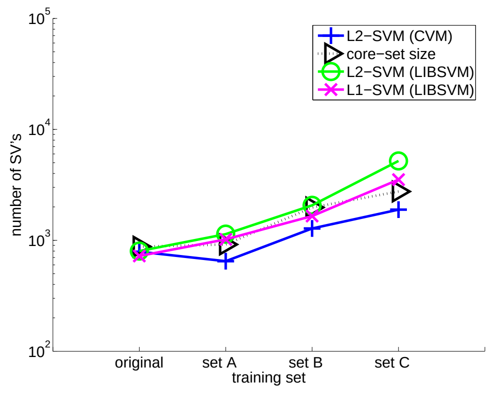
6.6 Relatively Small Data Sets: UCI Adult Data

Following (Platt, 1999), we use training sets[12] with up to 32,562 patterns. Experiments are performed with $C = 0.1$ and on a 3.2GHz Pentium–4 machine with 512MB RAM. As can be seen in Figure 9, CVM is still among the most accurate methods. However, as this data set is relatively small, more training patterns do carry more classification information. Hence, as discussed in Section 6.2, the number of iterations, the core set size and consequently the CPU time all increase with the number of training patterns. From another perspective, recall that the worst case core set size is $2/\varepsilon$, independent of m (Section 4.3). For the value of $\varepsilon = 10^{-6}$ used here, $2/\varepsilon = 2 \times 10^6$. Besides, although we have seen that the actual size of the core set is often much smaller than this worst case value, however, when $m \ll 2/\varepsilon$, the number of core vectors can still be dependent on m. More-

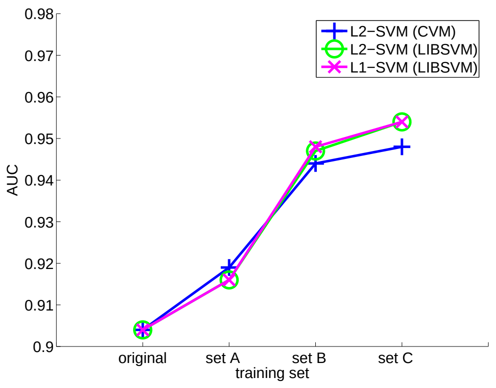
12. ftp://ftp.ics.uci.edu/pub/machine-learning-databases/adult



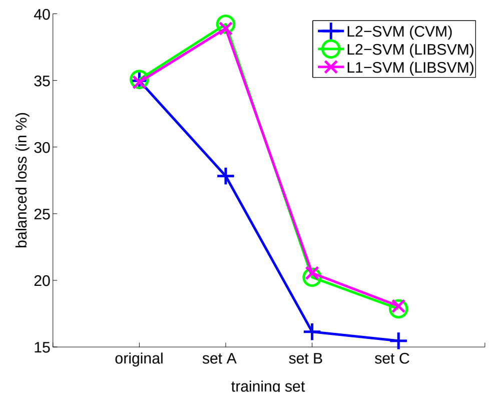
(a) CPU time.



(b) number of support vectors.



(c) AUC.



(d) ℓ_{bal}.

Figure 7: Results on the extended MIT face data set. Note that the CPU time and number of support vectors are in log scale.

method		# training patterns input to SVM	# test errors	SVM training time (in sec)	other processing time (in sec)
random sampling	0.001%	47	25,713	0.000991	500.02
	0.01%	515	25,030	0.120689	502.59
	0.1%	4,917	25,531	6.944	504.54
	1%	49,204	25,700	604.54	509.19
	5%	245,364	25,587	15827.3	524.31
active learning		747	21,634	94192.213	
CB-SVM		4,090	20,938	7.639	4745.483
CVM		4,898,431	19,513	1.4	

Table 3: Results on the KDDCUP-99 intrusion detection data set by CVM and methods reported in (Yu et al., 2003). Here, "other processing time" refers to the (1) sampling time for SVM with random sampling; and (2) clustering time for CB-SVM. For SVM with active learning and CVM, the total training time required is reported. Note that Yu et al. (2003) used a 800MHz Pentium-3 machine with 906MB RAM while we use a 3.2GHz Pentium–4 machine with 2GB RAM. Hence, the time measurements are for reference only and cannot be directly compared.

AUC	ℓ_{bal}	# core vectors	# support vectors
0.977	0.042	55	20

Table 4: More performance measures of CVM on the KDDCUP-99 intrusion detection data.

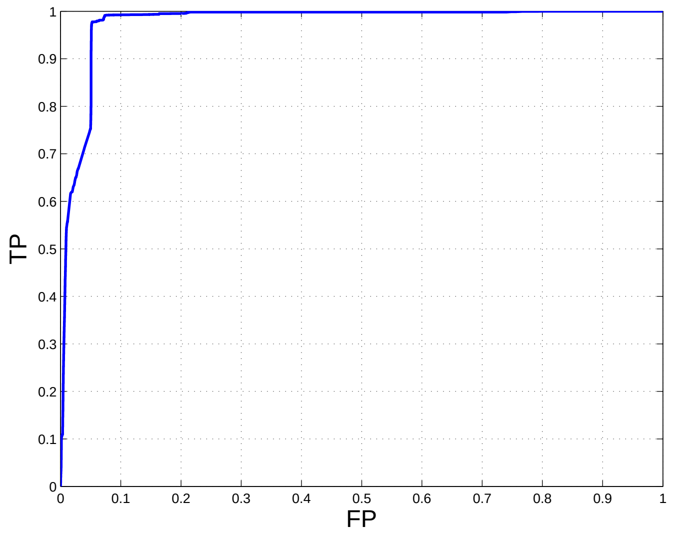
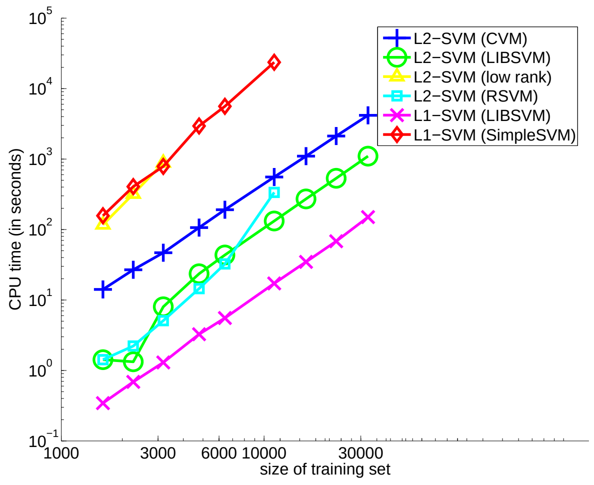
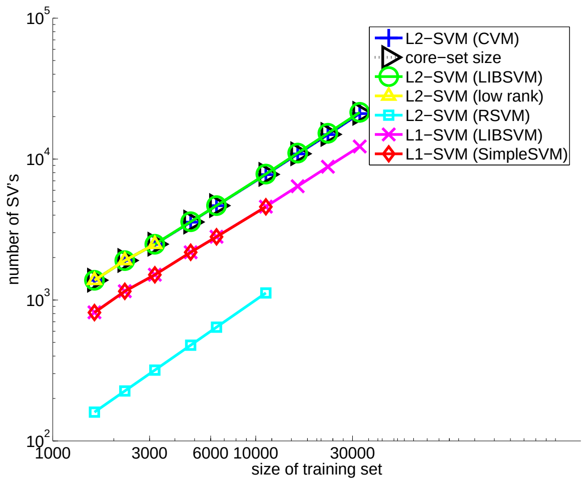


Figure 8: ROC of the the KDDCUP-99 intrusion detection data using CVM.

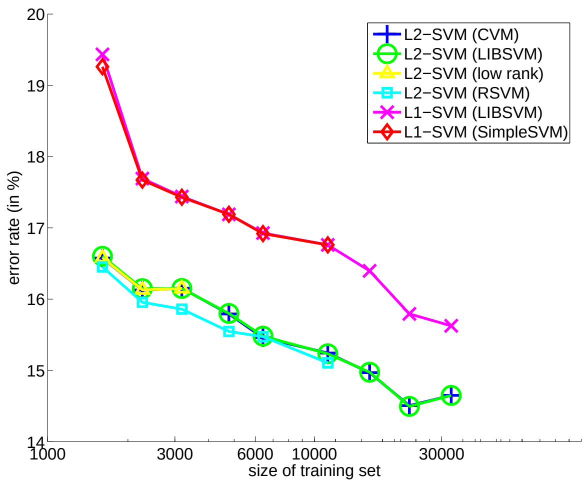
over, as has been observed in the previous sections, the CVM is slower than the more sophisticated LIBSVM on processing these smaller data sets.



(a) CPU time.



(b) number of support vectors.



(c) testing error.

Figure 9: Results on the UCI adult data set (The other implementations have to terminate early because of not enough memory and/or the training time is too long). Note that the CPU time, number of SV's and size of training set are in log scale.

7. Conclusion

In this paper, we exploit the "approximateness" in practical SVM implementations to scale-up SVM training. We formulate kernel methods (including the soft-margin one-class and two-class SVMs) as equivalent MEB problems, and then obtain approximately optimal solutions efficiently with the use of core sets. The proposed CVM procedure is simple, and does not require sophisticated heuristics as in other decomposition methods. Moreover, despite its simplicity, CVM has small asymptotic time and space complexities. In particular, for a fixed ε, its asymptotic time complexity is *linear* in the training set size m while its space complexity is *independent* of m. This can be further improved when probabilistic speedup is used. Experimentally, it is as accurate as existing SVM implementations, but is much faster and produces far fewer support vectors (and thus faster testing) on large data sets. On the other hand, on relatively small data sets where $m \ll 2/\varepsilon$, SMO can be faster. Besides, although we have fixed the value of ε in the experiments, one could also vary the value of ε to adjust the tradeoff between efficiency and approximation quality. In general, with a smaller ε, the CVM solution becomes closer to the exact optimal solution, but at the expense of higher time and space complexities. Our experience suggests that a fixed value of $\varepsilon = 10^{-6}$ is acceptable for most tasks.

The introduction of CVM opens new doors for applying kernel methods to data-intensive applications involving very large data sets. The use of approximation algorithms also brings immense opportunities to scaling up other kernel methods. For example, we have obtained preliminary success in extending support vector regression using the CVM technique. In the future, we will also apply CVM-like approximation algorithms to other kernel-related learning problems such as imbalanced learning, ranking and clustering. The iterative recruitment of core vectors is also similar to incremental procedures (Cauwenberghs and Poggio, 2001; Fung and Mangasarian, 2002), and this connection will be further explored. Besides, although the CVM can obtain much fewer support vectors than standard SVM implementations on large data sets, the number of support vectors may still be too large for real-time testing. As the core vectors in CVM are added incrementally and never removed, it is thus possible that some of them might be redundant. We will consider post-processing methods to further reduce the number of support vectors. Finally, all the training patterns are currently stored in the main memory. We anticipate that even larger data sets can be handled, possibly with reduced speed, when traditional scale-up techniques such as out-of-core storage and low-rank approximation are also incorporated.

Acknowledgements

This research has been partially supported by the Research Grants Council of the Hong Kong Special Administrative Region. The author would also like to thank the anonymous reviewers for their constructive comments on an earlier version of this paper.

References

D. Achlioptas, F. McSherry, and B. Schölkopf. Sampling techniques for kernel methods. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

G. H. Bakir, J. Weston, and L. Bottou. Breaking SVM complexity with cross-training. In *Advances in Neural Information Processing Systems 17*, Cambridge, MA, 2005. MIT Press.

D. Boley and D. Cao. Training support vector machine using adaptive clustering. In *Proceedings of the SIAM International Conference on Data Mining*, pages 126–137, Lake Buena Vista, FL, USA, April 2004.

A. P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.

M. Bădoiu and K. L. Clarkson. Optimal core sets for balls. In *DIMACS Workshop on Computational Geometry*, 2002.

M. Bădoiu, S. Har-Peled, and P. Indyk. Approximate clustering via core sets. In *Proceedings of 34th Annual ACM Symposium on Theory of Computing*, pages 250–257, Montréal, Québec, Canada, 2002.

G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, Cambridge, MA, 2001. MIT Press.

T. M. Chan. Approximating the diameter, width, smallest enclosing cylinder, and minimum-width annulus. In *Proceedings of the Sixteenth Annual Symposium on Computational Geometry*, pages 300–309, Clear Water Bay, Hong Kong, 2000.

C.-C. Chang and C.-J. Lin. *LIBSVM: a Library for Support Vector Machines*, 2004. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

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. S. Chu, I. W. Tsang, and J. T. Kwok. Scaling up support vector data description by using core-sets. In *Proceedings of the International Joint Conference on Neural Networks*, pages 425–430, Budapest, Hungary, July 2004.

R. Collobert, S. Bengio, and Y. Bengio. A parallel mixture of SVMs for very large scale problems. *Neural Computation*, 14(5):1105–1114, May 2002.

S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243–264, December 2001.

T. Friess, N. Cristianini, and C. Campbell. The Kernel-Adatron algorithm: a fast and simple learning procedure for support vector machines. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 188–196, Madison, Wisconsin, USA, July 1998.

G. Fung and O. L. Mangasarian. Incremental support vector machine classification. In R. Grossman, H. Mannila, and R. Motwani, editors, *Proceedings of the Second SIAM International Conference on Data Mining*, pages 247–260, Arlington, Virginia, USA, 2002.

G. Fung and O. L. Mangasarian. Finite Newton method for Lagrangian support vector machine classification. *Neurocomputing*, 55:39–55, 2003.

M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

S. Har-Peled and Y. Wang. Shape fitting with outliers. *SIAM Journal on Computing*, 33(2):269–285, 2004.

B. Heisele, T. Poggio, and M. Pontil. Face detection in still gray images. A.I. memo 1687, Center for Biological and Computational Learning, MIT, Cambridge, MA, 2000.

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 Learning*, pages 169–184. MIT Press, Cambridge, MA, 1999.

W.-C. Kao, K.-M. Chung, C.-L. Sun, and C.-J. Lin. Decomposition methods for linear support vector machines. *Neural Computation*, 16:1689–1704, 2004.

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, January 2000.

P. Kumar, J. S. B. Mitchell, and A. Yildirim. Approximate minimum enclosing balls in high dimensions using core-sets. *ACM Journal of Experimental Algorithmics*, 8, January 2003.

Y.-J. Lee and O. L. Mangasarian. RSVM: Reduced support vector machines. In *Proceeding of the First SIAM International Conference on Data Mining*, 2001.

O. L. Mangasarian and D. R. Musicant. Active set support vector machine classification. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 577–583, Cambridge, MA, 2001a. MIT Press.

O. L. Mangasarian and D. R. Musicant. Lagrangian support vector machines. *Journal of Machine Learning Research*, 1:161–177, 2001b.

N. Megiddo. Linear-time algorithms for linear programming in R^3 and related problems. *SIAM Journal on Computing*, 12:759–776, 1983.

F. Nielsen and R. Nock. Approximating smallest enclosing balls. In *Proceedings of International Conference on Computational Science and Its Applications*, volume 3045, pages 147–157, 2004.

E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. In *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, pages 276–285, Amelia Island, FL, USA, 1997a.

E. Osuna, R. Freund, and F. Girosi. Training support vector machines: an application to face detection. In *Proceedings of Computer Vision and Pattern Recognition*, pages 130–136, San Juan, Puerto Rico, June 1997b.

D. Pavlov, D. Chudova, and P. Smyth. Towards scalable support vector machines using squashing. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 295–299, Boston, Massachusetts, USA, 2000a.

D. Pavlov, J. Mao, and B. Dom. Scaling-up support vector machines using boosting algorithm. In *Proceedings of the International Conference on Pattern Recognition*, volume 2, pages 2219–2222, Barcelona, Spain, September 2000b.

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*, pages 185–208. MIT Press, Cambridge, MA, 1999.

F. P. Preparata. *Computational Geometry: An Introduction*. Springer-Verlag, 1985.

D. Roobaert. DirectSVM: a simple support vector machine perceptron. In *Proceedings of IEEE International Workshop on Neural Networks for Signal Processing*, pages 356–365, Sydney, Australia, December 2000.

G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 839–846, San Francisco, CA, USA, 2000. Morgan Kaufmann.

B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, July 2001.

B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

A. Schwaighofer and V. Tresp. The Bayesian committee support vector machine. In G. Dorffner, H. Bischof, and K. Hornik, editors, *Proceedings of the International Conference on Artificial Neural Networks*, pages 411–417. Springer Verlag, 2001.

A. Smola and B. Schölkopf. Sparse greedy matrix approximation for machine learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 911–918, Stanford, CA, USA, June 2000.

A. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14 (3):199–222, August 2004.

K.-K. Sung. *Learning and Example Selection for Object and Pattern Recognition*. PhD thesis, Artificial Intelligence Laboratory and Center for Biological and Computational Learning, MIT, Cambridge, MA, 1996.

J. J. Sylvester. A question in the geometry of situation. *Quarterly Journal on Mathematics*, 1:79, 1857.

D. M. J. Tax and R. P. W. Duin. Support vector domain description. *Pattern Recognition Letters*, 20 (14):1191–1199, 1999.

S. Tong and D. Koller. Support vector machine active learning with applications to text classification. In *Proceedings of the 17th International Conference on Machine Learning*, pages 999–1006, San Francisco, CA, USA, 2000. Morgan Kaufmann.

V. Tresp. Scaling kernel-based systems to large data sets. *Data Mining and Knowledge Discovery*, 5(3):197–211, 2001.

I. W. Tsang, J. T. Kwok, and P.-M. Cheung. Very large SVM training using core vector machines. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, Barbados, January 2005.

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

V. V. Vazirani. *Approximation Algorithms*. Springer, 2001.

P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, volume 1, pages 1063–6919, 2001.

S. V. N. Vishwanathan, A. J. Smola, and M. N. Murty. SimpleSVM. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 760–767, Washington, D.C., USA, August 2003.

E. Welzl. Smallest enclosing disks (balls and ellipsoids). In H. Maurer, editor, *New Results and New Trends in Computer Science*, pages 359–370. Springer-Verlag, 1991.

J. Weston, B. Schölkopf, E. Eskin, C. Leslie, and S. W. Noble. Dealing with large diagonals in kernel matrices. *Principles of Data Mining and Knowledge Discovery, Springer Lecture Notes in Computer Science 243*, 2002.

C. K. I. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, Cambridge, MA, 2001. MIT Press.

C. Yang, R. Duraiswami, and L. Davis. Efficient kernel machines using the improved fast Gauss transform. In *Advances in Neural Information Processing Systems 17*, Cambridge, MA, 2005. MIT Press.

H. Yu, J. Yang, and J. Han. Classifying large data sets using SVM with hierarchical clusters. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 306–315, Washington DC, USA, 2003.

T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An efficient data clustering method for very large databases. In H. V. Jagadish and I. S. Mumick, editors, *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 103–114, Montreal, Quebec, Canada, June 1996. ACM Press.

Journal of Machine Learning Research 6 (2005) 393–425 Submitted 6/04; Published 4/05

Generalization Bounds for the Area Under the ROC Curve[*]

Shivani Agarwal SAGARWAL@CS.UIUC.EDU
Department of Computer Science
University of Illinois at Urbana-Champaign
201 North Goodwin Avenue
Urbana, IL 61801, USA

Thore Graepel THOREG@MICROSOFT.COM
Ralf Herbrich RHERB@MICROSOFT.COM
Microsoft Research
7 JJ Thomson Avenue
Cambridge CB3 0FB, UK

Sariel Har-Peled SARIEL@CS.UIUC.EDU
Dan Roth DANR@CS.UIUC.EDU
Department of Computer Science
University of Illinois at Urbana-Champaign
201 North Goodwin Avenue
Urbana, IL 61801, USA

Editor: Michael I. Jordan

Abstract

We study generalization properties of the area under the ROC curve (AUC), a quantity that has been advocated as an evaluation criterion for the bipartite ranking problem. The AUC is a different term than the error rate used for evaluation in classification problems; consequently, existing generalization bounds for the classification error rate cannot be used to draw conclusions about the AUC. In this paper, we define the expected accuracy of a ranking function (analogous to the expected error rate of a classification function), and derive distribution-free probabilistic bounds on the deviation of the empirical AUC of a ranking function (observed on a finite data sequence) from its expected accuracy. We derive both a large deviation bound, which serves to bound the expected accuracy of a ranking function in terms of its empirical AUC on a test sequence, and a uniform convergence bound, which serves to bound the expected accuracy of a learned ranking function in terms of its empirical AUC on a training sequence. Our uniform convergence bound is expressed in terms of a new set of combinatorial parameters that we term the bipartite rank-shatter coefficients; these play the same role in our result as do the standard VC-dimension related shatter coefficients (also known as the growth function) in uniform convergence results for the classification error rate. A comparison of our result with a recent uniform convergence result derived by Freund et al. (2003) for a quantity closely related to the AUC shows that the bound provided by our result can be considerably tighter.

Keywords: generalization bounds, area under the ROC curve, ranking, large deviations, uniform convergence

*. Parts of the results contained in this paper were presented at the *18th Annual Conference on Neural Information Processing Systems* in December, 2004 (Agarwal et al., 2005a) and at the *10th International Workshop on Artificial Intelligence and Statistics* in January, 2005 (Agarwal et al., 2005b).

1. Introduction

In many learning problems, the goal is not simply to classify objects into one of a fixed number of classes; instead, a *ranking* of objects is desired. This is the case, for example, in information retrieval problems, where one is interested in retrieving documents from some database that are 'relevant' to a given query or topic. In such problems, one wants to return to the user a list of documents that contains relevant documents at the top and irrelevant documents at the bottom; in other words, one wants a ranking of the documents such that relevant documents are ranked higher than irrelevant documents.

The problem of ranking has been studied from a learning perspective under a variety of settings (Cohen et al., 1999; Herbrich et al., 2000; Crammer and Singer, 2002; Freund et al., 2003). Here we consider the setting in which objects come from two categories, positive and negative; the learner is given examples of objects labeled as positive or negative, and the goal is to learn a ranking in which positive objects are ranked higher than negative ones. This captures, for example, the information retrieval problem described above; in this case, the training examples given to the learner consist of documents labeled as relevant (positive) or irrelevant (negative). This form of ranking problem corresponds to the 'bipartite feedback' case of Freund et al. (2003); for this reason, we refer to it as the *bipartite* ranking problem.

Formally, the setting of the bipartite ranking problem is similar to that of the binary classification problem. In both problems, there is an instance space $\mathcal{X}$ from which instances are drawn, and a set of two class labels $\mathcal{Y}$ which we take without loss of generality to be $\mathcal{Y} = \{-1, +1\}$. One is given a finite sequence of labeled training examples $S = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_M, y_M)) \in (\mathcal{X} \times \mathcal{Y})^M$, and the goal is to learn a function based on this training sequence. However, the form of the function to be learned in the two problems is different. In classification, one seeks a binary-valued function $h : \mathcal{X} \rightarrow \mathcal{Y}$ that predicts the class of a new instance in $\mathcal{X}$. On the other hand, in ranking, one seeks a *real-valued* function $f : \mathcal{X} \rightarrow \mathbb{R}$ that induces a ranking over $\mathcal{X}$; an instance that is assigned a higher value by f is ranked higher than one that is assigned a lower value by f.

What is a good classification or ranking function? Intuitively, a good classification function should classify most instances correctly, while a good ranking function should rank most instances labeled as positive higher than most instances labeled as negative. At first thought, these intuitions might suggest that one problem could be reduced to the other; that a good solution to one could be used to obtain a good solution to the other. Indeed, several approaches to learning ranking functions have involved using a standard classification algorithm that produces a classification function h of the form $h(\mathbf{x}) = \theta(f_h(\mathbf{x}))$ for some real-valued function $f_h : \mathcal{X} \rightarrow \mathbb{R}$, where

$$\theta(u) \quad = \quad \begin{cases} 1 & \text{if } u > 0 \\ -1 & \text{otherwise} \end{cases} , \tag{1}$$

and then taking f_h to be the desired ranking function.[1] However, despite the apparently close relation between classification and ranking, on formalizing the above intuitions about evaluation criteria for classification and ranking functions, it turns out that a good classification function may not always translate into a good ranking function.

1. In Herbrich et al. (2000) the problem of learning a ranking function is also reduced to a classification problem, but on *pairs* of instances.

1.1 Evaluation of (Binary) Classification Functions

In classification, one generally assumes that examples (both training examples and future, unseen examples) are drawn randomly and independently according to some (unknown) underlying distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$. The mathematical quantity typically used to evaluate a classification function $h : \mathcal{X} \rightarrow \mathcal{Y}$ is then the *expected error rate* (or simply *error rate*) of h, denoted by $L(h)$ and defined as

$$L(h) \quad = \quad \mathbf{E}_{XY \sim \mathcal{D}} \left\{ \mathbf{I}_{\{h(X) \neq Y\}} \right\} , \tag{2}$$

where $\mathbf{I}_{\{\cdot\}}$ denotes the indicator variable whose value is one if its argument is true and zero otherwise. The error rate $L(h)$ is simply the probability that an example drawn randomly from $\mathcal{X} \times \mathcal{Y}$ (according to $\mathcal{D}$) will be misclassified by h; the quantity $(1 - L(h))$ thus measures our intuitive notion of 'how often instances are classified correctly by h'. In practice, since the distribution $\mathcal{D}$ is not known, the true error rate of a classification function cannot be computed exactly. Instead, the error rate must be estimated using a finite data sample. A widely used estimate is the *empirical error rate*: given a finite sequence of labeled examples $T = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)) \in (\mathcal{X} \times \mathcal{Y})^N$, the empirical error rate of a classification function h with respect to T, which we denote by $\hat{L}(h;T)$, is given by

$$\hat{L}(h;T) \quad = \quad \frac{1}{N} \sum_{i=1}^{N} \mathbf{I}_{\{h(\mathbf{x}_i) \neq y_i\}} . \tag{3}$$

When the examples in T are drawn randomly and independently from $\mathcal{X} \times \mathcal{Y}$ according to $\mathcal{D}$, the sequence T constitutes a random sample. Much work in learning theory research has concentrated on developing bounds on the probability that an error estimate obtained from such a random sample will have a large deviation from the true error rate. While the true error rate of a classification function may not be exactly computable, such generalization bounds allow us to compute confidence intervals within which the true value of the error rate is likely to be contained with high probability.

1.2 Evaluation of (Bipartite) Ranking Functions

Evaluating a ranking function has proved to be somewhat more difficult. One empirical quantity that has been used for this purpose is the average precision, which relates to recall-precision curves. The average precision is often used in applications that contain very few positive examples, such as information retrieval. Another empirical quantity that has recently gained some attention as being well-suited for evaluating ranking functions relates to receiver operating characteristic (ROC) curves. ROC curves were originally developed in signal detection theory for analysis of radar images (Egan, 1975), and have been used extensively in various fields such as medical decision-making. Given a ranking function $f : \mathcal{X} \rightarrow \mathbb{R}$ and a finite data sequence $T = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)) \in (\mathcal{X} \times \mathcal{Y})^N$, the ROC curve of f with respect to T is obtained as follows. First, a set of $N+1$ classification functions $h_i : \mathcal{X} \rightarrow \mathcal{Y}$, where $0 \leq i \leq N$, is constructed from f:

$$h_i(\mathbf{x}) \quad = \quad \theta(f(\mathbf{x}) - b_i) ,$$

where $\theta(\cdot)$ is as defined by Eq. (1) and

$$b_i \quad = \quad \begin{cases} f(\mathbf{x}_i) & \text{if } 1 \leq i \leq N \\ \left(\min_{1 \leq j \leq N} f(\mathbf{x}_j) \right) - 1 & \text{if } i = 0 . \end{cases}$$

The classification function h_0 classifies all instances in T as positive, while for $1 \leq i \leq N$, h_i classifies all instances ranked higher than $\mathbf{x}_i$ as positive, and all others (including $\mathbf{x}_i$) as negative. Next, for each classification function h_i, one computes the (empirical) true positive and false positive rates on T, denoted by tpr_i and fpr_i respectively:

$$tpr_i = \frac{\text{number of positive examples in } T \text{ classified correctly by } h_i}{\text{total number of positive examples in } T},$$

$$fpr_i = \frac{\text{number of negative examples in } T \text{ misclassified as positive by } h_i}{\text{total number of negative examples in } T}.$$

Finally, the points (fpr_i, tpr_i) are plotted on a graph with the false positive rate on the x-axis and the true positive rate on the y-axis; the ROC curve is then obtained by connecting these points such that the resulting curve is monotonically increasing. It is the *area under the ROC curve* (AUC) that has been used as an indicator of the quality of the ranking function f (Cortes and Mohri, 2004; Rosset, 2004). An AUC value of one corresponds to a perfect ranking on the given data sequence (*i.e.*, all positive instances in T are ranked higher than all negative instances); a value of zero corresponds to the opposite scenario (*i.e.*, all negative instances in T are ranked higher than all positive instances).

The AUC can in fact be expressed in a simpler form: if the sample T contains m positive and n negative examples, then it is not difficult to see that the AUC of f with respect to T, which we denote by $\hat{A}(f;T)$, is given simply by the following Wilcoxon-Mann-Whitney statistic (Cortes and Mohri, 2004):

$$\hat{A}(f;T) = \frac{1}{mn} \sum_{\{i:y_i=+1\}} \sum_{\{j:y_j=-1\}} \mathbf{I}_{\{f(\mathbf{x}_i)>f(\mathbf{x}_j)\}} + \frac{1}{2}\mathbf{I}_{\{f(\mathbf{x}_i)=f(\mathbf{x}_j)\}}. \quad (4)$$

In this simplified form, it becomes clear that the AUC of f with respect to T is simply the fraction of positive-negative pairs in T that are ranked correctly by f, assuming that ties are broken uniformly at random.[2]

There are two important observations to be made about the AUC defined above. The first is that the error rate of a classification function is not necessarily a good indicator of the AUC of a ranking function derived from it; different classification functions with the same error rate may produce ranking functions with very different AUC values. For example, consider two classification functions h_1, h_2 given by $h_i(\mathbf{x}) = \theta(f_i(\mathbf{x})), i = 1,2$, where the values assigned by f_1, f_2 to the instances in a sample $T \in (\mathcal{X} \times \mathcal{Y})^8$ are as shown in Table 1. Clearly, $\hat{L}(h_1;T) = \hat{L}(h_2;T) = 2/8$, but $\hat{A}(f_1;T) = 12/16$ while $\hat{A}(f_2;T) = 8/16$. The exact relationship between the (empirical) error rate of a classification function h of the form $h(\mathbf{x}) = \theta(f_h(\mathbf{x}))$ and the AUC value of the corresponding ranking function f_h with respect to a given data sequence was studied in detail by Cortes and Mohri (2004). In particular, they showed that when the number of positive examples m in the given data sequence is equal to the number of negative examples n, the average AUC value over all possible rankings corresponding to classification functions with a fixed (empirical) error rate ℓ is given by $(1-\ell)$, but the standard deviation among the AUC values can be large for large ℓ. As the proportion of positive instances $m/(m+n)$ departs from $1/2$, the average AUC value corresponding to an error rate ℓ departs from $(1-\ell)$, and the standard deviation increases further. The AUC is thus a different term than the error rate, and therefore requires separate analysis.

2. In (Cortes and Mohri, 2004), a slightly simpler form of the Wilcoxon-Mann-Whitney statistic is used, which does not account for ties.

$\mathbf{x}_i$	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$	$\mathbf{x}_7$	$\mathbf{x}_8$
y_i	-1	-1	-1	-1	+1	+1	+1	+1
$f_1(\mathbf{x}_i)$	-2	-1	3	4	1	2	5	6
$f_2(\mathbf{x}_i)$	-2	-1	5	6	1	2	3	4

Table 1: Values assigned by two functions f_1, f_2 to eight instances in a hypothetical example. The corresponding classification functions have the same (empirical) error rate, but the AUC values of the ranking functions are different. See text for details.

The second important observation about the AUC is that, as defined above, it is an empirical quantity that evaluates a ranking function with respect to a particular data sequence. What does the empirical AUC tell us about the expected performance of a ranking function on future examples? This is the question we address in this paper. The question has two parts, both of which are important for machine learning practice. First, what can be said about the expected performance of a ranking function based on its empirical AUC on an independent test sequence? Second, what can be said about the expected performance of a learned ranking function based on its empirical AUC on the training sequence from which it is learned? The first part of the question concerns the large deviation behaviour of the AUC; the second part concerns its uniform convergence behaviour. Both are addressed in this paper.

We start by defining the expected ranking accuracy of a ranking function (analogous to the expected error rate of a classification function) in Section 2. Section 3 contains our large deviation result, which serves to bound the expected accuracy of a ranking function in terms of its empirical AUC on an independent test sequence. Our conceptual approach in deriving the large deviation result for the AUC is similar to that of (Hill et al., 2002), in which large deviation properties of the average precision were considered. Section 4 contains our uniform convergence result, which serves to bound the expected accuracy of a learned ranking function in terms of its empirical AUC on a training sequence. Our uniform convergence bound is expressed in terms of a new set of combinatorial parameters that we term the bipartite rank-shatter coefficients; these play the same role in our result as do the standard shatter coefficients (also known as the growth function) in uniform convergence results for the classification error rate. A comparison of our result with a recent uniform convergence result derived by Freund et al. (2003) for a quantity closely related to the AUC shows that the bound provided by our result can be considerably tighter. We conclude with a summary and some open questions in Section 5.

2. Expected Ranking Accuracy

We begin by introducing some additional notation. As in classification, we shall assume that all examples are drawn randomly and independently according to some (unknown) underlying distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$. The notation $\mathcal{D}_{+1}$ and $\mathcal{D}_{-1}$ will be used to denote the class-conditional distributions $\mathcal{D}_{X|Y=+1}$ and $\mathcal{D}_{X|Y=-1}$, respectively. We use an underline to denote a sequence, *e.g.*, $\underline{y} \in \mathcal{Y}^N$ to denote a sequence of elements in $\mathcal{Y}$. We shall find it convenient to decompose a data sequence $T = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)) \in (\mathcal{X} \times \mathcal{Y})^N$ into two components, $T_X = (\mathbf{x}_1, \ldots, \mathbf{x}_N) \in \mathcal{X}^N$ and $T_Y = (y_1, \ldots, y_N) \in \mathcal{Y}^N$. Several of our results will involve the conditional distribution $\mathcal{D}_{T_X|T_Y=\underline{y}}$ for some label sequence $\underline{y} = (y_1, \ldots, y_N) \in \mathcal{Y}^N$; this distribution is simply $\mathcal{D}_{y_1} \times \ldots \times \mathcal{D}_{y_N}$.[3] If the distribution is clear from the context it will be dropped in the notation of expectations and probabilities, *e.g.*, $\mathbf{E}_{XY} \equiv \mathbf{E}_{XY\sim\mathcal{D}}$. As a final note of convention, we use $T \in (\mathcal{X} \times \mathcal{Y})^N$ to denote a general data sequence (*e.g.*, an independent test sequence), and $S \in (\mathcal{X} \times \mathcal{Y})^M$ to denote a training sequence.

We define below a quantity that we term the expected ranking accuracy; the purpose of this quantity will be to serve as an evaluation criterion for ranking functions (analogous to the use of the expected error rate as an evaluation criterion for classification functions).

Definition 1 (Expected ranking accuracy) *Let $f : \mathcal{X} \to \mathbb{R}$ be a ranking function on $\mathcal{X}$. Define the* expected ranking accuracy *(or simply* ranking accuracy*) of f, denoted by $A(f)$, as follows:*

$$A(f) \quad = \quad \mathbf{E}_{X\sim\mathcal{D}_{+1}, X'\sim\mathcal{D}_{-1}} \left\{ \mathbf{I}_{\{f(X)>f(X')\}} + \frac{1}{2}\mathbf{I}_{\{f(X)=f(X')\}} \right\} . \tag{5}$$

The ranking accuracy $A(f)$ defined above is simply the probability that an instance drawn randomly according to $\mathcal{D}_{+1}$ will be ranked higher by f than an instance drawn randomly according to $\mathcal{D}_{-1}$, assuming that ties are broken uniformly at random; the quantity $A(f)$ thus measures our intuitive notion of 'how often instances labeled as positive are ranked higher by f than instances labeled as negative'. As in the case of classification, the true ranking accuracy depends on the underlying distribution of the data and cannot be observed directly. Our goal shall be to derive generalization bounds that allow the true accuracy of a ranking function to be estimated from its empirical AUC with respect to a finite data sample. The following simple lemma shows that this makes sense, for given a fixed label sequence, the empirical AUC of a ranking function f is an unbiased estimator of the expected ranking accuracy of f:

Lemma 2 *Let $f : \mathcal{X} \to \mathbb{R}$ be a ranking function on $\mathcal{X}$, and let $\underline{y} = (y_1, \ldots, y_N) \in \mathcal{Y}^N$ be a finite label sequence. Then*

$$\mathbf{E}_{T_X|T_Y=\underline{y}} \left\{ \hat{A}(f;T) \right\} \quad = \quad A(f) .$$

Proof Let m be the number of positive labels in $\underline{y}$, and n the number of negative labels in $\underline{y}$. Then from the definition of empirical AUC (Eq. (4)) and linearity of expectation, we have

$$\begin{aligned} \mathbf{E}_{T_X|T_Y=\underline{y}} \left\{ \hat{A}(f;T) \right\} &= \frac{1}{mn} \sum_{\{i:y_i=+1\}} \sum_{\{j:y_j=-1\}} \mathbf{E}_{X_i\sim\mathcal{D}_{+1}, X_j\sim\mathcal{D}_{-1}} \left\{ \mathbf{I}_{\{f(X_i)>f(X_j)\}} + \frac{1}{2}\mathbf{I}_{\{f(X_i)=f(X_j)\}} \right\} \\ &= \frac{1}{mn} \sum_{\{i:y_i=+1\}} \sum_{\{j:y_j=-1\}} A(f) \\ &= A(f) . \end{aligned}$$

■

3. Note that, since the AUC of a ranking function f with respect to a data sequence $T \in (\mathcal{X} \times \mathcal{Y})^N$ is independent of the actual ordering of examples in the sequence, our results involving the conditional distribution $\mathcal{D}_{T_X|T_Y=\underline{y}}$ for some label sequence $\underline{y} = (y_1, \ldots, y_N) \in \mathcal{Y}^N$ depend only on the number m of positive labels in $\underline{y}$ and the number n of negative labels in $\underline{y}$. We choose to state our results in terms of the distribution $\mathcal{D}_{T_X|T_Y=\underline{y}} \equiv \mathcal{D}_{y_1} \times \ldots \times \mathcal{D}_{y_N}$ only because this is more general than stating them in terms of $\mathcal{D}_{+1}^m \times \mathcal{D}_{-1}^n$.

We are now ready to present the main results of this paper, namely, a large deviation bound in Section 3 and a uniform convergence bound in Section 4. We note that our results are all distribution-free, in the sense that they hold for any distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$.

3. Large Deviation Bound for the AUC

In this section we are interested in bounding the probability that the empirical AUC of a ranking function f with respect to a (random) test sequence T will have a large deviation from its expected ranking accuracy. In other words, we are interested in bounding probabilities of the form

$$\mathbf{P}\left\{\left|\hat{A}(f;T) - A(f)\right| \geq \varepsilon\right\}$$

for given $\varepsilon > 0$. Our main tool in deriving such a large deviation bound will be the following powerful concentration inequality of McDiarmid (1989), which bounds the deviation of any function of a sample for which a single change in the sample has limited effect:

Theorem 3 (McDiarmid, 1989) *Let $X_1, \ldots, X_N$ be independent random variables with X_k taking values in a set A_k for each k. Let $\phi : (A_1 \times \cdots \times A_N) \to \mathbb{R}$ be such that*

$$\sup_{x_i \in A_i, x'_k \in A_k} \left|\phi(x_1, \ldots, x_N) - \phi(x_1, \ldots, x_{k-1}, x'_k, x_{k+1}, \ldots, x_N)\right| \quad \leq \quad c_k\,.$$

Then for any $\varepsilon > 0$,

$$\mathbf{P}\left\{\left|\phi(X_1, \ldots, X_N) - \mathbf{E}\{\phi(X_1, \ldots, X_N)\}\right| \geq \varepsilon\right\} \quad \leq \quad 2e^{-2\varepsilon^2 / \sum_{k=1}^N c_k^2}\,.$$

Note that when $X_1, \ldots, X_N$ are independent bounded random variables with $X_k \in [a_k, b_k]$ with probability one, and $\phi(X_1, \ldots, X_N) = \sum_{k=1}^N X_k$, McDiarmid's inequality (with $c_k = b_k - a_k$) reduces to Hoeffding's inequality. Next we define the following quantity which appears in several of our results:

Definition 4 (Positive skew) *Let $\underline{y} = (y_1, \ldots, y_N) \in \mathcal{Y}^N$ be a finite label sequence of length $N \in \mathbb{N}$. Define the* positive skew *of $\underline{y}$, denoted by $\rho(\underline{y})$, as follows:*

$$\rho(\underline{y}) \quad = \quad \frac{1}{N} \sum_{\{i : y_i = +1\}} 1\,. \qquad (6)$$

The following is the main result of this section:

Theorem 5 *Let $f : \mathcal{X} \to \mathbb{R}$ be a fixed ranking function on $\mathcal{X}$ and let $\underline{y} = (y_1, \ldots, y_N) \in \mathcal{Y}^N$ be any label sequence of length $N \in \mathbb{N}$. Let m be the number of positive labels in $\underline{y}$, and $n = N - m$ the number of negative labels in $\underline{y}$. Then for any $\varepsilon > 0$,*

$$\begin{aligned} \mathbf{P}_{T_X | T_Y = \underline{y}}\left\{\left|\hat{A}(f;T) - A(f)\right| \geq \varepsilon\right\} \quad &\leq \quad 2e^{-2mn\varepsilon^2/(m+n)} \\ &= \quad 2e^{-2\rho(\underline{y})(1-\rho(\underline{y}))N\varepsilon^2}\,. \end{aligned}$$

Proof Given the label sequence $\underline{y}$, the random variables $X_1, \ldots, X_N$ are independent, with each X_k taking values in $\mathcal{X}$. Now, define $\phi : \mathcal{X}^N \to \mathbb{R}$ as follows:

$$\phi(\mathbf{x}_1, \ldots, \mathbf{x}_N) = \hat{A}(f; ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N))) .$$

Then, for each k such that $y_k = +1$, we have the following for all $\mathbf{x}_i, \mathbf{x}'_k \in \mathcal{X}$:

$$\begin{aligned} & \left|\phi(\mathbf{x}_1, \ldots, \mathbf{x}_N) - \phi(\mathbf{x}_1, \ldots, \mathbf{x}_{k-1}, \mathbf{x}'_k, \mathbf{x}_{k+1} \ldots, \mathbf{x}_N)\right| \\ &= \frac{1}{mn}\left| \sum_{\{j : y_j = -1\}} \left(\left(\mathbf{I}_{\{f(\mathbf{x}_k) > f(\mathbf{x}_j)\}} + \frac{1}{2}\mathbf{I}_{\{f(\mathbf{x}_k) = f(\mathbf{x}_j)\}} \right) - \right.\right. \\ &\qquad \left.\left. \left(\mathbf{I}_{\{f(\mathbf{x}'_k) > f(\mathbf{x}_j)\}} + \frac{1}{2}\mathbf{I}_{\{f(\mathbf{x}'_k) = f(\mathbf{x}_j)\}} \right) \right) \right| \\ &\le \frac{1}{mn} n \\ &= \frac{1}{m} . \end{aligned}$$

Similarly, for each k such that $y_k = -1$, one can show for all $\mathbf{x}_i, \mathbf{x}'_k \in \mathcal{X}$:

$$\left|\phi(\mathbf{x}_1, \ldots, \mathbf{x}_N) - \phi(\mathbf{x}_1, \ldots, \mathbf{x}_{k-1}, \mathbf{x}'_k, \mathbf{x}_{k+1} \ldots, \mathbf{x}_N)\right| \le \frac{1}{n} .$$

Thus, taking $c_k = 1/m$ for k such that $y_k = +1$ and $c_k = 1/n$ for k such that $y_k = -1$, and applying McDiarmid's theorem, we get for any $\varepsilon > 0$,

$$\begin{aligned} \mathbf{P}_{T_X | T_Y = \underline{y}} \left\{ \left| \hat{A}(f;T) - \mathbf{E}_{T_X | T_Y = \underline{y}} \left\{ \hat{A}(f;T) \right\} \right| \ge \varepsilon \right\} &\le 2e^{-2\varepsilon^2 / (m(\frac{1}{m})^2 + n(\frac{1}{n})^2)} \\ &= 2e^{-2mn\varepsilon^2/(m+n)} . \end{aligned}$$

The result follows from Lemma 2. ■

We note that the result of Theorem 5 can be strengthened so that the conditioning is only on the numbers m and n of positive and negative labels, and not on the specific label vector $\underline{y}$. From Theorem 5, we can derive a confidence interval interpretation of the bound that gives, for any $0 < \delta \le 1$, a confidence interval based on the empirical AUC of a ranking function (on a random test sequence) which is likely to contain the true ranking accuracy with probability at least $1 - \delta$. More specifically, we have:

Corollary 6 *Let $f : \mathcal{X} \to \mathbb{R}$ be a fixed ranking function on $\mathcal{X}$ and let $\underline{y} = (y_1, \ldots, y_N) \in \mathcal{Y}^N$ be any label sequence of length $N \in \mathbb{N}$. Then for any $0 < \delta \le 1$,*

$$\mathbf{P}_{T_X | T_Y = \underline{y}} \left\{ \left| \hat{A}(f;T) - A(f) \right| \ge \sqrt{\frac{\ln\left(\frac{2}{\delta}\right)}{2\rho(\underline{y})(1 - \rho(\underline{y}))N}} \right\} \le \delta .$$

Proof This follows directly from Theorem 5 by setting $2e^{-2\rho(\underline{y})(1-\rho(\underline{y}))N\varepsilon^2} = \delta$ and solving for ε. ■

We note that a different approach for deriving confidence intervals for the AUC has recently been taken by Cortes and Mohri (2005); in particular, their confidence intervals for the AUC are constructed from confidence intervals for the classification error rate.

Theorem 5 also allows us to obtain an expression for a test sample size that is sufficient to obtain, for given $0 < \varepsilon, \delta \leq 1$, an ε-accurate estimate of the ranking accuracy with δ-confidence:

Corollary 7 *Let $f : \mathcal{X} \to \mathbb{R}$ be a fixed ranking function on $\mathcal{X}$ and let $0 < \varepsilon, \delta \leq 1$. Let $\underline{y} = (y_1, \ldots, y_N) \in \mathcal{Y}^N$ be any label sequence of length $N \in \mathbb{N}$. If*

$$N \quad \geq \quad \frac{\ln\left(\frac{2}{\delta}\right)}{2\rho(\underline{y})(1-\rho(\underline{y}))\varepsilon^2},$$

then

$$\mathbf{P}_{T_X|T_Y=\underline{y}}\left\{\left|\hat{A}(f;T) - A(f)\right| \geq \varepsilon\right\} \quad \leq \quad \delta.$$

Proof This follows directly from Theorem 5 by setting $2e^{-2\rho(\underline{y})(1-\rho(\underline{y}))N\varepsilon^2} \leq \delta$ and solving for N. ■

The confidence interval of Corollary 6 can in fact be generalized to remove the conditioning on the label vector completely:

Theorem 8 *Let $f : \mathcal{X} \to \mathbb{R}$ be a fixed ranking function on $\mathcal{X}$ and let $N \in \mathbb{N}$. Then for any $0 < \delta \leq 1$,*

$$\mathbf{P}_{T \sim \mathcal{D}^N}\left\{\left|\hat{A}(f;T) - A(f)\right| \geq \sqrt{\frac{\ln\left(\frac{2}{\delta}\right)}{2\rho(T_Y)(1-\rho(T_Y))N}}\right\} \quad \leq \quad \delta.$$

Proof For $T \in (\mathcal{X} \times \mathcal{Y})^N$ and $0 < \delta \leq 1$, define the proposition

$$\Phi(T,\delta) \quad \equiv \quad \left\{\left|\hat{A}(f;T) - A(f)\right| \geq \sqrt{\frac{\ln\left(\frac{2}{\delta}\right)}{2\rho(T_Y)(1-\rho(T_Y))N}}\right\}.$$

Then for any $0 < \delta \leq 1$, we have

$$\begin{aligned}
\mathbf{P}_T\{\Phi(T,\delta)\} &= \mathbf{E}_T\left\{\mathbf{I}_{\Phi(T,\delta)}\right\} \\
&= \mathbf{E}_{T_Y}\left\{\mathbf{E}_{T_X|T_Y=\underline{y}}\left\{\mathbf{I}_{\Phi(T,\delta)}\right\}\right\} \\
&= \mathbf{E}_{T_Y}\left\{\mathbf{P}_{T_X|T_Y=\underline{y}}\{\Phi(T,\delta)\}\right\} \\
&\leq \mathbf{E}_{T_Y}\{\delta\} \qquad \text{(by Corollary 6)} \\
&= \delta.
\end{aligned}$$

■

Note that the above 'trick' works only once we have gone to a confidence interval; an attempt to generalize the bound of Theorem 5 in a similar way gives an expression in which the final expectation is not easy to evaluate. Interestingly, the above proof does not even require a factorized distribution $\mathcal{D}_{T_Y}$ since it is built on a result for any fixed label sequence $\underline{y}$. We note that the above technique could also be applied to generalize the results of Hill et al. (2002) in a similar manner.

3.1 Comparison with Bounds from Statistical Literature

The AUC, in the form of the Wilcoxon-Mann-Whitney statistic, has been studied extensively in the statistical literature. In particular, Lehmann (1975) derives an exact expression for the variance of the Wilcoxon-Mann-Whitney statistic which can be used to obtain large deviation bounds for the AUC. Below we compare the large deviation bound we have derived above with these bounds obtainable from the statistical literature. We note that the expression derived by Lehmann (1975) is for a simpler form of the Wilcoxon-Mann-Whitney statistic that does not account for ties; therefore, in this section we assume the AUC and the expected ranking accuracy are defined without the terms that account for ties (the large deviation result we have derived above applies also in this setting).

Let $f : \mathcal{X} \to \mathbb{R}$ be a fixed ranking function on $\mathcal{X}$ and let $\underline{y} = (y_1, \ldots, y_N) \in \mathcal{Y}^N$ be any label sequence of length $N \in \mathbb{N}$. Let m be the number of positive labels in $\underline{y}$, and $n = N - m$ the number of negative labels in $\underline{y}$. Then the variance of the AUC of f is given by the following expression (Lehmann, 1975):

$$\begin{aligned}\sigma_A^2 &= \mathbf{Var}_{T_X|T_Y=\underline{y}}\left\{\hat{A}(f;T)\right\} \\ &= \frac{A(f)(1-A(f)) + (m-1)(p_1 - A(f)^2) + (n-1)(p_2 - A(f)^2)}{mn}, \end{aligned} \tag{7}$$

where

$$p_1 = \mathbf{P}_{X_1^+, X_2^+ \sim \mathcal{D}_{+1}, X_1^- \sim \mathcal{D}_{-1}} \left\{ \left\{ f(X_1^+) > f(X_1^-) \right\} \cap \left\{ f(X_2^+) > f(X_1^-) \right\} \right\} \tag{8}$$

$$p_2 = \mathbf{P}_{X_1^+ \sim \mathcal{D}_{+1}, X_1^-, X_2^- \sim \mathcal{D}_{-1}} \left\{ \left\{ f(X_1^+) > f(X_1^-) \right\} \cap \left\{ f(X_1^+) > f(X_2^-) \right\} \right\}. \tag{9}$$

Next we recall the following classical inequality:

Theorem 9 (Chebyshev's inequality) *Let X be a random variable. Then for any $\varepsilon > 0$,*

$$\mathbf{P}\left\{|X - \mathbf{E}\{X\}| \geq \varepsilon\right\} \leq \frac{\mathbf{Var}\{X\}}{\varepsilon^2}.$$

The expression for the variance σ_A^2 of the AUC can be used with Chebyshev's inequality to give the following bound: for any $\varepsilon > 0$,

$$\mathbf{P}_{T_X|T_Y=\underline{y}}\left\{\left|\hat{A}(f;T) - A(f)\right| \geq \varepsilon\right\} \leq \frac{\sigma_A^2}{\varepsilon^2}. \tag{10}$$

This leads to the following confidence interval: for any $0 < \delta \leq 1$,

$$\mathbf{P}_{T_X|T_Y=\underline{y}}\left\{\left|\hat{A}(f;T) - A(f)\right| \geq \frac{\sigma_A}{\sqrt{\delta}}\right\} \leq \delta. \tag{11}$$

It has been established that the AUC follows an asymptotically normal distribution. Therefore, for large N, one can use a normal approximation to obtain a tighter bound:

$$\mathbf{P}_{T_X|T_Y=\underline{y}}\left\{\left|\hat{A}(f;T) - A(f)\right| \geq \varepsilon\right\} \leq 2(1 - \Phi(\varepsilon/\sigma_A)), \tag{12}$$

where $\Phi(\cdot)$ denotes the standard normal cumulative distribution function given by $\Phi(u) = \int_0^u e^{-z^2/2}\, dz/\sqrt{2\pi}$. The resulting confidence interval is given by

$$\mathbf{P}_{T_X|T_Y=\underline{y}}\left\{\left|\hat{A}(f;T) - A(f)\right| \geq \sigma_A \Phi^{-1}(1-\delta/2)\right\} \leq \delta. \tag{13}$$

The quantities p_1 and p_2 that appear in the expression for σ_A^2 in Eq. (7) depend on the underlying distributions $\mathcal{D}_{+1}$ and $\mathcal{D}_{-1}$; for example, Hanley and McNeil (1982) derive expressions for p_1 and p_2 in the case when the scores $f(X^+)$ assigned to positive instances X^+ and the scores $f(X^-)$ assigned to negative instances X^- both follow negative exponential distributions. Distribution-independent bounds can be obtained by using the fact that the variance σ_A^2 is at most (Cortes and Mohri, 2005; Dantzig, 1915; Birnbaum and Klose, 1957)

$$\sigma_{\max}^2 \;=\; \frac{A(f)(1-A(f))}{\min(m,n)} \;\leq\; \frac{1}{4\min(m,n)}\,. \tag{14}$$

A comparison of the resulting bounds with the large deviation bound we have derived above using McDiarmid's inequality is shown in Figure 1. The McDiarmid bound is tighter than the bound obtained using Chebyshev's inequality. It is looser than the bound obtained using the normal approximation; however, since the normal approximation is valid only for large N, for smaller values of N the McDiarmid bound is safer.

Of course, it should be noted that this comparison holds only in the distribution-free setting. In practice, depending on the underlying distribution, the actual variance of the AUC may be much smaller than $\sigma_{\max}^2$; indeed, in the best case, the variance could be as small as

$$\sigma_{\min}^2 \;=\; \frac{A(f)(1-A(f))}{mn} \;\leq\; \frac{1}{4mn}\,. \tag{15}$$

Therefore, one may be able to obtain tighter confidence intervals with Eqs. (11) and (13) by estimating the actual variance of the AUC. For example, one may attempt to estimate the quantities p_1, p_2 and $A(f)$ that appear in the expression in Eq. (7) directly from the data, or one may use resampling methods such as the bootstrap (Efron and Tibshirani, 1993), in which the variance is estimated from the sample variance observed over a number of bootstrap samples obtained from the data. The confidence intervals obtained using such estimates are only approximate (*i.e.*, the $1-\delta$ confidence is not guaranteed), but they can often be useful in practice.

3.2 Comparison with Large Deviation Bound for Classification Error Rate

Our use of McDiarmid's inequality in deriving the large deviation bound for the AUC of a ranking function is analogous to the use of Hoeffding's inequality in deriving a similar large deviation bound for the error rate of a classification function (see, for example, Devroye et al., 1996, Chapter 8). The need for the more general inequality of McDiarmid in our derivation arises from the fact that the empirical AUC, unlike the empirical error rate, cannot be expressed as a sum of independent random variables. In the notation of Section 1, the large deviation bound for the classification error rate obtained via Hoeffding's inequality states that for a fixed classification function $h : \mathcal{X} \rightarrow \mathcal{Y}$ and for any $N \in \mathbb{N}$ and any $\varepsilon > 0$,

$$\mathbf{P}_{T\sim\mathcal{D}^N}\left\{\left|\hat{L}(h;T) - L(h)\right| \geq \varepsilon\right\} \;\leq\; 2e^{-2N\varepsilon^2}\,. \tag{16}$$

Comparing Eq. (16) to the bound of Theorem 5, we see that the AUC bound differs from the error rate bound by a factor of $\rho(\underline{y})(1-\rho(\underline{y}))$ in the exponent. This difference translates into a $1/(\rho(\underline{y})(1-\rho(\underline{y})))$ factor difference in the resulting sample size bounds; in other words, for given $0 < \varepsilon, \delta \leq 1$, the test sample size sufficient to obtain an ε-accurate estimate of the expected accuracy of a ranking function with δ-confidence is $1/(\rho(\underline{y})(1-\rho(\underline{y})))$ times larger than the corresponding

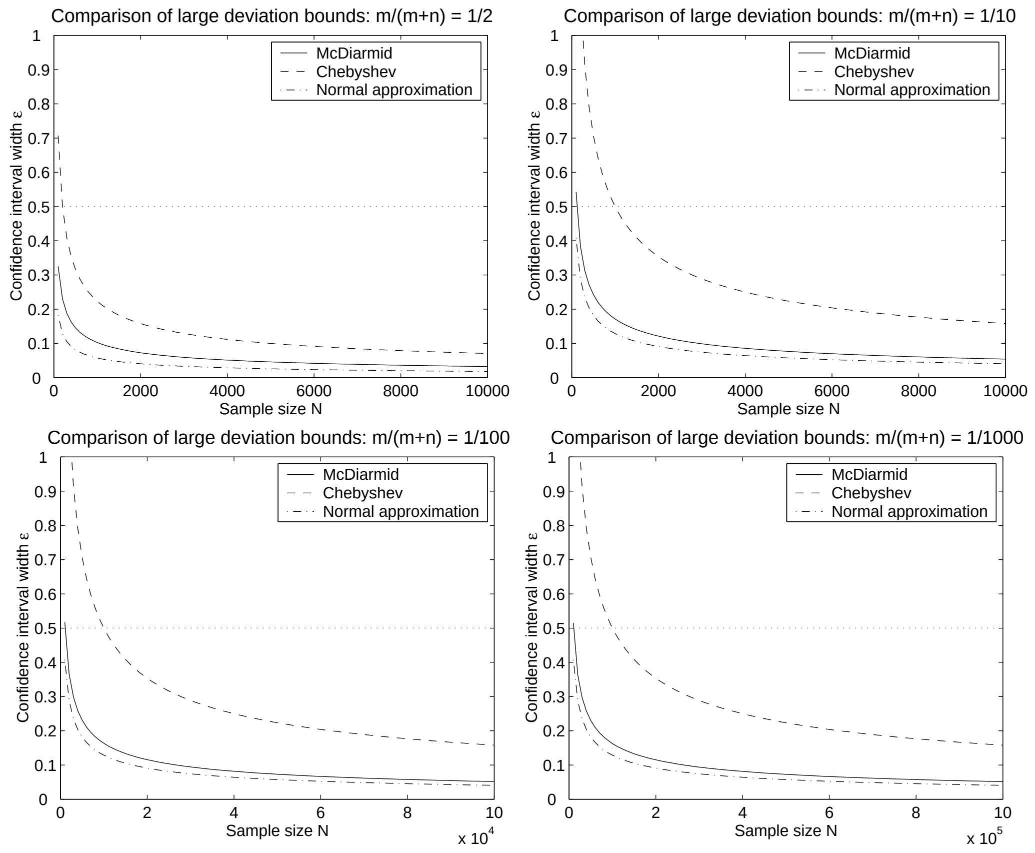


Figure 1: A comparison of our large deviation bound, derived using McDiarmid's inequality, with large deviation bounds obtainable from the statistical literature (see Section 3.1). The plots are for $\delta = 0.01$ and show how the confidence interval size ε given by the different bounds varies with the sample size $N = m + n$, for various values of $m/(m+n)$.

test sample size sufficient to obtain an ε-accurate estimate of the expected error rate of a classification function with the same confidence. For $\rho(\underline{y}) = 1/2$, this means a sample size larger by a factor of 4; as the positive skew $\rho(\underline{y})$ departs from $1/2$, the factor grows larger (see Figure 2).

Again, it should be noted that the above conclusion holds only in the distribution-free setting. Indeed, the variance σ_L^2 of the error rate (which follows a binomial distribution) is given by

$$\sigma_L^2 \;=\; \mathbf{Var}_{T\sim\mathcal{D}^N}\left\{\hat{L}(h;T)\right\} \;=\; \frac{L(h)(1-L(h))}{N} \;\leq\; \frac{1}{4N}\,. \tag{17}$$

Comparing to Eqs. (14) and (15), we see that although this is smaller than the worst-case variance of the AUC, in the best case, the variance of the AUC can be considerably smaller, leading to a tighter bound for the AUC and therefore a smaller sufficient test sample size.

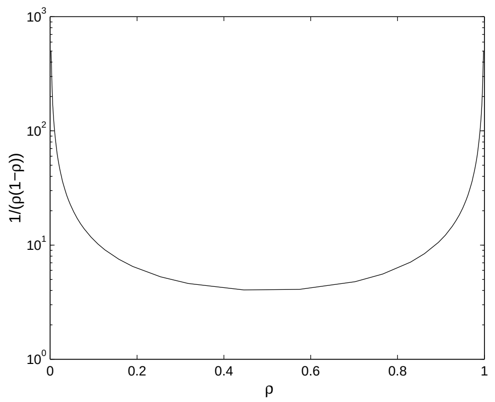


Figure 2: The test sample size bound for the AUC, for positive skew $\rho \equiv \rho(\underline{y})$ for some label sequence $\underline{y}$, is larger than the corresponding test sample size bound for the error rate by a factor of $1/(\rho(1-\rho))$ (see text for discussion).

3.3 Bound for Learned Ranking Functions Chosen from Finite Function Classes

The large deviation result of Theorem 5 bounds the expected accuracy of a ranking function in terms of its empirical AUC on an independent test sequence. A simple application of the union bound allows the result to be extended to bound the expected accuracy of a learned ranking function in terms of its empirical AUC on the training sequence from which it is learned, in the case when the learned ranking function is chosen from a finite function class. More specifically, we have:

Theorem 10 *Let $\mathcal{F}$ be a finite class of real-valued functions on $\mathcal{X}$ and let $f_S \in \mathcal{F}$ denote the ranking function chosen by a learning algorithm based on the training sequence S. Let $\underline{y} = (y_1, \ldots, y_M) \in \mathcal{Y}^M$ be any label sequence of length $M \in \mathbb{N}$. Then for any $\epsilon > 0$,*

$$\mathbf{P}_{S_X|S_Y=\underline{y}}\left\{\left|\hat{A}(f_S;S) - A(f_S)\right| \geq \epsilon\right\} \quad \leq \quad 2|\mathcal{F}|e^{-2\rho(\underline{y})(1-\rho(\underline{y}))M\epsilon^2}\,.$$

Proof For any $\epsilon > 0$, we have

$$\begin{aligned}
&\mathbf{P}_{S_X|S_Y=\underline{y}}\left\{\left|\hat{A}(f_S;S) - A(f_S)\right| \geq \epsilon\right\} \\
&\leq \quad \mathbf{P}_{S_X|S_Y=\underline{y}}\left\{\max_{f\in\mathcal{F}}\left|\hat{A}(f;S) - A(f)\right| \geq \epsilon\right\} \\
&\leq \quad \sum_{f\in\mathcal{F}} \mathbf{P}_{S_X|S_Y=\underline{y}}\left\{\left|\hat{A}(f;S) - A(f)\right| \geq \epsilon\right\} \qquad \text{(by the union bound)} \\
&\leq \quad 2|\mathcal{F}|e^{-2\rho(\underline{y})(1-\rho(\underline{y}))M\epsilon^2} \qquad \text{(by Theorem 5)}\,.
\end{aligned}$$

■

As before, we can derive from Theorem 10 expressions for confidence intervals and sufficient training sample size; we give these below without proof:

Corollary 11 *Let $\mathcal{F}$ be a finite class of real-valued functions on $\mathcal{X}$ and let $f_S \in \mathcal{F}$ denote the ranking function chosen by a learning algorithm based on the training sequence S. Let $\underline{y} = (y_1, \ldots, y_M) \in \mathcal{Y}^M$ be any label sequence of length $M \in \mathbb{N}$. Then for any $0 < \delta \leq 1$,*

$$\mathbf{P}_{S_X|S_Y=\underline{y}} \left\{ \left| \hat{A}(f_S;S) - A(f_S) \right| \geq \sqrt{\frac{\ln|\mathcal{F}| + \ln\left(\frac{2}{\delta}\right)}{2\rho(\underline{y})(1-\rho(\underline{y}))M}} \right\} \quad \leq \quad \delta.$$

Corollary 12 *Let $\mathcal{F}$ be a finite class of real-valued functions on $\mathcal{X}$ and let $f_S \in \mathcal{F}$ denote the ranking function chosen by a learning algorithm based on the training sequence S. Let $\underline{y} = (y_1, \ldots, y_M) \in \mathcal{Y}^M$ be any label sequence of length $M \in \mathbb{N}$. Then for any $0 < \varepsilon, \delta \leq 1$, if*

$$M \quad \geq \quad \frac{1}{2\rho(\underline{y})(1-\rho(\underline{y}))\varepsilon^2} \left(\ln|\mathcal{F}| + \ln\left(\frac{2}{\delta}\right) \right),$$

then

$$\mathbf{P}_{S_X|S_Y=\underline{y}} \left\{ \left| \hat{A}(f_S;S) - A(f_S) \right| \geq \varepsilon \right\} \quad \leq \quad \delta.$$

Theorem 13 *Let $\mathcal{F}$ be a finite class of real-valued functions on $\mathcal{X}$ and let $f_S \in \mathcal{F}$ denote the ranking function chosen by a learning algorithm based on the training sequence S. Let $M \in \mathbb{N}$. Then for any $0 < \delta \leq 1$,*

$$\mathbf{P}_{S \sim \mathcal{D}^M} \left\{ \left| \hat{A}(f_S;S) - A(f_S) \right| \geq \sqrt{\frac{\ln|\mathcal{F}| + \ln\left(\frac{2}{\delta}\right)}{2\rho(S_Y)(1-\rho(S_Y))M}} \right\} \quad \leq \quad \delta.$$

The above results apply only to ranking functions learned from finite function classes. The general case, when the learned ranking function may be chosen from a possibly infinite function class, is the subject of the next section.

4. Uniform Convergence Bound for the AUC

In this section we are interested in bounding the probability that the empirical AUC of a learned ranking function f_S with respect to the (random) training sequence S from which it is learned will have a large deviation from its expected ranking accuracy, when the function f_S is chosen from a possibly infinite function class $\mathcal{F}$. The standard approach for obtaining such bounds is via uniform convergence results. In particular, we have for any $\varepsilon > 0$,

$$\mathbf{P} \left\{ \left| \hat{A}(f_S;S) - A(f_S) \right| \geq \varepsilon \right\} \quad \leq \quad \mathbf{P} \left\{ \sup_{f \in \mathcal{F}} \left| \hat{A}(f;S) - A(f) \right| \geq \varepsilon \right\}.$$

Therefore, to bound probabilities of the form on the left hand side above, it is sufficient to derive a uniform convergence result that bounds probabilities of the form on the right hand side. Our uniform convergence result for the AUC is expressed in terms of a new set of combinatorial parameters, termed the *bipartite rank-shatter coefficients*, that we define below.

$\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$	$\begin{bmatrix}\frac{1}{2} & 0\\ 0 & 1\end{bmatrix}$	$\begin{bmatrix}1 & \frac{1}{2}\\ 0 & 1\end{bmatrix}$	$\begin{bmatrix}1 & 0\\ 0 & \frac{1}{2}\end{bmatrix}$	$\begin{bmatrix}1 & 0\\ \frac{1}{2} & 1\end{bmatrix}$	$\begin{bmatrix}\frac{1}{2} & \frac{1}{2}\\ 0 & 1\end{bmatrix}$	$\begin{bmatrix}\frac{1}{2} & 0\\ 0 & \frac{1}{2}\end{bmatrix}$	$\begin{bmatrix}\frac{1}{2} & 0\\ \frac{1}{2} & 1\end{bmatrix}$	$\begin{bmatrix}1 & \frac{1}{2}\\ 0 & \frac{1}{2}\end{bmatrix}$	$\begin{bmatrix}1 & \frac{1}{2}\\ \frac{1}{2} & 1\end{bmatrix}$	$\begin{bmatrix}1 & 0\\ \frac{1}{2} & \frac{1}{2}\end{bmatrix}$	$\begin{bmatrix}1 & \frac{1}{2}\\ \frac{1}{2} & \frac{1}{2}\end{bmatrix}$	$\begin{bmatrix}\frac{1}{2} & 0\\ \frac{1}{2} & \frac{1}{2}\end{bmatrix}$	$\begin{bmatrix}\frac{1}{2} & \frac{1}{2}\\ \frac{1}{2} & 1\end{bmatrix}$	$\begin{bmatrix}\frac{1}{2} & \frac{1}{2}\\ 0 & \frac{1}{2}\end{bmatrix}$
$\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$	$\begin{bmatrix}\frac{1}{2} & 1\\ 1 & 0\end{bmatrix}$	$\begin{bmatrix}0 & \frac{1}{2}\\ 1 & 0\end{bmatrix}$	$\begin{bmatrix}0 & 1\\ 1 & \frac{1}{2}\end{bmatrix}$	$\begin{bmatrix}0 & 1\\ \frac{1}{2} & 0\end{bmatrix}$	$\begin{bmatrix}\frac{1}{2} & \frac{1}{2}\\ 1 & 0\end{bmatrix}$	$\begin{bmatrix}\frac{1}{2} & 1\\ 1 & \frac{1}{2}\end{bmatrix}$	$\begin{bmatrix}\frac{1}{2} & 1\\ \frac{1}{2} & 0\end{bmatrix}$	$\begin{bmatrix}0 & \frac{1}{2}\\ 1 & \frac{1}{2}\end{bmatrix}$	$\begin{bmatrix}0 & \frac{1}{2}\\ \frac{1}{2} & 0\end{bmatrix}$	$\begin{bmatrix}0 & 1\\ \frac{1}{2} & \frac{1}{2}\end{bmatrix}$	$\begin{bmatrix}0 & \frac{1}{2}\\ \frac{1}{2} & \frac{1}{2}\end{bmatrix}$	$\begin{bmatrix}\frac{1}{2} & 1\\ \frac{1}{2} & \frac{1}{2}\end{bmatrix}$	$\begin{bmatrix}\frac{1}{2} & \frac{1}{2}\\ \frac{1}{2} & 0\end{bmatrix}$	$\begin{bmatrix}\frac{1}{2} & \frac{1}{2}\\ 1 & \frac{1}{2}\end{bmatrix}$

Table 2: Sub-matrices that cannot appear in a bipartite rank matrix.

4.1 Bipartite Rank-Shatter Coefficients

We define first the notion of a bipartite rank matrix; this is used in our definition of bipartite rank-shatter coefficients.

Definition 14 (Bipartite rank matrix) *Let $f : \mathcal{X} \to \mathbb{R}$ be a ranking function on $\mathcal{X}$, let $m,n \in \mathbb{N}$, and let $\underline{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_m) \in \mathcal{X}^m$, $\underline{\mathbf{x}}' = (\mathbf{x}'_1, \ldots, \mathbf{x}'_n) \in \mathcal{X}^n$. Define the* bipartite rank matrix *of f with respect to $\underline{\mathbf{x}}, \underline{\mathbf{x}}'$, denoted by $\mathbf{B}_f(\underline{\mathbf{x}}, \underline{\mathbf{x}}')$, to be the matrix in $\{0, \frac{1}{2}, 1\}^{m \times n}$ whose (i,j)-th element is given by*

$$\left[\mathbf{B}_f(\underline{\mathbf{x}}, \underline{\mathbf{x}}')\right]_{ij} \quad = \quad \mathbf{I}_{\{f(\mathbf{x}_i) > f(\mathbf{x}'_j)\}} + \frac{1}{2}\mathbf{I}_{\{f(\mathbf{x}_i) = f(\mathbf{x}'_j)\}} \tag{18}$$

for all $i \in \{1, \ldots, m\}$, $j \in \{1, \ldots, n\}$.

Definition 15 (Bipartite rank-shatter coefficient) *Let $\mathcal{F}$ be a class of real-valued functions on $\mathcal{X}$, and let $m,n \in \mathbb{N}$. Define the (m,n)-th* bipartite rank-shatter coefficient *of $\mathcal{F}$, denoted by $r(\mathcal{F}, m, n)$, as follows:*

$$r(\mathcal{F}, m, n) \quad = \quad \max_{\underline{\mathbf{x}} \in \mathcal{X}^m, \underline{\mathbf{x}}' \in \mathcal{X}^n} \left| \left\{ \mathbf{B}_f(\underline{\mathbf{x}}, \underline{\mathbf{x}}') \mid f \in \mathcal{F} \right\} \right| . \tag{19}$$

Clearly, for finite $\mathcal{F}$, we have $r(\mathcal{F}, m, n) \leq |\mathcal{F}|$ for all m,n. In general, $r(\mathcal{F}, m, n) \leq 3^{mn}$ for all m,n. In fact, not all 3^{mn} matrices in $\{0, \frac{1}{2}, 1\}^{m \times n}$ can be realized as bipartite rank matrices. Therefore, we have

$$r(\mathcal{F}, m, n) \leq \psi(m,n),$$

where $\psi(m,n)$ is the number of matrices in $\{0, \frac{1}{2}, 1\}^{m \times n}$ that can be realized as a bipartite rank matrix. The number $\psi(m,n)$ can be characterized in the following ways:

Theorem 16 *Let $\psi(m,n)$ be the number of matrices in $\{0, \frac{1}{2}, 1\}^{m \times n}$ that can be realized as a bipartite rank matrix $\mathbf{B}_f(\underline{\mathbf{x}}, \underline{\mathbf{x}}')$ for some $f : \mathcal{X} \to \mathbb{R}$, $\underline{\mathbf{x}} \in \mathcal{X}^m$, $\underline{\mathbf{x}}' \in \mathcal{X}^n$. Then*

1. *$\psi(m,n)$ is equal to the number of complete mixed acyclic (m,n)-bipartite graphs (where a mixed graph is one which may contain both directed and undirected edges, and where we define a cycle in such a graph as a cycle that contains at least one directed edge and in which all directed edges have the same directionality along the cycle).*
2. *$\psi(m,n)$ is equal to the number of matrices in $\{0, \frac{1}{2}, 1\}^{m \times n}$ that do not contain a sub-matrix of any of the forms shown in Table 4.1.*

Proof
Part 1. Let $\mathcal{G}(m,n)$ denote the set of all complete mixed (m,n)-bipartite graphs. Clearly, $|\mathcal{G}(m,n)| = 3^{mn}$, since there are mn edges and three possibilities for each edge. Let $V = \{v_1, \ldots, v_m\}$, $V' =$

$\{v'_1,\ldots,v'_n\}$ be sets of m and n vertices respectively, and for any matrix $\mathbf{B}=[b_{ij}]\in\{0,\frac{1}{2},1\}^{m\times n}$, let $E(\mathbf{B})$ denote the set of edges between V and V' given by $E(\mathbf{B})=\{(v_i\leftarrow v'_j)\mid b_{ij}=1\}\cup\{(v_i\rightarrow v'_j)\mid b_{ij}=0\}\cup\{(v_i - v'_j)\mid b_{ij}=\frac{1}{2}\}$. Define the mapping $G:\{0,\frac{1}{2},1\}^{m\times n}\rightarrow\mathcal{G}(m,n)$ as follows:

$$G(\mathbf{B})=(V\cup V',E(\mathbf{B}))\,.$$

Then clearly, G is a bijection that puts the sets $\{0,\frac{1}{2},1\}^{m\times n}$ and $\mathcal{G}(m,n)$ into one-to-one correspondence. We show that a matrix $\mathbf{B}\in\{0,\frac{1}{2},1\}^{m\times n}$ can be realized as a bipartite rank matrix if and only if the corresponding bipartite graph $G(\mathbf{B})\in\mathcal{G}(m,n)$ is acyclic.

First suppose $\mathbf{B}=\mathbf{B}_f(\underline{\mathbf{x}},\underline{\mathbf{x}}')$ for some $f:\mathcal{X}\rightarrow\mathbb{R}$, $\underline{\mathbf{x}}\in\mathcal{X}^m$, $\underline{\mathbf{x}}'\in\mathcal{X}^n$, and let if possible $G(\mathbf{B})$ contain a cycle, say

$$(v_{i_1}\leftarrow v'_{j_1} - v_{i_2} - v'_{j_2} - \ldots - v_{i_k} - v'_{j_k} - v_{i_1})\,.$$

Then, from the definition of a bipartite rank matrix, we get

$$f(\mathbf{x}_{i_1})<f(\mathbf{x}'_{j_1})=f(\mathbf{x}_{i_2})=f(\mathbf{x}'_{j_2})=\ldots=f(\mathbf{x}_{i_k})=f(\mathbf{x}'_{j_k})=f(\mathbf{x}_{i_1})\,,$$

which is a contradiction.

To prove the other direction, let $\mathbf{B}\in\{0,\frac{1}{2},1\}^{m\times n}$ be such that $G(\mathbf{B})$ is acyclic. Let $G'(\mathbf{B})$ denote the directed graph obtained by collapsing together vertices in $G(\mathbf{B})$ that are connected by an undirected edge. Then it is easily verified that $G'(\mathbf{B})$ does not contain any directed cycles, and therefore there exists a complete order on the vertices of $G'(\mathbf{B})$ that is consistent with the partial order defined by the edges of $G'(\mathbf{B})$ (topological sorting; see, for example, Cormen et al., 2001, Section 22.4). This implies a unique order on the vertices of $G(\mathbf{B})$ (in which vertices connected by undirected edges are assigned the same position in the ordering). For any $\underline{\mathbf{x}}\in\mathcal{X}^m$, $\underline{\mathbf{x}}'\in\mathcal{X}^n$, identifying $\underline{\mathbf{x}},\underline{\mathbf{x}}'$ with the vertex sets V,V' of $G(\mathbf{B})$ therefore gives a unique order on $\mathbf{x}_1,\ldots,\mathbf{x}_m,\mathbf{x}'_1,\ldots,\mathbf{x}'_n$. It can be verified that defining $f:\mathcal{X}\rightarrow\mathbb{R}$ such that it respects this order then gives $\mathbf{B}=\mathbf{B}_f(\underline{\mathbf{x}},\underline{\mathbf{x}}')$.

Part 2. Consider again the bijection $G:\{0,\frac{1}{2},1\}^{m\times n}\rightarrow\mathcal{G}(m,n)$ defined in Part 1 above. We show that a matrix $\mathbf{B}\in\{0,\frac{1}{2},1\}^{m\times n}$ does not contain a sub-matrix of any of the forms shown in Table 4.1 if and only if the corresponding bipartite graph $G(\mathbf{B})\in\mathcal{G}(m,n)$ is acyclic; the desired result then follows by Part 1 of the theorem.

We first note that the condition that $\mathbf{B}\in\{0,\frac{1}{2},1\}^{m\times n}$ not contain a sub-matrix of any of the forms shown in Table 4.1 is equivalent to the condition that the corresponding mixed (m,n)-bipartite graph $G(\mathbf{B})\in\mathcal{G}(m,n)$ not contain any 4-cycles.

Now, to prove the first direction, let $\mathbf{B}\in\{0,\frac{1}{2},1\}^{m\times n}$ not contain a sub-matrix of any of the forms shown in Table 4.1. As noted above, this means $G(\mathbf{B})$ does not contain any 4-cycles. Let, if possible, $G(\mathbf{B})$ contain a cycle of length $2k$, say

$$(v_{i_1}\leftarrow v'_{j_1} - v_{i_2} - v'_{j_2} - \ldots - v_{i_k} - v'_{j_k} - v_{i_1})\,.$$

Now consider v_{i_1},v'_{j_2}. Since $G(\mathbf{B})$ is a complete bipartite graph, there must be an edge between these vertices. If $G(\mathbf{B})$ contained the edge $(v_{i_1}\rightarrow v'_{j_2})$, it would contain the 4-cycle

$$(v_{i_1}\leftarrow v'_{j_1} - v_{i_2} - v'_{j_2}\leftarrow v_{i_1})\,,$$

which would be a contradiction. Similarly, if $G(\mathbf{B})$ contained the edge $(v_{i_1} - v'_{j_2})$, it would contain the 4-cycle

$$(v_{i_1}\leftarrow v'_{j_1} - v_{i_2} - v'_{j_2} - v_{i_1})\,,$$

which would again be a contradiction. Therefore, $G(\mathbf{B})$ must contain the edge $(v_{i_1} \leftarrow v'_{j_2})$. However, this means $G(\mathbf{B})$ must contain a $2(k-1)$-cycle, namely,

$$(v_{i_1} \leftarrow v'_{j_2} — v_{i_3} — v'_{j_3} — \ldots — v_{i_k} — v'_{j_k} — v_{i_1})\,.$$

By a recursive argument, we eventually get that $G(\mathbf{B})$ must contain a 4-cycle, which is a contradiction.

To prove the other direction, let $\mathbf{B} \in \{0, \frac{1}{2}, 1\}^{m \times n}$ be such that $G(\mathbf{B})$ is acyclic. Then it follows trivially that $G(\mathbf{B})$ does not contain a 4-cycle, and therefore, by the above observation, $\mathbf{B}$ does not contain a sub-matrix of any of the forms shown in Table 4.1. ■

We discuss further properties of the bipartite rank-shatter coefficients in Section 4.3; we first present below our uniform convergence result in terms of these coefficients.

4.2 Uniform Convergence Bound

The following is the main result of this section:

Theorem 17 *Let $\mathcal{F}$ be a class of real-valued functions on $\mathcal{X}$, and let $\underline{y} = (y_1, \ldots, y_M) \in \mathcal{Y}^M$ be any label sequence of length $M \in \mathbb{N}$. Let m be the number of positive labels in $\underline{y}$, and $n = M - m$ the number of negative labels in $\underline{y}$. Then for any* $\varepsilon > 0$,

$$\begin{aligned}\mathbf{P}_{S_X|S_Y=\underline{y}}\left\{\sup_{f\in\mathcal{F}}\left|\hat{A}(f;S) - A(f)\right| \geq \varepsilon\right\} &\leq 4\cdot r(\mathcal{F}, 2m, 2n)\cdot e^{-mn\varepsilon^2/8(m+n)} \\ &= 4\cdot r\left(\mathcal{F}, 2\rho(\underline{y})M, 2(1-\rho(\underline{y}))M\right)\cdot e^{-\rho(\underline{y})(1-\rho(\underline{y}))M\varepsilon^2/8},\end{aligned}$$

where $\rho(\underline{y})$ denotes the positive skew of $\underline{y}$ defined in Eq. (6).

The proof is adapted from proofs of uniform convergence for the classification error rate (see, for example, Anthony and Bartlett, 1999; Devroye et al., 1996). The main difference is that since the AUC cannot be expressed as a sum of independent random variables, more powerful inequalities are required. In particular, a result of Devroye (1991) is required to bound the variance of the AUC that appears after an application of Chebyshev's inequality; the application of this result to the AUC requires the same reasoning that was used to apply McDiarmid's inequality in deriving the large deviation result of Theorem 5. Similarly, McDiarmid's inequality is required in the final step of the proof where Hoeffding's inequality sufficed in the case of classification. Complete details of the proof are given in Appendix A.

As in the case of the large deviation bound of Section 3, we note that the result of Theorem 17 can be strengthened so that the conditioning is only on the numbers m and n of positive and negative labels, and not on the specific label vector $\underline{y}$. From Theorem 17, we can derive a confidence interval interpretation of the bound as follows:

Corollary 18 *Let $\mathcal{F}$ be a class of real-valued functions on $\mathcal{X}$, and let $\underline{y} = (y_1, \ldots, y_M) \in \mathcal{Y}^M$ be any label sequence of length $M \in \mathbb{N}$. Let m be the number of positive labels in $\underline{y}$, and $n = M - m$ the number of negative labels in $\underline{y}$. Then for any* $0 < \delta \leq 1$,

$$\mathbf{P}_{S_X|S_Y=\underline{y}}\left\{\sup_{f\in\mathcal{F}}\left|\hat{A}(f;S) - A(f)\right| \geq \sqrt{\frac{8(m+n)\left(\ln r(\mathcal{F}, 2m, 2n) + \ln\left(\frac{4}{\delta}\right)\right)}{mn}}\right\} \leq \delta.$$

Proof This follows directly from Theorem 17 by setting $4 \cdot r(\mathcal{F}, 2m, 2n) \cdot e^{-mn\varepsilon^2/8(m+n)} = \delta$ and solving for ε. ■

Again, as in the case of the large deviation bound, the confidence interval above can be generalized to remove the conditioning on the label vector completely:

Theorem 19 *Let $\mathcal{F}$ be a class of real-valued functions on $\mathcal{X}$, and let $M \in \mathbb{N}$. Then for any* $0 < \delta \leq 1$,

$$\mathbf{P}_{S \sim \mathcal{D}^M} \left\{ \sup_{f \in \mathcal{F}} \left| \hat{A}(f;S) - A(f) \right| \geq \sqrt{\frac{8\left(\ln r\left(\mathcal{F}, 2\rho(S_Y)M, 2(1-\rho(S_Y))M\right) + \ln\left(\frac{4}{\delta}\right)\right)}{\rho(S_Y)(1-\rho(S_Y))M}} \right\} \quad \leq \quad \delta .$$

4.3 Properties of Bipartite Rank-Shatter Coefficients

As discussed in Section 4.1, we have $r(\mathcal{F}, m, n) \leq \psi(m,n)$, where $\psi(m,n)$ is the number of matrices in $\{0, \frac{1}{2}, 1\}^{m \times n}$ that can be realized as a bipartite rank matrix. The number $\psi(m,n)$ is strictly smaller than 3^{mn}; indeed, $\psi(m,n) = O(e^{(m+n)(\ln(m+n)+1)})$. (To see this, note that the number of distinct bipartite rank matrices of size $m \times n$ is bounded above by the total number of permutations of $(m+n)$ objects, allowing for objects to be placed at the same position. This number is equal to $(m+n)!\,2^{(m+n-1)} = O(e^{(m+n)(\ln(m+n)+1)})$.) Nevertheless, $\psi(m,n)$ is still very large; in particular, $\psi(m,n) \geq 3^{\max(m,n)}$. (To see this, note that choosing any column vector in $\{0, \frac{1}{2}, 1\}^m$ and replicating it along the n columns or choosing any row vector in $\{0, \frac{1}{2}, 1\}^n$ and replicating it along the m rows results in a matrix that does not contain a sub-matrix of any of the forms shown in Table 4.1. The conclusion then follows from Theorem 16 (Part 2).)

For the bound of Theorem 17 to be meaningful, one needs an upper bound on $r(\mathcal{F}, m, n)$ that is at least slightly smaller than $e^{mn/8(m+n)}$. Below we provide one method for deriving upper bounds on $r(\mathcal{F}, m, n)$; taking $\mathcal{Y}^* = \{-1, 0, +1\}$, we extend slightly the standard VC-dimension related shatter coefficients studied in binary classification to $\mathcal{Y}^*$-valued function classes, and then derive an upper bound on the bipartite rank-shatter coefficients $r(\mathcal{F}, m, n)$ of a class of ranking functions $\mathcal{F}$ in terms of the shatter coefficients of a class of $\mathcal{Y}^*$-valued functions derived from $\mathcal{F}$.

Definition 20 (Shatter coefficient) *Let $\mathcal{Y}^* = \{-1, 0, +1\}$, and let $\mathcal{H}$ be a class of $\mathcal{Y}^*$-valued functions on $\mathcal{X}$. Let $N \in \mathbb{N}$. Define the N-*th shatter coefficient *of $\mathcal{H}$, denoted by $s(\mathcal{H}, N)$, as follows:*

$$s(\mathcal{H}, N) \quad = \quad \max_{\mathbf{x} \in \mathcal{X}^N} \left| \{ (h(\mathbf{x}_1), \ldots, h(\mathbf{x}_N)) \mid h \in \mathcal{H} \} \right| .$$

Clearly, $s(\mathcal{H}, N) \leq 3^N$ for all N. Next we define a series of $\mathcal{Y}^*$-valued function classes derived from a given ranking function class. Only the second function class is used in this section; the other two are needed in Section 4.4. Note that we take

$$\text{sign}(u) = \begin{cases} +1 & \text{if } u > 0 \\ 0 & \text{if } u = 0 \\ -1 & \text{if } u < 0 . \end{cases}$$

Definition 21 (Function classes) *Let $\mathcal{F}$ be a class of real-valued functions on $\mathcal{X}$. Define the following classes of $\mathcal{Y}^*$-valued functions derived from $\mathcal{F}$:*

$$\begin{aligned}
&1. \quad \bar{\mathcal{F}} &=& \quad \{\bar{f}:\mathcal{X}\to\mathcal{Y}^* \mid \bar{f}(\mathbf{x}) = \text{sign}(f(\mathbf{x})) \text{ for some } f\in\mathcal{F}\} && (20)\\
&2. \quad \tilde{\mathcal{F}} &=& \quad \{\tilde{f}:\mathcal{X}\times\mathcal{X}\to\mathcal{Y}^* \mid \tilde{f}(\mathbf{x},\mathbf{x}') = \text{sign}(f(\mathbf{x})-f(\mathbf{x}')) \text{ for some } f\in\mathcal{F}\} && (21)\\
&3. \quad \check{\mathcal{F}} &=& \quad \{\check{f}_{\mathbf{z}}:\mathcal{X}\to\mathcal{Y}^* \mid \check{f}_{\mathbf{z}}(\mathbf{x}) = \text{sign}(f(\mathbf{x})-f(\mathbf{z})) \text{ for some } f\in\mathcal{F},\mathbf{z}\in\mathcal{X}\} && (22)
\end{aligned}$$

The following result gives an upper bound on the bipartite rank-shatter coefficients of a class of ranking functions $\mathcal{F}$ in terms of the standard shatter coefficients of $\tilde{\mathcal{F}}$:

Theorem 22 *Let $\mathcal{F}$ be a class of real-valued functions on $\mathcal{X}$, and let $\tilde{\mathcal{F}}$ be the class of $\mathcal{Y}^*$-valued functions on $\mathcal{X}\times\mathcal{X}$ defined by Eq. (21). Then for all $m,n\in\mathbb{N}$,*

$$r(\mathcal{F},m,n) \quad \leq \quad s(\tilde{\mathcal{F}},mn)\,.$$

Proof For any $m,n\in\mathbb{N}$, we have[4]

$$\begin{aligned}
r(\mathcal{F},m,n) \quad &= \quad \max_{\underline{\mathbf{x}}\in\mathcal{X}^m,\underline{\mathbf{x}}'\in\mathcal{X}^n} \left|\left\{ \left[\mathbf{I}_{\{f(\mathbf{x}_i)>f(\mathbf{x}'_j)\}} + \frac{1}{2}\mathbf{I}_{\{f(\mathbf{x}_i)=f(\mathbf{x}'_j)\}}\right] \;\middle|\; f\in\mathcal{F}\right\}\right|\\
&= \quad \max_{\underline{\mathbf{x}}\in\mathcal{X}^m,\underline{\mathbf{x}}'\in\mathcal{X}^n} \left|\left\{ \left[\mathbf{I}_{\{\tilde{f}(\mathbf{x}_i,\mathbf{x}'_j)=+1\}} + \frac{1}{2}\mathbf{I}_{\{\tilde{f}(\mathbf{x}_i,\mathbf{x}'_j)=0\}}\right] \;\middle|\; \tilde{f}\in\tilde{\mathcal{F}}\right\}\right|\\
&= \quad \max_{\underline{\mathbf{x}}\in\mathcal{X}^m,\underline{\mathbf{x}}'\in\mathcal{X}^n} \left|\left\{ \left[\tilde{f}(\mathbf{x}_i,\mathbf{x}'_j)\right] \;\middle|\; \tilde{f}\in\tilde{\mathcal{F}}\right\}\right|\\
&\leq \quad \max_{\mathbf{X},\mathbf{X}'\in\mathcal{X}^{m\times n}} \left|\left\{ \left[\tilde{f}(\mathbf{x}_{ij},\mathbf{x}'_{ij})\right] \;\middle|\; \tilde{f}\in\tilde{\mathcal{F}}\right\}\right|\\
&= \quad \max_{\underline{\mathbf{x}},\underline{\mathbf{x}}'\in\mathcal{X}^{mn}} \left|\left\{ \left(\tilde{f}(\mathbf{x}_1,\mathbf{x}'_1),\ldots,\tilde{f}(\mathbf{x}_{mn},\mathbf{x}'_{mn})\right) \;\middle|\; \tilde{f}\in\tilde{\mathcal{F}}\right\}\right|\\
&= \quad s(\tilde{\mathcal{F}},mn)\,.
\end{aligned}$$

■

Below we make use of the above result to derive polynomial upper bounds on the bipartite rank-shatter coefficients for linear and higher-order polynomial ranking functions. We note that the same method can be used to establish similar upper bounds for other algebraically well-behaved function classes.

Lemma 23 *For $d\in\mathbb{N}$, let $\mathcal{F}_{\text{lin}(d)}$ denote the class of linear ranking functions on $\mathbb{R}^d$:*

$$\mathcal{F}_{\text{lin}(d)} \quad = \quad \{f:\mathbb{R}^d\to\mathbb{R} \mid f(\mathbf{x}) = \mathbf{w}\cdot\mathbf{x}+b \text{ for some } \mathbf{w}\in\mathbb{R}^d, b\in\mathbb{R}\}\,.$$

Then for all $N\in\mathbb{N}$,

$$s(\tilde{\mathcal{F}}_{\text{lin}(d)},N) \;\leq\; \left(\frac{2eN}{d}\right)^d.$$

4. We use the notation $\left[a_{ij}\right]$ to denote a matrix whose $(i,j)^{\text{th}}$ element is a_{ij}. The dimensions of such a matrix should be clear from context.

Proof We have,

$$\tilde{\mathcal{F}}_{\text{lin}(d)} \quad = \quad \{\tilde{f} : \mathbb{R}^d \times \mathbb{R}^d \to \mathcal{Y}^* \mid \tilde{f}(\mathbf{x},\mathbf{x}') = \text{sign}(\mathbf{w}\cdot(\mathbf{x}-\mathbf{x}')) \text{ for some } \mathbf{w} \in \mathbb{R}^d\}\,.$$

Let $(\mathbf{x}_1,\mathbf{x}'_1),\ldots,(\mathbf{x}_N,\mathbf{x}'_N)$ be any N points in $\mathbb{R}^d \times \mathbb{R}^d$, and consider the 'dual' weight space corresponding to $\mathbf{w} \in \mathbb{R}^d$. Each point $(\mathbf{x}_i,\mathbf{x}'_i)$ defines a hyperplane $(\mathbf{x}_i - \mathbf{x}'_i)$ in this space; the N points thus give rise to an arrangement of N hyperplanes in $\mathbb{R}^d$. It is easily seen that the number of sign patterns $(\tilde{f}(\mathbf{x}_1,\mathbf{x}'_1),\ldots,\tilde{f}(\mathbf{x}_N,\mathbf{x}'_N))$ that can be realized by functions $\tilde{f} \in \tilde{\mathcal{F}}_{\text{lin}(d)}$ is equal to the total number of faces of this arrangement (Matoušek, 2002), which is at most (Buck, 1943)

$$\sum_{k=0}^{d}\sum_{i=d-k}^{d}\binom{i}{d-k}\binom{N}{i} \;=\; \sum_{i=0}^{d}2^i\binom{N}{i} \;\le\; \left(\frac{2eN}{d}\right)^d .$$

Since the N points were arbitrary, the result follows. ■

Theorem 24 *For $d \in \mathbb{N}$, let $\mathcal{F}_{\text{lin}(d)}$ denote the class of linear ranking functions on $\mathbb{R}^d$ (defined in Lemma 23 above). Then for all $m,n \in \mathbb{N}$,*

$$r(\mathcal{F}_{\text{lin}(d)},m,n) \;\le\; \left(\frac{2emn}{d}\right)^d .$$

Proof This follows immediately from Lemma 23 and Theorem 22. ■

Lemma 25 *For $d,q \in \mathbb{N}$, let $\mathcal{F}_{\text{poly}(d,q)}$ denote the class of polynomial ranking functions on $\mathbb{R}^d$ with degree less than or equal to q. Then for all $N \in \mathbb{N}$,*

$$s(\tilde{\mathcal{F}}_{\text{poly}(d,q)},N) \;\le\; \left(\frac{2eN}{C(d,q)}\right)^{C(d,q)} ,$$

where

$$C(d,q) \quad = \quad \sum_{i=1}^{q}\left(\binom{d}{i}\sum_{j=1}^{q}\binom{j-1}{i-1}\right). \tag{23}$$

Proof We have,

$$\tilde{\mathcal{F}}_{\text{poly}(d,q)} \quad = \quad \{\tilde{f} : \mathbb{R}^d \times \mathbb{R}^d \to \mathcal{Y}^* \mid \tilde{f}(\mathbf{x},\mathbf{x}') = \text{sign}(f(\mathbf{x}) - f(\mathbf{x}')) \text{ for some } f \in \mathcal{F}_{\text{poly}(d,q)}\}\,.$$

Let $(\mathbf{x}_1,\mathbf{x}'_1),\ldots,(\mathbf{x}_N,\mathbf{x}'_N)$ be any N points in $\mathbb{R}^d \times \mathbb{R}^d$. For any $f \in \mathcal{F}_{\text{poly}(d,q)}$, $(f(\mathbf{x}) - f(\mathbf{x}'))$ is a linear combination of $C(d,q)$ basis functions of the form $(g_k(\mathbf{x}) - g_k(\mathbf{x}'))$, $1 \le k \le C(d,q)$, each $g_k(\mathbf{x})$ being a product of 1 to q components of $\mathbf{x}$. Denote $\mathbf{g}(\mathbf{x}) = (g_1(\mathbf{x}),\ldots,g_{C(d,q)}(\mathbf{x})) \in \mathbb{R}^{C(d,q)}$. Then each point $(\mathbf{x}_i,\mathbf{x}'_i)$ defines a hyperplane $(\mathbf{g}(\mathbf{x}_i) - \mathbf{g}(\mathbf{x}'_i))$ in $\mathbb{R}^{C(d,q)}$; the N points thus give rise to an arrangement of N hyperplanes in $\mathbb{R}^{C(d,q)}$. It is easily seen that the number of sign patterns

$(\tilde{f}(\mathbf{x}_1,\mathbf{x}'_1),\ldots,\tilde{f}(\mathbf{x}_N,\mathbf{x}'_N))$ that can be realized by functions $\tilde{f} \in \tilde{\mathcal{F}}_{\mathrm{poly}(d,q)}$ is equal to the total number of faces of this arrangement (Matoušek, 2002), which is at most (Buck, 1943)

$$\left(\frac{2eN}{C(d,q)}\right)^{C(d,q)}.$$

Since the N points were arbitrary, the result follows. ■

Theorem 26 *For $d,q \in \mathbb{N}$, let $\mathcal{F}_{\mathrm{poly}(d,q)}$ denote the class of polynomial ranking functions on $\mathbb{R}^d$ with degree less than or equal to q. Then for all $m,n \in \mathbb{N}$,*

$$r(\mathcal{F}_{\mathrm{poly}(d,q)},m,n) \;\le\; \left(\frac{2emn}{C(d,q)}\right)^{C(d,q)},$$

where $C(d,q)$ is as defined in Eq. (23).

Proof This follows immediately from Lemma 25 and Theorem 22. ■

4.4 Comparison with Uniform Convergence Bound of Freund et al.

Freund et al. (2003) recently derived a uniform convergence bound for a quantity closely related to the AUC, namely the ranking loss for the bipartite ranking problem. As pointed out by Cortes and Mohri (2004), the bipartite ranking loss is equal to one minus the AUC; the uniform convergence bound of Freund et al. (2003) therefore implies a uniform convergence bound for the AUC.[5] Although the result in (Freund et al., 2003) is given only for function classes considered by their RankBoost algorithm, their technique is generally applicable. We state their result below, using our notation, for the general case (*i.e.*, function classes not restricted to those considered by RankBoost), and then offer a comparison of our bound with theirs. As in (Freund et al., 2003), the result is given in the form of a confidence interval.[6]

Theorem 27 (Generalization of Freund et al. (2003), Theorem 3) *Let $\mathcal{F}$ be a class of real-valued functions on $\mathcal{X}$, and let $\underline{y} = (y_1,\ldots,y_M) \in \mathcal{Y}^M$ be any label sequence of length $M \in \mathbb{N}$. Let m be the number of positive labels in $\underline{y}$, and $n = M - m$ the number of negative labels in $\underline{y}$. Then for any $0 < \delta \le 1$,*

$$\mathbf{P}_{S_X|S_Y=\underline{y}}\left\{\sup_{f\in\mathcal{F}}\left|\hat{A}(f;S)-A(f)\right| \ge 2\sqrt{\frac{\ln s(\check{\mathcal{F}},2m)+\ln\left(\frac{12}{\delta}\right)}{m}} + 2\sqrt{\frac{\ln s(\check{\mathcal{F}},2n)+\ln\left(\frac{12}{\delta}\right)}{n}}\right\} \;\le\; \delta,$$

where $\check{\mathcal{F}}$ is the class of $\mathcal{Y}^*$-valued functions on $\mathcal{X}$ defined by Eq. (22).

5. As in the AUC definition of (Cortes and Mohri, 2004), the ranking loss defined in (Freund et al., 2003) does not account for ties; this is easily remedied.

6. The result in (Freund et al., 2003) was stated in terms of the VC dimension, but the basic result can be stated in terms of shatter coefficients. Due to our AUC definition which accounts for ties, the standard shatter coefficients are replaced here with the extended shatter coefficients defined above for $\mathcal{Y}^*$-valued function classes.

The proof follows that of Freund et al. (2003); for completeness, we give details in Appendix B. We now compare the uniform convergence bound derived in Section 4.2 with that of Freund et al. for a simple function class for which the quantities involved in both bounds (namely, $r(\mathcal{F},2m,2n)$ and $s(\check{\mathcal{F}},2m),s(\check{\mathcal{F}},2n)$) can be characterized exactly. Specifically, consider the function class $\mathcal{F}_{\text{lin}(1)}$ of linear ranking functions on $\mathbb{R}$, given by

$$\mathcal{F}_{\text{lin}(1)} \quad = \quad \{f:\mathbb{R}\to\mathbb{R} \mid f(x)=wx+b \text{ for some } w\in\mathbb{R}, b\in\mathbb{R}\}\,.$$

Although $\mathcal{F}_{\text{lin}(1)}$ is an infinite function class, it is easy to verify that $r(\mathcal{F}_{\text{lin}(1)},m,n)=3$ for all $m,n\in\mathbb{N}$. (To see this, note that for any set of $m+n$ distinct points in $\mathbb{R}$, one can obtain exactly three different ranking behaviours with functions in $\mathcal{F}_{\text{lin}(1)}$: one by setting $w>0$, another by setting $w<0$, and the third by setting $w=0$.) On the other hand, $s(\check{\mathcal{F}}_{\text{lin}(1)},N)=4N+1$ for all $N\geq 2$, since $\check{\mathcal{F}}_{\text{lin}(1)}=\bar{\mathcal{F}}_{\text{lin}(1)}$ (see Eq. (20)) and, as is easily verified, the number of sign patterns on $N\geq 2$ distinct points in $\mathbb{R}$ that can be realized by functions in $\bar{\mathcal{F}}_{\text{lin}(1)}$ is $4N+1$. We thus get from our result (Corollary 18) that

$$\mathbf{P}_{S_X|S_Y=\underline{y}}\left\{\sup_{f\in\mathcal{F}_{\text{lin}(1)}}\left|\hat{A}(f;S)-A(f)\right|\geq\sqrt{\frac{8(m+n)\left(\ln 3+\ln\left(\frac{4}{\delta}\right)\right)}{mn}}\right\} \quad\leq\quad \delta,$$

and from the result of Freund et al. (Theorem 27) that

$$\mathbf{P}_{S_X|S_Y=\underline{y}}\left\{\sup_{f\in\mathcal{F}_{\text{lin}(1)}}\left|\hat{A}(f;S)-A(f)\right|\geq 2\sqrt{\frac{\ln(8m+1)+\ln\left(\frac{12}{\delta}\right)}{m}}+2\sqrt{\frac{\ln(8n+1)+\ln\left(\frac{12}{\delta}\right)}{n}}\right\} \;\leq\; \delta.$$

The above bounds are plotted in Figure 3 for $\delta=0.01$ and various values of $m/(m+n)$. As can be seen, the bound provided by our result is considerably tighter.

4.5 Correctness of Functional Shape of Bound

Although our bound seems to be tighter than the previous bound of Freund et al. (2003), it is still, in general, too loose to make quantitative predictions. Nevertheless, the bound can serve as a useful analysis tool if it displays a correct functional dependence on the training sample size parameters m and n. In this section we give an empirical assessment of the correctness of the functional shape of our bound.

We generated data points in $d=16$ dimensions ($\mathcal{X}=\mathbb{R}^{16}$) as follows. We took $\mathcal{D}_{+1}$ and $\mathcal{D}_{-1}$ to be mixtures of two 16-dimensional Gaussians each, where each of the elements of both the means and the (diagonal) covariances of the Gaussians were chosen randomly from a uniform distribution on the interval $(0,1)$. A test sequence was generated by drawing 2500 points from $\mathcal{D}_{+1}$ and 2500 points from $\mathcal{D}_{-1}$.[7] Training sequences of varying sizes were then generated by drawing m points from $\mathcal{D}_{+1}$ and n points from $\mathcal{D}_{-1}$ for various values of m and n. For each training sequence, a linear ranking function in $\mathcal{F}_{\text{lin}(16)}$ was learned using the RankBoost algorithm of Freund et al. (2003) (the

7. To sample points from Gaussian mixtures we made use of the NETLAB toolbox written by Ian Nabney and Christopher Bishop, available from `http://www.ncrg.aston.ac.uk/netlab/`.

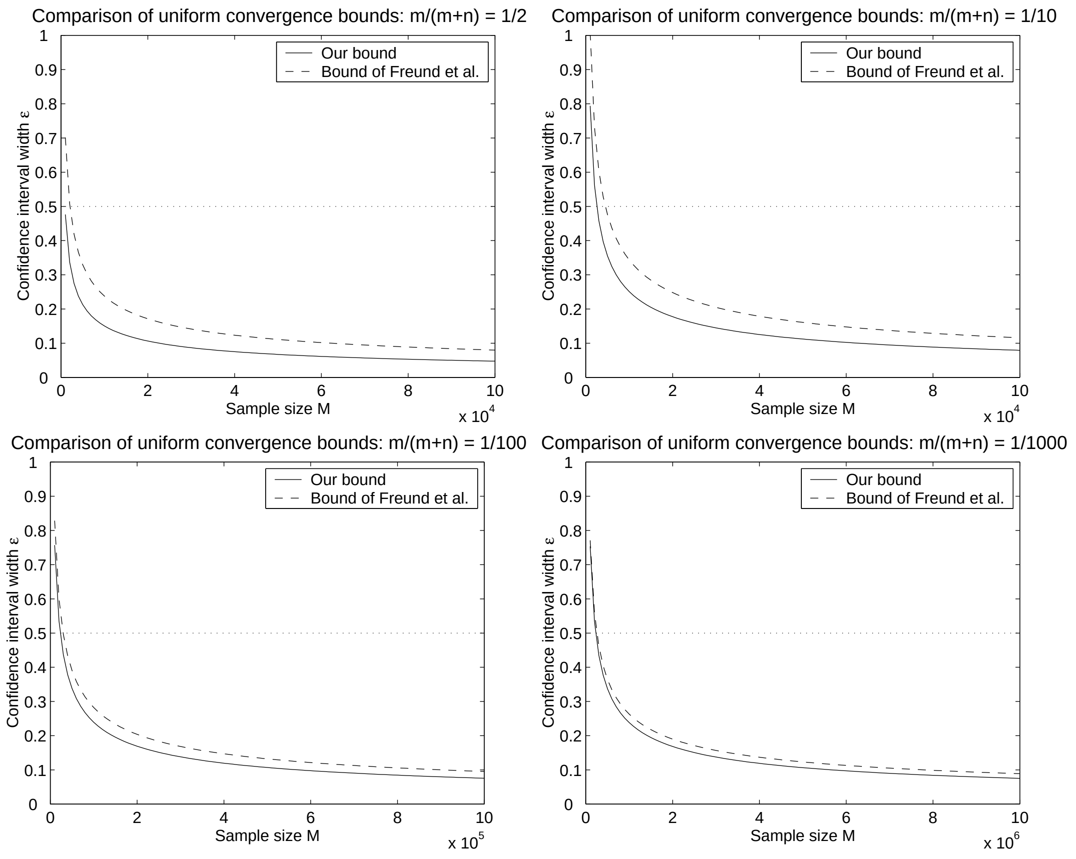


Figure 3: A comparison of our uniform convergence bound with that of Freund et al. (2003) for the class of linear ranking functions on $\mathbb{R}$. The plots are for $\delta = 0.01$ and show how the confidence interval size ε given by the two bounds varies with the sample size $M = m+n$, for various values of $m/(m+n)$. In all cases where the bounds are meaningful ($\varepsilon < 0.5$), our bound is tighter.

algorithm was run for $T = 20$ rounds). The training AUC of the learned ranking function, its AUC on the independent test sequence, and the lower bound on its expected ranking accuracy obtained from our uniform convergence result (using Corollary 18, at a confidence level $\delta = 0.01$) were then calculated. Since we do not have a means to characterize $r(\mathcal{F}_{\text{lin}(16)}, m, n)$ exactly, we used the (loose) bound provided by Theorem 24 in calculating the lower bound on the expected accuracy. The results, averaged over 10 trials (draws of the training sequence) for each pair of values of m and n, are shown in Figure 4. As can be seen, the shape of the bound is in correspondence with that of the test AUC, suggesting that the bound does indeed display a correct functional dependence.

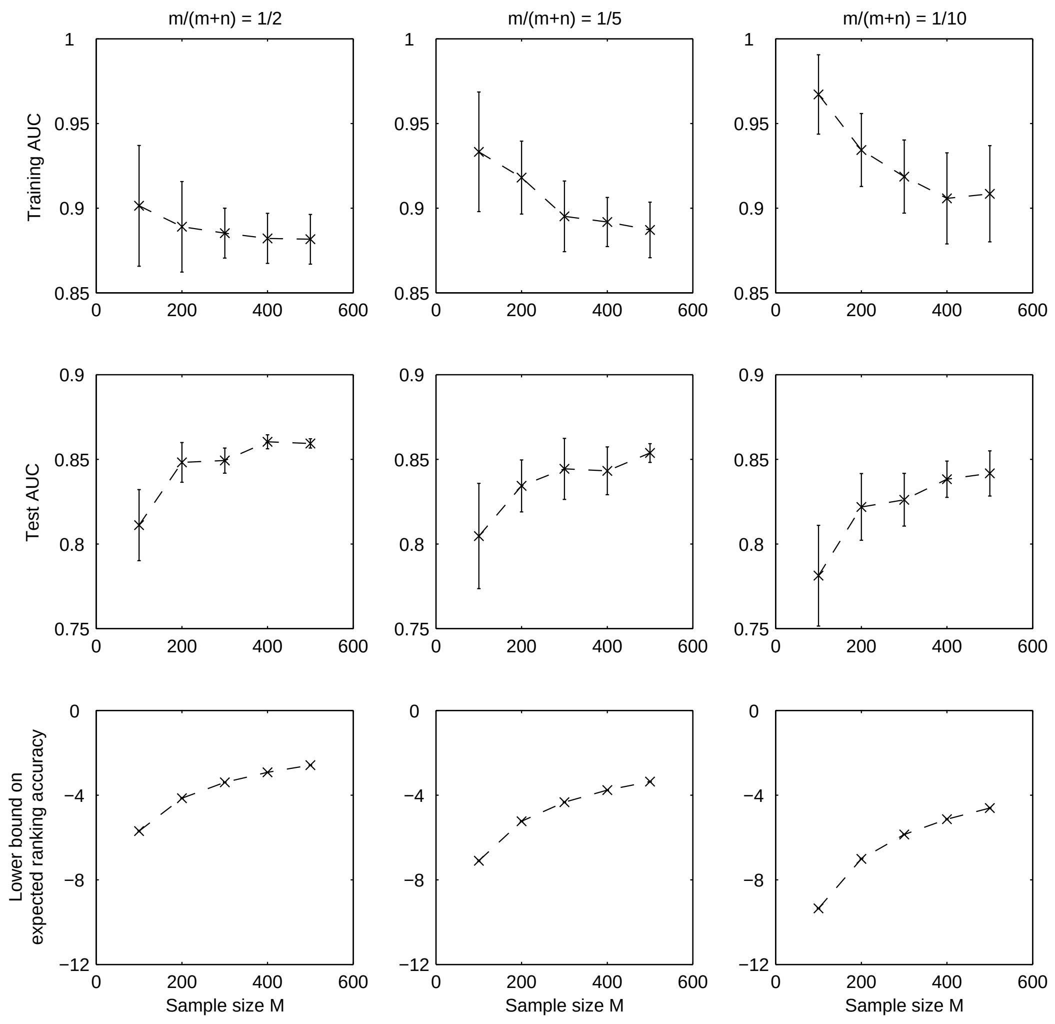


Figure 4: The training AUC (top row), test AUC (middle row), and lower bound on expected ranking accuracy (bottom row) of linear ranking functions learned from training sequences of different sizes $M = m + n$ (see Section 4.5). The plots show mean values over 10 trials for each pair of values of m and n; the error bars show standard deviations (note that there are also error bars on the values of the lower bound; these have the same size as the error bars on the training AUC, but are invisible due to the difference in scale of the plots). Although the bound is quantitatively loose, its shape is in correspondence with that of the test AUC (and therefore correct).

5. Conclusion and Open Questions

We have derived geralization bounds for the area under the ROC curve (AUC), a quantity used as an evaluation criterion for the bipartite ranking problem. We have derived both a large deviation bound, which serves to bound the expected accuracy of a ranking function in terms of its empirical AUC on a test sequence, and a uniform convergence bound, which serves to bound the expected accuracy of a learned ranking function in terms of its empirical AUC on a training sequence. Both our bounds are distribution-free.

Our large deviation result for the AUC parallels the classical large deviation result for the classification error rate obtained via Hoeffding's inequality. A comparison with the large deviation result for the error rate suggests that, in the distribution-free setting, the test sample size required to obtain an ε-accurate estimate of the expected accuracy of a ranking function with δ-confidence is larger than the test sample size required to obtain a similar estimate of the expected error rate of a classification function.

Our uniform convergence bound for the AUC is expressed in terms of a new set of combinatorial parameters that we have termed the bipartite rank-shatter coefficients. These coefficients define a new measure of complexity for real-valued function classes and play the same role in our result as do the standard VC-dimension related shatter coefficients in uniform convergence results for the classification error rate.

For the case of linear ranking functions on $\mathbb{R}$, for which we could compute the bipartite rank-shatter coefficients exactly, we have shown that our uniform convergence bound is considerably tighter than a recent uniform convergence bound derived by Freund et al. (2003), which is expressed directly in terms of standard shatter coefficients from results for classification. This suggests that the bipartite rank-shatter coefficients we have introduced may be a more appropriate complexity measure for studying the bipartite ranking problem. However, in order to take advantage of our results, one needs to be able to characterize these coefficients for the class of ranking functions of interest. The biggest open question that arises from our study is, for what other function classes $\mathcal{F}$ can the bipartite rank-shatter coefficients $r(\mathcal{F},m,n)$ be characterized? We have derived in Theorem 22 a general upper bound on the bipartite rank-shatter coefficients of a function class $\mathcal{F}$ in terms of the standard shatter coefficients of the function class $\tilde{\mathcal{F}}$ (see Eq. (21)); this allows us to establish a polynomial upper bound on the bipartite rank-shatter coefficients for linear and higher-order polynomial ranking functions on $\mathbb{R}^d$ and other algebraically well-behaved function classes. However, this upper bound is inherently loose (see proof of Theorem 22). Is it possible to find tighter upper bounds on $r(\mathcal{F},m,n)$ than that given by Theorem 22?

Our study also raises several other interesting questions. First, can we establish analogous complexity measures and generalization bounds for other forms of ranking problems (*i.e.*, other than bipartite)? Second, do there exist data-dependent bounds for ranking, analogous to existing margin bounds for classification? Finally, it also remains an open question whether tighter (or alternative) generalization bounds for the AUC can be derived using different proof techniques. Possible routes for deriving alternative bounds for the AUC could include the theory of compression bounds (Littlestone and Warmuth, 1986; Graepel et al., 2005).

Acknowledgments

We would like to thank the anonymous reviewers for many useful suggestions and for pointing us to the statistical literature on ranks. We are also very grateful to an anonymous reviewer of an earlier version of part of this work for helping us identify an important mistake in our earlier results. This research was supported in part by NSF ITR grants IIS 00-85980 and IIS 00-85836 and a grant from the ONR-TRECC program.

Appendix A. Proof of Theorem 17

We shall need the following result of Devroye (1991), which bounds the variance of any fuction of a sample for which a single change in the sample has limited effect:

Theorem 28 (Devroye, 1991; Devroye et al., 1996, Theorem 9.3) *Let $X_1, \ldots, X_N$ be independent random variables with X_k taking values in a set A_k for each k. Let $\phi : (A_1 \times \cdots \times A_N) \to \mathbb{R}$ be such that*

$$\sup_{x_i \in A_i, x'_k \in A_k} \left| \phi(x_1, \ldots, x_N) - \phi(x_1, \ldots, x_{k-1}, x'_k, x_{k+1}, \ldots, x_N) \right| \quad \leq \quad c_k \,.$$

Then

$$\mathbf{Var}\left\{ \phi(X_1, \ldots, X_N) \right\} \quad \leq \quad \frac{1}{4} \sum_{k=1}^{N} c_k^2 \,.$$

Proof [of Theorem 17]
The proof is adapted from proofs of uniform convergence for the classification error rate given in (Anthony and Bartlett, 1999; Devroye et al., 1996). It consists of four steps.

Step 1. Symmetrization by a ghost sample.

For each $k \in \{1, \ldots, M\}$, define the random variable $\tilde{X}_k$ such that $X_k, \tilde{X}_k$ are independent and identically distributed. Let $\tilde{S}_X = (\tilde{X}_1, \ldots, \tilde{X}_M)$, and denote by $\tilde{S}$ the joint sequence $(\tilde{S}_X, \underline{y})$. Then for any $\varepsilon > 0$ satisfying $mn\varepsilon^2/(m+n) \geq 2$, we have

$$\mathbf{P}_{S_X|S_Y=\underline{y}} \left\{ \sup_{f \in \mathcal{F}} \left| \hat{A}(f;S) - A(f) \right| \geq \varepsilon \right\} \quad \leq \quad 2\mathbf{P}_{S_X \tilde{S}_X|S_Y=\underline{y}} \left\{ \sup_{f \in \mathcal{F}} \left| \hat{A}(f;S) - \hat{A}(f;\tilde{S}) \right| \geq \frac{\varepsilon}{2} \right\} .$$

To see this, let $f_S^* \in \mathcal{F}$ be a function for which $|\hat{A}(f_S^*;S) - A(f_S^*)| \geq \varepsilon$ if such a function exists, and let f_S^* be a fixed function in $\mathcal{F}$ otherwise. Then

$$\begin{aligned}
&\mathbf{P}_{S_X \tilde{S}_X|S_Y=\underline{y}} \left\{ \sup_{f \in \mathcal{F}} \left| \hat{A}(f;S) - \hat{A}(f;\tilde{S}) \right| \geq \frac{\varepsilon}{2} \right\} \\
\geq \quad & \mathbf{P}_{S_X \tilde{S}_X|S_Y=\underline{y}} \left\{ \left| \hat{A}(f_S^*;S) - \hat{A}(f_S^*;\tilde{S}) \right| \geq \frac{\varepsilon}{2} \right\} \\
\geq \quad & \mathbf{P}_{S_X \tilde{S}_X|S_Y=\underline{y}} \left\{ \left\{ \left| \hat{A}(f_S^*;S) - A(f_S^*) \right| \geq \varepsilon \right\} \cap \left\{ \left| \hat{A}(f_S^*;\tilde{S}) - A(f_S^*) \right| \leq \frac{\varepsilon}{2} \right\} \right\} \\
= \quad & \mathbf{E}_{S_X|S_Y=\underline{y}} \left\{ \mathbf{I}_{\{|\hat{A}(f_S^*;S) - A(f_S^*)| \geq \varepsilon\}} \mathbf{P}_{\tilde{S}_X|S_X, S_Y=\underline{y}} \left\{ \left| \hat{A}(f_S^*;\tilde{S}) - A(f_S^*) \right| \leq \frac{\varepsilon}{2} \right\} \right\} . \qquad (24)
\end{aligned}$$

The conditional probability inside can be bounded using Chebyshev's inequality (and Lemma 2):

$$\mathbf{P}_{\tilde{S}_X|S_X,S_Y=\underline{y}}\left\{\left|\hat{A}(f_S^*;\tilde{S})-A(f_S^*)\right|\leq\frac{\varepsilon}{2}\right\} \quad\geq\quad 1-\frac{\mathbf{Var}_{\tilde{S}_X|S_X,S_Y=\underline{y}}\left\{\hat{A}(f_S^*;\tilde{S})\right\}}{\varepsilon^2/4}\,.$$

Now, by the same reasoning as in the proof of Theorem 5, a change in the value of a single random variable $\tilde{X}_k$ can cause a change of at most $1/m$ in $\hat{A}(f_S^*;\tilde{S})$ for $k:y_k=+1$, and a change of at most $1/n$ for $k:y_k=-1$. Thus, by Theorem 28, we have

$$\mathbf{Var}_{\tilde{S}_X|S_X,S_Y=\underline{y}}\left\{\hat{A}(f_S^*;\tilde{S})\right\} \;\leq\; \frac{1}{4}\left(\sum_{\{i:y_i=+1\}}\left(\frac{1}{m}\right)^2+\sum_{\{j:y_j=-1\}}\left(\frac{1}{n}\right)^2\right) \;=\; \frac{m+n}{4mn}\,.$$

This gives

$$\mathbf{P}_{\tilde{S}_X|S_X,S_Y=\underline{y}}\left\{\left|\hat{A}(f_S^*;\tilde{S})-A(f_S^*)\right|\leq\frac{\varepsilon}{2}\right\} \;\geq\; 1-\frac{m+n}{mn\varepsilon^2} \;\geq\; \frac{1}{2}\,,$$

whenever $mn\varepsilon^2/(m+n)\geq 2$. Thus, from Eq. (24) and the definition of f_S^*, we have

$$\begin{aligned}
\mathbf{P}_{S_X\tilde{S}_X|S_Y=\underline{y}}\left\{\sup_{f\in\mathcal{F}}\left|\hat{A}(f;S)-\hat{A}(f;\tilde{S})\right|\geq\frac{\varepsilon}{2}\right\} \quad &\geq\quad \frac{1}{2}\mathbf{E}_{S_X|S_Y=\underline{y}}\left\{\mathbf{I}_{\{|\hat{A}(f_S^*;S)-A(f_S^*)|\geq\varepsilon\}}\right\}\\
&=\quad \frac{1}{2}\mathbf{P}_{S_X|S_Y=\underline{y}}\left\{\left|\hat{A}(f_S^*;S)-A(f_S^*)\right|\geq\varepsilon\right\}\\
&\geq\quad \frac{1}{2}\mathbf{P}_{S_X|S_Y=\underline{y}}\left\{\sup_{f\in\mathcal{F}}\left|\hat{A}(f;S)-A(f)\right|\geq\varepsilon\right\}.
\end{aligned}$$

Step 2. Permutations.

Let Γ_M be the set of all permutations of $\{X_1,\ldots,X_M,\tilde{X}_1,\ldots,\tilde{X}_M\}$ that swap X_k and $\tilde{X}_k$, for all k in some subset of $\{1,\ldots,M\}$. In other words, for all $\sigma\in\Gamma_M$ and $k\in\{1,\ldots,M\}$, either $\sigma(X_k)=X_k$, in which case $\sigma(\tilde{X}_k)=\tilde{X}_k$, or $\sigma(X_k)=\tilde{X}_k$, in which case $\sigma(\tilde{X}_k)=X_k$. Now, define

$$\begin{aligned}
\beta_f(X_1,\ldots,X_M,\tilde{X}_1,\ldots,\tilde{X}_M) \quad\equiv\quad \frac{1}{mn}\sum_{\{i:y_i=+1\}}\sum_{\{j:y_j=-1\}}\Bigg(&\left(\mathbf{I}_{\{f(X_i)>f(X_j)\}}+\frac{1}{2}\mathbf{I}_{\{f(X_i)=f(X_j)\}}\right)\\
&-\left(\mathbf{I}_{\{f(\tilde{X}_i)>f(\tilde{X}_j)\}}+\frac{1}{2}\mathbf{I}_{\{f(\tilde{X}_i)=f(\tilde{X}_j)\}}\right)\Bigg).
\end{aligned}$$

Then clearly, since $X_k,\tilde{X}_k$ are i.i.d. for each k, for any $\sigma\in\Gamma_M$ we have that the distribution of

$$\sup_{f\in\mathcal{F}}\left|\beta_f(X_1,\ldots,X_M,\tilde{X}_1,\ldots,\tilde{X}_M)\right|$$

is the same as the distribution of

$$\sup_{f\in\mathcal{F}}\left|\beta_f(\sigma(X_1),\ldots,\sigma(X_M),\sigma(\tilde{X}_1),\ldots,\sigma(\tilde{X}_M))\right|.$$

Therefore, using $\mathcal{U}(D)$ to denote the uniform distribution over a discrete set D, we have the following:

$$
\begin{aligned}
&\mathbf{P}_{S_X\tilde{S}_X|S_Y=\underline{y}}\left\{\sup_{f\in\mathcal{F}}\left|\hat{A}(f;S)-\hat{A}(f;\tilde{S})\right|\geq\frac{\varepsilon}{2}\right\}\\
&= \mathbf{P}_{S_X\tilde{S}_X|S_Y=\underline{y}}\left\{\sup_{f\in\mathcal{F}}\left|\beta_f(X_1,\ldots,X_M,\tilde{X}_1,\ldots,\tilde{X}_M)\right|\geq\frac{\varepsilon}{2}\right\}\\
&= \frac{1}{|\Gamma_M|}\sum_{\sigma\in\Gamma_M}\mathbf{P}_{S_X\tilde{S}_X|S_Y=\underline{y}}\left\{\sup_{f\in\mathcal{F}}\left|\beta_f(\sigma(X_1),\ldots,\sigma(X_M),\sigma(\tilde{X}_1),\ldots,\sigma(\tilde{X}_M))\right|\geq\frac{\varepsilon}{2}\right\}\\
&= \frac{1}{|\Gamma_M|}\sum_{\sigma\in\Gamma_M}\mathbf{E}_{S_X\tilde{S}_X|S_Y=\underline{y}}\left\{\mathbf{I}_{\left\{\sup_{f\in\mathcal{F}}\left|\beta_f(\sigma(X_1),\ldots,\sigma(X_M),\sigma(\tilde{X}_1),\ldots,\sigma(\tilde{X}_M))\right|\geq\frac{\varepsilon}{2}\right\}}\right\}\\
&= \mathbf{E}_{S_X\tilde{S}_X|S_Y=\underline{y}}\left\{\frac{1}{|\Gamma_M|}\sum_{\sigma\in\Gamma_M}\mathbf{I}_{\left\{\sup_{f\in\mathcal{F}}\left|\beta_f(\sigma(X_1),\ldots,\sigma(X_M),\sigma(\tilde{X}_1),\ldots,\sigma(\tilde{X}_M))\right|\geq\frac{\varepsilon}{2}\right\}}\right\}\\
&= \mathbf{E}_{S_X\tilde{S}_X|S_Y=\underline{y}}\left\{\mathbf{P}_{\sigma\sim\mathcal{U}(\Gamma_M)}\left\{\sup_{f\in\mathcal{F}}\left|\beta_f(\sigma(X_1),\ldots,\sigma(X_M),\sigma(\tilde{X}_1),\ldots,\sigma(\tilde{X}_M))\right|\geq\frac{\varepsilon}{2}\right\}\right\}\\
&\leq \max_{\underline{\mathbf{x}},\underline{\tilde{\mathbf{x}}}\in\mathcal{X}^M}\mathbf{P}_{\sigma\sim\mathcal{U}(\Gamma_M)}\left\{\sup_{f\in\mathcal{F}}\left|\beta_f(\sigma(\mathbf{x}_1),\ldots,\sigma(\mathbf{x}_M),\sigma(\tilde{\mathbf{x}}_1),\ldots,\sigma(\tilde{\mathbf{x}}_M))\right|\geq\frac{\varepsilon}{2}\right\}.
\end{aligned}
$$

Step 3. Reduction to a finite class.

We wish to bound the quantity on the right hand side above. From the definition of bipartite rank matrices (Definition 14), it follows that for any $\underline{\mathbf{x}},\underline{\tilde{\mathbf{x}}}\in\mathcal{X}^M$, as f ranges over $\mathcal{F}$, the number of different random variables

$$\left|\beta_f(\sigma(\mathbf{x}_1),\ldots,\sigma(\mathbf{x}_M),\sigma(\tilde{\mathbf{x}}_1),\ldots,\sigma(\tilde{\mathbf{x}}_M))\right|$$

is at most the number of different bipartite rank matrices $\mathbf{B}_f(\underline{\mathbf{z}},\underline{\mathbf{z}}')$ that can be realized by functions in $\mathcal{F}$, where $\underline{\mathbf{z}}\in\mathcal{X}^{2m}$ contains $\mathbf{x}_i,\tilde{\mathbf{x}}_i$ for $i:y_i=+1$ and $\underline{\mathbf{z}}'\in\mathcal{X}^{2n}$ contains $\mathbf{x}_j,\tilde{\mathbf{x}}_j$ for $j:y_j=-1$. This number, by definition, cannot exceed $r(\mathcal{F},2m,2n)$ (see the definition of bipartite rank-shatter coefficients, Definition 15). Therefore, the supremum in the above probability is a maximum of at most $r(\mathcal{F},2m,2n)$ random variables. Thus, by the union bound, we get for any $\underline{\mathbf{x}},\underline{\tilde{\mathbf{x}}}\in\mathcal{X}^M$,

$$
\begin{aligned}
&\mathbf{P}_{\sigma\sim\mathcal{U}(\Gamma_M)}\left\{\sup_{f\in\mathcal{F}}\left|\beta_f(\sigma(\mathbf{x}_1),\ldots,\sigma(\mathbf{x}_M),\sigma(\tilde{\mathbf{x}}_1),\ldots,\sigma(\tilde{\mathbf{x}}_M))\right|\geq\frac{\varepsilon}{2}\right\}\\
&\leq r(\mathcal{F},2m,2n)\cdot\sup_{f\in\mathcal{F}}\mathbf{P}_{\sigma\sim\mathcal{U}(\Gamma_M)}\left\{\left|\beta_f(\sigma(\mathbf{x}_1),\ldots,\sigma(\mathbf{x}_M),\sigma(\tilde{\mathbf{x}}_1),\ldots,\sigma(\tilde{\mathbf{x}}_M))\right|\geq\frac{\varepsilon}{2}\right\}.
\end{aligned}
$$

Step 4. McDiarmid's inequality.

Notice that for any $\underline{\mathbf{x}},\underline{\tilde{\mathbf{x}}}\in\mathcal{X}^M$, we can write

$$
\begin{aligned}
&\mathbf{P}_{\sigma\sim\mathcal{U}(\Gamma_M)}\left\{\left|\beta_f(\sigma(\mathbf{x}_1),\ldots,\sigma(\mathbf{x}_M),\sigma(\tilde{\mathbf{x}}_1),\ldots,\sigma(\tilde{\mathbf{x}}_M))\right|\geq\frac{\varepsilon}{2}\right\}\\
&= \mathbf{P}_{\underline{W}\sim\mathcal{U}\left(\prod_{k=1}^{M}\{\mathbf{x}_k,\tilde{\mathbf{x}}_k\}\right)}\left\{\left|\beta_f(W_1,\ldots,W_M,\tilde{W}_1,\ldots,\tilde{W}_M)\right|\geq\frac{\varepsilon}{2}\right\},
\end{aligned}
$$

where $\underline{W} = (W_1,\ldots,W_M)$ and $\tilde{W}_k = \begin{cases} \tilde{\mathbf{x}}_k, & \text{if } W_k = \mathbf{x}_k \\ \mathbf{x}_k, & \text{if } W_k = \tilde{\mathbf{x}}_k \end{cases}$.
Now, for $f : \mathcal{X} \to \mathbb{R}$ and $\mathbf{x},\mathbf{x}' \in \mathcal{X}$, let

$$\alpha(f;\mathbf{x},\mathbf{x}') \quad \equiv \quad \mathbf{I}_{\{f(\mathbf{x})>f(\mathbf{x}')\}} + \frac{1}{2}\mathbf{I}_{\{f(\mathbf{x})=f(\mathbf{x}')\}} \,.$$

Then for any $f \in \mathcal{F}$,

$$\begin{aligned}
&\mathbf{E}_{\underline{W}\sim\mathcal{U}\left(\prod_{k=1}^{M}\{\mathbf{x}_k,\tilde{\mathbf{x}}_k\}\right)}\left\{\beta_f(W_1,\ldots,W_M,\tilde{W}_1,\ldots,\tilde{W}_M)\right\} \\
&\quad = \frac{1}{mn}\sum_{\{i:y_i=+1\}}\sum_{\{j:y_j=-1\}} \mathbf{E}_{W_i\sim\mathcal{U}(\{\mathbf{x}_i,\tilde{\mathbf{x}}_i\}),W_j\sim\mathcal{U}\left(\{\mathbf{x}_j,\tilde{\mathbf{x}}_j\}\right)}\left\{\alpha(f;W_i,W_j)-\alpha(f;\tilde{W}_i,\tilde{W}_j)\right\} \\
&\quad = \frac{1}{mn}\sum_{\{i:y_i=+1\}}\sum_{\{j:y_j=-1\}} \frac{1}{4}\Bigg[\Big(\alpha(f;\mathbf{x}_i,\mathbf{x}_j)-\alpha(f;\tilde{\mathbf{x}}_i,\tilde{\mathbf{x}}_j)\Big)+\Big(\alpha(f;\tilde{\mathbf{x}}_i,\mathbf{x}_j)-\alpha(f;\mathbf{x}_i,\tilde{\mathbf{x}}_j)\Big)+ \\
&\qquad\qquad \Big(\alpha(f;\mathbf{x}_i,\tilde{\mathbf{x}}_j)-\alpha(f;\tilde{\mathbf{x}}_i,\mathbf{x}_j)\Big)+\Big(\alpha(f;\tilde{\mathbf{x}}_i,\tilde{\mathbf{x}}_j)-\alpha(f;\mathbf{x}_i,\mathbf{x}_j)\Big)\Bigg] \\
&\quad = 0\,.
\end{aligned}$$

Also, it can be verified that for any $f \in \mathcal{F}$, a change in the value of a single random variable W_k can bring a change of at most $2/m$ in the value of

$$\beta_f(W_1,\ldots,W_M,\tilde{W}_1,\ldots,\tilde{W}_M)$$

for $k : y_k = +1$, and a change of at most $2/n$ for $k : y_k = -1$. Therefore, by McDiarmid's inequality (Theorem 3), it follows that for any $f \in \mathcal{F}$,

$$\begin{aligned}
&\mathbf{P}_{\underline{W}\sim\mathcal{U}\left(\prod_{k=1}^{M}\{\mathbf{x}_k,\tilde{\mathbf{x}}_k\}\right)}\left\{\left|\beta_f(W_1,\ldots,W_M,\tilde{W}_1,\ldots,\tilde{W}_M)\right| \geq \frac{\varepsilon}{2}\right\} \\
&\quad \leq 2e^{-2\varepsilon^2/4(m(\frac{2}{m})^2+n(\frac{2}{n})^2)} \\
&\quad = 2e^{-mn\varepsilon^2/8(m+n)}\,.
\end{aligned}$$

Putting everything together, we get that

$$\mathbf{P}_{S_X|S_Y=\underline{y}}\left\{\sup_{f\in\mathcal{F}}\left|\hat{A}(f;S)-A(f)\right| \geq \varepsilon\right\} \quad \leq \quad 4\cdot r(\mathcal{F},2m,2n)\cdot e^{-mn\varepsilon^2/8(m+n)}\,,$$

for $mn\varepsilon^2/(m+n) \geq 2$. In the other case, *i.e.*, for $mn\varepsilon^2/(m+n) < 2$, the bound is greater than one and therefore holds trivially. ■

Appendix B. Proof of Theorem 27

We shall need to extend the notion of error rate to $\mathcal{Y}^*$-valued functions (recall that $\mathcal{Y}^* = \{-1,0,+1\}$). Given a function $h : \mathcal{X} \to \mathcal{Y}^*$ and a data sequence $T = ((\mathbf{x}_1,y_1),\ldots,(\mathbf{x}_N,y_N)) \in (\mathcal{X}\times\mathcal{Y})^N$, let the empirical error rate of h with respect to T be denoted by $\hat{L}^*(h;T)$ and defined as

$$\hat{L}^*(h;T) \quad = \quad \frac{1}{N}\sum_{i=1}^{N}\left\{\mathbf{I}_{\{h(\mathbf{x}_i)\neq 0\}}\mathbf{I}_{\{h(\mathbf{x}_i)\neq y_i\}} + \frac{1}{2}\mathbf{I}_{\{h(\mathbf{x}_i)=0\}}\right\}. \qquad (25)$$

Similarly, for an underlying distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, let the expected error rate of h be denoted by $L^*(h)$ and defined as

$$L^*(h) \quad = \quad \mathbf{E}_{XY\sim\mathcal{D}}\left\{\mathbf{I}_{\{h(X)\neq 0\}}\mathbf{I}_{\{h(X)\neq Y\}} + \frac{1}{2}\mathbf{I}_{\{h(X)=0\}}\right\}. \tag{26}$$

Then, following the proof of a similar result given in (Vapnik, 1982) for binary-valued functions, it can be shown that if $\mathcal{H}$ is a class of $\mathcal{Y}^*$-valued functions on $\mathcal{X}$ and $M \in \mathbb{N}$, then for any $\varepsilon > 0$,

$$\mathbf{P}_{S\sim\mathcal{D}^M}\left\{\sup_{h\in\mathcal{H}}\left|\hat{L}^*(h;S) - L^*(h)\right| \geq \varepsilon\right\} \quad \leq \quad 6s(\mathcal{H},2M)e^{-M\varepsilon^2/4}. \tag{27}$$

Proof [of Theorem 27]
To keep notation concise, for $f : \mathcal{X}\to\mathbb{R}$ and $\mathbf{x},\mathbf{x}' \in \mathcal{X}$, let

$$\eta(f;\mathbf{x},\mathbf{x}') \quad \equiv \quad \mathbf{I}_{\{f(\mathbf{x})<f(\mathbf{x}')\}} + \frac{1}{2}\mathbf{I}_{\{f(\mathbf{x})=f(\mathbf{x}')\}},$$

and for $h : \mathcal{X}\to\mathcal{Y}^*$, $\mathbf{x} \in \mathcal{X}$, $y \in \mathcal{Y}$, let

$$\nu(h;\mathbf{x},y) \quad \equiv \quad \mathbf{I}_{\{h(\mathbf{x})\neq 0\}}\mathbf{I}_{\{h(\mathbf{x})\neq y\}} + \frac{1}{2}\mathbf{I}_{\{h(\mathbf{x})=0\}}.$$

Now, given $S_Y = \underline{y}$, we have for all $f \in \mathcal{F}$

$$\begin{aligned}
&\left|\hat{A}(f;S) - A(f)\right| \\
&\quad = \quad \left|(1-\hat{A}(f;S)) - (1-A(f))\right| \\
&\quad = \quad \left|\frac{1}{mn}\sum_{\{i:y_i=+1\}}\sum_{\{j:y_j=-1\}}\eta(f;X_i,X_j) - \mathbf{E}_{X\sim\mathcal{D}_{+1},X'\sim\mathcal{D}_{-1}}\left\{\eta(f;X,X')\right\}\right| \\
&\quad = \quad \left|\frac{1}{mn}\sum_{\{i:y_i=+1\}}\sum_{\{j:y_j=-1\}}\eta(f;X_i,X_j) - \frac{1}{m}\sum_{\{i:y_i=+1\}}\mathbf{E}_{X'\sim\mathcal{D}_{-1}}\left\{\eta(f;X_i,X')\right\}\right. \\
&\qquad\qquad \left. + \frac{1}{m}\sum_{\{i:y_i=+1\}}\mathbf{E}_{X'\sim\mathcal{D}_{-1}}\left\{\eta(f;X_i,X')\right\} - \mathbf{E}_{X\sim\mathcal{D}_{+1},X'\sim\mathcal{D}_{-1}}\left\{\eta(f;X,X')\right\}\right| \\
&\quad = \quad \left|\frac{1}{m}\sum_{\{i:y_i=+1\}}\left(\frac{1}{n}\sum_{\{j:y_j=-1\}}\eta(f;X_i,X_j) - \mathbf{E}_{X'\sim\mathcal{D}_{-1}}\left\{\eta(f;X_i,X')\right\}\right)\right. \\
&\qquad\qquad \left. + \mathbf{E}_{X'\sim\mathcal{D}_{-1}}\left\{\frac{1}{m}\sum_{\{i:y_i=+1\}}\eta(f;X_i,X') - \mathbf{E}_{X\sim\mathcal{D}_{+1}}\left\{\eta(f;X,X')\right\}\right\}\right| \\
&\quad \leq \quad \frac{1}{m}\sum_{\{i:y_i=+1\}}\left|\frac{1}{n}\sum_{\{j:y_j=-1\}}\eta(f;X_i,X_j) - \mathbf{E}_{X'\sim\mathcal{D}_{-1}}\left\{\eta(f;X_i,X')\right\}\right| \\
&\qquad\qquad + \mathbf{E}_{X'\sim\mathcal{D}_{-1}}\left\{\left|\frac{1}{m}\sum_{\{i:y_i=+1\}}\eta(f;X_i,X') - \mathbf{E}_{X\sim\mathcal{D}_{+1}}\left\{\eta(f;X,X')\right\}\right\}\right|
\end{aligned}$$

$$
\begin{aligned}
&\leq \sup_{f'\in\mathcal{F},\mathbf{z}\in\mathcal{X}} \left| \frac{1}{n} \sum_{\{j:y_j=-1\}} \eta(f';\mathbf{z},X_j) - \mathbf{E}_{X'\sim\mathcal{D}_{-1}}\left\{\eta(f';\mathbf{z},X')\right\} \right| \\
&\quad + \sup_{f'\in\mathcal{F},\mathbf{z}\in\mathcal{X}} \left| \frac{1}{m} \sum_{\{i:y_i=+1\}} \eta(f';X_i,\mathbf{z}) - \mathbf{E}_{X\sim\mathcal{D}_{+1}}\left\{\eta(f';X,\mathbf{z})\right\} \right| \\
&= \sup_{\check{f}_\mathbf{z}\in\check{\mathcal{F}}} \left| \frac{1}{n} \sum_{\{j:y_j=-1\}} \nu(\check{f}_\mathbf{z};X_j,-1) - \mathbf{E}_{X'\sim\mathcal{D}_{-1}}\left\{\nu(\check{f}_\mathbf{z};X',-1)\right\} \right| \\
&\quad + \sup_{\check{f}_\mathbf{z}\in\check{\mathcal{F}}} \left| \frac{1}{m} \sum_{\{i:y_i=+1\}} \nu(\check{f}_\mathbf{z};X_i,+1) - \mathbf{E}_{X\sim\mathcal{D}_{+1}}\left\{\nu(\check{f}_\mathbf{z};X,+1)\right\} \right| .
\end{aligned}
$$

If we augment the notation $L^*(h)$ used to denote the expected error rate with the distribution, *e.g.*, $L^*_{\mathcal{D}}(h)$, we thus get

$$
\sup_{f\in\mathcal{F}} \left|\hat{A}(f;S)-A(f)\right| \;\leq\; \sup_{\check{f}_\mathbf{z}\in\check{\mathcal{F}}} \left|\hat{L}^*(\check{f}_\mathbf{z};S^{(n)}_{-1}) - L^*_{\mathcal{D}_{-1}}(\check{f}_\mathbf{z})\right| + \sup_{\check{f}_\mathbf{z}\in\check{\mathcal{F}}} \left|\hat{L}^*(\check{f}_\mathbf{z};S^{(m)}_{+1}) - L^*_{\mathcal{D}_{+1}}(\check{f}_\mathbf{z})\right| , \tag{28}
$$

where $S^{(m)}_{+1}$ and $S^{(n)}_{-1}$ denote the subsequences of S containing the m positive and n negative examples, respectively. Now, from the confidence interval interpretation of the result given in Eq. (27), we have

$$
\mathbf{P}_{S^{(m)}_{+1}\sim\mathcal{D}^m_{+1}} \left\{ \sup_{\check{f}_\mathbf{z}\in\check{\mathcal{F}}} \left|\hat{L}^*(\check{f}_\mathbf{z};S^{(m)}_{+1}) - L^*_{\mathcal{D}_{+1}}(\check{f}_\mathbf{z})\right| \geq 2\sqrt{\frac{\ln s(\check{\mathcal{F}},2m)+\ln\left(\frac{12}{\delta}\right)}{m}} \right\} \;\leq\; \frac{\delta}{2}, \tag{29}
$$

$$
\mathbf{P}_{S^{(n)}_{-1}\sim\mathcal{D}^n_{-1}} \left\{ \sup_{\check{f}_\mathbf{z}\in\check{\mathcal{F}}} \left|\hat{L}^*(\check{f}_\mathbf{z};S^{(n)}_{-1}) - L^*_{\mathcal{D}_{-1}}(\check{f}_\mathbf{z})\right| \geq 2\sqrt{\frac{\ln s(\check{\mathcal{F}},2n)+\ln\left(\frac{12}{\delta}\right)}{n}} \right\} \;\leq\; \frac{\delta}{2}. \tag{30}
$$

Combining Eqs. (28-30) gives the desired result. ■

References

Shivani Agarwal, Thore Graepel, Ralf Herbrich, and Dan Roth. A large deviation bound for the area under the ROC curve. In *Advances in Neural Information Processing Systems 17*. MIT Press, 2005a.

Shivani Agarwal, Sariel Har-Peled, and Dan Roth. A uniform convergence bound for the area under the ROC curve. In *Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics*, 2005b.

Martin Anthony and Peter Bartlett. *Learning in Neural Networks: Theoretical Foundations*. Cambridge University Press, 1999.

Z. W. Birnbaum and O. M. Klose. Bounds for the variance of the Mann-Whitney statistic. *Annals of Mathematical Statistics*, 38, 1957.

R. C. Buck. Partition of space. *American Mathematical Monthly*, 50:2541–544, 1943.

William W. Cohen, Robert E. Schapire, and Yoram Singer. Learning to order things. *Journal of Artificial Intelligence Research*, 10:243–270, 1999.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, second edition, 2001.

Corinna Cortes and Mehryar Mohri. AUC optimization vs. error rate minimization. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, 2004.

Corinna Cortes and Mehryar Mohri. Confidence intervals for the area under the ROC curve. In *Advances in Neural Information Processing Systems 17*. MIT Press, 2005.

Koby Crammer and Yoram Singer. Pranking with ranking. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 641–647. MIT Press, 2002.

D. Van Dantzig. On the consistency and power of Wilcoxon's two sample test. In *Koninklijke Nederlandse Akademie van Weterschappen, Series A*, volume 54, 1915.

Luc Devroye. Exponential inequalities in nonparametric estimation. In G. Roussas, editor, *Nonparametric Functional Estimation and Related Topics*, NATO ASI Series, pages 31–44. Kluwer Academic Publishers, 1991.

Luc Devroye, László Györfi, and Gábor Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer-Verlag, New York, 1996.

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

James P. Egan. *Signal Detection Theory and ROC Analysis*. Academic Press, 1975.

Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.

Thore Graepel, Ralf Herbrich, and John Shawe-Taylor. PAC-Bayesian compression bounds on the prediction error of learning algorithms for classification. *Machine Learning*, 2005. To appear.

James A. Hanley and Barbara J. McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143:29–36, 1982.

Ralf Herbrich, Thore Graepel, and Klaus Obermayer. Large margin rank boundaries for ordinal regression. *Advances in Large Margin Classifiers*, pages 115–132, 2000.

Simon I. Hill, Hugo Zaragoza, Ralf Herbrich, and Peter J. W. Rayner. Average precision and the problem of generalisation. In *Proceedings of the ACM SIGIR Workshop on Mathematical and Formal Methods in Information Retrieval*, 2002.

Erich L. Lehmann. *Nonparametrics: Statistical Methods Based on Ranks*. Holden-Day, San Francisco, California, 1975.

Nick Littlestone and Manfred Warmuth. Relating data compression and learnability. Technical report, University of California Santa Cruz, 1986.

Jiří Matoušek. *Lectures on Discrete Geometry*. Springer-Verlag, New York, 2002.

Colin McDiarmid. On the method of bounded differences. In *Surveys in Combinatorics 1989*, pages 148–188. Cambridge University Press, 1989.

Saharon Rosset. Model selection via the AUC. In *Proceedings of the 21st International Conference on Machine Learning*, 2004.

Vladimir N. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, New York, 1982.

Journal of Machine Learning Research 6 (2005) 427–451 Submitted 9/04; Published 4/05

Learning with Decision Lists of Data-Dependent Features

Mario Marchand MARIO.MARCHAND@IFT.ULAVAL.CA
Département IFT-GLO
Université Laval
Québec, Canada, G1K-7P4

Marina Sokolova SOKOLOVA@SITE.UOTTAWA.CA
School of Information Technology and Engineering
University of Ottawa
Ottawa, Ontario K1N-6N5, Canada

Editor: Manfred K. Warmuth

Abstract

We present a learning algorithm for decision lists which allows features that are constructed from the data and allows a trade-off between accuracy and complexity. We provide bounds on the generalization error of this learning algorithm in terms of the number of errors and the size of the classifier it finds on the training data. We also compare its performance on some natural data sets with the set covering machine and the support vector machine. Furthermore, we show that the proposed bounds on the generalization error provide effective guides for model selection.

Keywords: decision list machines, set covering machines, sparsity, data-dependent features, sample compression, model selection, learning theory

1. Introduction

The set covering machine (SCM) has recently been proposed by Marchand and Shawe-Taylor (2001, 2002) as an alternative to the support vector machine (SVM) when the objective is to obtain a sparse classifier with good generalization. Given a feature space, the SCM attempts to find the smallest conjunction (or disjunction) of features that gives a small training error. In contrast, the SVM attempts to find the maximum soft-margin separating hyperplane on all the features. Hence, the two learning machines are fundamentally different in what they are aiming to achieve on the training data.

To investigate if it is worthwhile to consider larger classes of functions than just the conjunctions and disjunctions that are used in the SCM, we focus here on the class of decision lists (Rivest, 1987) because this class strictly includes both conjunctions and disjunctions while being strictly included in the class of linear threshold functions (Ehrenfeucht and Haussler, 1989; Blum and Singh, 1990; Marchand and Golea, 1993).

From a theoretical point of view, the class of decision lists has been extensively studied (Rivest, 1987; Dhagat and Hellerstein, 1994; Eiter et al., 2002; Anthony, 2004) and a few learning algorithms have been proposed. The first learning algorithm, due to Rivest (1987), PAC learns the class of decision lists (also known as 1-decision lists) over the input attributes but may return a classifier that depends on all the input attributes even when the number r of relevant attributes is much smaller than the total number n of attributes. Dhagat and Hellerstein (1994) and Kivinen et al. (1992) have proposed an attribute efficient algorithm that outputs a decision list of $O(r^k \log^k m)$ attributes for a training set of m examples where k denotes the number of *alternations* of the target decision list (see the definition in Section 2). However, both of these algorithms are unattractive for the practitioner because they do not provide an accuracy-complexity tradeoff. Indeed, real-world data are often noisy and, therefore, simpler functions that make some training errors might be better than more complex functions that make no training errors. Since the amount of noise is problem-specific, a learning algorithm should provide to the user a means to control the tradeoff between accuracy and the complexity of a classifier. Ideally, the user should be able to choose from a wide range of functions that includes very simple functions (like constants), that almost always underfit the data, and very complex functions that often overfit the data. But this latter requirement for decision lists can be generally achieved only if the set of features used for the decision list is data-dependent. It is only with a data-dependent set of features that a restricted class of functions like decision lists can almost always overfit any training data set (that does not contain too many pairs of identical examples with opposite classification labels). Hence, in this paper, we present a learning algorithm for decision lists which can be used with any set of features, including those that are defined with respect to the training data, and that provides some "model selection parameters" (also called learning parameters) for allowing the user to choose the proper tradeoff between accuracy and complexity.

We denote by *decision list machine* (DLM) any classifier which computes a decision list of Boolean-valued features, including features that are possibly constructed from the data. In this paper, we use the set of features known as data-dependent balls (Marchand and Shawe-Taylor, 2001; Sokolova et al., 2003) and the set of features known as data-dependent half-spaces (Marchand et al., 2003). We show, on some natural data sets, that the DLM can provide better generalization than the SCM with the same set of features.

We will see that the proposed learning algorithm for the DLM with data-dependent features is effectively compressing the training data into a small subset of examples which is called the *compression set*. Hence, we will show that the DLM with data-dependent features is an example of a sample-compression algorithm and we will thus propose a general risk bound that depends on the number of examples that are used in the final classifier and the size of the information message needed to identify the final classifier from the compression set. The proposed bound will apply to any compression set-dependent distribution of messages (see the definition in Section 4) and allows for the message set to be of variable size (in contrast with the sample compression bound of Littlestone and Warmuth (1986) that requires fixed size). We will apply this general risk bound to DLMs by making appropriate choices for a compression set-dependent distribution of messages and we will show, on natural data sets, that these specialized risk bounds are generally slightly more effective than K-fold cross validation for selecting a good DLM model.

This paper extends the previous preliminary results of Sokolova et al. (2003).

2. The Decision List Machine

Let $\mathbf{x}$ denote an arbitrary n-dimensional vector of the input space $\mathcal{X}$ which is an arbitrary subset of $\mathbb{R}^n$. We consider binary classification problems for which the training set $S = P \cup N$ consists of a set P of positive training examples and a set N of negative training examples. We define a *feature* as an arbitrary Boolean-valued function that maps $\mathcal{X}$ onto $\{0,1\}$. Given any set $\mathcal{H} = \{h_i(\mathbf{x})\}_{i=1}^{|\mathcal{H}|}$ of features $h_i(\mathbf{x})$ and any training set S, the learning algorithm returns a small subset $\mathcal{R} \subset \mathcal{H}$ of features. Given that subset $\mathcal{R}$, and an arbitrary input vector $\mathbf{x}$, the output $f(\mathbf{x})$ of the decision list machine (DLM) is given by the following rule

If $(h_1(\mathbf{x}))$ then b_1

Else If $(h_2(\mathbf{x}))$ then b_2

$\ldots$

Else If $(h_r(\mathbf{x}))$ then b_r

Else b_{r+1},

where each $b_i \in \{0,1\}$ defines the output of $f(\mathbf{x})$ if and only if h_i is the first feature to be satisfied on $\mathbf{x}$ (*i.e.* the smallest i for which $h_i(\mathbf{x}) = 1$). The constant b_{r+1} (where $r = |\mathcal{R}|$) is known as the *default value*. Note that f computes a disjunction of the h_is whenever $b_i = 1$ for $i = 1 \ldots r$ and $b_{r+1} = 0$. To compute a conjunction of h_is, we simply place in f the negation of each h_i with $b_i = 0$ for $i = 1 \ldots r$ and $b_{r+1} = 1$. Note, however, that a DLM f that contains one or many *alternations* (*i.e.* a pair (b_i, b_{i+1}) for which $b_i \neq b_{i+1}$ for $i < r$) cannot be represented as a (pure) conjunction or disjunction of h_is (and their negations). Hence, the class of decision lists strictly includes conjunctions and disjunctions.

We can also easily verify that decision lists are a proper subset of linear threshold functions in the following way. Given a DLM with r features as above, we assign a weight value w_i to each h_i in the DLM in order to satisfy

$$|w_i| > \sum_{j=i+1}^{r} |w_j| \quad \forall i \in \{1,\ldots,r\}.$$

Let us satisfy these constraints with $|w_i| = 2^{r-i}$ for $i \in \{1,\ldots,r\}$. Then, for each i, we set $w_i = +|w_i|$ if $b_i = 1$, otherwise we set $w_i = -|w_i|$ if $b_i = 0$. For the threshold θ we use $\theta = -1/2$ if $b_{r+1} = 1$ and $\theta = +1/2$ if $b_{r+1} = 0$. With this prescription, given any example $\mathbf{x}$, we always have that

$$\mathrm{sgn}\left(\sum_{i=1}^{r} w_i h_i(\mathbf{x}) - \theta\right) = 2b_k - 1,$$

where k is the smallest integer for which $h_k(\mathbf{x}) = 1$ in the DLM or $k = r+1$ if $h_i(\mathbf{x}) = 0\ \forall i \in \{1,\ldots,r\}$. Hence, with this prescription, the output of the linear threshold function is the same as the output of the DLM for all input $\mathbf{x}$. Finally, to show that the subset is proper we simply point out that a majority vote of three features is a particular case of a linear threshold function that cannot be represented as a decision list since the output of the majority vote cannot be determined from the value of a single feature.

From our definition of the DLM, it seems natural to use the following greedy algorithm for building a DLM from a training set. For a given set $S' = P' \cup N'$ of examples (where $P' \subseteq P$ and $N' \subseteq N$) and a given set $\mathcal{H}$ of features, consider only the features $h_i \in \mathcal{H}$ which either have $h_i(\mathbf{x}) = 0$ for all $\mathbf{x} \in P'$ or $h_i(\mathbf{x}) = 0$ for all $\mathbf{x} \in N'$. Let Q_i be the subset of examples on which $h_i = 1$ (our constraint on the choice of h_i implies that Q_i contains only examples having the same class label). We say that h_i is *covering* Q_i. The greedy algorithm starts with $S' = S$ and an empty DLM. Then it finds a h_i with the largest $|Q_i|$ and appends (h_i, b) to the DLM (where b is the class label of the examples in Q_i). It then removes Q_i from S' and repeats to find the h_k with the largest $|Q_k|$ until either P' or N' is empty. It finally assigns b_{r+1} to the class label of the remaining non-empty set of examples.

Following Rivest (1987), this greedy algorithm is assured to build a DLM that makes no training errors whenever *there exists* a DLM on a set $\mathcal{E} \subseteq \mathcal{H}$ of features that makes zero training errors. However, this constraint is not really required in practice since we do want to permit the user of a learning algorithm to control the tradeoff between the accuracy achieved on the training data and the complexity (here the size) of the classifier. Indeed, a small DLM which makes a few errors on the training set might give better generalization than a larger DLM (with more features) which makes zero training errors. One way to include this flexibility is to early-stop the greedy algorithm when there remains a few more training examples to be covered. But a further reduction in the size of the DLM can be accomplished by considering features h_i that cover examples of both classes. Indeed, if Q_i denotes the subset of S' on which $h_i = 1$ (as before), let P_i denote the subset of P' that belongs to Q_i and let N_i be the subset of N' that belongs to Q_i (thus $Q_i = P_i \cup N_i$). In the previous greedy algorithm, we were considering only features h_i for which either P_i or N_i was empty. Now we are willing to consider features for which neither P_i nor N_i is empty whenever $\max(|P_i|, |N_i|)$ is substantially larger than before. In other words, we want now to consider features that may err on a few examples whenever they can cover many more examples. We therefore define the *usefulness* U_i of feature h_i by

$$U_i \stackrel{\text{def}}{=} \max\{|P_i| - p_n|N_i|,\ |N_i| - p_p|P_i|\},$$

where p_n denotes the *penalty* of making an error on a negative example whereas p_p denotes the penalty of making an error on a positive example. Indeed, whenever we add to a DLM a feature h_i for which P_i and N_i are both non empty, the output b_i associated with h_i will be 1 if $|P_i| - p_n|N_i| \geq |N_i| - p_p|P_i|$ or 0 otherwise. Hence, the DLM will necessarily incorrectly classify the examples in N_i if $b_i = 1$ or the examples in P_i if $b_i = 0$.

Hence, to include this flexibility in choosing the proper tradeoff between complexity and accuracy, each greedy step will be modified as follows. For a given training set $S' = P' \cup N'$, we will select a feature h_i with the largest value of U_i and append $(h_i, 1)$ to the DLM if $|P_i| - p_n|N_i| \geq |N_i| - p_p|P_i|$, otherwise, we append $(h_i, 0)$ to the DLM. If $(h_i, 1)$ was appended, we will then remove from S' every example in P_i (since they are correctly classified by the current DLM) *and* we will also remove from S' every example in N_i (since a DLM with this feature is already misclassifying N_i, and, consequently, the training error of the DLM will not increase if later features err on the examples in N_i). Similarly if $(h_i, 0)$ was appended, we will then remove from S' the examples in $Q_i = N_i \cup P_i$. Hence, we recover the simple greedy algorithm when $p_p = p_n = \infty$.

The formal description of our learning algorithm is presented in Figure 1. Note that we always set $b_{r+1} = \neg b_r$ since, otherwise, we could remove the rth feature without changing the classifier's output f for any input $\mathbf{x}$.

The penalty parameters p_p and p_n and the early stopping point s of **BuildDLM** are the model-selection parameters that give the user the ability to control the proper tradeoff between the training accuracy and the size of the DLM. Their values could be determined either by using K-fold cross-validation, or by computing the risk bounds proposed below. It therefore generalizes the learning algorithm of Rivest (1987) by providing this complexity-accuracy tradeoff and by permitting the use of any kind of Boolean-valued features, including those that are constructed from the training data.

```
Algorithm BuildDLM($S, p_p, p_n, s, \mathcal{H}$)

Input: A set $S$ of examples, the penalty values $p_p$ and $p_n$, a stopping point $s$, and a set
$\mathcal{H} = \{h_i(\mathbf{x})\}_{i=1}^{|\mathcal{H}|}$ of Boolean-valued features.

Output: A decision list $f$ consisting of an ordered set $\mathcal{R} = \{(h_i, b_i)\}_{i=1}^{r}$ of features $h_i$ with their
corresponding output values $b_i$, and a default value $b_{r+1}$.

Initialization: $\mathcal{R} = \emptyset,\ r = 0,\ S' = S,\ b_0 = \neg a$ (where $a$ is the label of the majority class).

  1. For each $h_i \in \mathcal{H}$, let $Q_i = P_i \cup N_i$ be the subset of $S'$ for which $h_i = 1$ (where $P_i$ consists of
     positive examples and $N_i$ consists of negative examples). For each $h_i$ compute $U_i$, where:

         $U_i \overset{\text{def}}{=} \max\{|P_i| - p_n|N_i|,\ |N_i| - p_p|P_i|\}$

  2. Let $h_k$ be a feature with the largest value of $U_k$. If $Q_k = \emptyset$ then go to step 6 (no progress
     possible).

  3. If ($|P_k| - p_n|N_k| \geq |N_k| - p_p|P_k|$) then append $(h_k, 1)$ to $\mathcal{R}$. Else append $(h_k, 0)$ to $\mathcal{R}$.

  4. Let $S' = S' - Q_k$ and let $r = r + 1$.

  5. If ($r < s$ and $S'$ contains examples of both classes) then go to step 1

  6. Set $b_{r+1} = \neg b_r$. Return $f$.
```

Figure 1: The learning algorithm for the decision list machine

The time complexity of **BuildDLM** is trivially bounded as follows. Assuming a time of at most t for evaluating one feature on one example, it takes a time of at most $|\mathcal{H}|mt$ to find the first feature of the DLM for a training set of m examples. For the data-dependent set of features presented in Section 3.1, it is (almost) always possible to find a feature that covers at least one example. In that case, it takes a time of $O(|\mathcal{H}|mst)$ to find s features. Note that the algorithm must stop if, at some greedy step, there does not exists a feature that covers at least one training example.

Generally, we can further reduce the size of the DLM by observing that any feature h_i with $b_i = b_{r+1}$ can be deleted from the DLM if there does not exist a training example $\mathbf{x}$ with label $y = b_{r+1}$ and another feature h_j with $j > i$ and $b_j \neq b_i$ for which $h_i(\mathbf{x}) = h_j(\mathbf{x}) = 1$ (since, in that case, feature h_i can be moved to the end of the DLM without changing the output for any correctly classified training example). The algorithm **PruneDLM** of Figure 2 deletes all such nodes from the DLM.

We typically use both algorithms in the following way. Given a training set, we first run **Build-DLM** without early stopping (*i.e.*, with parameter s set to infinity) to generate what we call a *template* DLM. Then we consider all the possible DLMs that can be obtained by truncating this template. More precisely, if the template DLM contains r features, we build $r+1$ possible DLMs: the DLM that contains zero features (a constant function), the DLM that contains the first feature only,

```
Algorithm PruneDLM($(S, f)$)

Input: A set $S$ of examples, a decision list $f$ consisting of an ordered set $\mathcal{R} = \{(h_i, b_i)\}_{i=1}^{r}$ of
features $h_i$ with their corresponding output values $b_i$, and a default value $b_{r+1}$.

Output: The same decision list $f$ with, possibly, some features removed.

Initialization: $l = r$

  1. Let $(h_k, b_k) \in \mathcal{R}$ be the pair with the largest value of $k$ such that $b_k = b_{r+1}$ and $k < l$.
  2. If $(\nexists (h_j, b_j) \in \mathcal{R} : j > k,\ b_j \neq b_k,\ h_j(\mathbf{x}) = h_k(\mathbf{x})$ for some $(\mathbf{x}, y) \in S$ with $y = b_{r+1})$ then
     delete $(h_k, b_k)$ from $\mathcal{R}$.
  3. $l = k$.
  4. If $(l > 1)$ then go to step 1; else stop.
```

Figure 2: The pruning algorithm for the decision list machine

the DLM that contains the first two features, and so on, up to the DLM that contains all r features. Then we run **PruneDLM** on all these DLMs to try to reduce them further. Finally all these DLMs are tested on a provided testing set.

It is quite easy to build artificial data sets for which **PruneDLM** decreases substantially the size of the DLM. However, for the natural data sets used in Section 7, **PruneDLM** almost never deleted any node from the DLM returned by **BuildDLM**.

3. Data-Dependent Features

The set of *data-dependent balls* (Marchand and Shawe-Taylor, 2001) and *data-dependent half-spaces* (Marchand et al., 2003) were introduced for their usage with the SCM. We now need to adapt their definitions for using them with the DLM.

3.1 Balls and Holes

Let $d : \mathcal{X}^2 \rightarrow \mathbb{R}$ be a metric for our input space $\mathcal{X}$. Let $h_{\mathbf{c},\rho}$ be a feature identified by a *center* $\mathbf{c}$ and a *radius* ρ. Feature $h_{\mathbf{c},\rho}$ is said to be a *ball* iff $h_{\mathbf{c},\rho}(\mathbf{x}) = 1\ \forall \mathbf{x}\colon d(\mathbf{x}, \mathbf{c}) \leq \rho$ and 0 otherwise. Similarly, feature $h_{\mathbf{c},\rho}$ is said to be a *hole* iff $h_{\mathbf{c},\rho}(\mathbf{x}) = 1\ \forall \mathbf{x}\colon d(\mathbf{x}, \mathbf{c}) > \rho$ and 0 otherwise. Hence, a ball is a feature that covers the examples that are located inside the ball; whereas a hole covers the examples that are located outside. In general, both types of features will be used in the DLM.

Partly to avoid computational difficulties, we are going to restrict the centers of balls and holes to belong to the training set, *i.e.*, each center $\mathbf{c}$ must be chosen among $\{\mathbf{x}_i : (\mathbf{x}_i, y_i) \in S\}$ for a given training set S. Moreover, given a center $\mathbf{c}$, the set of relevant radius values are given by the positions of the other training examples, *i.e.*, the relevant radius values belong to $\{d(\mathbf{c}, \mathbf{x}_i) : (\mathbf{x}_i, y_i) \in S\}$. Hence, each ball and hole is identified by only two training examples: a center $\mathbf{c}$ and a *border* $\mathbf{b}$ that identifies the radius with $d(\mathbf{c}, \mathbf{b})$. Therefore, a DLM made of these two-example features is effectively compressing the training data into the smaller set of examples used for its features. This is the other reason why we have constrained the centers and radii to these values. Hence, given a training set S of m examples, the set $\mathcal{H}$ of features used by the DLM will contain $O(m^2)$ balls and holes. This is a data-dependent set of features since the features are defined with respect to the training data S.

Whenever a ball (or hole) $h_{\mathbf{c},\rho}$ is chosen to be appended to the DLM, we must also provide an output value b which will be the output of the DLM on example $\mathbf{x}$ when $h_{\mathbf{c},\rho}$ is the first feature of the DLM that has $h_{\mathbf{c},\rho}(\mathbf{x}) = 1$. In this paper we always choose b to be the class label of $\mathbf{c}$ if $h_{\mathbf{c},\rho}$ is a ball. If $h_{\mathbf{c},\rho}$ is a hole, then we always choose b to be the negation of the class label of $\mathbf{c}$. We have not explored the possibility of using balls and holes with an output *not* given by the class label of its center because, as we will see later, this would have required an additional information bit in order to reconstruct the ball (or hole) from its center and border and, consequently, would have given a looser generalization error bound without providing additional discriminative power (*i.e.*, power to fit the data) that seemed "natural".

To avoid having examples directly on the decision surface of the DLM, the radius ρ of a ball of center $\mathbf{c}$ will always be given by $\rho = d(\mathbf{c},\mathbf{b}) - \varepsilon$ for some training example $\mathbf{b}$ chosen for the border and some fixed and very small positive value ε. Similarly, the radius of a hole of center $\mathbf{c}$ will always be given by $\rho = d(\mathbf{c},\mathbf{b}) + \varepsilon$. We have not chosen to assign the radius values "in between" two training example since this would have required three examples per ball and hole and would have decreased substantially the tightness of our generalization error bound without providing a significant increase of discriminative power.

With these choices for centers and radii, it is straightforward to see that, for any penalty values p_p and p_n, the set of balls having the largest usefulness U always contains a ball with a center and border of opposite class labels whereas the set of holes having the largest usefulness always contains a hole having a center and border of the same class label. Hence, we will only consider such balls and holes in the set of features for the DLM. For a training set of m_p positive examples and m_n negative examples we have exactly $2m_pm_n$ such balls and $m_p^2 + m_n^2$ such holes. We thus provide to **BuildDLM** a set $\mathcal{H}$ of at most $(m_p + m_n)^2$ features.

Finally, note that this set of features has the property that there always exists a DLM of these features that correctly classifies all the training set S provided that S does not contain a pair of contradictory examples, *i.e.*, $(\mathbf{x},y)$ and $(\mathbf{x}',y')$ such that $\mathbf{x} = \mathbf{x}'$ and $y \neq y'$. Therefore, this feature set gives to the user the ability to choose the proper tradeoff between training accuracy and function size.

3.2 Half-Spaces

With the use of kernels, each input vector $\mathbf{x}$ is implicitly mapped into a high-dimensional vector $\boldsymbol{\phi}(\mathbf{x})$ such that $\boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{x}') = k(\mathbf{x},\mathbf{x}')$ (the kernel trick). We consider the case where each feature is a half-space constructed from a set of 3 points $\{\boldsymbol{\phi}_a,\boldsymbol{\phi}_b,\boldsymbol{\phi}_c\}$ where each $\boldsymbol{\phi}_l$ is the image of an input example $\mathbf{x}_l$ taken from the training set S. We consider the case where $\mathbf{x}_a$ and $\mathbf{x}_b$ have opposite class labels and the class label of $\mathbf{x}_c$ is the same as the class label of $\mathbf{x}_b$. The weight vector $\mathbf{w}$ of such a half-space $h^c_{a,b}$ is defined by $\mathbf{w} \stackrel{\text{def}}{=} \boldsymbol{\phi}_a - \boldsymbol{\phi}_b$ and its threshold t by $t \stackrel{\text{def}}{=} \mathbf{w} \cdot \boldsymbol{\phi}_c + \varepsilon$ where ε is a small positive real number. We use $\varepsilon > 0$ to avoid having examples directly on the decision surface of the DLM. Hence

$$h^c_{a,b}(\mathbf{x}) \stackrel{\text{def}}{=} \operatorname{sgn}\{\mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}) - t\} = \operatorname{sgn}\{k(\mathbf{x}_a,\mathbf{x}) - k(\mathbf{x}_b,\mathbf{x}) - t\},$$

where

$$t = k(\mathbf{x}_a, \mathbf{x}_c) - k(\mathbf{x}_b, \mathbf{x}_c) + \varepsilon.$$

Whenever a half-space $h^c_{a,b}$ is chosen to be appended to the DLM, we must also provide an output value b which will be the output of the DLM on example $\mathbf{x}$ when $h^c_{a,b}$ is the first feature of the DLM having $h^c_{a,b}(\mathbf{x}) = 1$. From our definition above, we choose b to be the class label of $\boldsymbol{\phi}_a$. Hence, a DLM made of these three-example features is effectively compressing the training set into the smaller set of examples used for its features.

Given a training set S of $m = m_p + m_n$ examples, the set $\mathcal{H}$ of features considered by the DLM will contain at most $m \cdot m_p \cdot m_n$ half-spaces. However, in contrast with the set of balls and holes, we are not guaranteed to always be able to cover all the training set S with these half-spaces.

Finally, note that this set of features (in the linear kernel case $k(\mathbf{x}, \mathbf{x}') = \mathbf{x} \cdot \mathbf{x}'$) was already proposed by Hinton and Revow (1996) for decision tree learning but no formal analysis of their learning method has been given.

4. A Sample Compression Risk Bound

Since our learning algorithm tries to build a DLM with the smallest number of data-dependent features, and since each feature is described in terms of small number of training examples (two for balls and holes and three for half-spaces), we can thus think of our learning algorithm as compressing the training set into a small subset of examples that we call the *compression set*.

Hence, in this section, we provide a general risk bound that depends on the number of examples that are used in the final classifier and the size of the information message needed to identify the final classifier from the compression set. Such a risk bound was first obtained by Littlestone and Warmuth (1986). The bound provided here allows the message set to be of variable size (whereas previous bounds require fixed size). In the next section, we will compare this bound with other well known bounds. Later, we apply this general risk bound to DLMs by making appropriate choices for a compression set-dependent distribution of messages. Finally, we will show, on natural data sets, that these specialized risk bounds provide an effective guide for choosing the model-selection parameters of **BuildDLM**.

Recall that we consider binary classification problems where the input space $\mathcal{X}$ consists of an arbitrary subset of $\mathbb{R}^n$ and the output space $\mathcal{Y} = \{0, 1\}$. An example $\mathbf{z} \overset{\text{def}}{=} (\mathbf{x}, y)$ is an input-output pair where $\mathbf{x} \in \mathcal{X}$ and $y \in \mathcal{Y}$. We are interested in learning algorithms that have the following property. Given a training set $S = \{\mathbf{z}_1, \ldots, \mathbf{z}_m\}$ of m examples, the classifier $A(S)$ returned by algorithm A is described entirely by two *complementary sources of information*: a subset $\mathbf{z_i}$ of S, called the *compression set*, and a *message string* σ which represents the additional information needed to obtain a classifier from the compression set $\mathbf{z_i}$. This implies that there exists a *reconstruction function* $\mathcal{R}$, associated to A, that outputs a classifier $\mathcal{R}(\sigma, \mathbf{z_i})$ when given an arbitrary compression set $\mathbf{z_i}$ and message string σ chosen from the set $\mathcal{M}(\mathbf{z_i})$ of all distinct messages that can be supplied to $\mathcal{R}$ with the compression set $\mathbf{z_i}$. It is only when such an $\mathcal{R}$ exists that the classifier returned by $A(S)$ is *always* identified by a compression set $\mathbf{z_i}$ and a message string σ.

Given a training set S, the compression set $\mathbf{z_i}$ is defined by a vector $\mathbf{i}$ of indices such that

$$\begin{aligned} \mathbf{i} &\overset{\text{def}}{=} (i_1, i_2, \ldots, i_{|\mathbf{i}|}) \qquad (1) \\ \text{with} &: \quad i_j \in \{1, \ldots, m\} \; \forall j \\ \text{and} &: \quad i_1 < i_2 < \ldots < i_{|\mathbf{i}|}, \end{aligned}$$

where $|\mathbf{i}|$ denotes the number of indices present in $\mathbf{i}$.

The classical perceptron learning rule and support vector machines are examples of learning algorithms where the final classifier can be reconstructed solely from a compression set (Graepel et al., 2000, 2001). In contrast, we will see in the next section that the reconstruction function for DLMs needs both a compression set and a message string.

We seek a tight risk bound for arbitrary reconstruction functions that holds uniformly for all compression sets and message strings. For this, we adopt the PAC setting where each example $\mathbf{z}$ is drawn according to a fixed, but unknown, probability distribution D on $\mathcal{X} \times \mathcal{Y}$. The risk $R(f)$ of any classifier f is defined as the probability that it misclassifies an example drawn according to D:

$$R(f) \overset{\text{def}}{=} \Pr_{(\mathbf{x},y)\sim D}\left(f(\mathbf{x}) \neq y\right) = \mathbf{E}_{(\mathbf{x},y)\sim D} I(f(\mathbf{x}) \neq y),$$

where $I(a) = 1$ if predicate a is true and 0 otherwise. Given a training set $S = \{\mathbf{z}_1, \ldots, \mathbf{z}_m\}$ of m examples, the *empirical risk* $R_S(f)$ on S, of any classifier f, is defined according to

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

Let $\mathbf{Z}^m$ denote the collection of m random variables whose instantiation gives a training sample $S = \mathbf{z}^m = \{\mathbf{z}_1, \ldots, \mathbf{z}_m\}$. Let us denote $\Pr_{\mathbf{Z}^m \sim D^m}(\cdot)$ by $\mathbf{P}_{\mathbf{Z}^m}(\cdot)$. The basic method to find a bound on the true risk of a learning algorithm A, is to bound P' where

$$P' \quad \overset{\text{def}}{=} \quad \mathbf{P}_{\mathbf{Z}^m}\left(R(A(\mathbf{Z}^m)) > \varepsilon\right). \tag{2}$$

Our goal is to find the smallest value for ε such that $P' \leq \delta$ since, in that case, we have

$$\mathbf{P}_{\mathbf{Z}^m}\left(R(A(\mathbf{Z}^m)) \leq \varepsilon\right) \geq 1 - \delta.$$

Recall that classifier $A(\mathbf{z}^m)$ is described entirely in terms of a compression set $\mathbf{z_i} \subset \mathbf{z}^m$ and a message string $\sigma \in \mathcal{M}(\mathbf{z_i})$. Let I be the set of all 2^m vectors of indices $\mathbf{i}$ as defined by Equation 1. Let $\mathcal{M}(\mathbf{z_i})$ be the set of all messages σ that can be attached to compression set $\mathbf{z_i}$. We assume that the empty message is always present in $\mathcal{M}(\mathbf{z_i})$ so that we always have $|\mathcal{M}(\mathbf{z_i})| \geq 1$. Since any $\mathbf{i} \in I$ and $\sigma \in \mathcal{M}(\mathbf{z_i})$ could *a priori* be reached by classifier $A(\mathbf{z}^m)$, we bound P' by the following probability

$$P' \quad \leq \quad \mathbf{P}_{\mathbf{Z}^m}\left(\exists \mathbf{i} \in I \colon \exists \sigma \in \mathcal{M}(\mathbf{Z_i}) \colon R(\mathcal{R}(\sigma, \mathbf{Z_i})) > \varepsilon\right) \overset{\text{def}}{=} P'',$$

where $\mathbf{Z_i}$ are the random variables whose instantiation gives $\mathbf{z_i}$ and where ε depends on $\mathbf{Z_i}, \sigma$ and the amount of training errors. In the sequel, we denote by $\bar{\mathbf{i}}$ the vector of indices made of all the indices not present in $\mathbf{i}$. Since $\mathbf{P}_{\mathbf{Z}^m}(\cdot) = \mathbf{E}_{\mathbf{Z_i}}\mathbf{P}_{\mathbf{Z_{\bar{i}}}|\mathbf{Z_i}}(\cdot)$, we have (by the union bound)

$$\begin{aligned} P'' \quad &\leq \quad \sum_{\mathbf{i} \in I} \mathbf{E}_{\mathbf{Z_i}} \mathbf{P}_{\mathbf{Z_{\bar{i}}}|\mathbf{Z_i}}\left(\exists \sigma \in \mathcal{M}(\mathbf{Z_i}) \colon R(\mathcal{R}(\sigma, \mathbf{Z_i})) > \varepsilon\right) \\ &\leq \quad \sum_{\mathbf{i} \in I} \mathbf{E}_{\mathbf{Z_i}} \sum_{\sigma \in \mathcal{M}(\mathbf{Z_i})} \mathbf{P}_{\mathbf{Z_{\bar{i}}}|\mathbf{Z_i}}\left(R(\mathcal{R}(\sigma, \mathbf{Z_i})) > \varepsilon\right). \end{aligned} \tag{3}$$

We will now stratify $\mathbf{P}_{\mathbf{Z_{\bar{i}}}|\mathbf{Z_i}}(R(\mathcal{R}(\sigma, \mathbf{Z_i})) > \varepsilon)$ in terms of the errors that $\mathcal{R}(\sigma, \mathbf{Z_i})$ can make on the *training examples that are not used for the compression set*. Let $I_{\mathbf{i}}$ denote the set of vectors of indices where each index is not present in $\mathbf{i}$. Given a training sample $\mathbf{z}^m$ and a compression set $\mathbf{z_i}$, we denote by $\mathbf{R}_{\mathbf{z}_{\bar{\mathbf{i}}}}(f)$ the vector of indices pointing to the examples in $\mathbf{z}_{\bar{\mathbf{i}}}$ which are misclassified by f. We have

$$\mathbf{P}_{\mathbf{Z}_{\bar{\mathbf{i}}}|\mathbf{Z_i}}\left(R(\mathcal{R}(\sigma,\mathbf{Z_i}))>\varepsilon\right)=\sum_{\mathbf{j}\in I_{\mathbf{i}}}\mathbf{P}_{\mathbf{Z}_{\bar{\mathbf{i}}}|\mathbf{Z_i}}\left(R(\mathcal{R}(\sigma,\mathbf{Z_i}))>\varepsilon,\mathbf{R}_{\mathbf{Z}_{\bar{\mathbf{i}}}}(\mathcal{R}(\sigma,\mathbf{Z_i}))=\mathbf{j}\right). \tag{4}$$

But now, since the classifier $\mathcal{R}(\sigma,\mathbf{Z_i})$ is fixed when $(\sigma,\mathbf{Z_i})$ is fixed, and since each $\mathbf{Z}_i$ is independent and identically distributed according to the same (but unknown) distribution D, we have

$$\mathbf{P}_{\mathbf{Z}_{\bar{\mathbf{i}}}|\mathbf{Z_i}}\left(R(\mathcal{R}(\sigma,\mathbf{Z_i}))>\varepsilon,\mathbf{R}_{\mathbf{Z}_{\bar{\mathbf{i}}}}(\mathcal{R}(\sigma,\mathbf{Z_i}))=\mathbf{j}\right)\le(1-\varepsilon)^{m-|\mathbf{i}|-|\mathbf{j}|}. \tag{5}$$

Hence, by using Equations 3, 4, and 5, we have

$$P'' \le \sum_{\mathbf{i}\in I}\sum_{\mathbf{j}\in I_{\mathbf{i}}}\mathbf{E}_{\mathbf{Z_i}}\sum_{\sigma\in\mathcal{M}(\mathbf{Z_i})}[1-\varepsilon(\sigma,\mathbf{Z_i},\mathbf{j})]^{m-|\mathbf{i}|-|\mathbf{j}|}, \tag{6}$$

where we have now shown explicitly the dependence of ε on $\mathbf{Z_i}$, σ, and $\mathbf{j}$.

Given any compression set $\mathbf{z_i}$, let us now use any function $P_{\mathcal{M}(\mathbf{z_i})}(\sigma)$ which has the property that

$$\sum_{\sigma\in\mathcal{M}(\mathbf{z_i})}P_{\mathcal{M}(\mathbf{z_i})}(\sigma)\le 1 \tag{7}$$

and can, therefore, be interpreted as compression set-dependent distribution of messages when it sums to one. Let us then choose ε such that

$$\binom{m}{|\mathbf{i}|}\binom{m-|\mathbf{i}|}{|\mathbf{j}|}[1-\varepsilon(\sigma,\mathbf{Z_i},|\mathbf{j}|)]^{m-|\mathbf{i}|-|\mathbf{j}|}=P_{\mathcal{M}(\mathbf{Z_i})}(\sigma)\cdot\zeta(|\mathbf{i}|)\cdot\zeta(|\mathbf{j}|)\cdot\delta, \tag{8}$$

where, for any non-negative integer a, we define

$$\zeta(a)\stackrel{\text{def}}{=}\frac{6}{\pi^2}(a+1)^{-2}. \tag{9}$$

In that case, we have indeed that $P'\le\delta$ since $\sum_{i=1}^{\infty}i^{-2}=\pi^2/6$. Any choice for $\zeta(a)$ is allowed as long as it is a non negative function who's sum is bounded by 1.

The solution to Equation 8 is given by

$$\begin{aligned}\varepsilon(\sigma,\mathbf{Z_i},|\mathbf{j}|,\delta) \quad = \quad & 1-\exp\left(\frac{-1}{m-|\mathbf{i}|-|\mathbf{j}|}\left[\ln\binom{m}{|\mathbf{i}|}+\ln\binom{m-|\mathbf{i}|}{|\mathbf{j}|}+\ln\left(\frac{1}{P_{\mathcal{M}(\mathbf{Z_i})}(\sigma)}\right)+\right.\right.\\ & \left.\left.\ln\left(\frac{1}{\zeta(|\mathbf{i}|)\zeta(|\mathbf{j}|)\delta}\right)\right]\right). \end{aligned} \tag{10}$$

We have therefore shown the following theorem:

Theorem 1 *For any $\delta\in(0,1]$ and for any sample compression learning algorithm with a reconstruction function $\mathcal{R}$ that maps arbitrary subsets of a training set and information messages to classifiers, we have*

$$\mathbf{P}_{\mathbf{Z}^m}\left\{\forall\mathbf{i}\in I,\sigma\in\mathcal{M}(\mathbf{Z_i})\colon R(\mathcal{R}(\sigma,\mathbf{Z_i}))\le\varepsilon(\sigma,\mathbf{Z_i},|\mathbf{j}|,\delta)\right\}\ge 1-\delta.$$

Although the risk bound given by Theorem 1 (and Equation 10) increases with the amount $|\mathbf{j}|$ of training errors made on the examples that do not belong to the compression set $\mathbf{z_i}$, it is interesting to note that it is *independent* of the amount of errors made on the compression set. However, a reconstruction function will generally need less additional information when it is constrained to produce a classifier making no errors with the compression set. Hence, the above risk bound will generally be smaller for sample-compression learning algorithms that always return a classifier making no errors on the compression set. But this constraint might, in turn, force the learner to produce classifiers with larger compression sets.

Finally note that the risk bound is small for classifiers making a small number $|\mathbf{j}|$ of training errors, having a small compression set size $|\mathbf{i}|$, and having a message string σ with large prior "probability" $P_{\mathcal{M}(\mathbf{z_i})}(\sigma)$. This "probability" is usually larger for short message strings since larger message strings are usually much more numerous at sharing the same "piece" (or fraction) of probability.

5. Comparison with Other Risk Bounds

Although the risk bound of Theorem 1 is basically a sample compression bound it, nevertheless, applies to a much broader class of learning algorithms than just sample compression learning algorithms. Indeed the risk bound depends on two complementary sources of information used to identify the classifier: the sample compression set $\mathbf{z_i}$ and the message string σ. In fact, the bound still holds when the sample compression set vanishes and when the classifier $h = \mathcal{R}(\sigma)$ is described entirely in terms of a message string σ. It is therefore worthwhile to compare the risk bound of Theorem 1 to other well-known bounds.

5.1 Comparison with Data-Independent Bounds

The risk bound of Theorem 1 can be qualified as "data-dependent" when the learning algorithm is searching among a class of functions (classifiers) described in terms of a subset $\mathbf{z_i}$ of the training set. Nevertheless, the bound still holds when the class of functions is "data-independent" and when individual functions of this class are identified only in terms of a (data-independent) message σ. In that limit, $|\mathbf{i}| = 0$ and the risk bound ε depends only on σ and the number $|\mathbf{j}| = k$ of training errors:

$$\varepsilon(\sigma, k, \delta) = 1 - \exp\left(\frac{-1}{m-k}\left[\ln\binom{m}{k} + \ln\left(\frac{1}{P_{\mathcal{M}}(\sigma)}\right) + \ln\left(\frac{1}{\zeta(k)\delta}\right)\right]\right). \tag{11}$$

Since here each classifier h is given by $\mathcal{R}(\sigma)$ for some $\sigma \in \mathcal{M}$, we can consider $\mathcal{M}$ as defining a data-independent set of classifiers. This set may contain infinitely many classifiers but it must be countable. Indeed all that is required is

$$\sum_{\sigma \in \mathcal{M}} P(\sigma) \leq 1$$

for any fixed prior P over $\mathcal{M}$. If we further restrict the learning algorithm to produce a classifier that always make no training errors ($k = 0$) and if we choose $P(\sigma) = 1/|\mathcal{M}| \quad \forall \sigma \in \mathcal{M}$ for some finite set $\mathcal{M}$, we obtain the famous Occam's razor bound (Blumer et al., 1987)

$$\varepsilon(\delta) = 1 - \exp\left(\frac{-1}{m}\left[\ln\left(\frac{|\mathcal{M}|}{\delta}\right)\right]\right) \leq \frac{1}{m}\left[\ln\left(\frac{|\mathcal{M}|}{\delta}\right)\right], \tag{12}$$

where we have used $1-\exp(-x) \leq x$. Hence the bound of Equation 11 is a generalization of the Occam's razor bound to the case of an arbitrary (but fixed) prior $P(\sigma)$ over a countably infinite set $\mathcal{M}$ of classifiers which are possibly making some training errors. Consequently, the bound of Theorem 1 is a generalization of the Occam's razor bound to the case where the classifiers are identified by two complementary sources of information: the message string σ and the compression set $\mathbf{z_i}$.

The proposed bound is obtained by using a union bound over the possible compression subsets of the training set and over the possible messages $\sigma \in \mathcal{M}(\mathbf{z_i})$. This bound therefore fails when we consider a continuous set of classifiers. In view of the fact that the set of DLMs of data-dependent features is a subset of the same class of functions but with features that are not constrained to be identified by pairs or triples of training examples, why not use the well-known Vapnik-Chervonenkis (VC) bounds (Vapnik, 1998) or Rademacher bounds (Mendelson, 2002) to characterize the learning algorithms discussed in this paper? The reason is that the proposed algorithms are indeed constrained to use a data-dependent set of features identified by pairs and triples of training examples. The risk bound of Theorem 1 therefore reflects more the set of possible classifiers that can be produced by the proposed algorithms than the VC or Rademacher bounds which are suited for algorithms that can produce any classifier of a continuous set.

5.2 Comparison with Other Sample Compression Risk Bounds

The risk bound of Theorem 1 can be reduced to the sample compression bounds of Littlestone and Warmuth (1986) if we perform the following changes and specializations:

- We restrict the set $\mathcal{M}$ of possible messages to be a finite set which is the same for all possible compression sets $\mathbf{z_i}$.
- For the distribution of messages, we use[1]

$$P_{\mathcal{M}}(\sigma) = \frac{1}{|\mathcal{M}|} \quad \forall \sigma \in \mathcal{M}.$$

- Theorem 1 is valid for any function ζ that satisfies $\sum_{i=0}^{m} \zeta(i) \leq 1$. Here we will use $\zeta(a) = 1/(m+1) \quad \forall a \in \{0,\ldots,m\}$. This choice increases the bound since $6\pi^{-2}(a+1)^{-2} > 1/(m+1)$ for $a < \sqrt{6(m+1)/\pi} - 1$.
- We use the approximation $1-\exp(-x) \leq x$ to obtain a looser (but somewhat easier to understand) bound.

With these restrictions and changes we obtain the following bounds for $|\mathbf{j}| = 0$ and $|\mathbf{j}| \geq 0$:

$$\begin{aligned}
\varepsilon(|\mathbf{i}|,\delta) &\leq \frac{1}{m-|\mathbf{i}|}\left[\ln\binom{m}{|\mathbf{i}|} + \ln\left(\frac{|\mathcal{M}|}{\delta}\right) + \ln(m+1)\right] \quad \text{for } |\mathbf{j}| = 0, && (13)\\
\varepsilon(|\mathbf{i}|,|\mathbf{j}|,\delta) &\leq \frac{1}{m-|\mathbf{i}|-|\mathbf{j}|}\left[\ln\binom{m}{|\mathbf{i}|} + \ln\binom{m-|\mathbf{i}|}{|\mathbf{j}|} + \right.\\
&\qquad \left. \ln\left(\frac{|\mathcal{M}|}{\delta}\right) + 2\ln(m+1)\right] \quad \text{for } |\mathbf{j}| \geq 0. && (14)
\end{aligned}$$

1. The case of $|\mathcal{M}| = 1$ (no message strings used) is also treated by Floyd and Warmuth (1995).

Apart from the $\ln(m+1)$ terms, these bounds are the same as the sample compression bounds of Littlestone and Warmuth (1986). The $\ln(m+1)$ terms are absent from the Littlestone and Warmuth compression bounds because their bounds hold uniformly for all compression sets of a *fixed size* $|\mathbf{i}|$ and for all configurations of training error points of a fixed amount $|\mathbf{j}|$. A $\ln(m+1)$ term occurs in the bound of Equation 13 from the extra requirement to hold uniformly for all compression set sizes. Still an extra $\ln(m+1)$ term occurs in Equation 14 from the extra requirement to hold uniformly for all amounts $|\mathbf{j}|$ of training errors. The bound of Theorem 1 holds uniformly for all compression sets of arbitrary sizes and for all configurations of training error points of an arbitrary amount. But instead of using $\zeta(a) = 1/(m+1) \quad \forall a \in \{0,\ldots,m\}$ we have used the tighter form given by Equation 9.

It is also interesting to compare the bounds of Equations 13 and 14 with the sample compression bounds given by Theorems 5.17 and 5.18 of Herbrich (2002). The bound of Equation 13 is the same as the bound of Theorem 5.17 of Herbrich (2002) when $|\mathcal{M}| = 1$ (no messages used). When the classifier is allowed to make training errors, the bound of Equation 14 is tighter than the lossy compression bound of Theorem 5.18 of Herbrich (2002) when $|\mathbf{j}| \ll m$ since the latter have used the Hoeffding inequality which becomes tight only when $|\mathbf{j}|$ is close to $m/2$.

Consequently, the bound of Theorem 1 is tighter than the above-mentioned sample compression bounds for three reasons. First, the approximation $1-\exp(-x) \leq x$ was not performed. Second, the function $\zeta(a)$ of Equation 9 was used instead of the looser factor of $1/(m+1)$. Third, in contrast with the other sample compression bounds, the bound of Theorem 1 is valid for any *a priori* defined sample compression-dependent distribution of messages $P_{\mathcal{M}(\mathbf{z_i})}(\sigma)$.

This last characteristic may be the most important contribution of Theorem 1. Indeed, we feel that it is important to allow the set of possible messages and the message set size to depend on the sample compression $\mathbf{z_i}$ since the class labels of the compression set examples give information about the set of possible data-dependent features that can be constructed from $\mathbf{z_i}$. Indeed, it is conceivable that for some $\mathbf{z_i}$, very little extra information may be needed to identify the classifier whereas for some other $\mathbf{z_i}$, more information may be needed. Consider, for example, the case where the compression set consists of two examples that are used by the reconstruction function $\mathcal{R}$ to obtain a single-ball classifier. For the reconstruction function of the set covering machine (described in the next section), a ball border must be a positive example whereas both positive and negative examples are allowed for ball centers. In that case, if the two examples in the compression set have a positive label, the reconstruction function needs a message string of at least one bit that indicates which example is the ball center. If the two examples have opposite class labels, then the negative example is necessarily the ball center and no message at all is needed to reconstruct the classifier. More generally, the set of messages that we use for all types of DLMs proposed in this paper depends on some properties of $\mathbf{z_i}$ like its number $n(\mathbf{z_i})$ of negative examples. Without such a dependency on $\mathbf{z_i}$, the set of possible messages $\mathcal{M}$ could be unnecessarily too large and would then loosen the risk bound.

5.3 Comparison with the Set Covering Machine Risk Bound

The risk bound for the set covering machine (SCM) (Marchand and Shawe-Taylor, 2001, 2002) is not a general-purposed sample compression risk bound as the one provided by Theorem 1. It does exploit the fact that the final classifier is partly identified by a small subset of the training examples (the compression set) but, instead of using messages to provide the additional information needed to obtain a classifier, it partitions the compression set into three disjoint sets. Hence, we cannot compare directly the bound of Theorem 1 with the SCM risk bound since the latter is much more specialized than the former. Instead we will show how we can apply the general risk bound of Theorem 1 to the case of the SCM just by choosing an appropriate sample compression-dependent distribution of messages $P_{\mathcal{M}(\mathbf{z_i})}(\sigma)$.

Recall that the task of the SCM is to construct the smallest possible conjunction of (Boolean-valued) features. We discuss here only the conjunction case. The disjunction case is treated similarly just by exchanging the role of the positive with the negative examples.

For the case of data-dependent balls and holes, each feature is identified by a training example called a *center* $(\mathbf{x}_c, y_c)$, and another training example called a *border* $(\mathbf{x}_b, y_b)$. Given any metric d, the output $h(\mathbf{x})$ on any input example $\mathbf{x}$ of such a feature is given by

$$h(\mathbf{x}) = \begin{cases} y_c & \text{if } d(\mathbf{x},\mathbf{x}_c) \le d(\mathbf{x},\mathbf{x}_b) \\ \neg y_c & \text{otherwise.} \end{cases}$$

In this case, given a compression set $\mathbf{z_i}$, we need to specify the examples in $\mathbf{z_i}$ that are used for a border point without being used as a center. As explained by Marchand and Shawe-Taylor (2001), no additional amount of information is required to pair each center with its border point whenever the reconstruction function $\mathcal{R}$ is constrained to produce a classifier that always correctly classifies the compression set. Furthermore, as argued by Marchand and Shawe-Taylor (2001), we can limit ourselves to the case where each border point is a positive example. In that case, each message $\sigma \in \mathcal{M}(\mathbf{z_i})$ just needs to specify the positive examples that are a border point without being a center. Let $n(\mathbf{z_i})$ and $p(\mathbf{z_i})$ be, respectively, the number of negative and the number of positive examples in compression set $\mathbf{z_i}$. Let $b(\sigma)$ be the number of border point examples specified in message σ and let $\zeta(a)$ be defined by Equation 9. We can then use

$$P_{\mathcal{M}(\mathbf{Z_i})}(\sigma) = \zeta(b(\sigma)) \cdot \binom{p(\mathbf{z_i})}{b(\sigma)}^{-1} \tag{15}$$

since, in that case, we have for any compression set $\mathbf{z_i}$:

$$\sum_{\sigma \in \mathcal{M}(\mathbf{z_i})} P_{\mathcal{M}(\mathbf{z_i})}(\sigma) = \sum_{b=0}^{p(\mathbf{z_i})} \zeta(b) \sum_{\sigma : b(\sigma) = b} \binom{p(\mathbf{z_i})}{b(\sigma)}^{-1} \le 1.$$

With this distribution $P_{\mathcal{M}(\mathbf{z_i})}$, the risk bound of Theorem 1 specializes to

$$\begin{aligned} \varepsilon(\sigma, \mathbf{Z_i}, |\mathbf{j}|, \delta) \quad = \quad & 1 - \exp\left(\frac{-1}{m - |\mathbf{i}| - |\mathbf{j}|} \left[\ln \binom{m}{|\mathbf{i}|} + \ln \binom{m - |\mathbf{i}|}{|\mathbf{j}|} + \ln \binom{p(\mathbf{z_i})}{b(\sigma)} + \right.\right. \\ & \left.\left. \ln \left(\frac{1}{\zeta(|\mathbf{i}|)\zeta(|\mathbf{j}|)\zeta(b(\sigma))\delta} \right) \right] \right) . \end{aligned} \tag{16}$$

In contrast, the SCM risk bound of Marchand and Shawe-Taylor (2001) is equal to

$$\begin{aligned} \varepsilon'(\sigma, \mathbf{Z_i}, |\mathbf{j}|, \delta) \quad = \quad & 1 - \exp\left(\frac{-1}{m - |\mathbf{i}| - |\mathbf{j}|} \left[\ln \binom{m}{|\mathbf{i}|} + \ln \binom{m - |\mathbf{i}|}{|\mathbf{j}|} + \right.\right. \\ & \left.\left. \ln \binom{c_p(\mathbf{z_i}) + c_n(\mathbf{z_i}) + b(\mathbf{z_i})}{c_p(\mathbf{z_i})} + \ln \left(\frac{m^2 |\mathbf{i}|}{\delta} \right) \right] \right) , \end{aligned} \tag{17}$$

where $c_p(\mathbf{z_i})$ and $c_n(\mathbf{z_i})$ denote, respectively, the number of positive centers and the number of negative centers in $\mathbf{z_i}$ and where $b(\mathbf{z_i})$ denotes the the number of borders in $\mathbf{z_i}$.

Hence, we observe only two differences between these two bounds. First, the (larger) $\ln(m^2|\mathbf{i}|/\delta)$ term of Marchand and Shawe-Taylor (2001) has been replaced by the (smaller) $\ln(1/\zeta(|\mathbf{i}|)\zeta(|\mathbf{j}|)\zeta(b(\sigma))\delta)$ term. Second, the coefficient

$$\binom{c_p(\mathbf{z_i})+c_n(\mathbf{z_i})+b(\mathbf{z_i})}{c_p(\mathbf{z_i})}$$

has been replaced by the smaller coefficient

$$\binom{p(\mathbf{z_i})}{b(\sigma)}.$$

We can verify that this last coefficient is indeed smaller since

$$\binom{p(\mathbf{z_i})}{b(\sigma)} = \binom{c_p(\mathbf{z_i})+b(\mathbf{z_i})}{b(\mathbf{z_i})} = \binom{c_p(\mathbf{z_i})+b(\mathbf{z_i})}{c_p(\mathbf{z_i})} \leq \binom{c_p(\mathbf{z_i})+c_n(\mathbf{z_i})+b(\mathbf{z_i})}{c_p(\mathbf{z_i})}.$$

Consequently, the risk bound of Theorem 1, applied to the SCM with the distribution given by Equation 15, is smaller than the SCM risk bound of Marchand and Shawe-Taylor (2001).

6. Risks Bounds for Decision List Machines

To apply the risk bound of Theorem 1, we need to define a distribution of message strings[2] $P_{\mathcal{M}(\mathbf{z_i})}(\sigma)$ for each type of DLM that we will consider. Once that distribution is known, we only need to insert it in Equation 10 to obtain the risk bound. Note that the risk bound does not depend on how we actually code σ (for some receiver, in a communication setting). It only depends on the *a priori* probabilities assigned to each possible realization of σ.

6.1 DLMs Containing Only Balls

Even in this simplest case, the compression set $\mathbf{z_i}$ alone does not contain enough information to identify a DLM classifier (the hypothesis). To identify unambiguously the hypothesis we need to provide also a message string σ.

Recall that, in this case, $\mathbf{z_i}$ contains ball centers and border points. By construction, each center is always correctly classified by the hypothesis. Moreover, each center can only be the center of one ball since the center is removed from the data when a ball is added to the DLM. But a center can be the border of a previous ball in the DLM and a border can be the border of more than one ball (since the border of a ball is not removed from the data when that ball is added to the DLM). Hence, σ needs to specify the border points in $\mathbf{z_i}$ that are a border without being the center of another ball. Let σ_1 be the part of the message string σ that will specify that information and let $P_1(\sigma_1)$ be the probabilities that we assign to each possible realization of σ_1. Since we expect that most of the compression sets will contain roughly the same number of centers and borders, we assign, to each example of $\mathbf{z_i}$, an equal *a priori* probability to be a center or a border. Hence we use

$$P_1(\sigma_1) = \frac{1}{2^{|\mathbf{i}|}} \quad \forall \sigma_1.$$

2. We will refer to $P_{\mathcal{M}(\mathbf{z_i})}$ as the “distribution” of messages even though its summation over the possible realizations of σ might be less than one (as specified by Equation 7).

Once σ_1 is specified, the centers and borders of $\mathbf{z_i}$ are identified. But to identify each ball we need to pair each center with a border point (which could possibly be the center of another ball). For this operation, recall that the border and the center of each ball must have opposite class labels. Let σ_2 be the part of the message string σ that specifies that pairing information and let $P_2(\sigma_2|\sigma_1)$ be the probabilities that we assign to each possible realization of σ_2 once σ_1 is given. Let $n(\mathbf{z_i})$ and $p(\mathbf{z_i})$ be, respectively, the number of negative and the number of positive examples in compression set $\mathbf{z_i}$. Consider now a positive center example $\mathbf{x}$ of $\mathbf{z_i}$. Since a border point can be used for more that one ball and a center can also be used as a border, we assign an equal probability of $1/n(\mathbf{z_i})$ to each negative example of $\mathbf{z_i}$ to be paired with $\mathbf{x}$. Similarly, we assign an equal probability of $1/p(\mathbf{z_i})$ to each positive example to be paired with a negative center of $\mathbf{z_i}$. Let $c_p(\mathbf{z_i})$ and $c_n(\mathbf{z_i})$ be, respectively, the number of positive centers and negative centers in $\mathbf{z_i}$ (this is known once σ_1 is specified). For an arbitrary compression set $\mathbf{z_i}$, we thus assign the following *a priori* probability to each possible pairing information string σ_2:

$$P_2(\sigma_2|\sigma_1) = \left(\frac{1}{n(\mathbf{z_i})}\right)^{c_p(\mathbf{z_i})} \left(\frac{1}{p(\mathbf{z_i})}\right)^{c_n(\mathbf{z_i})} \quad \forall\sigma_2 \mid n(\mathbf{z_i}) \neq 0 \text{ and } p(\mathbf{z_i}) \neq 0.$$

This probability is, indeed, correctly defined only under the condition that $n(\mathbf{z_i}) \neq 0$ and $p(\mathbf{z_i}) \neq 0$. However, since the border and center of each ball must have opposite labels, this condition is the same as $|\mathbf{i}| \neq 0$. When $|\mathbf{i}| = 0$, we can just assign 1 to $P_2(\sigma_2|\sigma_1)$. By using the indicator function $I(a)$ defined previously, we can thus write $P_2(\sigma_2|\sigma_1)$ more generally as

$$P_2(\sigma_2|\sigma_1) = \left(\frac{1}{n(\mathbf{z_i})}\right)^{c_p(\mathbf{z_i})} \left(\frac{1}{p(\mathbf{z_i})}\right)^{c_n(\mathbf{z_i})} I(|\mathbf{i}| \neq 0) \; + \; I(|\mathbf{i}| = 0) \quad \forall\sigma_2.$$

Once σ_1 and σ_2 are known, each ball of the DLM is known. However, to place these balls in the DLM, we need to specify their order. Let $r(\mathbf{z_i}) \overset{\text{def}}{=} c_p(\mathbf{z_i}) + c_n(\mathbf{z_i})$ be the number of balls in the DLM (this is known once σ_1 and σ_2 are specified). Let σ_3 be the part of the message string σ that specifies this ordering information and let $P_3(\sigma_3|\sigma_2,\sigma_1)$ be the probabilities that we assign to each possible realization of σ_3 once σ_1 and σ_2 are given. For an arbitrary compression set $\mathbf{z_i}$, we assign an equal *a priori* probability to each possible ball ordering by using

$$P_3(\sigma_3|\sigma_2,\sigma_1) = \frac{1}{r(\mathbf{z_i})!} \quad \forall\sigma_3.$$

The distribution of messages is then given by $P_1(\sigma_1)P_2(\sigma_2|\sigma_1)P_3(\sigma_3|\sigma_2,\sigma_1)$. Hence

$$P_{\mathcal{M}(\mathbf{z_i})}(\sigma) = \frac{1}{2^{|\mathbf{i}|}} \cdot \left[\left(\frac{1}{n(\mathbf{z_i})}\right)^{c_p(\mathbf{z_i})} \left(\frac{1}{p(\mathbf{z_i})}\right)^{c_n(\mathbf{z_i})} I(|\mathbf{i}| \neq 0) \; + \; I(|\mathbf{i}| = 0)\right] \cdot \frac{1}{r(\mathbf{z_i})!} \quad \forall\sigma. \tag{18}$$

6.2 DLMs Containing Balls and Holes

The use of holes in addition to balls introduces a few more difficulties that are taken into account by sending a few more bits of information to the reconstruction function. The most important change is that the center of a hole can be used more than once since the covered examples are outside the hole. Hence, the number of features can now exceed the number of centers but it is always smaller than $|\mathbf{i}|^2$. Indeed, in the worst case, each pair of (distinct) examples taken from the compression set $\mathbf{z_i}$ could be used for two holes: giving a total of $|\mathbf{i}|(|\mathbf{i}|-1)$ features. The first part σ_1 of the (complete) message string σ will specify the number $r(\mathbf{z_i})$ of features present in compression set $\mathbf{z_i}$. Since we always have $r(\mathbf{z_i}) < |\mathbf{i}|^2$ for $|\mathbf{i}| > 0$, we could give equal *a priori* probability for each value of $r \in \{0, \ldots, |\mathbf{i}|^2\}$. However since we want to give a slight preference to smaller DLMs, we choose to assign a probability equal to $\zeta(r)$ (defined by Equation 9) for all possible values of r. Hence

$$P_1(\sigma_1) = \zeta(r(\mathbf{z_i})) \quad \forall \sigma_1.$$

The second part σ_2 of σ specifies, for each feature, if the feature is a ball or a hole. For this, we give equal probability to each of the $r(\mathbf{z_i})$ features to be a ball or a hole. Hence

$$P_2(\sigma_2|\sigma_1) = 2^{-r(\mathbf{z_i})} \quad \forall \sigma_2.$$

Finally, the third part σ_3 of σ specifies, sequentially for each feature, the center and border point. For this, we give an equal probability of $1/|\mathbf{i}|$ to each example in $\mathbf{z_i}$ of being chosen (whenever $|\mathbf{i}| \neq 0$). Consequently

$$P_3(\sigma_3|\sigma_2,\sigma_1) = |\mathbf{i}|^{-2r(\mathbf{z_i})} I(|\mathbf{i}| \neq 0) + I(|\mathbf{i}| = 0) \quad \forall \sigma_3.$$

The distribution of messages is then given by $P_1(\sigma_1)P_2(\sigma_2|\sigma_1)P_3(\sigma_3|\sigma_2,\sigma_1)$. Hence

$$P_{\mathcal{M}(\mathbf{z_i})}(\sigma) = \zeta(r(\mathbf{z_i})) \cdot 2^{-r(\mathbf{z_i})} \cdot \left[|\mathbf{i}|^{-2r(\mathbf{z_i})} I(|\mathbf{i}| \neq 0) + I(|\mathbf{i}| = 0) \right] \quad \forall \sigma. \tag{19}$$

6.3 Constrained DLMs Containing Only Balls

A *constrained* DLM is a DLM that has the property of correctly classifying each example of its compression set $\mathbf{z_i}$ with the exception of the compression set examples who's output is determined by the default value. This implies that **BuildDLM** must be modified to ensure that this constraint is satisfied. This is achieved by considering, at each greedy step, only the features h_i with an output b_i and covering set Q_i that satisfy the following property. Every training example $(\mathbf{x}, y) \in Q_i$ that is either a border point of a previous feature (ball or hole) in the DLM or a center of a previous hole in the DLM must have $y = b_i$ and thus be correctly classified by h_i.

We will see that this constraint will enable us to provide less information to the reconstruction function (to identify a classifier) and will thus yield tighter risk bounds. However, this constraint might, in turn, force **BuildDLM** to produce classifiers containing more features. Hence, we do not know *a priori* which version will produce classifiers having a smaller risk.

Let us first describe the simpler case where only balls are permitted.

As before, we use a string σ_1, with the same probability $P_1(\sigma_1) = 2^{-|\mathbf{i}|}$ $\forall \sigma_1$ to specify if each example of the compression set $\mathbf{z_i}$ is a center or a border point. This gives us the set of centers which coincides with the set of balls since each center can only be used once for this type of DLM.

Next we use a string σ_2 to specify the ordering of each center (or ball) in the DLM. As before we assign equal *a priori* probability to each possible ordering. Hence $P_2(\sigma_2|\sigma_1) = 1/r(\mathbf{z_i})!$ $\forall \sigma_2$ where $r(\mathbf{z_i})$ denotes the number of balls for $\mathbf{z_i}$ (an information given by σ_1).

But now, since each feature was constrained to correctly classify the examples of $\mathbf{z_i}$ that it covers (and which were not covered by the features above), we do not need any additional information to specify the border for each center. Indeed, for this task we use the following algorithm. Given a compression set $\mathbf{z_i}$, let P and N denote, respectively, the set of positive and the set of negative examples in $\mathbf{z_i}$. We start with $P' = P, N' = N$ and do the following, sequentially, from the first center (or ball) to the last. If center $\mathbf{c}$ is positive, then its border $\mathbf{b}$ is given by $\text{argmin}_{\mathbf{x} \in N'} d(\mathbf{c}, \mathbf{x})$ and we remove from P' (to find the border of the other balls) the center $\mathbf{c}$ and all other positive examples covered by that feature and used by the previous features. If center $\mathbf{c}$ is negative, then its border $\mathbf{b}$ is given by $\text{argmin}_{\mathbf{x} \in P'} d(\mathbf{c}, \mathbf{x})$ and we remove from N' the center $\mathbf{c}$ and all other negative examples covered by that feature and used by the previous features.

The distribution of messages is then given by

$$P_{\mathcal{M}(\mathbf{z_i})}(\sigma) = \frac{1}{2^{|\mathbf{i}|}} \cdot \frac{1}{r(\mathbf{z_i})!} \quad \forall \sigma. \tag{20}$$

6.4 Constrained DLMs Containing Balls and Holes

As for the case of Section 6.2, we use a string σ_1 to specify the number $r(\mathbf{z_i})$ of features present in compression set $\mathbf{z_i}$. We also use a string σ_2 to specify, for each feature, if the feature is a ball or a hole. The probabilities $P_1(\sigma_1)$ and $P_2(\sigma_2|\sigma_1)$ used are the same as those defined in Section 6.2. Here, however, we only need to specify the center of each feature, since, as we will see below, no additional information is needed to find the border of each feature when the DLM is constrained to classify correctly each example in $\mathbf{z_i}$. Consequently

$$P_{\mathcal{M}(\mathbf{z_i})}(\sigma) = \zeta(r(\mathbf{z_i})) \cdot 2^{-r(\mathbf{z_i})} \cdot \left[|\mathbf{i}|^{-r(\mathbf{z_i})} I(|\mathbf{i}| \neq 0) + I(|\mathbf{i}| = 0) \right] \quad \forall \sigma. \tag{21}$$

To specify the border of each feature, we use the following algorithm. Given a compression set $\mathbf{z_i}$, let P and N denote, respectively, the set of positive and the set of negative examples in $\mathbf{z_i}$. We start with $P' = P, N' = N$ and do the following, sequentially, from the first feature to the last. If the feature is a ball with a positive center $\mathbf{c}$, then its border is given by $\text{argmin}_{\mathbf{x} \in N'} d(\mathbf{c}, \mathbf{x})$ and we remove from P' the center $\mathbf{c}$ and all other positive examples covered by that feature and used by the previous features. If the feature is a hole with a positive center $\mathbf{c}$, then its border is given by $\text{argmax}_{\mathbf{x} \in P' - \{\mathbf{c}\}} d(\mathbf{c}, \mathbf{x})$ and we remove from N' all the negative examples covered by that feature and used by the previous features. If the feature is a ball with a negative center $\mathbf{c}$, then its border is given by $\text{argmin}_{\mathbf{x} \in P'} d(\mathbf{c}, \mathbf{x})$ and we remove from N' the center $\mathbf{c}$ and all other negative examples covered by that feature and used by the previous features. If the feature is a hole with a negative center $\mathbf{c}$, then its border is given by $\text{argmax}_{\mathbf{x} \in N' - \{\mathbf{c}\}} d(\mathbf{c}, \mathbf{x})$ and we remove from P' all the positive examples covered by that feature and used by the previous features.

6.5 Constrained DLMs with Half-Spaces

Recall that each half-space is specified by weight vector $\mathbf{w}$ and a threshold value t. The weight vector is identified by a pair $(\mathbf{x}_a, \mathbf{x}_b)$ of examples having opposite class labels and the threshold is specified by a third example $\mathbf{x}_c$ of the same class label as example $\mathbf{x}_a$.

The first part of the message will be a string σ_1 that specifies the number $r(\mathbf{z_i})$ of half-spaces used in the compression set $\mathbf{z_i}$. As before, let $p(\mathbf{z_i})$ and $n(\mathbf{z_i})$ denote, respectively, the number of positive examples and the number of negative examples in the compression set $\mathbf{z_i}$. Let $P(\mathbf{z_i})$ and $N(\mathbf{z_i})$ denote, respectively, the set of positive examples and the set of negative examples in the compression set $\mathbf{z_i}$. From these definitions, each pair $(\mathbf{x}_a, \mathbf{x}_b) \in P(\mathbf{z_i}) \times N(\mathbf{z_i}) \cup N(\mathbf{z_i}) \times P(\mathbf{z_i})$ can provide one weight vector. Moreover, since a half-space may not cover any point used for its construction, each weight vector may be used for several half-spaces in the DLM. But half-spaces having the same weight vector $\mathbf{w}$ must have a different threshold since, otherwise, they would cover the same set of examples. Hence the total number of half-spaces in the DLM is at most $|\mathbf{i}|p(\mathbf{z_i})n(\mathbf{z_i})$. Therefore, for the string σ_1 that specifies the number $r(\mathbf{z_i})$ of half-spaces used in the compression set $\mathbf{z_i}$, we could assign the same probability to each number between zero and $|\mathbf{i}|p(\mathbf{z_i})n(\mathbf{z_i})$. However, as before, we want to give preference to DLMs having a smaller number of half-spaces. Hence we choose to assign a probability equal to $\zeta(r)$ (defined by Equation 9) for all possible values of r. Therefore

$$P_1(\sigma_1) = \zeta(r(\mathbf{z_i})) \quad \forall \sigma_1.$$

Next, the second part σ_2 of σ specifies, sequentially for each half-space, the pair $(\mathbf{x}_a, \mathbf{x}_b) \in P(\mathbf{z_i}) \times N(\mathbf{z_i}) \cup N(\mathbf{z_i}) \times P(\mathbf{z_i})$ used for its weight vector $\mathbf{w}$. For this we assign an equal probability of $1/2p(\mathbf{z_i})n(\mathbf{z_i})$ for each possible $\mathbf{w}$ of each half-space. Hence

$$P_2(\sigma_2|\sigma_1) = \left(\frac{1}{2p(\mathbf{z_i})n(\mathbf{z_i})}\right)^{r(\mathbf{z_i})} \quad \forall \sigma_2 \mid n(\mathbf{z_i}) \neq 0 \text{ and } p(\mathbf{z_i}) \neq 0.$$

The condition that $n(\mathbf{z_i}) \neq 0$ and $p(\mathbf{z_i}) \neq 0$ is equivalent to $|\mathbf{i}| \neq 0$ since, for any half-space, $\mathbf{x}_a$ and $\mathbf{x}_b$ must have opposite labels. Hence, more generally, we have

$$P_2(\sigma_2|\sigma_1) = \left(\frac{1}{2p(\mathbf{z_i})n(\mathbf{z_i})}\right)^{r(\mathbf{z_i})} I(|\mathbf{i}| \neq 0) \;+\; I(|\mathbf{i}| = 0) \quad \forall \sigma_2.$$

Finally, as for the other constrained DLMs, we do not need any additional message string to identify the threshold point $\mathbf{x}_c \in \mathbf{z_i}$ for each $\mathbf{w}$ of the DLM. Indeed, for this task we can perform the following algorithm. Let P and N denote, respectively, the set of positive and the set of negative examples in $\mathbf{z_i}$. We start with $P' = P, N' = N$ and do the following, sequentially, from the first half-space to the last. Let $\mathbf{w} = \boldsymbol{\phi}(\mathbf{x}_a) - \boldsymbol{\phi}(\mathbf{x}_b)$ be the weight vector of the current half-space. If $\mathbf{x}_a \in P$ then, for the threshold point $\mathbf{x}_c$, we choose $\mathbf{x}_c = \operatorname*{argmax}_{\mathbf{x} \in N'} \mathbf{w} \cdot \mathbf{x}$ and we remove from P' the positive examples covered by this half-space and used by the previous half-spaces. Else if $\mathbf{x}_a \in N$ then, for the threshold point $\mathbf{x}_c$, we choose $\mathbf{x}_c = \operatorname*{argmax}_{\mathbf{x} \in P'} \mathbf{w} \cdot \mathbf{x}$ and we remove from N' the negative examples covered by this half-space and used by the previous half-spaces.

Consequently, the distribution of message strings is given by

$$P_{\mathcal{M}(\mathbf{z_i})}(\sigma) = \zeta(r(\mathbf{z_i})) \cdot \left[\left(\frac{1}{2p(\mathbf{z_i})n(\mathbf{z_i})}\right)^{r(\mathbf{z_i})} I(|\mathbf{i}| \neq 0) \;+\; I(|\mathbf{i}| = 0) \right] \quad \forall \sigma. \tag{22}$$

6.6 Unconstrained DLMs with Half-Spaces

As for the case of Section 6.5, we use a string σ_1 to specify the number $r(\mathbf{z_i})$ of half-spaces present in compression set $\mathbf{z_i}$. We also use a string σ_2 to specify, sequentially for each half-space, the pair $(\mathbf{x}_a, \mathbf{x}_b) \in P(\mathbf{z_i}) \times N(\mathbf{z_i}) \cup N(\mathbf{z_i}) \times P(\mathbf{z_i})$ used for its weight vector $\mathbf{w}$. Hence, the probabilities $P_1(\sigma_1)$ and $P_2(\sigma_2|\sigma_1)$ used are the same as those defined in Section 6.5. But here, in addition, we need to specify the threshold point $\mathbf{x}_c$ for each $\mathbf{w}$. For this, we give an equal probability of $1/|\mathbf{i}|$ to each example in $\mathbf{z_i}$ of being chosen (when $|\mathbf{i}| \neq 0$). Consequently, the distribution of messages is given by

$$P_{\mathcal{M}(\mathbf{z_i})}(\sigma) = \zeta(r(\mathbf{z_i})) \cdot \left[\left(\frac{1}{2p(\mathbf{z_i})n(\mathbf{z_i})}\right)^{r(\mathbf{z_i})} |\mathbf{i}|^{-r(\mathbf{z_i})} I(|\mathbf{i}| \neq 0) \;+\; I(|\mathbf{i}| = 0) \right] \quad \forall \sigma. \tag{23}$$

7. Empirical Results on Natural Data

We have tested the DLM on several “natural” data sets which were obtained from the machine learning repository at UCI. For each data set, we have removed all examples that contained attributes with unknown values and we have removed examples with contradictory labels (this occurred only for a few examples in the Haberman data set). The remaining number of examples for each data set is reported in Table 3. No other preprocessing of the data (such as scaling) was performed. For all these data sets, we have used the 10-fold cross-validation error as an estimate of the generalization error. The values reported are expressed as the total number of errors (*i.e.* the sum of errors over all testing sets). We have ensured that each training set and each testing set, used in the 10-fold cross validation process, was the same for each learning machine (*i.e.* each machine was trained on the same training sets and tested on the same testing sets).

Table 1 and Table 2 show the DLM sizes s and penalty values that gave the smallest 10-fold cross-validation error separately for the following types of DLMs that we have studied in Section 6:

DLM$_b$: unconstrained DLMs with balls (only).

DLM$^*_b$: constrained DLMs with balls (only).

DLM$_{bh}$: unconstrained DLMs with balls and holes.

DLM$^*_{bh}$: constrained DLMs with balls and holes.

DLM$_{hsp}$: unconstrained DLMs with half-spaces.

DLM$^*_{hsp}$: constrained DLMs with half-spaces.

For each of these DLMs, the learning algorithm used was **BuildDLM**. We have observed that **PruneDLM** had no effect on all these data sets, except for Credit where it was sometimes able to remove one feature.

In Table 3, we have compared the performance of the DLM with the set covering machine (SCM) using the same sets of data-dependent features, and the support vector machine (SVM) equipped with a radial basis function kernel of variance $1/\gamma$ and a soft-margin parameter C.

We have used the L_2 metric for the data-dependent features for both DLMs and SCMs to obtain a fair comparison with SVMs. Indeed, the argument of the radial basis function kernel is given by the L_2 metric between two input vectors. For the SVM, the values of s refer to the average number of support vectors obtained from the 10 different training sets of 10-fold cross-validation. For the SCM, the value of T indicates the type of features it used and whether the SCM was a conjunction (c) or a disjunction (d). The values of p and s for the SCM refer to the penalty value and the number of features that gave the smallest 10-fold cross-validation error. We emphasize that, for all learning machines, the values of the learning parameters reported in Tables 1, 2, and 3 are the ones that gave the smallest 10-fold cross-validation error when chosen among a very large list of values. Although this overestimates the performance of every learning algorithm, it was used here to compare equally fairly (or equally unfairly) every learning machine. We will report below the results for DLMs when the testing sets are not used to determine the best values of the learning parameters.

In addition to our estimate of the generalization error, we have also reported in Table 3, a (rough) estimate of the standard deviation of the error. This estimate was obtained in the following way. We first compute the standard deviation of the generalization error (per example) over the 10 different

Data Set	DLM_b		DLM_b^*		DLM_{bh}		DLM_{bh}^*	
	$\{p_p,p_n,s\}$	err	$\{p_p,p_n,s\}$	err	$\{p_p,p_n,s\}$	err	$\{p_p,p_n,s\}$	err
BreastW	0.7, 2.1, 6	18	0.7, 0, 1	18	2, 1, 2	13	1.6, 1, 2	13
Bupa	1.5, 3.7, 14	104	2.5, 1.5, 21	107	2, 2.1, 4	110	2.5, 1.5, 17	107
Credit	1.7, 2.1, 6	188	0.7, 0.3, 42	195	2.1, 1.4, 11	187	1.3, ∞, 33	195
Glass	2.5, ∞, 8	33	2.5, 3.5, 8	32	4, 4.5, 7	29	∞, 3.7, 7	29
Haberman	∞, 4.5, 23	74	0.5, 4, 3	70	3.7, 3.4, 12	64	1.7, 3.7, 6	65
Heart	1.7, 2.7, 17	94	1.6, 2.3, 8	89	1.5, 2.6, 10	95	2, 2, 9	101
Pima	1.5, 2.2, 5	184	2.5, 2.6, 74	184	1, 1.5, 2	190	1, 1.5, 6	189
Votes	2.5, 1.5, 6	34	2, ∞, 14	36	4.5, ∞, 14	35	1.8, ∞, 23	37

Table 1: Optimal 10-fold cross-validation results for DLMs with balls (and holes).

Data Set	DLM_{hsp}		DLM_{hsp}^*	
	$\{p_p,p_n,s\}$	err	$\{p_p,p_n,s\}$	err
BreastW	1, ∞, 1	18	1.7,∞, 1	20
Bupa	2.7,1.5,15	107	0.9,2,6	102
Credit	3.5,2.5,26	141	1.9,1.5,7	151
Glass	2.1,0.7,4	35	2,1.3,4	37
Haberman	1.8,3,5	66	1.5,1.1,2	70
Heart	0.8,1,1	85	0.8,0.5,3	83
Pima	1.7,1, 4	169	1.9,2,8	183
Votes	4.4, ∞, 13	24	3.3, ∞, 13	33

Table 2: Optimal 10-fold cross-validation results for DLMs with half-spaces.

testing sets and then divide by $\sqrt{10}$ (since the variance of the average of n iid random variables, each with variance σ^2, is σ^2/n). Finally we multiply this estimate by the number of examples in the data set.

In terms of generalization error, there is no substantial difference among all the types of DLMs presented in Tables 1 and 2 for most of the data sets—except for Credit where DLMs with half-spaces have a significantly lower error rate.

From the results in Tables 1 and 2, we notice that the effect of constraining the DLM to correctly classify the compression set[3] generally increases the size of the DLM. The increase is substantial for the Credit and Pima data sets (except for half-spaces where the opposite behavior is observed). It is surprising that a DLM_b^* with 74 balls has the same error rate as a DLM_b with 5 balls on the Pima data set. In contrast, constraining an SCM to correctly classify the compression set had virtually no effect on the size of the classifier (for these data sets).

The most striking feature in all the results is the level of sparsity achieved by the SCM and the DLM in comparison with the SVM. This difference is always huge. The other important feature is that DLMs often produce slightly better generalization than SCMs and SVMs. Hence, DLMs can provide a good alternative to SCMs and SVMs.

Recall that the results reported in Tables 1, 2, and 3 are, in fact, the 10-fold cross validation estimate of the generalization error that is achieved by the model selection strategy that correctly guesses the best values for p_p, p_n and s. This model-selection strategy is, in that sense, optimal (but not realizable). Hence, we refer to the scores obtained in theses Tables as those obtained by the

3. Recall from Section 6 that this was done to obtain a tighter risk bound.

Data Set		SVM		SCM			DLM		
Name	exs	{ γ,C,s }	err	T	{ p,s }	err$\pm\sigma$	T	{ p_p,p_n,s }	err$\pm\sigma$
BreastW	683	0.005, 2, 58	19	b*,c	1.8, 2	15$\pm$3.9	bh*	1.6, 1, 2	**13**$\pm$3.4
Bupa	345	0.002, 0.2, 266	107	hsp*,c	1.4,1	103 $\pm$ 6.2	hsp*	0.9,2,6	**102** $\pm$7.8
Credit	653	0.0006, 32, 423	190	hsp*,d	1.2, 3	148$\pm$10.2	hsp	3.5,2.5,26	**141**$\pm$13.5
Glass	214	0.8, 1.2, 130	34	b*,d	∞, 3	36$\pm$6.3	bh*	∞, 3.7, 7	**29**$\pm$6.7
Haberman	294	0.01, 0.6, 146	71	hsp*,d	0.7,1	68$\pm$ 5.9	bh	3.7, 3.4, 12	**64**$\pm$3.9
Heart	297	0.001, 2, 204	87	hsp*,d	1.3,1	87$\pm$7.1	hsp*	0.8,0.5,3	**83**$\pm$6.9
Pima	768	0.002, 1, 526	203	hsp*,c	1.5, 3	175$\pm$4.9	b	1.7,1,4	**169**$\pm$5
Votes	435	0.2, 1.7, 125	**22**	hsp*, d	∞,13	34$\pm$5.8	hsp	4.4,∞,13	24$\pm$5.2

Table 3: Optimal 10-fold cross-validation results for SVMs, SCMs, and DLMs.

"optimal" model-selection strategy. To investigate the extent to which a bound can perform model selection, we use the proposed risk bound to select a DLM among those obtained for a list of at least 1000 pairs of penalty values (which always included the optimal pair of penalty values) and for all possible sizes s. We have compared these results with the K-fold cross-validation model selection method. This latter method, widely used in practice, consists of using K-fold cross-validation to find the best stopping point s and the best penalty values p_p and p_n (among the same list of values used for the previous model selection method) on a given training set and then use these best values on the full training set to find the best DLM. Both model selection methods were tested with K-fold cross-validation. The results are reported in Tables 4, 5 and 6 for all the types of DLMs that we have considered in Section 6. In these tables, "MSfromCV" stands for "model selection from cross-validation" and "MSfromBound" stands for "model selection from bound". For all these results we have used $K = 10$, except for DLM$^*_{hsp}$ where we have used $K = 5$. Except for a few cases, we see that model selection by using the bounds of Section 6 is generally slightly more effective than using K-fold cross validation (and takes substantially less computation time).

Data Set	DLM$_b$				DLM$^*_b$			
	MSfromCV		MSfromBound		MSfromCV		MSfromBound	
	s	err$\pm\sigma$	s	err$\pm\sigma$	s	err$\pm\sigma$	s	err$\pm\sigma$
BreastW	1.8	18$\pm$5.01	2	18$\pm$4.4	1	20 $\pm$4.9	10	20$\pm$6.4
Bupa	15.3	123$\pm$8.5	15.3	112$\pm$5.6	15.3	115$\pm$7.6	27	117$\pm$7.3
Credit	25.7	231$\pm$ 12.3	20.2	194$\pm$5.6	3.5	203 $\pm$13.2	6.8	215$\pm$20.2
Glass	7.2	44$\pm$ 4.1	15.9	36$\pm$6.5	8.1	44$\pm$7.9	14.1	34$\pm$6.1
Haberman	6.3	80$\pm$9.8	37	89$\pm$15.1	3.8	89 $\pm$13.9	13.4	104$\pm$7.8
Heart	8.7	104$\pm$8.5	20.6	97$\pm$6	10.8	103$\pm$8.5	30.6	95$\pm$8.7
Pima	25.5	213$\pm$10.5	13.7	198$\pm$10.2	28.2	230$\pm$10.4	50	192$\pm$9.3
Votes	13.4	48$\pm$6.3	7.2	38$\pm$8.6	13.4	48$\pm$6.3	16.5	40$\pm$7.3

Table 4: Model selection results for DLMs with balls (only).

8. Conclusion and Open Problems

We have introduced a new learning algorithm for decision lists and have shown that it can provide a favorable alternative to the SCM on some "natural" data sets. Compared with SVMs, the proposed

Data Set	DLM_{bh}				DLM^*_{bh}			
	MSfromCV		MSfromBound		MSfromCV		MSfromBound	
	s	err$\pm\sigma$	s	err$\pm\sigma$	s	err$\pm\sigma$	s	err$\pm\sigma$
BreastW	1.8	15$\pm$3.7	2.2	14$\pm$4.1	1.8	18$\pm$4.9	2.3	16$\pm$4.9
Bupa	8.6	124$\pm$7.9	12.9	119$\pm$7.7	15.2	123$\pm$8.4	23.7	118$\pm$8.2
Credit	6.3	209$\pm$14.7	17.6	206$\pm$11.5	25.7	231$\pm$12.1	40.9	208$\pm$9.5
Glass	7.4	37$\pm$8.4	13.4	35$\pm$6.4	7.2	44$\pm$4.6	21.7	30$\pm$6
Haberman	6	74$\pm$3	23.6	68$\pm$4.4	6.3	80$\pm$9.6	20.1	65$\pm$6.1
Heart	9.1	112$\pm$8.2	14.9	104$\pm$6.6	10.9	103$\pm$10.5	24.1	105$\pm$6.4
Pima	2.8	204$\pm$8.9	4	212$\pm$9.5	7.8	212$\pm$8.7	11.1	203$\pm$11.2
Votes	13.5	44$\pm$6.8	19.5	35$\pm$4.8	11.8	48$\pm$5.4	14.5	40$\pm$7.4

Table 5: Model selection results for DLMs with balls and holes.

Data Set	DLM_{hsp}				DLM^*_{hsp}			
	MSfromCV		MSfromBound		MSfromCV		MSfromBound	
	s	err$\pm\sigma$	s	err$\pm\sigma$	s	err$\pm\sigma$	s	err$\pm\sigma$
BreastW	1.6	22$\pm$4.6	7	18$\pm$6	8.2	20$\pm$2.6	9	18 $\pm$ 6.9
Bupa	3.1	117$\pm$2.9	11.8	117$\pm$6.2	3.2	110$\pm$3.7	5.2	117 $\pm$ 9.3
Credit	17.7	152$\pm$13.9	28.3	152 $\pm$17.8	6.4	165$\pm$9.2	7.6	163 $\pm$ 10
Glass	2.2	39$\pm$6	2.9	34$\pm$6.3	2.8	38$\pm$4.7	3.4	34 $\pm$ 5.9
Haberman	5.8	78$\pm$5.8	11.2	68$\pm$8.9	2.4	69$\pm$2.3	4	68 $\pm$ 3.4
Heart	1.6	87$\pm$ 8	3.6	89$\pm$8.2	1.6	99$\pm$3.5	3	120 $\pm$ 2.2
Pima	2.7	178$\pm$11.2	4.3	182 $\pm$12.5	10.5	187$\pm$6.7	15.8	175 $\pm$ 12.7
Votes	9.5	32$\pm$5.4	16.3	26$\pm$5.4	5.6	38$\pm$3.3	13.8	33 $\pm$ 4.3

Table 6: Model selection results for DLMs with half-spaces.

learning algorithm for DLMs produces substantially sparser classifiers with comparable, and often better, generalization.

We have proposed a general risk bound that depends on the number of examples that are used in the final classifier and the size of the information message needed to identify the final classifier from the compression set. The proposed bound is significantly tighter than the one provided by Littlestone and Warmuth (1986) and Floyd and Warmuth (1995) and applies to any compression set-dependent distribution of messages. We have applied this general risk bound to DLMs by making appropriate choices for the compression set-dependent distribution of messages and have shown, on natural data sets, that these specialized risk bounds are generally slightly more effective than K-fold cross validation for selecting a good DLM model.

The next important step is to find risk bounds that hold for asymmetrical loss functions. Indeed, this is the type of loss function which is most appropriate for many natural data sets and we cannot use, in these circumstances, the risk bounds proposed here since they are valid only for the symmetric loss case. Other important issues are the investigation of other metrics and other data-dependent sets of features.

This paper shows that it is sometimes worthwhile to use a decision list of data-dependent features instead of a conjunction or a disjunction of the same set of features. Hence, we may ask if it is worthwhile to consider the larger class of linear threshold functions. With data-dependent features, we want to use (or adapt) algorithms that are efficient when irrelevant features abound. In these cases, the winnow (Littlestone, 1988) and the multi-layered winnow (Nevo and El-Yaniv,

2002) algorithms are obvious candidates. However, these algorithms do not return a sparse solution since many features will be assigned a non-negligible weight value. Moreover, our (preliminary) numerical experiments with the winnow algorithm indicate that this algorithm is simply too slow to be used with $O(m^2)$ features for $m \geq 700$. More generally, we think that this research direction is not attractive in view of the fact that it is (generally) very hard to find a linear threshold function with few non-zero weight values.

Acknowledgments

Work supported by NSERC Discovery grant 122405 - 2002 and NSERC doctoral scholarship 266896 - 2003. We thank John Shawe-Taylor for his comments concerning the proof of Theorem 1. We thank two of the reviewers for many helpful comments.

References

Martin Anthony. Generalization error bounds for threshold decision lists. *Journal of Machine Learning Research*, 5:189–217, 2004.

Avrim Blum and Mona Singh. Learning functions of k terms. In *COLT '90: Proceedings of the third annual workshop on Computational learning theory*, pages 144–153. Morgan Kaufmann Publishers Inc., 1990. ISBN 1-55860-146-5.

Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Occam's razor. *Information Processing Letters*, 24:377–380, 1987.

Aditi Dhagat and Lisa Hellerstein. PAC learning with irrelevant attributes. In *Proc. of the 35rd Annual Symposium on Foundations of Computer Science*, pages 64–74. IEEE Computer Society Press, Los Alamitos, CA, 1994.

Andrzej Ehrenfeucht and David Haussler. Learning decision trees from random examples. *Information and Computation*, 82:231–246, 1989.

Thomas Eiter, Toshihide Ibaraki, and Kazuhiso Makino. Decision lists and related boolean functions. *Theoretical Computer Science*, 270:493–524, 2002.

Sally Floyd and Manfred K. Warmuth. Sample compression, learnability, and the Vapnik-Chervonenkis dimension. *Machine Learning*, 21(3):269–304, 1995.

Thore Graepel, Ralf Herbrich, and John Shawe-Taylor. Generalisation error bounds for sparse linear classifiers. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, pages 298–303, 2000.

Thore Graepel, Ralf Herbrich, and Robert C. Williamson. From margin to sparsity. In *Advances in neural information processing systems*, volume 13, pages 210–216, 2001.

Ralf Herbrich. *Learning Kernel Classifiers*. MIT Press, Cambridge, Massachusetts, 2002.

Geoffrey Hinton and Michael Revow. Using pairs of data-points to define splits for decision trees. *Advances in Neural Information Processing Systems 8*, pages 507–513, 1996.

Jyrki Kivinen, Heikki Mannila, and Esko Ukkonen. Learning hierarchical rule sets. In *Proceedings of the fifth annual ACM workshop on Computational Learning Theory*, pages 37–44. Morgan Kaufmann, 1992.

Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4):285–318, 1988.

Nick Littlestone and Manfred K. Warmuth. Relating data compression and learnability. Technical report, University of California Santa Cruz, 1986.

Mario Marchand and Mostefa Golea. On learning simple neural concepts: from halfspace intersections to neural decision lists. *Network: Computation in Neural Systems*, 4:67–85, 1993.

Mario Marchand, Mohak Shah, John Shawe-Taylor, and Marina Sokolova. The set covering machine with data-dependent half-spaces. In *Proceedings of the Twentieth International Conference on Machine Learning(ICML'2003)*, pages 520–527. Morgan Kaufmann, 2003.

Mario Marchand and John Shawe-Taylor. Learning with the set covering machine. *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001)*, pages 345–352, 2001.

Mario Marchand and John Shawe-Taylor. The set covering machine. *Journal of Machine Learning Reasearch*, 3:723–746, 2002.

Shahar Mendelson. Rademacher averages and phase transitions in Glivenko-Cantelli class. *IEEE Transactions on Information Theory*, 48:251–263, 2002.

Ziv Nevo and Ran El-Yaniv. On online learning of decision lists. *Journal of Machine Learning Reasearch*, 3:271–301, 2002.

Ronald L. Rivest. Learning decision lists. *Machine Learning*, 2:229–246, 1987.

Marina Sokolova, Mario Marchand, Nathalie Japkowicz, and John Shawe-Taylor. The decision list machine. In *Advances in Neural Information Processing Systems(NIPS'2002)*, volume 15, pages 921–928. The MIT Press, 2003.

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

Journal of Machine Learning Research 6 (2005) 453–482 Submitted 6/04; Published 4/05

Estimating Functions for Blind Separation When Sources Have Variance Dependencies

Motoaki Kawanabe NABE@FIRST.FHG.DE
Fraunhofer FIRST.IDA
Kekuléstrasee 7
12489 Berlin, Germany

Klaus-Robert Müller KLAUS@FIRST.FHG.DE
Fraunhofer FIRST.IDA
Kekuléstrasse 7
12489 Berlin, Germany and
Department of Computer Science
University of Potsdam
August-Bebel-Strasse 89
14482 Potsdam, Germany

Editor: Aapo Hyvärinen

Abstract

A blind separation problem where the sources are not independent, but have variance dependencies is discussed. For this scenario Hyvärinen and Hurri (2004) proposed an algorithm which requires no assumption on distributions of sources and no parametric model of dependencies between components. In this paper, we extend the semiparametric approach of Amari and Cardoso (1997) to variance dependencies and study estimating functions for blind separation of such dependent sources. In particular, we show that many ICA algorithms are applicable to the variance-dependent model as well under mild conditions, although they should in principle not. Our results indicate that separation can be done based only on normalized sources which are adjusted to have stationary variances and is not affected by the dependent activity levels. We also study the asymptotic distribution of the quasi maximum likelihood method and the stability of the natural gradient learning in detail. Simulation results of artificial and realistic examples match well with our theoretical findings.

Keywords: blind source separation, variance dependencies, independent component analysis, semiparametric statistical models, estimating functions

1. Introduction

Blind methods of source separation have been successfully applied to many areas of science (e.g. Hyvärinen et al., 2001b; Olshausen and Field, 1996; Makeig et al., 1997; Vigario, 1997; Ziehe et al., 2000; Thi and Jutten, 1995; Cardoso, 1998a; Parra and Spence, 2000; Cardoso, 2003; Meinecke et al., 2005). The basic model assumes that the observed signals are linear superpositions of underlying hidden source signals. Let us denote the n source signals by the vector

$s(t) = (s_1(t), \ldots, s_n(t))^\top$, and the observed signals by $x(t) = (x_1(t), \ldots, x_m(t))^\top$. In this paper,[1] we will focus on real-valued signals. The mixing can be expressed as the equation

$$x(t) = As(t),$$

where $A = (a_{ij})$ denotes the mixing matrix. For simplicity, we consider the case where the number of source signals equals that of observed signals ($n = m$). Both the sources $s(t)$ and the mixing matrix A are unknown, and the goal is to estimate them based on the observation $x(t)$ alone.

In most blind source separation (BSS) methods, the source signals are assumed to be statistically independent. Blind source separation based on such a model is called independent component analysis (ICA). By using non-Gaussianity of the sources, the mixing matrix can be estimated and the source signals can be extracted under appropriate conditions. There are also further approaches of BSS, that are, for example, based on second-order statistics and algorithms exploiting nonstationarity. The second-order methods are applicable to the case where the source signals have (lagged) auto-correlation. Provided that components have nonstationary, smoothly changing variances, the model can be estimated as well by algorithms based on nonstationarity of signals.

Among many extensions of the basic ICA models, several researchers have studied the case where the source signals are not independent (for example, Cardoso, 1998b; Hyvärinen et al., 2001a; Bach and Jordan, 2002; Valpola et al., 2003, see also references in Hyvärinen and Hurri, 2004). The dependencies either need to be exactly known beforehand, or they are simultaneously estimated by the algorithms. Recently, a novel idea called double-blind approach was introduced by Hyvärinen and Hurri (2004). In contrast to previous work, their method requires no assumption on the distributions of the sources and no parametric model of the dependencies between the components. They simply assume that the sources are dependent only through their variances and that the sources have temporal correlation. In the Topographic ICA (Hyvärinen et al., 2001a), the dependencies of the sources are also caused only by their variances, but in contrast to the double blind case, they are determined by a prefixed neighborhood relation. It should be noted that for such dependent component models identifiability results have not been theoretically established so far, while identifiability of multidimensional ICA was proven by Theis (2004).

A statistical basis of ICA was established by Amari and Cardoso (1997). They pointed out that the ICA model is an example of semiparametric statistical models (Bickel et al., 1993; Amari and Kawanabe, 1997a,b) and studied estimating functions for it. In particular, they showed that the quasi maximum likelihood (QML) estimation and the natural gradient learning give a correct solution regardless of the true source densities which satisfy certain mild conditions. In this paper, we extend their approach to the BSS problem considered in Hyvärinen and Hurri (2004). Investigating estimating functions for the model, we show that many of ICA algorithms based on the independence assumption can achieve consistent solutions in a local sense, even if there exist variance dependencies, which is astonishing and seems somewhat counterintuitive. We remark that estimating functions are concerned with local consistency ('consistency' will denote its local version in the following) and in general have spurious solutions. For a few algorithms, even global consistency has been proven by different principles (for example, Hyvärinen and Hurri, 2004). Nevertheless, our result goes beyond existing ones, because it covers most types of BSS algorithms and can give asymptotic distributions. The main message of this paper is that most ICA algorithms can be proven to be consistent in our framework *although* the data is *not* independent. So they must effectively

1. This is an extended version of Kawanabe and Müller (2004) presented at ICA2004.

use some concept beyond independence. Thus our consistency results indicate that separation can be done based only on normalized sources which are adjusted to have stationary variances and is not affected by the dependent activity levels.

This paper is organized as follows. At first, we define the variance-dependent model in Section 2 and explain estimating functions, a useful tool for discussing semiparametric estimators in Section 3. In Section 4, we discuss the relation between estimating functions for the ICA model and those for the variance-dependent BSS model in general. It is shown that these algorithms work properly, *even if* there exist spatiotemporal variance dependencies. Among several ICA algorithms, the quasi maximum likelihood method and its online version, the natural gradient learning are discussed in detail. We study the asymptotic distributions of the quasi maximum likelihood method (Section 5.1) and the stability of the natural gradient learning (Section 5.2). We also give a brief summary about several other ICA algorithms from our viewpoint in Section 5.3. Detailed discussion can be found in Appendix A. The theoretical insights are underlined by several numerical simulations in Section 6. In particular, we carried out two experiments, where we extract two speech signals with high variance dependencies. It is sometimes believed that ICA algorithms work for mixture of acoustic signals or natural images because the data are sparse and often disjoint. Our results show that they can also separate even highly coherent signals, and our theoretical analysis can thus help to understand the reason.

2. Variance-Dependent BSS Model

Hyvärinen and Hurri (2004) formalized the probabilistic framework of variance-dependent blind separation. Let us assume that each source signal $s_i(t)$ is a product of non-negative activity level $v_i(t)$ and underlying i.i.d. signal $z_i(t)$, that is,

$$s_i(t) = v_i(t) z_i(t). \tag{1}$$

We remark that the sequences of the vectors $s = (s_1, \dots, s_n)^\top$, $v = (v_1, \dots, v_n)^\top$ and $z = (z_1, \dots, z_n)^\top$ are considered as multivariate random processes in this paper. In practice, the activity levels $v_i(t)$ are often dependent among different signals and each observed signal is expressed as

$$x_i(t) = \sum_{j=1}^{n} a_{ij} v_j(t) z_j(t), \qquad i = 1, \dots, n,$$

where $v_i(t)$ and $z_i(t)$ satisfy:

(i) $v_i(t)$ and $z_j(t')$ are independent for all i, j, t and t',

(ii) each $z_i(t)$ is i.i.d. in time for all i, the random vector $z = (z_1, \dots, z_n)^\top$ is mutually independent,

(iii) $z_i(t)$ have zero mean and unit variance for all i.

No assumption on the distribution of z_i is made except (iii). Regarding the general activity levels v_i's, $v_i(t)$ and $v_j(t)$ are allowed to be statistically dependent, and furthermore, no particular assumption on these dependencies is made (double blind situation). We refer to this framework as the variance-dependent BSS model in this paper. Figure 1 shows an example of the sources s used in the model. As stated in the assumption (ii) above, the normalized signals z_1 and z_2 are mutually independent.

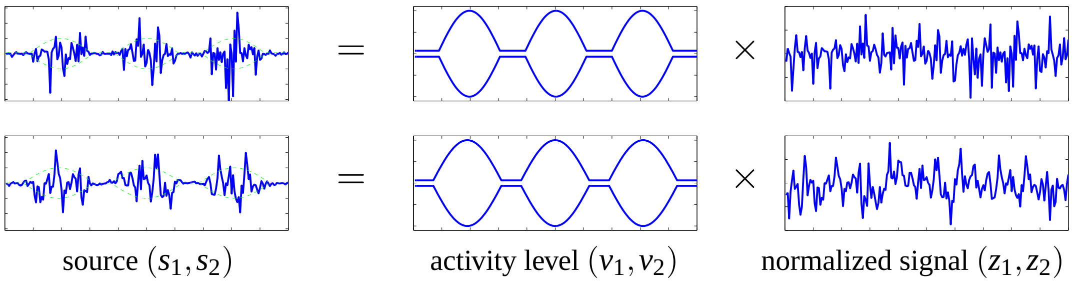


Figure 1: Sources (s_1, s_2) with variance dependencies in the variance-dependent BSS model. In the middle panels both v_i and $-v_i$ are plotted for clarity.

However, since the sequences z_1 and z_2 are multiplied by extremely dependent activity levels v_1 and v_2, respectively, the short-term variance of the source signals s_1 and s_2 are highly correlated.

Hyvärinen and Hurri (2004) proposed an algorithm which maximizes the objective function

$$J(W) = \sum_{i,j} [\,\widehat{\text{cov}}([w_i^\top u(t)]^2, [w_j^\top u(t-\Delta t)]^2)\,]^2,$$

where $\widehat{\text{cov}}$ denotes the sample covariance, $W = (w_1, \ldots, w_n)^\top$ is constrained to be orthogonal and where $u(t)$ is obtained by preprocessing the signal $x(t)$ by spatial whitening. It was proved that the objective function J is maximized when WA equals a signed permutation matrix, if the matrix $K = (K_{ij}) = (\,\text{cov}\{s_i^2(t), s_j^2(t-\Delta t)\}\,)$ is of full rank. This method shows good performance as long as there exist temporal variance dependencies and the data is not spoiled by outliers (see Meinecke et al., 2004).

It is important to remark that the nonstationary algorithm by Pham and Cardoso (2000) was also designed for the same source model (1), except that $v_i(t)$'s are assumed to be deterministic and slowly varying. However, it is straightforward to show validity of this algorithm, when $v_i(t)$'s are (slowly-varying) random sequences.

3. Semiparametric Statistical Models and Estimating Functions

Amari and Cardoso (1997) established a statistical basis of the ICA problem. They pointed out that the standard ICA model[2]

$$p(X|B, \rho_s) = |\det B|^T \prod_{t=1}^{T} \prod_{i=1}^{n} \rho_{s_i}\{b_i^\top x(t)\} \tag{2}$$

is an example of semiparametric statistical models (Bickel et al., 1993; Amari and Kawanabe, 1997a,b), where $B = (b_1, \ldots, b_n)^\top = A^{-1}$ is the demixing matrix to be estimated and $\rho_s(s) = \prod_{i=1}^{n} \rho_{s_i}(s_i)$ is the density of the sources s. Notations used in the following sections are also summarized in Table 1. As the function ρ_s in this model, semiparametric statistical models contain infinite dimensional or functional nuisance parameters which are difficult to estimate. Moreover, they even disturb inference on parameters of interest.

2. Since the sources are assumed to be i.i.d. in time, people consider the distribution of one sample x instead of the entire sequence X.

$x(t) = (x_1(t), \ldots, x_n(t))^\top$	observed data at t
$X = (x(1), \ldots, x(T))$	whole sequence of the observed data
$s(t) = (s_1(t), \ldots, x_n(t))^\top$	source signals at t
$v(t) = (v_1(t), \ldots, v_n(t))^\top$	general activity levels of the sources $s(t)$
$V = (v(1), \ldots, v(T))$	whole sequence of the activity levels
$z(t) = (z_1(t), \ldots, z_n(t))^\top$	normalized source signals by the activity levels $v(t)$
A	$n \times n$ mixing matrix
$B = (b_{ij}) = (b_1, \ldots, b_n)^\top$	demixing matrix which is equivalent to A^{-1}
$\rho_z(z) = \prod_{i=1}^{n} \rho_{z_i}(z_i)$	density of the normalized source signals z
$\rho_V(V)$	density of the entire sequence $V = (v(1), \ldots, v(T))$ of the activity levels
$y(t) = Bx(t)$	extracted sources by the demixing matrix B
$F(x, B)$ or $\bar{F}(X, B)$	estimating function which is an $n \times n$ matrix-valued function of the data and the parameter B
$\mathrm{vec}(F)$ $= (F_{11}, \ldots, F_{n1}, \ldots, F_{1n}, \ldots, F_{nn})^\top$	vectorization operator

Table 1: List of notations used in the variance-dependent BSS model

In the variance-dependent BSS model which we consider, the sources $s(t)$ are decomposed of two components, the normalized signals $z(t) = (z_1(t), \ldots, z_n(t))^\top$ and the general activity levels $v(t) = (v_1(t), \ldots, v_n(t))^\top$. Since the former has mutual independence like the ICA model, the density of the data X is factorized as

$$p(X|V; B, \rho_z) = |\det B|^T \prod_{t=1}^{T} \prod_{i=1}^{n} \frac{1}{v_i(t)} \, \rho_{z_i} \left\{ \frac{b_i^\top x(t)}{v_i(t)} \right\}, \tag{3}$$

when $V = (v(1), \ldots, v(T))$ is fixed. Therefore, the marginal distribution can be expressed as

$$p(X|B, \rho_z, \rho_V) = \int p(X|V; B, \rho_z) \rho_V(V) \mathrm{d}V, \tag{4}$$

where the density ρ_V of V becomes an extra nuisance function.

In order to construct valid estimators for such semiparametric models, estimating functions were introduced by Godambe (1976). Let us consider a general semiparametric model $p(x|\theta, \rho)$, where θ is an r-dimensional parameter of interest and ρ is a nuisance parameter. An r-dimensional vector valued function $f(x, \theta)$ is called an estimating function, when it satisfies the following conditions for any θ and ρ (Godambe, 1991),

$$\begin{aligned}
&\mathrm{E}[\, f(x,\theta)\ |\theta,\rho] = \mathbf{0}, \\
&|\det Q| \neq 0, \qquad \text{where } Q = \mathrm{E}\left[\frac{\partial}{\partial\theta} f(x,\theta)\ \middle|\, \theta,\rho\right], \\
&\mathrm{E}\left[\ \| f(x,\theta)\|^2\ \middle|\, \theta,\rho\right] < \infty,
\end{aligned}$$

where $\mathrm{E}[\cdot|\theta,\rho]$ denotes the expectation over x with the density $p(x|\theta,\rho)$ and $\|\cdot\|$ is the Euclidean norm. Suppose i.i.d. samples $x(1),\dots,x(T)$ are obtained from the model $p(x|\theta^*,\rho^*)$. If such a function exists, by solving the estimating equation

$$\sum_{t=1}^{T} f(x(t),\widehat{\theta}) = \mathbf{0}, \tag{5}$$

we can get an estimator $\widehat{\theta}$ with good asymptotic property. Such an estimator that is a solution of an estimating equation as (5) is called an M-estimator in statistics (Huber, 1981). It can be regarded as an extension of the maximum likelihood method for parametric models. The M-estimator $\widehat{\theta}$ is consistent regardless of the true nuisance parameter ρ^*, when the sample size T goes to infinity. Moreover, it is asymptotically Gaussian distributed, that is, $\widehat{\theta} \sim N(\theta^*, \mathrm{Av})$, where Av denotes the asymptotic variance computed by the following equation

$$\mathrm{Av} = \mathrm{Av}(\theta^*,\rho^*) = \frac{1}{T} Q^{-1}\, \mathrm{E}\left[\ f(x,\theta)\, f^{\top}(x,\theta)\ \middle|\, \theta^*,\rho^*\right]\ (Q^{-1})^{\top},$$

and $Q = Q(\theta^*,\rho^*) = \mathrm{E}\left[\ \frac{\partial}{\partial\theta} f(x,\theta)\ \middle|\, \theta^*,\rho^*\right]$. We remark that the asymptotic variance Av depends on the true parameters (θ^*,ρ^*), but not on the data $x(1),\dots,x(T)$. As we will explain in Section 4.2, notions of estimating functions and M-estimators were extended to non i.i.d cases.

Although estimating functions are useful for semiparametric models, it is non-trivial to find such functions. Amari and Kawanabe (1997a,b) studied this problem from a geometrical point of view and gave a guideline for discussing estimating functions.

The asymptotic result guarantees theoretically that the estimator $\widehat{\theta}$ derived from the estimating function converges to the true parameter θ^* under mild conditions. However, we should remark that the asymptotic variance Av of the estimator depends on the true nuisance parameter ρ^*. For example, when the matrix Q is almost singular at ρ^*, it can happen that the asymptotic variance Av becomes very large. This may cause some practical problem, that is, the estimate from finite samples can be no longer close to the true parameter. We will revisit this issue in Section 6.

Furthermore, online algorithms with similar consistency property can also be constructed from estimating functions,

$$\theta_{t+1} = \theta_t - \eta_t\, f(x(t),\theta_t), \tag{6}$$

$$\theta_{t+1} = \theta_t - \eta_t\, R(\theta_t)\, f(x(t),\theta_t), \tag{7}$$

where $R(\theta)$ is an $n\times n$ nonsingular matrix and depends only on θ. We remark that the functions $f(x,\theta)$ and $R(\theta) f(x,\theta)$ give the same estimating equation, if $R(\theta)$ has the inverse matrix and does not depend on the data x. Such functions are called equivalent estimating functions. It is also easy to see that the online algorithms (6) and (7) have the same equilibria points. However, their dynamics are different. The stability of such online learning was investigated by Amari et al. (1997).

4. General Properties of Estimating Functions for Blind Separation

In this section we will at first review estimating functions for the ICA model (2) (see also Amari and Cardoso, 1997; Cardoso, 1997) and then discuss our contribution, that is, by defining estimating functions for the variance-dependent BSS model (3) and (4).

4.1 Estimating Functions for Ordinary Blind Source Separation

In case of the ICA model, the parameter of interest is the $n \times n$ matrix $B = A^{-1}$ and hence it is convenient to write the estimating functions in $n \times n$ matrix form $F(x,B)$. The conditions of estimating functions are reshaped accordingly as

$$\mathrm{E}[F(x,B)\;|B,\rho_s] = 0, \tag{8}$$

$$|\det Q| \neq 0, \qquad \text{where } Q = \mathrm{E}\left[\left.\frac{\partial \mathrm{vec}\{F(x,B)\}}{\partial \mathrm{vec}(B)}\right| B,\rho_s\right], \tag{9}$$

$$\mathrm{E}\left[\, \|F(x,B)\|_F^2 \;\middle|\, B,\rho_s\right] < \infty, \tag{10}$$

where $\mathrm{vec}(F) = (F_{11},\dots,F_{n1},\dots,F_{1n},\dots,F_{nn})^\top$ is the vectorization of matrices and $\|\cdot\|_F$ denotes Frobenius norm. It should be noted that both in usual ICA models and in the variance-dependent BSS model, scales and orders of the sources cannot be determined, that is, two matrices B and PDB indicate the same distribution, when P and D are a permutation and a diagonal matrix respectively (Comon, 1994).[3] Therefore, we can find any matrix in the equivalence class, so for notational convenience we will fix scales as the constraints (25) later.[4]

One of the standard ICA algorithms originates from maximum likelihood estimation, which is asymptotically the best method if the density ρ_s is known. Because in the semiparametric model ρ_s is unknown and difficult to estimate, the idea is to use instead the maximum likelihood estimation under a prefixed density $\widetilde{\rho}_s$. The method is called the quasi maximum likelihood estimation, since the fixed $\widetilde{\rho}_s$ does not coincide with the true one. The estimator $\widehat{B}$ is derived from the equation

$$\sum_{t=1}^{T}\left[I - \varphi\{y(t)\}\, y^\top(t)\right] = 0, \tag{11}$$

where $y(t) = \widehat{B}x(t)$ is the estimator of the sources, $\varphi(y) = (\varphi_1(y_1),\dots,\varphi_n(y_n)\,)^\top$ and

$$\varphi_i(y_i) = -\frac{\mathrm{d}}{\mathrm{d}y_i}\log\widetilde{\rho}_{s_i}(y_i).$$

For the nonlinear function $\varphi_i(y_i)$,

$$\varphi_i(y_i) \;=\; \tanh(cy_i), \qquad c > 0, \tag{12}$$

$$\varphi_i(y_i) \;=\; y_i^3, \tag{13}$$

are often employed. The function $F(x,B) = I - \varphi\{Bx\}(Bx)^\top$ in (11) is an example of estimating functions for the ICA model, provided that it satisfies (9) and (10). It is trivial to show that it fulfills (8). Another example is the function

$$F(x,B) = Bx\,(Bx)^\top - I + (Bx)\, g^\top(Bx) - g(Bx)\;(Bx)^\top$$

3. It is clear that the variance-dependent BSS model has at least such indeterminacy. On the other hand, the identifiability in this case has not been proved so far.

4. We ignore the permutation indeterminacy P, since it's locally not problematic.

for FastICA (see (37) in Appendix A.1), where $g(\cdot)$ is a vector valued non-linear function as $\varphi(\cdot)$. We remark that this procedure can also be derived from minimum mutual information (Yang and Amari, 1997) and infomax principle (Bell and Sejnowski, 1995).

In general the quasi maximum likelihood estimator is no longer consistent because of misspecified distribution. However, in the ICA model (2), Amari and Cardoso (1997) found that the quasi maximum likelihood method and its online version (the natural gradient learning) give an asymptotically consistent estimator, provided that $F(x,B) = I - \varphi\{Bx\}(Bx)^\top$ satisfies (9) and (10). In particular, we remark that the assumed distribution $\widetilde{\rho}_s$ is not equal to the true one. This research has motivated us to investigate also such semiparametric procedures for the variance-dependent BSS model (3) and (4). In particular, we will show in Section 5.1 that the quasi maximum likelihood method (11) still gives a consistent estimator even under this extended situation.

4.2 Estimating Functions for Variance-Dependent Blind Source Separation

In the variance-dependent BSS model, in contrast to the ICA model studied by Amari and Cardoso (1997), the data sequence $X = (x(1),\dots,x(T))$ is not i.i.d. in time, but might have time dependencies. Therefore, we have to consider more general functions $\bar{F}(X,B)$ of the whole sequence X. General estimating functions $\bar{F}(X,B)$ must satisfy

$$\mathrm{E}[\bar{F}(X,B)\,|B,\rho_z,\rho_V] = 0, \tag{14}$$

$$|\det Q| \neq 0, \qquad \text{where } Q = \mathrm{E}\left[\left.\frac{\partial \mathrm{vec}\{\bar{F}(X,B)\}}{\partial \mathrm{vec}(B)}\right| B,\rho_z,\rho_V\right], \tag{15}$$

$$\mathrm{E}\left[\,\|\bar{F}(X,B)\|_F^2\,\middle|\, B,\rho_z,\rho_V\right] < \infty, \tag{16}$$

for all (B,ρ_z,ρ_V). An M-estimator $\widehat{B}$ can be derived from the estimating equation

$$\bar{F}(X,\widehat{B}) = 0. \tag{17}$$

Suppose that the data X is subject to $p(X|B^*,\rho_z^*,\rho_V^*)$ defined by (3) and (4).

Theorem 1 *If the function $\bar{F}(X,B)$ satisfies the conditions (14) – (16) and appropriate regularity conditions such as Condition 2.6 in Sørensen (1999), the M-estimator $\widehat{B}$ derived from the equation (17) is asymptotically Gaussian distributed* $\mathrm{vec}(\widehat{B}) \sim N(\mathrm{vec}(B^*),\mathrm{Av})$, *where*

$$\begin{aligned}
\mathrm{Av} &= \mathrm{Av}(B^*,\rho_z^*,\rho_V^*) = Q^{-1}\Sigma(Q^{-1})^\top, \qquad (18)\\
\Sigma &= \Sigma(B^*,\rho_z^*,\rho_V^*) = E\left[\mathrm{vec}\{\bar{F}(X,B^*)\}\,\mathrm{vec}\{\bar{F}(X,B^*)\}^\top \middle| B^*,\rho_z^*,\rho_V^*\right]\\
Q &= Q(B^*,\rho_z^*,\rho_V^*) = \mathrm{E}\left[\left.\frac{\partial \mathrm{vec}\{\bar{F}(X,B^*)\}}{\partial \mathrm{vec}(B)}\right| B^*,\rho_z^*,\rho_V^*\right].
\end{aligned}$$

Proof See Sørensen (1999).

Now, we investigate the relation between estimating functions for the ICA model and those for the variance-dependent BSS model. Let $F(x,B)$ be an estimating function for the ICA model. In the ICA context it is often the case that such estimating functions satisfy

$$\mathrm{E}[F_{ij}(x,DB)\,|B,\rho_s] = 0, \qquad i \neq j, \tag{19}$$

for any diagonal matrix D, that is, its off-diagonal parts (19) hold for all matrices equivalent to B. The scale factor D is determined usually by the diagonal parts of condition (8)

$$\mathrm{E}[F_{ii}(x,B)\ |B,\rho_s] = 0,$$

in the ordinary ICA model. We will soon present the equation for fixing the scale factor D in the variance-dependent BSS model.

Let us consider the function

$$\bar{F}(X,B) = \sum_{t=1}^{T} F(x(t),B), \tag{20}$$

which is used in estimating equations for the ICA model.[5] We can show that this function becomes a candidate of estimating functions for the variance-dependent BSS model.

Proposition 2 *The function $\bar{F}(X,B)$ defined in (20) satisfies condition (14), provided that $F(x,B)$ is an estimating function for the ICA model and fulfills (19). Furthermore, if the additional assumption*

$$\mathrm{E}\left[\ \|F(x(t),B)\|_F^2\ \middle|B,\rho_z,\rho_V\right] < \infty, \qquad \forall t \tag{21}$$

holds, condition (16) is also satisfied.

Proof Taking expectations of the off-diagonal terms of (14), we get

$$\begin{aligned}
\mathrm{E}\left[\ \bar{F}_{ij}(X,DB)\ \middle|B,\rho_z,\rho_V\right] &= \mathrm{E}\left[\sum_{t=1}^{T}\mathrm{E}\left[\ F_{ij}(x(t),DB)\ \middle|V;B,\rho_z\right]\middle|\rho_V\right] \\
&= \mathrm{E}\left[\sum_{t=1}^{T} E\left[\ F_{ij}(x(t),DB)\ \middle|B,\rho_{s|v(t)}\right]\middle|\rho_V\right]
\end{aligned}$$

where $\rho_{s|v(t)}$ is the density function of $s(t)$ when its activity level is fixed at $v(t)$, that is,

$$\rho_{s|v(t)}(s) = \prod_{i=1}^{n}\frac{1}{v_i(t)}\ \rho_{z_i}\left\{\frac{s_i}{v_i(t)}\right\}.$$

We remark that the expectation $\mathrm{E}[\cdot|V;B,\rho_z]$ ($\mathrm{E}[\cdot|B,\rho_{s|v(t)}]$) is taken over $z(t)$ (resp. $s(t)$) under fixed activity levels V, while $\mathrm{E}[\cdot|\rho_V]$ denotes the expectation over the activity level V. Because (19) holds for any ρ_s, we can prove

$$\mathrm{E}\left[\ \bar{F}_{ij}(X,DB)\ \middle|B,\rho_z,\rho_V\right] = 0,$$

for all diagonal matrices D. If we select the scale factor D such that the diagonal terms

$$\mathrm{E}[\bar{F}_{ii}(X,B)\ |B,\rho_z,\rho_V] = 0$$

hold, $\bar{F}$ satisfies the unbiasedness condition (14). We furthermore note that this scaling is different from that in the ICA model presented before, and the expectation $\mathrm{E}\left[\ F_{ii}(x(t),B)\ |B,\rho_{s|v(t)}\right]$ at each time t can be non-zero in general.

5. We remark that some of ICA/BSS algorithms (for example, TDSEP/SOBI) are not based on estimating functions in this class. Because it is not easy to discuss them in such a general form, we deal with other classes separately in Appendix A.

The left hand side of Eq. (16) can be expressed as

$$\begin{aligned}
&\mathrm{E}\left[\, \|\bar{F}(X,B)\|_F^2 \,\middle|\, B, \rho_z, \rho_V \right] \\
&= \mathrm{E}\left[\sum_{t,t'} \mathrm{E}\left[\mathrm{tr}\{F(x(t),B)F^\top(x(t'),B)\} \,\middle|\, V; B, \rho_z \right] \,\middle|\, \rho_V \right] \\
&= \mathrm{E}\left[\sum_{t} \mathrm{E}\left[\|F(x(t),B)\|_F^2 \,\middle|\, B, \rho_{s|v(t)} \right] \,\middle|\, \rho_V \right] \\
&\quad + \mathrm{E}\left[\sum_{t\neq t'} \sum_{i=1}^{n} \mathrm{E}\left[F_{ii}(x(t),B) \,\middle|\, B, \rho_{s|v(t)} \right] \mathrm{E}\left[F_{ii}(x(t'),B) \,\middle|\, B, \rho_{s|v(t')} \right] \,\middle|\, \rho_V \right], \qquad (22)
\end{aligned}$$

where we used the fact that $x(t)$ and $x(t')$ ($t \neq t'$) are independent for fixed V. From assumption (21), the first term of Eq. (22) is finite.

$$\begin{aligned}
&\sum_t \mathrm{E}\left[\, \mathrm{E}\left[\|F(x(t),B)\|_F^2 \,\middle|\, B, \rho_{s|v(t)} \right] \,\middle|\, \rho_V \right] \\
&= \sum_t \mathrm{E}\left[\|F(x(t),B)\|_F^2 \,\middle|\, B, \rho_z, \rho_V \right] < \infty \qquad (23)
\end{aligned}$$

We remark that condition (10) does not necessarily imply assumption (21). Let us define $c(v(t)) := \mathrm{E}\left[\, \|F(x(t),B)\|_F^2 \,\middle|\, B, \rho_{s|v(t)} \right]$. Since

$$\left| \mathrm{E}\left[F_{ii}(x(t),B) \,\middle|\, B, \rho_{s|v(t)} \right] \right| \leq \sqrt{\mathrm{E}\left[F_{ii}^2(x(t),B) \,\middle|\, B, \rho_{s|v(t)} \right]} \leq \sqrt{c(v(t))},$$

the second term of Eq. (22) (called r in the following) can be bounded as

$$\begin{aligned}
|r| &< n \sum_{t\neq t'} \mathrm{E}\left[\sqrt{c(v(t))}\sqrt{c(v(t'))} \,\middle|\, \rho_V \right] \\
&\leq n \left\{ \sum_t \sqrt{\mathrm{E}\left[c(v(t)) \,|\, \rho_V \right]} \right\}^2 \\
&\leq nT \sum_t \mathrm{E}\left[c(v(t)) \,|\, \rho_V \right].
\end{aligned}$$

Here we used Schwarz's inequality twice. Because of Eq. (23), this bound is also finite. □

The basic idea of this proof is that the situation becomes similar to the ordinary ICA model, if the activity levels V are fixed. Unfortunately, the other conditions are difficult to be proven in this general form. For example, the second condition can be transformed in the similar way as

$$\begin{aligned}
&\mathrm{E}\left[\frac{\partial \mathrm{vec}\{\bar{F}(X,B)\}}{\partial \mathrm{vec}(B)} \,\middle|\, B, \rho_z, \rho_V \right] \\
&= \sum_{t=1}^{T} \mathrm{E}\left[\mathrm{E}\left[\frac{\partial \mathrm{vec}\{F(x(t),B)\}}{\partial \mathrm{vec}(B)} \,\middle|\, B, \rho_{s|v(t)} \right] \,\middle|\, \rho_V \right]. \qquad (24)
\end{aligned}$$

Even if each term $\mathrm{E}\left[\frac{\partial \mathrm{vec}\{F(x(t),B)\}}{\partial \mathrm{vec}(B)} \,\middle|\, B, \rho_{s|v(t)} \right]$ is non-singular, it may still be possible that the sum (24) becomes singular. However this is in practice an extremely rare case.

5. Consistency Results for Variance Dependent Blind Source Separation Using the Estimating Function Framework

We will use the estimating function framework to prove consistency results for (i) the quasi maximum likelihood type methods (for example, Pham and Garrat, 1997; Bell and Sejnowski, 1995), (ii) the natural gradient learning for ICA (for example, Amari, 1998) and (iii) various other ICA algorithms such as FastICA (Hyvärinen and Oja, 1997), TDSEP/SOBI (Ziehe and Müller, 1998; Belouchrani et al., 1997), 'Sepagaus' (Pham and Cardoso, 2000) and JADE (Cardoso and Souloumiac, 1993).

5.1 Asymptotic Distribution of the Quasi Maximum Likelihood Estimator

In this section, it is shown that the quasi maximum likelihood method (11) as for example Pham and Garrat (1997); Bell and Sejnowski (1995) still gives a consistent estimator even under the extended model (3) and (4). For convenience, we fix the scales of the recovered signals as

$$\mathrm{E}\left[\sum_{t=1}^{T} \varphi_i\{b_i^\top x(t)\}\, b_i^\top x(t) \,\middle|\, B, \rho_z, \rho_V\right] = T, \tag{25}$$

for $i = 1, \ldots, n$. Then (14) is automatically fulfilled for the diagonal terms. We remark that by this constraints the length of b_i's may depend on the nuisance parameters (ρ_z, ρ_V), but this does not change the following discussion, because the scales can be fixed arbitrarily.

Since the function $F(x,B) = I - \varphi\{Bx\}(Bx)^\top$ obviously satisfies (19), we already know from Theorem 2 that the function

$$\bar{F}^{\mathrm{QML}}(X,B) = \sum_{t=1}^{T} \left[I - \varphi\{y(t)\}\, y^\top(t) \right]$$

satisfies the conditions (14) and (16) under the assumption

$$\mathrm{E}\left[\varphi_i^2\{y_i(t)\}\, y_j^2(t) \,\middle|\, B, \rho_z, \rho_V\right] < \infty, \qquad \forall i,\, j,\, t, \tag{26}$$

where $y(t)$ denotes the extracted sources $Bx(t)$. The additional assumption imposes mild restriction on the distribution of the activity levels V. For example, when the density ρ_V has extremely heavy tails, the left hand side of Eq. (16) becomes infinite, even if condition (10) is fulfilled. Thus, we need assumptions like (26) to exclude such unusual cases.

For better understanding, we directly analyze the off-diagonal terms of (14)

$$\begin{aligned}
&\mathrm{E}\left[\sum_{t=1}^{T} \varphi_i\{y_i(t)\}\, y_j(t) \,\middle|\, B, \rho_z, \rho_V\right] \\
&= \sum_{t=1}^{T} \mathrm{E}\left[\, \mathrm{E}\left[\varphi_i\{v_i(t) z_i(t)\}\ v_j(t) z_j(t) \,\middle|\, V; B, \rho_z\right] \,\middle|\, \rho_V\right] \\
&= \sum_{t=1}^{T} \mathrm{E}\left[\, \mathrm{E}\left[\varphi_i\{v_i(t) z_i(t)\} \,\middle|\, V; B, \rho_z\right] \mathrm{E}\left[\, v_j(t) z_j(t) \,\middle|\, V; B, \rho_z\right] \,\middle|\, \rho_V\right] \;=\; 0.
\end{aligned}$$

The second equality follows from the fact that z_i and z_j are independent for fixed V.[6]

6. This unbiasedness in fact holds under a wider condition $\mathrm{E}[s_i(t)|s_j(\cdot),\ j \neq i] = 0$.

To prove condition (15) and compute the asymptotic variance (18), we calculate the $n^2 \times n^2$ matrix Q. If we use the non-holonomic basis $d\chi = dBB^{-1}$ (Amari et al., 2000), Q is expressed as

$$Q = \mathrm{E}\left[\frac{\partial \mathrm{vec}(\bar{F}^{\mathrm{QML}})}{\partial \mathrm{vec}(\chi)} \,\middle|\, B, \rho_z, \rho_V \right] \bar{B}^{-1},$$

where $\bar{B} = (\bar{B}_{ij;kl})$ and $\bar{B}_{ij;kl} = \delta_{ik} b_{lj}$. Fortunately, the matrix $\mathrm{E}\left[\frac{\partial \mathrm{vec}(\bar{F}^{\mathrm{QML}})}{\partial \mathrm{vec}(\chi)}\right]$ turns out to have a simple structure such that only the following $2n^2 - n$ components are non-zero,

$$\begin{aligned}
\mathrm{E}\left[\frac{\partial \bar{F}_{ii}^{\mathrm{QML}}}{\partial \chi_{ii}}\right] &= -\sum_{t=1}^{T} \mathrm{E}\left[m_i\{v_i(t)\}\right] - T, \\
\begin{pmatrix} \mathrm{E}\left[\frac{\partial \bar{F}_{ij}^{\mathrm{QML}}}{\partial \chi_{ij}}\right] & \mathrm{E}\left[\frac{\partial \bar{F}_{ij}^{\mathrm{QML}}}{\partial \chi_{ji}}\right] \\ \mathrm{E}\left[\frac{\partial \bar{F}_{ji}^{\mathrm{QML}}}{\partial \chi_{ij}}\right] & \mathrm{E}\left[\frac{\partial \bar{F}_{ji}^{\mathrm{QML}}}{\partial \chi_{ji}}\right] \end{pmatrix} &= -\begin{pmatrix} \sum_{t=1}^{T} \mathrm{E}[k_i\{v_i(t)\} v_j^2(t)] & T \\ T & \sum_{t=1}^{T} \mathrm{E}[k_j\{v_j(t)\} v_i^2(t)] \end{pmatrix},
\end{aligned}$$

in which we employed the following quantities

$$\begin{aligned}
k_i\{v_i(t)\} &= \mathrm{E}\left[\, \dot{\varphi}_i\{v_i(t) z_i(t)\} \,|V; B, \rho_z\right], \\
m_i\{v_i(t)\} &= v_i^2(t)\, \mathrm{E}\left[\, \dot{\varphi}_i\{v_i(t) z_i(t)\}\, z_i^2(t) \,\middle|V; B, \rho_z\right],
\end{aligned}$$

and $\dot{\varphi}_i$ is the derivative of φ_i. Hence, it is not difficult to check non-singularity of this matrix, and if this is the case, the condition (15) holds. We can also explicitly calculate the inverse matrix $Q^{-1} = \bar{B}\left(\mathrm{E}\left[\frac{\partial \mathrm{vec}(\bar{F}^{\mathrm{QML}})}{\partial \mathrm{vec}(\chi)}\right]\right)^{-1}$ that appears in the asymptotic variance (18), because we only have to invert the 2×2 matrices.

Finally, the variance of the estimating function can be computed as

$$\begin{aligned}
&\mathrm{E}\left[\bar{F}_{ij}^{\mathrm{QML}} \bar{F}_{kl}^{\mathrm{QML}} | B, \rho_z, \rho_V\right] \\
&= \begin{cases} \sum_{t,t'} \mathrm{cov}\left[\varphi_i\{y_i(t)\} y_i(t),\ \varphi_k\{y_k(t')\} y_k(t')\right], & i = j,\ k = l \\ \sum_{t} \mathrm{E}\left[\varphi_i\{y_i(t)\}\, \varphi_k\{y_k(t)\}\, y_j^2(t)\right], & j = l,\ i \neq j \text{ or } k \neq l \\ \sum_{t} \mathrm{E}\left[\varphi_i\{y_i(t)\} y_i(\mathrm{t})\, \varphi_j\{y_j(t)\} y_j(t)\right], & i = l,\ j = k,\ i \neq j \end{cases}
\end{aligned}$$

which is slightly more complicated than the standard ICA model. Summing up the discussion above, we get the following theorem.

Theorem 3 *Suppose that the conditions*

$$\sum_{t=1}^{T} \mathrm{E}\left[m_i\{v_i(t)\}\right] + T \neq 0, \qquad \forall i, \tag{27}$$

$$\det \begin{pmatrix} \sum_{t=1}^{T} \mathrm{E}[k_i\{v_i(t)\} v_j^2(t)] & T \\ T & \sum_{t=1}^{T} \mathrm{E}[k_j\{v_j(t)\} v_i^2(t)] \end{pmatrix} \neq 0, \qquad \forall i \neq j, \tag{28}$$

and assumption (26) hold. Then the function $\bar{F}^{QML}(X,B)$ satisfies the conditions (14) – (16) and becomes an estimating function. In that case, the quasi maximum likelihood estimator $\widehat{B}^{\mathrm{QML}}$ derived from the equation $\bar{F}^{QML}(X,\widehat{B}^{\mathrm{QML}})=0$ is consistent regardless of the true nuisance functions (ρ_z^*,ρ_V^*) under appropriate regularity conditions.

5.2 Stability of the Natural Gradient Learning

In neural networks and machine learning, online leaning is often preferred to batch learning because of computational efficiency, less memory and adaptability (see, for example, Müller et al., 1998; Murata et al., 2002). The natural gradient learning (Amari, 1998)

$$B(t+1)=B(t)+\eta(t)\left[I-\varphi\{y(t)\}y^\top(t)\right]B(t), \tag{29}$$

is an online algorithm based on the quasi maximum likelihood method, where $y(t)=B(t)x(t)$ is the current estimator of the sources and $\eta(t)$ is an appropriate learning constant.

Following the discussion in Amari et al. (1997), we will study the stability of the natural gradient learning for the variance-dependent BSS model. For the sake of simplicity, they analyzed a continuous version of the algorithm (29)

$$\dot{B}(t)=\mu(t)\left[I-\varphi\{y(t)\}y^\top(t)\right]B(t), \tag{30}$$

where $\dot{B}(t)$ denotes time derivative of the matrix $B(t)$, $\mu(t)=\eta(t)/\tau$ and τ means the sampling period. Suppose that the marginal distributions of the activity levels $v(t)$ are identical in time. For example, when the sequence V is generated from an AR process, this holds approximately after it reaches the equilibrium distribution. Although the random variables $v(t)$'s (activity levels) have an identical marginal distribution in time, their realization can fluctuate from time to time and weak nonstationary structures can be found in the observed signals. Unfortunately, it is difficult to eliminate this rather strong assumption. If we apply the online algorithm (29) to data with highly nonstationary variances like speech, the scale factor of the demixing matrix B changes substantially from time to time and never converges. This makes the current stability analysis impossible. It might be possible to discuss these cases by considering only the equivalence class, but it is out of the scope of the current paper.

In order to fix the scales of the sources, we impose constraints

$$\mathrm{E}\left[\varphi_i\{b_i^\top x(t)\}\ b_i^\top x(t)\right]=1, \qquad \forall i. \tag{31}$$

Note that the marginal distribution of $x(t)$ is identical in time t and the equilibrium points B_0 of the equation (30) satisfy

$$\mathrm{E}\left[I-\varphi\{y_0(t)\}y_0^\top(t)\right]=0, \tag{32}$$

where $y_0(t)=B_0x(t)$. With a similar calculation as in Section 5.1, we can show that the function

$$F^{\mathrm{NG}}(x,B)=I-\varphi(y)y^\top$$

satisfies the unbiasedness condition (8) of estimating functions. This means that the true demixing matrix B^* satisfies the equilibrium equation (32), that is, B^* becomes an equilibrium point of the flow (30). However, it does not guaranteed that $B(t)$ converges to B^* even locally.

Let us fix the stochastic process $V = \{v(t),\ t \geq 0\}$ of the activity levels at first and consider the conditional expected version of the learning equation

$$\dot{B}(t) = \mu(t)\ \mathrm{E}\left[I - \varphi\{y(t)\} y^\top(t)\ \middle|\ V \right]\ B(t).$$

By linearizing it at the equilibrium point B^*, we have the variational equation

$$\mathrm{vec}\{\delta\dot{B}(t)\} = \mu(t)\ \frac{\partial \mathrm{vec}\{\ \mathrm{E}\left[F^{\mathrm{NG}}(x(t), B^*)\ \middle|V\right] B^*\}}{\partial \mathrm{vec}(B)}\ \mathrm{vec}\{\delta B(t)\},$$

where $\delta B(t)$ is a small perturbation. Therefore, we have to check the eigenvalues of the operators $\frac{\partial \mathrm{vec}\{\ \mathrm{E}\left[F^{\mathrm{NG}}(x(t),B^*)\ \middle|V\right] B^*\}}{\partial \mathrm{vec}(B)}$ for each $t \geq 0$. If all eigenvalues have negative real parts, then the equilibrium B^* is asymptotically stable for the fixed activity levels V. Since the matrix can be expressed as

$$\frac{\partial \mathrm{vec}\{\ \mathrm{E}\left[\ F^{\mathrm{NG}}(x(t), B^*)\ \middle|V\right]\ B^*\}}{\partial \mathrm{vec}(B)} = \bar{B}^*\ \frac{\partial \mathrm{vec}\left(\mathrm{E}\left[F^{\mathrm{NG}}\middle|V\right]\right)}{\partial \mathrm{vec}(\chi)}\ (\bar{B}^*)^{-1}, \tag{33}$$

where $\bar{B}^* = (\bar{B}^*_{ij;kl}) = (\delta_{ik} b^*_{lj})$, and derivative w.r.t. χ corresponds to the non-holonomic basis $d\chi = dBB^{-1}$. Because the left hand side of (33) is a similar transformation of $\frac{\partial \mathrm{vec}(\mathrm{E}[F^{\mathrm{NG}}|V])}{\partial \mathrm{vec}(\chi)}$, their eigenvalues are the same. Fortunately, as is the case of the quasi maximum likelihood, the matrix $\frac{\partial \mathrm{vec}(\mathrm{E}[F^{\mathrm{NG}}|V])}{\partial \mathrm{vec}(\chi)}$ has a simple structure such that only the following $2n^2 - n$ components are non-zero,

$$\begin{aligned}
\frac{\partial \mathrm{E}[F^{\mathrm{NG}}_{ii}|V]}{\partial \chi_{ii}} &= -m_i\{v_i(t)\} - 1 \\
\begin{pmatrix} \frac{\partial \mathrm{E}[F^{\mathrm{NG}}_{ij}|V]}{\partial \chi_{ij}} & \frac{\partial \mathrm{E}[F^{\mathrm{NG}}_{ij}|V]}{\partial \chi_{ji}} \\ \frac{\partial \mathrm{E}[F^{\mathrm{NG}}_{ji}|V]}{\partial \chi_{ij}} & \frac{\partial \mathrm{E}[F^{\mathrm{NG}}_{ji}|V]}{\partial \chi_{ji}} \end{pmatrix} &= -\begin{pmatrix} k_i\{v_i(t)\} v_j^2(t) & 1 \\ 1 & k_j\{v_j(t)\} v_i^2(t) \end{pmatrix}
\end{aligned}$$

Therefore, the matrix $\frac{\partial \mathrm{vec}(\mathrm{E}[F^{\mathrm{NG}}|V])}{\partial \mathrm{vec}(\chi)}$ at time t has eigenvalues only with negative real parts, if and only if

$$m_i\{v_i(t)\} + 1 > 0 \tag{34}$$
$$k_i\{v_i(t)\} > 0 \tag{35}$$
$$v_i^2(t)\, v_j^2(t)\, k_i\{v_i(t)\}\, k_j\{v_j(t)\} > 1 \tag{36}$$

for all i, $j\ (i \neq j)$.

Theorem 4 *If the stochastic process $V = \{v(t),\ t \geq 0\}$ of the activity levels satisfies the conditions (34) – (36) with probability* 1 *as for the true parameter $(B^*, \rho_z^*, \rho_V^*)$, then the true demixing matrix B^* becomes an asymptotically stable equilibrium of the flow (30) with probability* 1.

Although asymptotic stability could be proved under weaker conditions, we summarize the discussion as Theorem 4 for simplicity. In order to understand the result better, we revisit the examples presented in Amari et al. (1997). The conditions turn out to be much harder than those by Amari et al. (1997) because of the fluctuating activity levels.

Example 1. Let us consider the following odd activation function

$$\varphi_i(y_i) = |y_i|^p \mathrm{sign}(y_i)$$

for $p = 1, 2, \ldots$. The conditions (34) and (35) are automatically satisfied for any fixed $v_i(t) > 0$.

$$\begin{aligned} m_i\{v_i(t)\} &= p\, v_i^{p+1}(t)\ \mathrm{E}\left[|z_i(t)|^{p+1}\right] > 0 \\ k_i\{v_i(t)\} &= p\, v_i^{p-1}(t)\ \mathrm{E}\left[|z_i(t)|^{p-1}\right] > 0 \end{aligned}$$

The condition (36) becomes

$$p^2\, v_i^{p+1}(t)\, v_j^{p+1}(t)\ \mathrm{E}\left[|z_i(t)|^{p-1}\right]\ \mathrm{E}\left[|z_j(t)|^{p-1}\right] > 1.$$

By introducing Gray's norm

$$\gamma_{pi} = \frac{\mathrm{E}[\,|z_i|^{p+1}]}{\mathrm{E}[\,|z_i|^2]\,\mathrm{E}[\,|z_i|^{p-1}]}$$

and taking notice of the normalization constraints (31), that is, $\mathrm{E}[\,|z_i|^{p+1}] = \left(\mathrm{E}[v_i^{p+1}]\right)^{-1}$, finally we obtain

$$\gamma_{pi}\gamma_{pj} < p^2\ \frac{\min_t v_i^{p+1}(t)}{\mathrm{E}[v_i^{p+1}]}\ \frac{\min_t v_j^{p+1}(t)}{\mathrm{E}[v_j^{p+1}]}.$$

For the cubic function $\varphi_i(y_i) = y_i^3$, not as in the ICA model, the condition that all signals are sub-Gaussian

$$\gamma_{3i} = \frac{\mathrm{E}[\,|z_i|^4]}{\left(\mathrm{E}[\,|z_i|^2]\right)^2} < 3$$

is not enough, but the variation of activity levels v_i from (1) should be taken into account.

Example 2. Let us consider a symmetrical sigmoidal function

$$\varphi_i(y_i) = \tanh(\beta y_i).$$

The conditions (34) and (35) can be checked easily. Unfortunately, in this case we can only do a rather coarse analysis as follows. Let us assume $\beta \ll 1$ so that the approximation

$$\varphi_i(y_i) \approx \beta y_i - \frac{1}{3}(\beta y_i)^3 + \frac{2}{15}(\beta y_i)^5$$

holds with high probability. Then, we can express the condition (36) as

$$\beta^2 v_i^2(t)\, v_j^2(t)\ \mathrm{E}\left[1 - (\beta y_i)^2 + \frac{2}{3}(\beta y_i)^4 \middle| V\right]\ \mathrm{E}\left[1 - (\beta y_j)^2 + \frac{2}{3}(\beta y_j)^4 \middle| V\right] > 1.$$

Because $1 - t^2 + 2t^4/3 > t^4/3$, we get a stronger condition

$$\frac{\beta^{10}}{9}\ v_i^2(t)\, v_j^2(t)\ \mathrm{E}\left[y_i^4 \middle| V\right]\ \mathrm{E}\left[y_j^4 \middle| V\right] > 1.$$

From a rough approximation of (31), the relation $\beta \approx \left(\mathrm{E}[v_i^2]\right)^{-1}$ is derived. Therefore, if all approximations are accurate enough, we finally get a sufficient condition of (36) like

$$\gamma_{3i}\gamma_{3j} > \frac{9}{\beta^4}\left(\frac{\mathrm{E}[v_i^2]}{\min\limits_t v_i^2}\right)^3\left(\frac{\mathrm{E}[v_j^2]}{\min\limits_t v_j^2}\right)^3.$$

In contrast to the ordinal ICA model without variance dependence, the condition that all signals are super-Gaussian may not be enough, but each kurtosis γ_{3i} should be much larger than 3.[7]

5.3 Properties of Other BSS Algorithms

Although we concentrated on estimating functions of the form (20), we can deal with more general functions and investigate other ICA algorithms within the framework of estimating functions and asymptotic estimating functions (see also Cardoso, 1997). Such analysis helps to check whether these algorithms may give valid solutions regardless of the nuisance densities (ρ_z, ρ_V). We remark that our extension enables us to analyze algorithms based on temporal structure such as TDSEP/SOBI (Ziehe and Müller, 1998; Belouchrani et al., 1997). Since it is quite technical, the detailed discussion is put in Appendix A, where the unbiasedness condition (14) of estimating functions is examined for these algorithms under the variance-dependent BSS model. We briefly summarize the consequences in Table 2. Estimators by all algorithms listed below are derived from estimating equations which satisfy the unbiasedness condition at least asymptotically. When the other conditions are taken into account, TDSEP/SOBI never works for the variance-dependent BSS model, because sources have no lagged auto-correlations. ICA algorithms using non-Gaussianity such as FastICA and JADE are not working, if sources are Gaussian. The double blind algorithm (Hyvärinen and Hurri, 2004) cannot be applied to the case where the variance structures of sources are the same or there is no temporal variance-dependency. The nonstationary algorithm by Pham and Cardoso (2000) is not applicable to the case where time courses of the activity levels are proportional to each other. Of course, such a theoretical analysis tells us only about the possibility of failure. In practice, algorithms do not always return valid answers, because of local minima and numerical instability of their learning process. Nevertheless, this theoretical analysis can explain the results of our numerical experiments in the next section.

6. Numerical Experiments

We carried out experiments with several artificial and more realistic data sets for several BSS algorithms. The eight batch algorithms and the online versions of the quasi maximum likelihood methods listed in Table 3 were applied to those data sets. Note that our goal is not primarily an algorithm comparison but the experiments serve to demonstrate the correctness of our theoretical analysis.

For evaluating the results, we used the index defined by Amari et al. (1996)

$$\mathrm{AmariIndex}(B, A^*) = \sum_{i=1}^{n}\left\{\frac{\sum_{j=1}^{n}|C_{ij}|}{\max_k |C_{ik}|} - 1\right\} + \sum_{j=1}^{n}\left\{\frac{\sum_{i=1}^{n}|C_{ij}|}{\max_k |C_{kj}|} - 1\right\},$$

7. This different result corrects a calculation in Amari et al. (1997).

algorithm	unbiasedness	unavailable cases
FastICA Hyvärinen (1999)	yes	Sources are Gaussian.
double blind Hyvärinen and Hurri (2004)	asymptotically	Variance structures are same or there is no temporal variance-dependency.
JADE Cardoso and Souloumiac (1993)	asymptotically	Sources are Gaussian.
TDSEP/SOBI Ziehe and Müller (1998) Belouchrani et al. (1997)	yes	always (since we consider here only the case without auto-correlations)
nonstationary Pham and Cardoso (2000)	yes	Time course of the activity levels are proportional to each other.

Table 2: Availability of other ICA and BSS algorithms

QML(tanh)	quasi maximal likelihood method with the hyperbolic tangent nonlinearity
QML(pow3)	quasi maximal likelihood method with the cubic nonlinearity
Online(tanh)	online version of QML(tanh) with learning rate $\eta(t) = \frac{0.1}{(1+t/20)}$
Online(pow3)	online version of QML(pow3) with learning rate $\eta(t) = \frac{0.25}{(1+t/20)}$
’DoubleBlind’	the double blind algorithm by Hyvärinen and Hurri (2004)
JADE	JADE algorithm
FastICA(tanh)	FastICA with the hyperbolic tangent nonlinearity
FastICA(pow3)	FastICA with the cubic nonlinearity
TDSEP/SOBI	TDSEP/SOBI algorithm
’Sepagaus’	The ’sepagaus’ algorithm for nonstationary signals by Pham and Cardoso (2000)

Table 3: ICA and BSS algorithms used in the experiments

where A^* is the true mixing matrix and $C = BA^*$. If $B = PD(A^*)^{-1}$ with a permutation matrix P and a diagonal matrix D, then $\text{AmariIndex}(B, A^*) = 0$.

6.1 Artificial Data Sets

In all artificial data sets, five source signals of various types with length $T = 10000$ were generated and data after multiplying a random 5×5 mixing matrix were observed. We made 100 replications for each setting and compute the demixing matrix for each replication. The first data set was made according to the experiments in Hyvärinen and Hurri (2004). The activity levels $v(t)$ were generated from a multivariate AR(1) model, where outliers larger than three times standard deviations from the means were reduced to these bounds. The normalized signals z_i's were i.i.d. sub-Gaussian random variables which are signed fourth-order roots of zero-mean uniform variables. The medians of the 100 replications are summarized in the row 'ar_subG' of Table 4 with the measure of deviation (3rd-quantile $-$ 1st-quantile)/2. As was pointed out by Hyvärinen and Hurri (2004), only 'DoubleBlind' gave small AmariIndex. Because the marginal distribution of the source signal $s_i(t)$ looks like a Gaussian, all algorithms based on indices favouring non-Gaussianity failed. Even though the determinant in the left hand side of (28) is close to 0, all the assumptions are satisfied and the local consistency theorem is still valid. However, this does not directly mean that the estimated demixing matrix converges globally to the true one. In this case, many local optima can make the algorithms fail. This could also be understood from the fact that the contrast functions based on non-Gaussianity become almost flat and thus are very difficult to optimize. In the experiments, we observed that part of the true sources were often extracted correctly.

Although all the algorithms except for 'DoubleBlind' did not work for the first difficult example, the theoretical study in principle tells that many ICA and BSS algorithms are also applicable to the variance-dependent BSS problem. So in fact the failure of the algorithms except 'DoubleBlind' can be solely explained by the particular choice of the data set which is in contrast to prior findings in Hyvärinen and Hurri (2004). In the second example, uniform random variables were used as z_i's instead of sub-Gaussian ones. The marginal distribution of the source signal $s_i(t)$ looks Laplacian. Therefore, as was shown in the row 'ar_uni' of Table 4, the algorithms QML(tanh) and FastICA(tanh), which are suitable for super-Gaussian sources, always give correct answers. The algorithms 'DoubleBlind', JADE and FastICA(pow3) based on 4-th order moments also worked except several failures due to outliers. We got admissible results by the nonstationary BSS algorithm 'Sepagaus', if an appropriate smoothing window was chosen.

In the third and the fourth data, the activity levels $v_i(t)$ are sinusoidal functions with different frequencies.

$$v_i(t) = 1 + 0.9 \sin\left(\frac{(13+i)\pi t}{8000}\right), \qquad i = 1, \ldots, 5$$

For the normalized signals z_i, Laplacian and the sub-Gaussian i.i.d. random variables were used in the third and the fourth examples, respectively. In the super-Gaussian case (the row 'sin_supG' of Table 4), the six algorithms except QML(pow3) and TDSEP/SOBI worked properly. 'Sepagaus' showed best performance, and QML(tanh) and FastICA(tanh) based on the hyperbolic tangent nonlinearity gave better results than 'DoubleBlind', JADE and FastICA(pow3) with 4-th order moments. On the other hand, in the sub-Gaussian case (the row 'sin_subG' of Table 4), the six algorithms except QML(tanh) and TDSEP/SOBI returned admissible results. 'Sepagaus' also showed

	QML(tanh)	QML(pow3)	Online(tanh)	Online(pow3)	'DoubleBlind'
ar_subG	8.25 (1.85)	11.32 (2.84)	14.59 (2.29)	14.75 (2.32)	0.52 (0.10)
ar_uni	0.30 (0.04)	27.77 (0.32)	0.51 (0.08)	23.40 (1.88)	0.70 (0.16)
sin_supG	0.17 (0.02)	29.97 (0.26)	0.39 (0.05)	28.74 (0.96)	0.79 (0.13)
sin_subG	19.21 (0.24)	0.32 (0.05)	21.51 (2.08)	0.57 (0.30)	0.27 (0.03)
com_supG	0.39 (0.06)	28.37 (0.27)	0.64 (0.09)	25.67 (1.74)	6.45 (1.56)
com_subG	26.53 (0.55)	0.14 (0.02)	27.00 (2.41)	0.28 (0.05)	22.05 (1.96)
exp_supG	0.35 (0.05)	28.43 (0.45)	0.59 (0.07)	22.84 (2.06)	7.63 (1.88)
uni_subG	27.38 (0.17)	0.13 (0.02)	27.24 (1.27)	0.27 (0.04)	18.56 (1.66)
sss	0.03	3.82	0.06 (0.01)	2.79 (0.53)	0.02
v12	0.01	3.73	0.06	2.89 (0.04)	0.21

	JADE	FastICA(tanh)	FastICA(pow3)	TDSEP/SOBI	'Sepagaus'
ar_subG	10.79 (1.88)	9.25 (1.98)	12.52 (2.05)	15.07 (1.96)	1.19 (0.48)
ar_uni	0.66 (0.14)	0.38 (0.05)	0.73 (0.14)	14.92 (2.37)	0.85 (0.22)
sin_supG	0.43 (0.07)	0.23 (0.03)	0.41 (0.07)	15.31 (2.04)	0.08 (0.01)
sin_subG	0.31 (0.04)	0.68 (0.14)	0.33 (0.05)	15.70 (1.94)	0.08 (0.01)
com_supG	0.84 (0.16)	0.48 (0.07)	0.87 (0.14)	16.02 (2.05)	1.28 (0.19)
com_subG	26.49 (0.86)	27.04 (0.38)	26.65 (0.17)	16.23 (2.01)	27.08 (0.40)
exp_supG	1.24 (0.23)	0.44 (0.06)	1.20 (0.22)	16.47 (1.81)	1.28 (0.20)
uni_subG	0.17 (0.03)	0.18 (0.03)	0.18 (0.03)	16.20 (1.78)	27.08 (0.33)
sss	0.02	0.19 (0.04)	0.09 (0.01)	0.01	0.01
v12	0.19	0.17 (0.02)	0.08 (0.09)	0.14	0.01

Table 4: AmariIndex of the estimators. The values are the medians of 100 replications with the measure of deviation, $(\text{3rd-quantile} - \text{1st-quantile})/2$

best performance, and all four algorithms with 4-th order moments showed better performance than the FastICA(tanh).

The double blind algorithm ('DoubleBlind') by Hyvärinen and Hurri (2004) does not work when (i) all v_i's have same temporal structure, and (ii) there exist no temporal dependencies in v_i's. 'Sepagaus' does not have a guarantee to separate sources either, because smoothed sequences of the activity levels are nearly proportional to each other (see Table 2). The fifth and sixth data set are examples of the case (i), where $v_i(t)$ are the same sinusoidal functions.

$$v_i(t) = 1 + 0.9 \sin\left(\frac{\pi t}{500}\right), \qquad i = 1, \ldots, 5$$

As in the third and the fourth examples, Laplace and the sub-Gaussian i.i.d. random variables were used for the normalized signals z_i. As in the row 'com_supG' of Table 4, the five algorithms except QML(pow3), 'DoubleBlind' and TDSEP/SOBI worked properly. Among them, QML(tanh) and FastICA(tanh) had better performance. 'DoubleBlind' gave poor results, because the matrix $\widetilde{K}_{ij} = \widehat{\text{cov}}\{s_i^2(\cdot), s_j^2(\cdot - 1)\}$ is almost singular. In the sub-Gaussian case, it looks quite difficult to distinguish the sources visually. Unfortunately, we could not demix them correctly except with QML(pow3) as shown in the row 'com_subG' of Table 4. In order to check why other algorithms with the local consistency did not work, we carried out extra experiments with larger sample size T. When $T = 200000$, the AmariIndices of the estimated demixing matrices by JADE are below 0.11, 92 times out of 100 repetition. On the other hand, both FastICA methods returned valid results almost always (AmariIndices are below 0.22, 89 times for FastICA(tanh) and 100 times for FastICA(pow3)), if $T = 50000$ and the algorithms start from the true demixing matrix. Therefore, we think that the global convergence is not achieved in these cases, because of finite sample size effects and local optima.

The seventh and the eighth data sets are examples of the case (ii), where $v(t)$ is i.i.d. in time t. In the former example, we transform 5 independent exponential random variables linearly such that v_i and v_j have correlation 0.9, and z_i's were i.i.d Laplace random variables. On the other hand, in the latter example, $v(t)$ was generated from 5 uniform random variables by the same linear transformation and z_i's were the i.i.d sub-Gaussian random variables. As one can see in the row 'exp_supG' of Table 4, the results are similar to the data set 'com_supG'. On the other hand, in the sub-Gaussian case summarized in the row 'uni_subG' of Table 4, QML(pow3), JADE, FastICA(tanh) and FastICA(pow3) gave correct results, but 'Sepagaus' showed very poor performance. We remark that in both cases, 'DoubleBlind' did not work as was expected.

We would now like to digest the results from Table 4 and relate them to our theoretical findings. We have shown that all algorithms except for TDSEP/SOBI have the local consistency for most of the given data. However, this does not directly mean that they converge globally to the true solution. Although we hope that algorithms with a local consistency work properly, we sometimes see significant deviations from this expectation in practice as in Table 4. The algorithmic failures are caused by local optima as pointed out above for the data set 'ar_subG', or more importantly to numerical stability and convergence issues. For example, since learning algorithms like gradient descent are used for QML(tanh) and QML(pow3), desired solutions (equilibria) turn out to be instable for sub-Gaussian (QML(tanh)) and super-Gaussian signals (QML(pow3)). In our data sets, 'ar_uni', all data sets with 'supG' and acoustic signals are super-Gaussian, while all data sets with 'subG' except 'ar_subG' are sub-Gaussian. One can see the clear pattern in the columns QML(tanh) and QML(pow3). The online version Online(tanh) and Online(pow3) had slightly degraded performance with appropriate learning rate, if the batch version QML(tanh) and QML(pow3) worked, respectively. On the other hand, although FastICA uses similar criteria for non-Gaussianity, it employs a kind of Newton's method and so the desired solutions are automatically better stabilized. In the columns FastICA(tanh) and FastICA(pow3), except for the difficult case 'com_subG' and nearly Gaussian case 'ar_subG', both algorithms succeeded.

6.2 Variance-Dependent Speech Signals

Next we will deal with more realistic data sets. Speech and audio signals have often been used as sources $s(t)$ even for experiments of the instantaneous ICA model. In order to check whether variance-dependency matters to many ICA and BSS algorithms, we applied BSS algorithms to speech signals which have strong variance-dependency.

In the first experiments 'sss',[8] we took two speech signals with length $T = 120976$, where one speaker says digits from 1 to 10 in English, and the other speaker counts at the same time in Spanish. We used the separated signals of their second demo as the sources, because their separation quality is good enough. Figure 2 shows the sources and the estimators of their activity levels with

8. The signals were downloaded from `http://inc2.ucsd.edu/~tewon/`.

an appropriate smoother. We inserted one short pause at different positions of both sequences to make correlation of the activity levels of the modified signals much larger (0.65). In the second experiments 'v12',[9] we took two speech signals from Japanese text ($T = 48000$). Figure 3 shows the sources and the estimators of the activity levels. We extended and shorten each syllable of the second sequence and tuned its amplitude such that the two sources have high variance-dependency. Correlation of the activity levels of the arranged signals becomes 0.74.

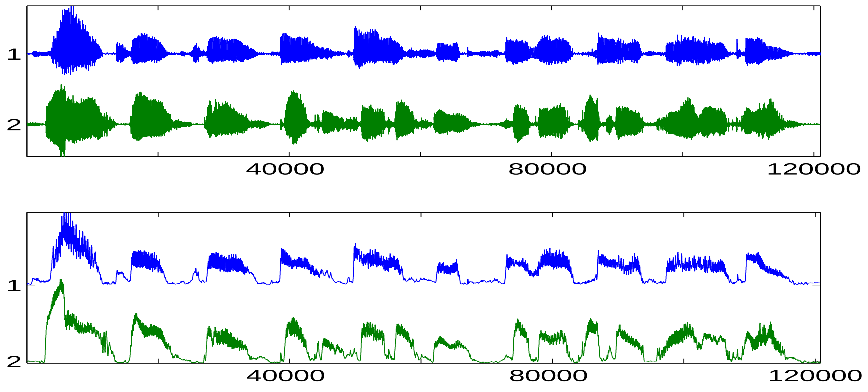


Figure 2: The sources of the data set 'sss' and the estimators of their activity levels. The upper panel contains the signals showing counting from 1 to 10 in English and Spanish. The lower panel shows their activity levels with an appropriate smoother.

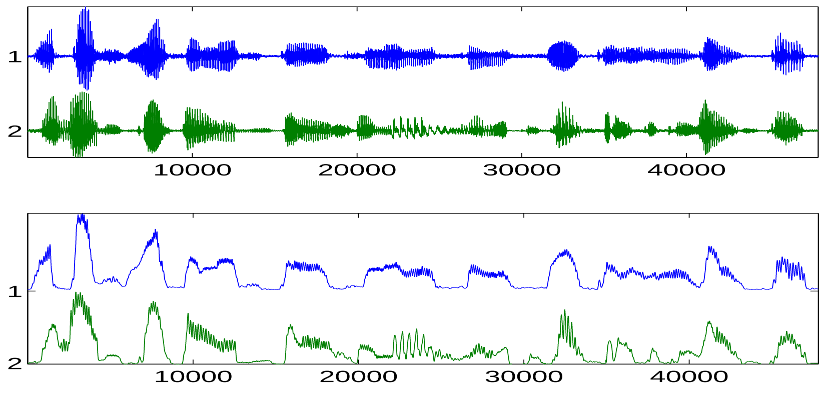


Figure 3: The sources of the data set 'v12' and the estimators of their activity levels. The upper panel are signals from Japanese sentences. The lower panel shows their activity levels with an appropriate smoother.

A 2×2 mixing matrix A was randomly generated 100 times and 100 different mixtures of the source signals were made. The results are summarized in the rows 'sss' and 'v12' of Table 4. In

9. The signals can be downloaded by `http://www.islab.brain.riken.go.jp/~mura/ica/v1.wav` and `v2.wav`.

both experiments, QML(tanh), JADE, TDSEP/SOBI and 'Sepagaus' always worked, while FastICA(tanh) and FastICA(pow3) gave admissible results except for several cases. Although TDSEP/SOBI is not applicable to the variance-dependent BSS model, it also returned correct results. This means that the speech signals are not perfectly matching the model Eq. (4), but the sources have furthermore a lagged autocorrelation. QML(tanh) always returned wrong answers, because speech is usually super-Gaussian.

7. Conclusions

In this paper, we discussed semiparametric estimation for blind source separation, when sources have variance dependencies. Hyvärinen and Hurri (2004) introduced the double blind setting where, in addition to source distributions, dependencies between components are not restricted by any parametric model. In the presence of these two nuisance parameters (densities of activity level and underlying signal), they proposed an algorithm based on lagged 4-th order cumulants. Although their algorithm works well in many cases, it fails if (i) all v_i's have similar temporal structure, or (ii) there exist no temporal dependencies in v_i's. Furthermore it also suffers from outliers.

Extending the semiparametric approach (Amari and Cardoso, 1997) under variance dependencies, we investigated estimating functions for the variance-dependent BSS model. In particular, we proved that the quasi maximum likelihood estimator is derived from an estimating function, and is hence consistent regardless of the true nuisance densities (which satisfy certain mild conditions). We also analyzed other ICA algorithms within the framework of (asymptotic) estimating functions and showed that many of them can separate sources with coherent variances. This is in contrast to previous understanding of the mechanisms underlying ICA algorithms. Theoretically we have shown that at least asymptotically all BSS algorithms except for TDSEP/SOBI have the local consistency, thus they should succeed on a given mixed data. However, local consistency does not necessarily guarantee global convergence to the true solution and we sometimes see significant deviations from this expectation in practice. The algorithmic failures are due to many local optima and more importantly due to numerical stability and convergence issues.

Although almost all ICA and BSS algorithms could not give correct answers in the numerical experiment of Hyvärinen and Hurri (2004), we showed here that this was mainly a matter of the specific choice of the data set. In fact, most ICA and BSS algorithms also work well in many other benchmark examples that use dependent data. In particular, we carried out two experiments with highly variance-dependent speech signals. Despite the dependence typically found in speech, most ICA and BSS algorithms yield excellent separation results and our theoretical analysis can help to understand the reason for this fact. We conjecture that it is not the coarse amplitude structure (e.g. from dependence) that matters for BSS but the statistical fine structure of the signals.

In this paper, we only tested existing ICA and BSS algorithms and pointed out that some of them are applicable to the variance-dependent BSS model. Future research will go one step further and construct more efficient or robust semiparametric algorithms. Note also that in practice, it is important to analyze how to select the best BSS method for a specific, say, variance-dependent data set. We think that suitable methods might be developed along the lines of Meinecke et al. (2002) or Harmeling et al. (2004).

Acknowledgments

The authors acknowledge A. Ziehe, S. Harmeling, F. Meinecke and N. Murata for valuable discussions, and A. Hyvärinen and three anonymous reviewers for useful comments to improve this paper. We furthermore thank the PASCAL Network of Excellence (EU #506778) and DFG (SFB 618) for partial funding.

Appendix A. Comments on Other Selected BSS Algorithms

We will discuss in the following the local consistency of ICA/BSS algorithms except the quasi maximum likelihood method.

A.1 FastICA

FastICA is one of the standard algorithms for blind source separation. Let $u(t) = C^{-1/2}x(t)$ be the whitened data, where $C = \frac{1}{T}\sum_{t=1}^{T} x(t)x^\top(t)$ is the sample covariance. FastICA gives the demixing matrix $W = (w_1, \ldots, w_n)^\top$ which maximizes the total non-Gaussianity

$$\sum_{i=1}^{n} \frac{1}{T} \sum_{t=1}^{T} G\{w_i^\top u(t)\}$$

under the orthogonality condition $WW^\top = I$. We use, in the following the notation W for the demixing matrix after whitening in order to distinguish it from the total demixing matrix $B = WC^{-1/2}$ including whitening process. Here G is a nonlinear function which is introduced to approximate the negentropy (Hyvärinen et al., 2001b). By solving the constrained optimization problem, we see that the estimator of W must satisfy the estimating equation

$$\sum_{t=1}^{T} \Big[y(t)y^\top(t) - I + y(t)g^\top\{y(t)\} - g\{y(t)\}y^\top(t) \Big] = 0 \tag{37}$$

where $y(t) = Wu(t)$. If we write the total demixing matrix as $B = WC^{-1/2}$, $y(t)$ can be expressed as $Bx(t)$. The vector function $g(y)$ consists of the derivatives $g(y_i) = G'(y_i)$, that is, $g(y) = (g(y_1), \ldots, g(y_n))^\top$. The functions (12) and (13) are also used as the function g. We remark that the equation (37) is equivalent to

$$\sum_{t=1}^{T} \Big[y(t)g^\top\{y(t)\} - g\{y(t)\}y^\top(t) \Big] = 0, \tag{38}$$

$$\sum_{t=1}^{T} \Big[y(t)y^\top(t) - I \Big] = 0, \tag{39}$$

because the left hand side of (38) is antisymmetric, while that of (39) is symmetric. If we determine the scales of the sources such that

$$\mathrm{E}\left[\sum_{t=1}^{T}\{b_i^{\top}x(t)\}^2\middle| B,\rho_z,\rho_V\right]=T,\qquad i=1,\ldots,n, \tag{40}$$

then it is easy to show that the expectations of the left hand side of (38) and (39) vanish regardless of the nuisance functions ρ_z and ρ_V, in the same way as for the quasi maximum likelihood method. This means that the left hand side of (37) satisfies the unbiasedness condition (14) of estimating functions. If the other regularity conditions hold, it becomes an estimating function and the estimator $\widehat{B}$ derived from it converges to the correct demixing matrix $B^*=(A^*)^{-1}$ with a permutation matrix P and a diagonal matrix D. Although the estimating function is similar to that of the quasi maximum likelihood, FastICA algorithm is based on the Newton's algorithm, and therefore, it has globally more stable dynamics than the natural gradient learning.

A.2 The Double Blind Algorithm by Hyvärinen and Hurri (2004)

Hyvärinen and Hurri (2004) proposed an algorithm for separating sources under the double blind situation. The estimator is obtained by maximizing

$$J(W)=\sum_{i,j}\left[\widehat{\mathrm{cov}}\{y_i^2(\cdot),y_j^2(\cdot-\Delta t)\}\right]^2,$$

under the orthogonality condition $WW^{\top}=I$, where

$$\widehat{\mathrm{cov}}\{y_i^2(\cdot),y_j^2(\cdot-\Delta t)\}=\frac{1}{T-\Delta t}\sum_{t=\Delta t+1}^{T}y_i^2(t)y_j^2(t-\Delta t)-1.$$

Let us assume that

$$\begin{aligned}
&\widehat{\mathrm{cum}}\{s_i(\cdot),s_j(\cdot),s_k(\cdot-\Delta t),s_l(\cdot-\Delta t)\}\\
:=\;&\frac{1}{T-\Delta t}\sum_{t=\Delta t+1}^{T}s_i(t)s_j(t)s_k(t-\Delta t)s_l(t-\Delta t)-\frac{1}{T^2}\sum_{t=1}^{T}s_i(t)s_j(t)\sum_{t=1}^{T}s_k(t)s_l(t)\\
&\quad-\frac{1}{(T-\Delta t)^2}\sum_{t=\Delta t+1}^{T}s_i(t)s_k(t-\Delta t)\sum_{t=\Delta t+1}^{T}s_j(t)s_l(t-\Delta t)\\
&\quad-\frac{1}{(T-\Delta t)^2}\sum_{t=\Delta t+1}^{T}s_i(t)s_l(t-\Delta t)\sum_{t=\Delta t+1}^{T}s_j(t)s_k(t-\Delta t)\\
=\;&\begin{cases}K_{ik}+o_p(1), & i=j,\ k=l\\ o_p(1), & \text{otherwise}\end{cases}
\end{aligned}$$

that is, the empirical cumulants of the source signal $s(t)=(A^*)^{-1}x(t)$ converge to their expectation, where

$$K_{ij}=\frac{1}{T-\Delta t}\sum_{t=\Delta t+1}^{T}\mathrm{E}\left[s_i^2(t)s_j^2(t-\Delta t)\right]-\frac{1}{T^2}\sum_{t=1}^{T}\mathrm{E}\left[s_i^2(t)\right]\sum_{t=1}^{T}\mathrm{E}\left[s_j^2(t)\right].$$

By ignoring higher-order terms, we get

$$J = \sum_{i,j,k,l} (q_{ik}^2 K_{kl} q_{jl}^2)^2$$

where $Q = (q_{ij}) = BA^*$ and $B = WC^{-1/2}$ indicates the demixing matrix without whitening. Provided that the matrix $K = (K_{ij})$ is non-singular, the quantity J is maximized when Q is a signed permutation matrix, that is, by maximizing the criterion J we can estimate the true demixing matrix $B^* = (A^*)^{-1}$ up to signed permutation matrices. This also means that the algorithm does not work if there is no temporal covariance dependencies (for example, the data sets 'exp_supG' and 'uni_subG' in our experiment), or all sources have exactly same temporal covariance dependencies (for example, the data sets 'com_supG' and 'com_subG' in our experiment).

Although the authors have already given its validity as mentioned above, we will check its estimating equation. By solving the constrained optimization problem, we see that the estimator is obtained from the estimating equation

$$\widehat{F}(X, \widehat{B}) = 0, \tag{41}$$

where

$$\begin{aligned} \widehat{F}_{ij}(X,B) &= \sum_{t=1}^{T} \{y_i(t)y_j(t) - \delta_{ij}\} + \sum_{t=\Delta t+1}^{T} \left[\sum_l (\widehat{K}_{il} - \widehat{K}_{jl}) y_l^2(t-\Delta t) y_i(t) y_j(t) \right. \\ &\qquad \left. + \sum_l (\widehat{K}_{li} - \widehat{K}_{lj}) y_l^2(t) y_i(t-\Delta t) y_j(t-\Delta t) \right]. \end{aligned} \tag{42}$$

and $\widehat{K}_{ij} = \widehat{\mathrm{cov}}\{y_i^2(\cdot), y_j(\cdot - \Delta t)\}$. By replacing $\widehat{K}_{ij}$ with K_{ij}, let us define the function

$$\begin{aligned} F_{ij}(X,B) &= \sum_{t=1}^{T} \{y_i(t)y_j(t) - \delta_{ij}\} + \sum_{t=\Delta t+1}^{T} \left[\sum_l (K_{il} - K_{jl}) y_l^2(t-\Delta t) y_i(t) y_j(t) \right. \\ &\qquad \left. + \sum_l (K_{li} - K_{lj}) y_l^2(t) y_i(t-\Delta t) y_j(t-\Delta t) \right]. \end{aligned} \tag{43}$$

Suppose that $F(X,B)$ is an estimating function which fulfills $F(X,B) = O_p(T^{1/2})$, when B is the true parameter. If the function $\widehat{F}(X,B)$ satisfies

$$\widehat{F}(X, \widetilde{B}) = F(X, \widetilde{B}) + o_p(T^{1/2}) \tag{44}$$

for any $\widetilde{B}$ such that $\|\widetilde{B} - B\| = O(T^{-1/2})$, it can be shown that the residual does not matter to the asymptotic property of the estimator and the solution $\widehat{B}$ of (41) is asymptotically equivalent to that of the equation $F(X,B) = 0$ (see Cardoso, 1997). In fact, we can prove (44) under mild conditions, that is, the difference between the functions (42) and (43) can be neglected. Therefore, we will check whether $F(X,B)$ actually satisfies the conditions of estimating functions. If we take the constraints (40) to determine the scales of the sources, the unbiasedness condition (14) follows from uncorrelatedness of the sources and

$$\begin{aligned} &\sum_{t=\Delta t+1}^{T} \sum_l (K_{il} - K_{jl}) \, \mathrm{E}\left[y_l^2(t-\Delta t) y_i(t) y_j(t) \right] \\ &\quad = \sum_{t=\Delta t+1}^{T} \sum_l (K_{il} - K_{jl}) \, \mathrm{E}\left[v_l^2(t-\Delta t) v_i(t) v_j(t) \right] \, \mathrm{E}\left[z_l^2(t-\Delta t) \right] \, \mathrm{E}\left[z_i(t) z_j(t) \right] = 0, \end{aligned}$$

$$\begin{aligned}
&\sum_{t=\Delta t+1}^{T} \sum_{l}(K_{li}-K_{lj})\,\mathrm{E}\left[y_l^2(t)y_i(t-\Delta t)y_j(t-\Delta t)\right] \\
&= \sum_{t=\Delta t+1}^{T} \sum_{l}(K_{li}-K_{lj})\,\mathrm{E}\left[v_l^2(t)v_i(t-\Delta t)v_j(t-\Delta t)\right]\,\mathrm{E}\left[z_l^2(t)\right]\,\mathrm{E}\left[z_i(t-\Delta t)z_j(t-\Delta t)\right] \\
&= 0.
\end{aligned}$$

We remark that the expectations are taken with respect to $p(X|B,\rho_z,\rho_V)$, and therefore $y(t)=Bx(t)=s(t)$ holds. If the other regularity condition holds, $\widehat{F}(X,B)$ turns out to be an asymptotic estimating function which is asymptotically equivalent to an estimating function and the estimator $\widehat{B}$ converges to the correct demixing matrix $B^*=(A^*)^{-1}$.

A.3 JADE

Although in a rigorous sense, the asymptotic properties of JADE should be analyzed as in the previous section (see also Cardoso, 1997), its consistency can be shown more easily (as suggested by one of the anonymous reviewers). Suppose that the contrast function of JADE

$$J_{\mathrm{JADE}}(W)=\sum_{ijkl\neq iikl}|\widehat{\mathrm{cum}}(y_i,y_j,y_k,y_l)|^2$$

uniformly converges to the ideal contrast function

$$J^*_{\mathrm{JADE}}(W)=\sum_{ijkl\neq iikl}|\mathrm{cum}(y_i,y_j,y_k,y_l)|^2$$

on the set of orthogonal matrices W such that $WW^\top=I$, where $\widehat{\mathrm{cum}}$ and cum denote the empirical and the expected cumulant tensor, respectively. Then, the minimum of the $J_{\mathrm{JADE}}(W)$ converges to that of $J^*_{\mathrm{JADE}}(W)$. If W is the true demixing matrix and y_i's are extracted signals with W, the components $K_{ijkl}:=\mathrm{cum}(y_i,y_j,y_k,y_l)$ of the expected cumulant are zero except for $i=j=k=l$ or $i=j\neq k=l$ or $i=l\neq j=k$. Thus, one needs only to show that the estimating equation is associated to the minimization of $2\sum_{i\neq j}|\mathrm{cum}(y_i,y_j,y_i,y_j)|^2$ under the orthogonality constraints which is satisfied when y_i equals the true sources (up to a scaling and a permutation). The estimating equation is

$$\sum_{t=1}^{T}\left\{\mathrm{E}[y_i(t)y_j(t)]-\delta_{ij} \right. \\ \left. +\sum_{k\neq i}K_{ikik}\mathrm{E}[y_k^2(t)y_i(t)y_j(t)]-\sum_{k\neq j}K_{jkjk}\mathrm{E}[y_k^2(t)y_i(t)y_j(t)]\right\}=0, \qquad (45)$$

which can be seen to be satisfied, when y_i's equal to the true sources. We remark that the same formula as (45) can be obtained after the rigorous analysis. The function which is associated with the asymptotic estimating function (see (43)) becomes

$$F_{ij}(X,W)=\sum_{t=1}^{T}\left\{y_i(t)y_j(t)-\delta_{ij}+\sum_{k\neq i}K_{ikik}y_k^2(t)y_i(t)y_j(t)-\sum_{k\neq j}K_{jkjk}y_k^2(t)y_i(t)y_j(t)\right\}.$$

A.4 TDSEP/SOBI

Let us define lagged covariance matrices of $x(t)$

$$\begin{aligned} R(\Delta t) &= \frac{1}{T}\sum_{t=\Delta t+1}^{T} \mathrm{E}\left[x(t)x^\top(t-\Delta t)\right] \\ &= A^*\left\{\frac{1}{T}\sum_{t=\Delta t+1}^{T} \mathrm{E}\left[s(t)s^\top(t-\Delta t)\right]\right\}(A^*)^\top. \end{aligned}$$

When the sources s_i's are mutually independent and have temporal covariance structure, the demixing matrix $PD(A^*)^{-1}$ can diagonalize all lagged covariance matrices $R(\Delta t)$, where P is a permutation matrix and D is a diagonal matrix. This property has been used in blind separation methods with second order statistics (Tong et al., 1991; Belouchrani et al., 1997; Ziehe and Müller, 1998).

In the variance-dependent BSS model, for any i, j, t and $\Delta t \geq 1$,

$$\mathrm{E}\left[s_i(t)s_j(t-\Delta t)\right] = \mathrm{E}\left[v_i(t)v_j(t-\Delta t)\right]\mathrm{E}\left[z_i(t)z_j(t-\Delta t)\right] = 0.$$

Therefore, $R(\Delta t) = 0$ for $\Delta t \geq 1$, that is, we cannot get any information about the mixing matrix A from lagged covariance matrices $R(\Delta t)$. This is why TDSEP does not work for this model.

References

S. Amari. *Differential Geometrical Methods in Statistics*. Springer Verlag, Berlin, 1985.

S. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998.

S. Amari and J.-F. Cardoso. Blind source separation—semiparametric statistical approach. *IEEE Trans. on Signal Processing*, 45(11):2692–2700, 1997.

S. Amari, T.-P. Chen, and A. Cichocki. Stability analysis of adaptive blind source separation. *Neural Networks*, 10(8):1345–1351, 1997.

S. Amari, T.-P. Chen, and A. Cichocki. Nonholonomic orthogonal learning algorithms for blind source separation. *Neural Computation*, 12:1463–1484, 2000.

S. Amari, A. Cichocki, and H. H. Yang. A new learning algorithm for blind source separation. In *Advances in Neural Information Processing Systems 8*, pages 757–763. MIT Press, 1996.

S. Amari and M. Kawanabe. Estimating functions in semiparametric statistical models. In I. V. Basawa et al., editor, *Selected Proceedings of the Symposium on Estimating Functions*, volume 32 of *IMS Lecture Notes–Monograph Series*, pages 65–81, 1997a.

S. Amari and M. Kawanabe. Information geometry of estimating functions in semiparametric statistical models. *Bernoulli*, 3:29–54, 1997b.

F. R. Bach and M. I. Jordan. Tree-dependent component analysis. In *Uncertainty in Artificial Intelligence: Proceedings of the Eighteenth Conference (UAI-2002)*, 2002.

A. J. Bell and T. J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7:1129–1159, 1995.

A. Belouchrani, K. Abed Meraim, J.-F. Cardoso, and E. Moulines. A blind source separation techinique based on second order statistics. *IEEE Trans. on Signal Processing*, 45(2):434–444, 1997.

P. J. Bickel, C. A. J. Klaassen, Y. Ritov, and J. A. Wellner. *Efficient and Adaptive Estimation for Semiparamtric Models*. John Hopkins Univ. Press, Baltimore, MD, 1993.

J.-F. Cardoso. Estimating equations for source separation. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP'97)*, volume 5, pages 3449–3452, Munich, Germany, 1997.

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

J.-F. Cardoso. Multidimensional independent component analysis. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP'98)*, Seattle, WA, 1998b.

J.-F. Cardoso. Independent component analysis of the cosmic microwave background. In *Proc.Int. Symposium on Independent Component Analysis and Blind Signal Separation (ICA2003)*, pages 1111–1116, Nara, Japan, 2003.

J.-F. Cardoso and A. Souloumiac. Blind beamforming for non Gaussian signals. *IEE Proceedings-F*, 140:362 – 370, 1993.

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

V. P. Godambe. Conditional likelihood and unconditional optimum estimating equations. *Biometrika*, 63:277–284, 1976.

V. P. Godambe, editor. *Estimating Functions*. Oxford Univ. Press, New York, 1991.

S. Harmeling, F. Meinecke, and K.-R. Müller. Injecting noise for analysing the stability of ica components. *Signal Processing*, 84:255–266, 2004.

P. J. Huber. *Robust Statistics*. Wiley, New York, 1981.

A. Hyvärinen. Fast and robust fixed-point algorithms for independent component analysis. *IEEE Trans. on Neural Networks*, 10(3):626–634, 1999.

A. Hyvärinen, P. O. Hoyer, and M. Inki. Topographic independent component analysis. *Neural Computation*, 13(7):1527–1558, 2001a.

A. Hyvärinen and J. Hurri. Blind separation of sources that have spatiotemporal variance dependencies. *Signal Processing*, 84, 2004. 247–254.

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

A. Hyvärinen and E. Oja. A fast fixed-point algorithm for independent component analysis. *Neural Computation*, 9(7):1483–1492, 1997.

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

M. Kawanabe and K.-R. Müller. Estimating functions for blind separation when source have variance-dependencies. In C. G. Puntonet and A. Prieto, editors, *Proc. Int. Workshop on Independent Component Analysis and Blind Signal Separation (ICA2004)*, pages 136 – 143, Granada, Spain, 2004.

M. Kawanabe and N. Murata. Independent component analysis in the presence of Gaussian noise based on estimating functions. In *Proc. Int. Workshop on Independent Component Analysis and Blind Signal Separation (ICA2000)*, pages 279–284, Helsinki, Finland, 2000.

S. Makeig, T-P. Jung, D. Ghahremani, A. J. Bell, and T. J. Sejnowski. Blind separation of event-related brain responses into independent components. *Proc. Natl. Acad. Sci. USA*, 94:10979–10984, 1997.

F. Meinecke, S. Harmeling, and K.-R. Müller. Robust ICA for super-Gaussian sources. In C. G. Puntonet and A. Prieto, editors, *Proc. Int. Workshop on Independent Component Analysis and Blind Signal Separation (ICA2004)*, pages 217 – 224, Granada, Spain, 2004.

F. Meinecke, A. Ziehe, M. Kawanabe, and K.-R. Müller. A resampling approach to estimate the stability of one- or multidimensional independent components. *IEEE Transactions on Biomedical Engineering*, 49(12):1514–1525, 2002.

F. Meinecke, A. Ziehe, J. Kurths, and K.-R. Müller. Measuring phase synchronization of superimposed signals. *Physical Review Letters*, 2005.

K.-R. Müller, N. Murata, A. Ziehe, and S.-I. Amari. *On-line learning in Switching and Drifting environments with application to blind source separation*, pages 93–110. On-line learning in neural networks. Cambridge University Press, 1998.

K.-R. Müller, R. Vigário, F. Meinecke, and A. Ziehe. Blind source separation techniques for decomposing evoked brain signals. *International Journal of Bifurcation and Chaos*, 14(2):773–791, 2004.

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.

B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field by learning a sparse code for natural images. *Nature*, 381:607–609, 1996.

L. Parra and C. Spence. Convolutive blind source separation of non-stationary sources. *IEEE Trans. on Speech and Audio Processing*, 8:320–327, 2000.

D.-T. Pham. Blind separation of instantaneous mixture sources via an independent component analysis. *IEEE Trans. on Signal Processing*, 44(11):2768–2779, 1996.

D.-T. Pham and J.-F. Cardoso. Blind separation of instantaneous mixtures of non-stationary sources. In *Proc. Int. Workshop on Independent Component Analysis and Blind Signal Separation (ICA2000)*, pages 187–193, Helsinki, Finland, 2000.

D.-T. Pham and P. Garrat. Blind separation of mixture of independent sources through a quasi-maximum likelihood approach. *IEEE Trans. on Signal Processing*, 45(7):1712–1725, 1997.

D.-T. Pham, P. Garrat, and C. Jutten. Separation of a mixture of independent sources through a maximum likelihood approach. In *Proc. EUSIPCO*, pages 771–774, 1992.

M. Sørensen. On asymptotics of estimating functions. *Brazilian Journal of Probability and Statistics*, 13:111–136, 1999.

F. J. Theis. Uniqueness of complex and multidimensional independent component analysis. *Signal Processing*, 84(5):951–956, 2004.

H.-Lan Nguyen Thi and Ch. Jutten. Blind source separation for convolutive mixtures. *Signal Processing*, 45:209–229, 1995.

L. Tong, R. W. Liu, V. Soon, and Y. F. Huang. Indeterminacy and identifiability of blind identification. *IEEE Transactions on Circuits and Systems*, 38:499–509, 1991.

H. Valpola, M. Harva, and J. Karhunen. Hierachical models of variance sources. In *Proc.Int. Symposium on Independent Component Analysis and Blind Signal Separation (ICA2003)*, Nara, Japan, 2003.

R. N. Vigario. Extraction of ocular artefacts from EEG using independent component analysis. *Electroencephalography and clinical Neurophysiology*, 103:395–404, 1997.

H. H. Yang and S.-I. Amari. Adaptive on-line learning algorithms for blind separation: Maximum entropy and minimum mutual information. *Neural Computation*, 9(7):1457–1482, 1997.

A. Ziehe and K.-R. Müller. TDSEP – an efficient algorithm for blind separation using time structure. In L. Niklasson, M. Bodén, and T. Ziemke, editors, *Proceedings of the 8th International Conference on Artificial Neural Networks (ICANN'98)*, pages 675 – 680, 1998.

A. Ziehe, K.-R. Müller, G. Nolte, B.-M. Mackert, and G. Curio. Artifact reduction in magnetoneurography based on time-delayed second order correlations. *IEEE Transactions on biomedical Engineering*, 47:75–87, 2000.

Journal of Machine Learning Research 6 (2005) 483–502 Submitted 9/04; Revised 1/05; Published 4/05

Characterization of a Family of Algorithms for Generalized Discriminant Analysis on Undersampled Problems

Jieping Ye JIEPING@CS.UMN.EDU
Department of Computer Science
University of Minnesota
Minneapolis, MN 55455, USA

Editor: Bin Yu

Abstract

A generalized discriminant analysis based on a new optimization criterion is presented. The criterion extends the optimization criteria of the classical Linear Discriminant Analysis (LDA) when the scatter matrices are singular. An efficient algorithm for the new optimization problem is presented.

The solutions to the proposed criterion form a family of algorithms for generalized LDA, which can be characterized in a closed form. We study two specific algorithms, namely Uncorrelated LDA (ULDA) and Orthogonal LDA (OLDA). ULDA was previously proposed for feature extraction and dimension reduction, whereas OLDA is a novel algorithm proposed in this paper. The features in the reduced space of ULDA are uncorrelated, while the discriminant vectors of OLDA are orthogonal to each other. We have conducted a comparative study on a variety of real-world data sets to evaluate ULDA and OLDA in terms of classification accuracy.

Keywords: dimension reduction, linear discriminant analysis, uncorrelated LDA, orthogonal LDA, singular value decomposition

1. Introduction

Many machine learning and data mining problems involve data in very high-dimensional spaces. We consider dimension reduction of high-dimensional, undersampled data, where the data dimension is much larger than the sample size. The high-dimensional, undersampled problems frequently occur in many applications including information retrieval (Berry et al., 1995; Deerwester et al., 1990), face recognition (Belhumeur et al., 1997; Swets and Weng, 1996; Turk and Pentland, 1991) and microarray data analysis (Dudoit et al., 2002).

Linear Discriminant Analysis (LDA) is a classical statistical approach for feature extraction and dimension reduction (Duda et al., 2000; Fukunaga, 1990; Hastie et al., 2001). LDA computes the optimal transformation (projection), which minimizes the within-class distance (of the data set) and maximizes the between-class distance simultaneously, thus achieving maximum discrimination. The optimal transformation can be readily computed by applying an eigen-decomposition on the scatter matrices of the given training data set. However classical LDA requires the total scatter matrix to be nonsingular. In many applications such as information retrieval, face recognition, and microarray data analysis, all scatter matrices in question can be singular since the data points are from a very high-dimensional space and in general the sample size does not exceed this dimension. This is known as the *singularity* or *undersampled* problems (Krzanowski et al., 1995).

In recent years, many approaches have been brought to bear on such high-dimensional, undersampled problems, including PCA+LDA (Belhumeur et al., 1997; Swets and Weng, 1996; Zhao et al., 1999), Regularized LDA (Friedman, 1989), Penalized LDA (Hastie et al., 1995), Pseudo-inverse LDA (Fukunaga, 1990; Raudys and Duin, 1998; Skurichina and Duin, 1996, 1999), and LDA/GSVD (Howland et al., 2003; Ye et al., 2004b). More details will be given in Section 2.

1.1 Contribution

In this paper, we present a new optimization criterion for discriminant analysis, which is applicable to undersampled problems. A detailed mathematical derivation for the proposed optimization problem is presented in Section 3.

The solutions to the proposed criterion characterize a family of algorithms for generalized LDA. Among the family of algorithms, we study two specific ones in detail, namely Uncorrelated LDA (ULDA) and Orthogonal LDA (OLDA). ULDA was developed in the past for feature extraction and dimension reduction, whereas OLDA is a novel LDA based algorithm proposed in this paper.

ULDA was recently proposed for extracting feature vectors with uncorrelated attributes (Jin et al., 2001a,b). A more recent work (Ye et al., 2004a) showed that classical LDA is equivalent to ULDA, in the sense that both classical LDA and ULDA produce the same transformation matrix when the total scatter matrix is nonsingular. Based on this equivalence, an efficient algorithm was presented in (Ye et al., 2004a) for computing the optimal discriminant vectors of ULDA. Interestingly, the solution in (Ye et al., 2004a) is a special case of the solutions to the proposed criterion in this paper (See Section 4).

OLDA is a novel dimension reduction algorithm proposed in this paper. The key property of OLDA is that the discriminant vectors of OLDA are orthogonal to each other, i.e., the transformation matrix of OLDA is orthogonal. There has been some early development on LDA based algorithms with orthogonal transformations. The algorithm is known as Foley-Sammon LDA (FSLDA). FSLDA was first proposed by Foley and Sammon for two-class problems (Foley and Sammon, 1975). It was then extended to the multi-class problems by Duchene and Leclercq (Duchene and Leclerq, 1988). The OLDA algorithm proposed in this paper provides an alternative, but simple and efficient way for computing orthogonal transformations in the framework of LDA.

We have conducted a comparative study on a variety of real-world data sets, including text documents, face images, and gene expression data to evaluate ULDA and OLDA, and compare with Regularized LDA (RLDA). Results have shown that OLDA is competitive with ULDA and RLDA in terms of classification accuracy.

The main contributions of this paper include:

- A generalization of the classical discriminant analysis to small sample size data using a new criterion, where the nonsingularity of the scatter matrices is not required;

- Mathematical derivation of the solutions to the new optimization criterion, based on the simultaneous diagonalization of the scatter matrices;

- Characterization of a family of algorithms for generalized LDA based on the proposed criterion and derivation of two specific algorithms, namely ULDA and OLDA.

1.2 Organization

The rest of the paper is organized as follows: We review classical LDA and several extensions in Section 2. A generalization of classical LDA using the new criterion is presented in Section 3. Two specific solutions to the proposed criterion, namely ULDA and OLDA, are discussed in Section 4. Experimental results are presented in Section 5. Finally, concluding discussions and future directions are presented in Section 6.

1.3 Notation

For convenience, we present in Table 1 the important notations used in the rest of the paper.

Notation	Description	Notation	Description
n	sample size	m	number of variables (dimensions)
k	number of classes	A	data matrix
A_i	data matrix of the i-th class	n_i	size of the i-th class
$c^{(i)}$	centroid of the i-th class	c	global centroid of the training set
S_b	between-class scatter matrix	S_w	within-class scatter matrix
S_t	total scatter matrix	G	transformation matrix
q	rank of the matrix S_b	t	rank of the matrix S_t

Table 1: Important notations used in the paper

2. Classical Discriminant Analysis

Given a data matrix $A \in \mathbb{R}^{m\times n}$, classical linear discriminant analysis computes a linear transformation $G \in \mathbb{R}^{m\times \ell}$ that maps each column a_i of A in the m-dimensional space to a vector y_i in the ℓ-dimensional space:

$$G : a_i \in \mathbb{R}^m \to y_i = G^T a_i \in \mathbb{R}^\ell \ (\ell < m).$$

Assume the original data is already clustered and ordering is imposed on the samples based on cluster membership. The goal of classical LDA is to find a transformation G such that the cluster structure of the original high-dimensional space is preserved in the reduced-dimensional space. Let the data matrix A be partitioned into k classes as $A = [A_1, \cdots, A_k]$, where $A_i \in \mathbb{R}^{m\times n_i}$, and $\sum_{i=1}^k n_i = n$.

In discriminant analysis (Fukunaga, 1990), three scatter matrices, called *within-class*, *between-class* and *total* scatter matrices are defined as follows:

$$\begin{aligned} S_w &= \frac{1}{n}\sum_{i=1}^k \sum_{x\in A_i} (x - c^{(i)})(x - c^{(i)})^T, \\ S_b &= \frac{1}{n}\sum_{i=1}^k \sum_{x\in A_i} (c^{(i)} - c)(c^{(i)} - c)^T = \frac{1}{n}\sum_{i=1}^k n_i (c^{(i)} - c)(c^{(i)} - c)^T, \\ S_t &= \frac{1}{n}\sum_{j=1}^n (a_j - c)(a_j - c)^T, \end{aligned} \tag{1}$$

where the *centroid* $c^{(i)}$ of the i-th class is defined as $c^{(i)} = \frac{1}{n_i} A_i e^{(i)}$ with

$$e^{(i)} = (1, 1, \cdots, 1)^T \in \mathbb{R}^{n_i},$$

and the *global centroid* c is defined as $c = \frac{1}{n} Ae$ with

$$e = (1, 1, \cdots, 1)^T \in \mathbb{R}^{n}.$$

It is easy to verify that $S_t = S_b + S_w$.

Define the matrices

$$\begin{aligned} H_w &= \frac{1}{\sqrt{n}}[A_1 - c^{(1)}(e^{(1)})^T, \cdots, A_k - c^{(k)}(e^{(k)})^T], \\ H_b &= \frac{1}{\sqrt{n}}[\sqrt{n_1}(c^{(1)} - c), \cdots, \sqrt{n_k}(c^{(k)} - c)], \\ H_t &= \frac{1}{\sqrt{n}}(A - ce^T). \end{aligned} \tag{2}$$

Then S_w, S_b, and S_t can be expressed as

$$S_w = H_w H_w^T, \quad S_b = H_b H_b^T, \quad S_t = H_t H_t^T.$$

The *traces* of the two scatter matrices S_w and S_b can be computed as follows:

$$\begin{aligned} \text{trace}(S_w) &= \frac{1}{n}\sum_{i=1}^{k}\sum_{x \in A_i}(x - c^{(i)})^T(x - c^{(i)}) = \frac{1}{n}\sum_{i=1}^{k}\sum_{x \in A_i}||x - c^{(i)}||^2 \\ \text{trace}(S_b) &= \frac{1}{n}\sum_{i=1}^{k} n_i (c^{(i)} - c)^T(c^{(i)} - c) = \frac{1}{n}\sum_{i=1}^{k} n_i ||c^{(i)} - c||^2. \end{aligned} \tag{3}$$

Hence, trace(S_w) measures the within-class cohesion, while trace(S_b) measures the between-class separation.

In the lower-dimensional space resulting from the linear transformation G, the scatter matrices become

$$S_w^L = G^T S_w G, \quad S_b^L = G^T S_b G, \quad S_t^L = G^T S_t G. \tag{4}$$

An optimal transformation G would maximize trace(S_b^L) and minimize trace(S_w^L) simultaneously, which is equivalent to maximizing trace(S_b^L) and minimizing trace(S_t^L) simultaneously, since $S_t^L = S_w^L + S_b^L$. A common optimization in classical discriminant analysis (Fukunaga, 1990) is

$$G = \arg\max_{G} \left\{ \text{trace}((S_t^L)^{-1} S_b^L) \right\}. \tag{5}$$

The optimization problem in Eq. (5) is equivalent to finding all the eigenvectors that satisfy $S_b x = \lambda S_t x$, for $\lambda \neq 0$ (Fukunaga, 1990). The solution can be obtained by applying an eigen-decomposition on the matrix $S_t^{-1} S_b$, if S_t is nonsingular. There are at most $k-1$ eigenvectors corresponding to nonzero eigenvalues, since the rank of the matrix S_b is bounded from above by $k-1$. Therefore, the reduced dimension by classical LDA is at most $k-1$. A stable way to solve this eigen-decomposition problem is to apply Singular Value Decomposition (SVD) (Golub and Loan, 1996) on the scatter matrices. Details can be found in (Swets and Weng, 1996).

Assuming normal distribution for each class with the common covariance matrix, classification based on maximum likelihood estimation results in a nearest class centroid rule, where the distance is measured in terms of the within-class Mahalanobis distance (Hastie et al., 2001). Assuming equal prior for all classes for simplicity, a test point h is classified as class j if

$$(h-c^{(j)})^T S_w^{-1}(h-c^{(j)}) \tag{6}$$

is minimized over $j=1,\cdots,k$. It was shown in (Hastie et al., 1995) that

$$\arg\min_j\{(h-c^{(j)})^T S_w^{-1}(h-c^{(j)})\} = \arg\min_j\{||G^T(h-c^{(j)})||^2\}, \tag{7}$$

where G is the optimal transformation solving the optimization problem in Eq. (5). Thus, classical LDA is equivalent to maximum likelihood classification assuming normal distribution for each class with the common covariance matrix. When the dimension m is much larger than the number of classes k, classification using the reduced representation, i.e., classification based on $\arg\min_j\{G^T(h-c^{(j)})\}$ may give considerable savings (Hastie et al., 1995).

Although relying on heavy assumptions which are not true in many applications, LDA has been proven to be effective. This is mainly due to the fact that a simple, linear model is more robust against noise, and most likely will not overfit. Generalization of LDA by fitting Gaussian mixtures to each class has been studied in (Hastie and Tibshirani, 1996).

Note that classical discriminant analysis requires the total scatter matrix S_t to be nonsingular, which may not hold for undersampled data. Several extensions, including two-stage PCA+LDA, Regularized LDA, Penalized LDA, Pseudo-inverse LDA, and LDA/GSVD were proposed in the past to deal with the singularity problems as follows.

A common way to deal with the singularity problems is to apply an intermediate dimension reduction stage such as PCA to reduce the dimension of the original data before classical LDA is applied. The algorithm is known as PCA+LDA (Belhumeur et al., 1997; Swets and Weng, 1996; Zhao et al., 1999). In this two-stage PCA+LDA algorithm, the discriminant stage is preceded by a dimension reduction stage using PCA. The dimension of the subspace transformed by PCA is chosen such as the "reduced" total scatter matrix in the subspace is nonsingular, so that classical LDA can be applied. A limitation of this approach is that the optimal value of the reduced dimension for PCA is difficult to determine. Moreover, the PCA stage may lose some useful information for discrimination.

A simple way to deal with the singularity of S_t is to apply the idea of regularization, by adding some constant values to the diagonal elements of S_t, as $S_t+\mu I_m$, for some $\mu>0$, where I_m is an identity matrix. It is easy to verify that $S_t+\mu I_m$ is positive definite, hence nonsingular. This approach is called Regularized LDA, or RLDA in short (Friedman, 1989). Regularization is a key in the theory of splines (Wahba, 1998) and is used widely in machine learning, such as Support Vector Machines (SVM) (Vapnik, 1998). It is evident that when $\mu\to\infty$, we lose the information on S_t, while very small values of μ may not be sufficiently effective. Cross-validation is commonly applied for estimating the optimal μ. More recent studies on RLDA can be found in (Dai and Yuen, 2003; Krzanowski et al., 1995).

The Penalized LDA (PLDA) is more general than Regularized LDA. PLDA penalizes the within-class scatter matrix as $S_w+\Omega$, for some penalty matrix Ω. Ω is symmetric and positive semidefinite. The penalties are designed to produce smoothness in the discriminant functions. Details on PLDA and the choices of penalties for different applications refer to (Hastie et al., 1995).

Pseudo-inverse is commonly applied to deal with the singularity problems, which is equivalent to approximating the solution using a least-squares solution method. The use of pseudo-inverse in discriminant analysis has been studied in the past. The *Pseudo Fisher Linear Discriminant* (PFLDA) (Fukunaga, 1990; Raudys and Duin, 1998; Skurichina and Duin, 1996, 1999) is based on the pseudo-inverse of the scatter matrices. The generalization error of PFLDA was studied in (Skurichina and Duin, 1996), when the size and dimension of the training data vary. Pseudo-inverses of the scatter matrices were also studied in (Krzanowski et al., 1995). Experiments in (Krzanowski et al., 1995) showed that the pseudo-inverse based methods are competitive with RLDA and PCA+LDA.

The LDA/GSVD algorithm (Howland et al., 2003; Ye et al., 2004b) is a more recent approach. The main technique applied is the Generalized Singular Value Decomposition (GSVD) (Golub and Loan, 1996). The criterion F_0 used in (Ye et al., 2004b) is:

$$F_0(G) = \text{trace}\left((S_b^L)^+ S_w^L\right), \tag{8}$$

where $(S_b^L)^+$ denotes the pseudo-inverse of the between-class scatter matrix. The definition of pseudo-inverse, as well as its computation via SVD, can be found in Appendix A.

LDA/GSVD aims to find the optimal transformation G that minimizes $F_0(G)$, subject to the constraint that $\text{rank}(G^T H_b) = q$, where q is the rank of S_b. The above constraint is enforced to preserve the dimension of the spaces spanned by the centroids in the original and transformed spaces. The optimal solution can be obtained by applying the GSVD. One limitation of this method is the high computational cost of GSVD, especially for large and high-dimensional data sets.

An overview of LDA on undersampled problems can be found in (Krzanowski et al., 1995).

The current paper focuses on linear discriminant analysis, which applies linear decision boundary. Discriminant analysis can also be studied in the non-linear fashion, so-called kernel discriminant analysis, by using the kernel trick (Schökopf and Smola, 2002). It is desirable if the data has weak linear separability. The interested readers can find more details on kernel discriminant analysis in (Baudat and Anouar, 2000; Hand, 1982; Lu et al., 2003; Schökopf and Smola, 2002).

3. Generalization of Discriminant Analysis

Classical discriminant analysis solves an eigen-decomposition problem when S_t is nonsingular. For undersampled problems, S_t is singular, since the sample size n may be smaller than its dimension m. In this section, we define a new criterion F_1, where the nonsingularity of S_t is not required.

The new criterion F_1 is a natural extension of the classical one in Eq. (5), where the inverse of a matrix is replaced by the pseudo-inverse (Golub and Loan, 1996). While the inverse of a matrix may not exist, the pseudo-inverse of any matrix is well defined. Moreover, when the matrix is invertible, its pseudo-inverse coincides with its inverse.

The new criterion F_1 is defined as

$$F_1(G) = \text{trace}\left((S_t^L)^+ S_b^L\right). \tag{9}$$

The optimal transformation matrix G is computed so that $F_1(G)$ is maximized. Note that in the following, the matrix G in $F_1(G)$ may be omitted if it is clear from the content.

In the rest of this section, we show how to solve the above maximization problem. It is based on the simultaneous diagonalization of the three scatter matrices. Details are given below.

3.1 Simultaneous Diagonalization of Scatter Matrices

In this section, we take a closer look at the relationship among three scatter matrices S_b, S_w, and S_t, and show how to diagonalize them simultaneously.

Let $H_t = U\Sigma V^T$ be the SVD of H_t, where H_t is defined in Eq. (2), U and V are orthogonal, $\Sigma = \begin{pmatrix} \Sigma_t & 0 \\ 0 & 0 \end{pmatrix}$, $\Sigma_t \in \mathbb{R}^{t\times t}$ is diagonal, and $t = \mathrm{rank}(S_t)$. Then

$$S_t = H_t H_t^T = U\Sigma V^T V\Sigma^T U^T = U\Sigma\Sigma^T U^T = U \begin{pmatrix} \Sigma_t^2 & 0 \\ 0 & 0 \end{pmatrix} U^T. \tag{10}$$

Let $U = (U_1, U_2)$ be a partition of U, such that $U_1 \in \mathbb{R}^{m\times t}$ and $U_2 \in \mathbb{R}^{m\times(m-t)}$. Since $S_t = S_b + S_w$, we have

$$\begin{aligned} \begin{pmatrix} \Sigma_t^2 & 0 \\ 0 & 0 \end{pmatrix} &= U^T(S_b + S_w)U \\ &= \begin{pmatrix} U_1^T \\ U_2^T \end{pmatrix} S_b(U_1, U_2) + \begin{pmatrix} U_1^T \\ U_2^T \end{pmatrix} S_w(U_1, U_2) \\ &= \begin{pmatrix} U_1^T S_b U_1 & U_1^T S_b U_2 \\ U_2^T S_b U_1 & U_2^T S_b U_2 \end{pmatrix} + \begin{pmatrix} U_1^T S_w U_1 & U_1^T S_w U_2 \\ U_2^T S_w U_1 & U_2^T S_w U_2 \end{pmatrix}. \end{aligned} \tag{11}$$

It follows that $U_2^T S_b U_2 + U_2^T S_w U_2 = 0$. Therefore, $U_2^T S_b U_2 = 0$ and $U_2^T S_w U_2 = 0$, since both are positive semidefinite. We thus have $U_1^T S_b U_2 = 0$ and $U_1^T S_w U_2 = 0$, since both matrices on the right hand size of Eq. (11) are positive semidefinite. That is,

$$U^T S_b U = \begin{pmatrix} U_1^T S_b U_1 & 0 \\ 0 & 0 \end{pmatrix}, \quad U^T S_w U = \begin{pmatrix} U_1^T S_w U_1 & 0 \\ 0 & 0 \end{pmatrix}. \tag{12}$$

From Eq. (11) and Eq. (12), we have $\Sigma_t^2 = U_1^T S_b U_1 + U_1^T S_w U_1$. It follows that

$$I_t = \Sigma_t^{-1} U_1^T S_b U_1 \Sigma_t^{-1} + \Sigma_t^{-1} U_1^T S_w U_1 \Sigma_t^{-1}. \tag{13}$$

Denote $B = \Sigma_t^{-1} U_1^T H_b$ and let $B = P\tilde{\Sigma}Q^T$ be the SVD of B, where P and Q are orthogonal and $\tilde{\Sigma}$ is diagonal. Then

$$\Sigma_t^{-1} U_1^T S_b U_1 \Sigma_t^{-1} = P\tilde{\Sigma}^2 P^T = P\Sigma_b P^T,$$

where

$$\Sigma_b \equiv \tilde{\Sigma}^2 = \mathrm{diag}(\lambda_1, \cdots, \lambda_t),$$

$$\lambda_1 \geq \cdots \geq \lambda_q > 0 = \lambda_{q+1} = \cdots = \lambda_t,$$

and $q = \mathrm{rank}(S_b)$.

It follows from Eq. (13) that

$$I_t = \Sigma_b + P^T \Sigma_t^{-1} U_1^T S_w U_1 \Sigma_t^{-1} P.$$

Hence

$$P^T \Sigma_t^{-1} U_1^T S_w U_1 \Sigma_t^{-1} P = I_t - \Sigma_b \equiv \Sigma_w$$

is also diagonal.

Combining all these together, we have

$$X^T S_b X = \begin{pmatrix} \Sigma_b & 0 \\ 0 & 0 \end{pmatrix} \equiv D_b, \quad X^T S_w X = \begin{pmatrix} \Sigma_w & 0 \\ 0 & 0 \end{pmatrix} \equiv D_w, \quad X^T S_t X = \begin{pmatrix} I_t & 0 \\ 0 & 0 \end{pmatrix} \equiv D_t, \tag{14}$$

where

$$X = U \begin{pmatrix} \Sigma_t^{-1} P & 0 \\ 0 & I \end{pmatrix}. \tag{15}$$

In summary, the matrix X in Eq. (15) simultaneously diagonalizes S_b, S_w, and S_t.

3.2 Maximization of the F_1 Criterion

In this section, we derive the generalized discriminant analysis by maximizing the F_1 criterion defined in Eq. (9). The main technique applied is the simultaneous diagonalization of scatter matrices from last section. We show in this section that the solutions to the proposed criterion F_1 can be characterized as $G = X_q M$, where X_q is the matrix consisting of the first q columns of X, defined in Eq. (15), $q = \text{rank}(S_b)$, and $M \in \mathbb{R}^{q\times q}$ is an arbitrary nonsingular matrix.

We first present two lemmas. The proof of Lemma 3.1 is straightforward from standard linear algebra and a generalization of Lemma 3.2 can be found in (Edelman et al., 1998).

Lemma 3.1 *For any matrix $A \in \mathbb{R}^{m\times n}$, the following equality holds: $(A^T A)^+ = A^+ (A^+)^T$.*

Lemma 3.2 *Let $A \in \mathbb{R}^{m\times m}$ be symmetric and positive semidefinite and let x_i be the eigenvector of A corresponding to the i-th largest eigenvalue λ_i. Then, for any $M \in \mathbb{R}^{m\times s} (s \le m)$ with orthonormal columns, the following inequality holds,*

$$\textit{trace}\left(M^T A M\right) \le \lambda_1 + \cdots + \lambda_s,$$

where the equality holds if $M = [x_1, \cdots, x_s]Q$, for any orthogonal matrix $Q \in \mathbb{R}^{s\times s}$.

The main result of this section is summarized in the following theorem.

Theorem 3.1 *Let X be the matrix defined in Eq. (15) and X_q be the matrix consisting of the first q columns of X, where $q = rank(S_b)$. Then $G = X_q M$, for any nonsingular M, maximizes F_1 defined in Eq. (9).*

Proof By the simultaneous diagonalization of the three scatter matrices in Eq. (14), we have

$$\begin{aligned} S_b^L &= G^T S_b G = G^T (X^{-1})^T (X^T S_b X) X^{-1} G = \tilde{G}^T D_b \tilde{G}, \\ S_t^L &= G^T S_t G = G^T (X^{-1})^T (X^T S_t X) X^{-1} G = \tilde{G}^T D_t \tilde{G}, \end{aligned} \tag{16}$$

where $\tilde{G} = X^{-1} G$.

Let $\tilde{G} = \begin{pmatrix} G_1 \\ G_2 \end{pmatrix}$ be a partition of $\tilde{G}$ so that $G_1 \in \mathbb{R}^{t\times \ell}$ and $G_2 \in \mathbb{R}^{(m-t)\times \ell}$. It follows that

$$S_b^L = \tilde{G}^T D_b \tilde{G} = G_1^T \Sigma_b G_1, \quad S_t^L = \tilde{G}^T D_t \tilde{G} = G_1^T G_1.$$

Hence

$$F_1 = \text{trace}\left((G_1^T G_1)^+ (G_1^T \Sigma_b G_1)\right) = \text{trace}\left((G_1 G_1^+)^T \Sigma_b (G_1 G_1^+)\right),$$

where the second equality follows from Lemma 3.1.

Recall that $\Sigma_b = \text{diag}(\lambda_1, \cdots, \lambda_t)$, where $\lambda_1 \geq \cdots \geq \lambda_q > 0 = \lambda_{q+1} = \cdots = \lambda_t$. Let $G_1 = R \begin{pmatrix} \Sigma_\delta & 0 \\ 0 & 0 \end{pmatrix} S^T$ be the SVD of G_1, where R and S are orthogonal, Σ_δ is diagonal, and $\delta = \text{rank}(G_1)$. Then $G_1^+ = S \begin{pmatrix} \Sigma_\delta^{-1} & 0 \\ 0 & 0 \end{pmatrix} R^T$, and $G_1 G_1^+ = R \begin{pmatrix} I_\delta & 0 \\ 0 & 0 \end{pmatrix} R^T$. It follows that

$$\begin{aligned} F_1 &= \text{trace}\left((G_1 G_1^+)^T \Sigma_b (G_1 G_1^+)\right) = \text{trace}\left(R \begin{pmatrix} I_\delta & 0 \\ 0 & 0 \end{pmatrix} R^T \Sigma_b R \begin{pmatrix} I_\delta & 0 \\ 0 & 0 \end{pmatrix} R^T\right) \\ &= \text{trace}\left(\left(\begin{pmatrix} I_\delta & 0 \\ 0 & 0 \end{pmatrix} R^T \Sigma_b R \begin{pmatrix} I_\delta & 0 \\ 0 & 0 \end{pmatrix}\right)\right) = \text{trace}\left(R_\delta^T \Sigma_b R_\delta\right) \leq \lambda_1 + \cdots + \lambda_q. \end{aligned}$$

where R_δ is the matrix consisting of the first δ columns of R, and the last inequality follows from Lemma 3.2. By Lemma 3.2 again, the above inequality becomes equality, if $R_\delta = \begin{pmatrix} W \\ 0 \end{pmatrix}$, for any orthogonal $W \in \mathbb{R}^{q \times q}$, $\delta = q$, and $\ell = q$. Under this choice of R_δ,

$$G_1 = R_q \Sigma_q S^T = \begin{pmatrix} W \Sigma_q S^T \\ 0 \end{pmatrix}.$$

We observe that the maximization of F_1 is independent of G_2, and simply set it to zero. Therefore, the maximum of F_1 is attained when

$$\tilde{G} = \begin{pmatrix} G_1 \\ G_2 \end{pmatrix} = \begin{pmatrix} W \Sigma_q S^T \\ 0 \end{pmatrix}.$$

Note that the orthogonal matrices W and S, and the diagonal matrix Σ_q are arbitrary. Hence, $M = W \Sigma_q S^T$ is an arbitrary nonsingular matrix. It follows that $G = X\tilde{G} = X_q M$, for any nonsingular M, maximizes F_1. This completes the proof of the theorem. ■

Remark 1 *Note that it is in general not true that $F_1(H) = F_1(HM)$, for any nonsingular M. However, Theorem 3.1 implies that for $H = X_q$, we have $F_1(H) = F_1(HM)$, for any nonsingular M.*

4. Uncorrelated LDA Versus Orthogonal LDA

From last section, $G = X_q M$, for any nonsingular M maximizes the F_1 criterion. A natural question is: How to choose the best M? In this section, we consider two specific choices of M, which lead to two distinct algorithms: Uncorrelated LDA and Orthogonal LDA.

4.1 Uncorrelated LDA

The simplest choice of M is the identity matrix, i.e., $M = I_q$. That is, $G = X_q$. It follows that $X_q^T S_t X_q = I_q$, i.e., the columns of the transformation G are S_t-orthogonal. Recall that two vectors x and y are S_t-orthogonal, if $x^T S_t y = 0$. The solution corresponds to the Uncorrelated LDA, originally proposed by Jin et al. (Jin et al., 2001a,b). The pseudo-code for ULDA is given in **Algorithm 1**.

Algorithm 1: Uncorrelated LDA

```
Input:   data matrix $A$
Output:  transformation matrix $G$
1. Form three matrices $H_b$, $H_w$, and $H_t$ as in Eq. (2);
2. Compute reduced SVD of $H_t$ as $H_t = U_1 \Sigma_t V_1^T$;
3. $B \leftarrow \Sigma_t^{-1} U_1^T H_b$;
4. Compute SVD of $B$ as $B = P \Sigma Q^T$; $q \leftarrow \text{rank}(B)$;
5. $X \leftarrow U_1 \Sigma_t^{-1} P$;
6. $G \leftarrow X_q$;
```

ULDA was originally proposed to compute the optimal discriminant vectors that are S_t-orthogonal. Specifically, suppose r vectors $\phi_1, \phi_2, \cdots, \phi_r$ are obtained, then the $(r+1)$-th vector ϕ_{r+1} of ULDA is the one that maximizes the Fisher criterion function

$$f(\phi) = \frac{\phi^T S_b \phi}{\phi^T S_w \phi}, \tag{17}$$

subject to the constraints:

$$\phi_{r+1}^T S_t \phi_i = 0, \quad i = 1, \cdots, r.$$

The algorithm in (Jin et al., 2001a) finds ϕ_i successively as follows: The j-th discriminant vector ϕ_j of ULDA is the eigenvector corresponding to the maximum eigenvalue of the following generalized eigenvalue problem:

$$U_j S_b \phi_j = \lambda_j S_w \phi_j,$$

where

$$\begin{aligned} U_1 &= I_m, \\ U_j &= I_m - S_t D_j^T (D_j S_t S_w^{-1} S_t D_j^T)^{-1} D_j S_t S_w^{-1} (j > 1), \\ D_j &= [\phi_1, \cdots, \phi_{j-1}]^T (j > 1), \end{aligned}$$

and I_m is the identity matrix.

A key property of ULDA is that the features in the reduced space are uncorrelated to each other, as stated in the following proposition.

Proposition 4.1 *Let the transformation matrix for ULDA be $G = [g_1, \cdots, g_d]$, for some $d > 0$. The original feature vector A is transformed into $Z = G^T A$, where the i-th feature component of Z is $Z_i = g_i^T A$. Assume that g_i and g_j are S_t-orthogonal to each other, i.e., $g_i^T S_t g_j = 0$, for $i \neq j$. Then the correlation between Z_i and Z_j is 0, for $i \neq j$. That is, Z_i and Z_j are uncorrelated to each other.*

Proof The covariance between Z_i and Z_j can be computed as

$$\text{Cov}(Z_i, Z_j) = E(Z_i - EZ_i)(Z_j - EZ_j) = g_i^T \{E(A - EA)(A - EA)^T\} g_j = g_i^T S_t g_j. \tag{18}$$

Hence, their correlation coefficient is

$$\text{Cor}(Z_i, Z_j) = \frac{g_i^T S_t g_j}{\sqrt{g_i^T S_t g_i}\sqrt{g_j^T S_t g_j}}. \tag{19}$$

Since $g_i^T S_t g_j = 0$, for $i \neq j$, we have $\mathrm{Cor}(Z_i, Z_j) = 0$, for $i \neq j$. This completes the proof of the proposition. ■

In (Ye et al., 2004a), an efficient algorithm for ULDA was proposed, based on the following optimization problem:

$$G = \arg\max_G \{\mathrm{trac}((S_t^L + \mu I_\ell)^{-1} S_b^L)\}. \tag{20}$$

Note that $S_t^L + \mu I_\ell$ is always nonsingular for $\mu > 0$, since S_t^L is positive semidefinite. One key result in (Ye et al., 2004a) shows that the optimal transformation G solving the optimization problem in Eq. (20) is independent of μ.

Interestingly, it can be shown that $G = X_q$ solves the optimization problem in Eq. (20) as stated in the following proposition. Detailed proof follows the one in (Ye et al., 2004a) and is thus omitted.

Proposition 4.2 *Let $G = X_q$, where X_q is the matrix consisting of the first q columns of X, and X is defined in Eq. (15). Then G solves the optimization problem in Eq. (20).*

4.1.1 RELATIONSHIP BETWEEN ULDA AND THE EIGEN-DECOMPOSITION OF $S_t^+ S_b$

In this section, we study the relationship between ULDA and the eigen-decomposition of $S_t^+ S_b$. More specifically, we show that the discriminant vectors of ULDA are eigenvectors of $S_t^+ S_b$ corresponding to nonzero eigenvalues. Recall that classical LDA computes the optimal discriminant vectors by solving an eigenvalue problem on $S_t^{-1} S_b$, assuming S_t is nonsingular (See Section 2). This equivalence result shows that ULDA is a natural extension of classical LDA by replacing inverse with pseudo-inverse, when dealing with singular S_t.

From Eq. (14), we have $X^T S_t X = D_t$, where

$$X = U \begin{pmatrix} \Sigma_t^{-1} P & 0 \\ 0 & 0 \end{pmatrix}, \text{ and } D_t = \begin{pmatrix} I_t & 0 \\ 0 & 0 \end{pmatrix}.$$

Note that P is orthogonal. It follows that

$$S_t = X^{-T} D_t X^{-1} = U \begin{pmatrix} \Sigma_t P & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} I_t & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} P^T \Sigma_t & 0 \\ 0 & 0 \end{pmatrix} U^T = U \begin{pmatrix} \Sigma_t^2 & 0 \\ 0 & 0 \end{pmatrix} U^T,$$

Hence,

$$S_t^+ = U \begin{pmatrix} \Sigma_t^{-2} & 0 \\ 0 & 0 \end{pmatrix} U^T.$$

It is easy to verify that

$$X D_t X^T = U \begin{pmatrix} \Sigma_t^{-2} & 0 \\ 0 & 0 \end{pmatrix} U^T.$$

It follows that

$$S_t^+ = X D_t X^T, \tag{21}$$

and

$$S_t^+ S_b = \left(X D_t X^T\right)\left(X^{-1} D_b X^{-1}\right) = X D_t D_b X^{-1}.$$

Therefore, the columns of X_q form the eigenvectors of $S_t^+ S_b$ corresponding to nonzero eigenvalues, since $D_t D_b$ is diagonal with q nonzero diagonal entries.

```
Algorithm 2: Orthogonal LDA
Input:   data matrix $A$
Output:  transformation matrix $G$
1. Compute the matrix $X_q$ as in ULDA (Steps 1–5 of **Algorithm 1**);
2. Compute QR decomposition of $X_q$ as $X_q = \tilde{Q}\tilde{R}$;
3. $G \leftarrow \tilde{Q}$;
```

4.2 Orthogonal LDA

LDA with orthogonal discriminant vectors is a natural alternative to ULDA. Let $X_q = \tilde{Q}\tilde{R}$ be the QR decomposition of X_q, then we can simply choose $M = \tilde{R}^{-1}$ so that the columns of $G = X_qM = \tilde{Q}$ are orthogonal to each other. The pseudo-code for OLDA is given in **Algorithm 2**.

Note that in the literature of LDA, Foley-Sammon LDA (FSLDA) is also known for its orthogonal discriminant vectors. FSLDA was first proposed by Foley and Sammon for two-class problems (Foley and Sammon, 1975). It was then extended to the multi-class problems by Duchene and Leclercq (Duchene and Leclerq, 1988). Specifically, suppose r vectors $\phi_1, \phi_2, \cdots, \phi_r$ are obtained, then the $(r+1)$-th vector ϕ_{r+1} of FSLDA is the one that maximizes the Fisher criterion function $f(\phi)$ defined in Eq. (17), subject to the constraints: $\phi_{r+1}^T \phi_i = 0$, $i = 1, \cdots, r$.

The algorithm in (Duchene and Leclerq, 1988) finds ϕ_i successively as follows: The j-th discriminant vector ϕ_j of FSLDA is the eigenvector corresponding to the maximum eigenvalue of the following matrix:

$$\left(I_m - S_w^{-1} D_j^T S_j^{-1} D_j\right) S_w^{-1} S_b,$$

where

$$D_j = [\phi_1, \cdots, \phi_{j-1}]^T \, (j > 1), \text{ and } S_j = D_j S_w^{-1} D_j^T.$$

The above FSLDA algorithm may be expensive for large and high-dimensional data sets. More details on the computation of FSLDA can be found in (Duchene and Leclerq, 1988).

It is worthwhile to point out that both ULDA and FSLDA use the same Fisher criterion function, and the main difference is that the optimal discriminant vectors generated by ULDA are S_t-orthogonal to each other, while the optimal discriminant vectors of FSLDA are orthogonal to each other.

The common point of the proposed OLDA algorithm and the FSLDA algorithm described above is that the transformation matrix has orthogonal columns. However, these two algorithms were derived from distinct perspectives.

4.3 Discussions

As discussed in Section 2, classical LDA is equivalent to maximum likelihood classification assuming normal distribution for each class with the common covariance matrix. Classification in classical LDA based on the maximum likelihood estimation is based on the Mahalanobis distance as follows: a test point h is classified as class j if

$$j = \arg\min_j (h - c^{(j)})^T S_w^{-1} (h - c^{(j)}), \quad (22)$$

which is equivalent to

$$j = \arg\min_j (h - c^{(j)})^T S_t^{-1} (h - c^{(j)}). \tag{23}$$

We show in the following that the classification in ULDA uses the following distance:

$$(h - c^{(j)})^T S_t^{+} (h - c^{(j)}). \tag{24}$$

The main result is summarized in the following theorem.

Theorem 4.1 *Let G be the optimal transformation matrix for ULDA, and let h be any test point. Then*

$$\arg\min_j \left\{ (h - c^{(j)})^T S_t^{+} (h - c^{(j)}) \right\} = \arg\min_j \left\{ ||G^T (h - c^{(j)})||^2 \right\}.$$

Proof Let X_i be the i-th column of X. From Eq. (21), we have

$$S_t^{+} = X D_t X^T = \sum_{i=1}^{t} X_i X_i^T = G G^T + \sum_{i=q+1}^{t} X_i X_i^T,$$

where G consists of the first q columns of X, and $q = \text{rank}(S_b)$.

Recall from Section 3.1 that X diagonalizes S_b and $X_i^T S_b X_i = 0$, for $i = q+1, \cdots, t$. Hence $H_b X_i = 0$, or $(c^{(j)})^T X_i = c X_i$, for all $j = 1, \cdots, k$. It follows that

$$\begin{aligned} (h - c^{(j)})^T S_t^{+} (h - c^{(j)}) &= (h - c^{(j)})^T G G^T (h - c^{(j)}) + \sum_{i=q+1}^{t} (h - c^{(j)})^T X_i X_i^T (h - c^{(j)}) \\ &= ||G^T (h - c^{(j)})||^2 + \sum_{i=q+1}^{t} (h - c)^T X_i X_i^T (h - c). \end{aligned} \tag{25}$$

The second term on the right hand side of Eq. (25) is independent of class j, hence

$$\arg\min_j \left\{ (h - c^{(j)})^T S_t^{+} (h - c^{(j)}) \right\} = \arg\min_j \left\{ ||G^T (h - c^{(j)})||^2 \right\}.$$

This completes the proof of the theorem.

∎

Theorem 4.1 shows that the classification rule in ULDA is a variant of the one used in classical LDA. ULDA can be considered as an extension of classical LDA for singular scatter matrices. The result does not extend to OLDA. However, with whitened total scatter matrix, that is if S_t is an identity matrix, OLDA is equivalent to ULDA.

Geometrically, both ULDA and OLDA project the data onto the subspace spanned by the centroids. ULDA removes the correlation among the features in the transformed space, which is theoretically sound but may be sensitive to the noise in the data. On the other hand, OLDA applies orthogonal transformation $\tilde{Q}$, by factoring out the $\tilde{R}$ matrix through the QR decomposition of $X_q = \tilde{Q}\tilde{R}$. The removal of $\tilde{R}$ in OLDA may contribute to the noise removal. Our experiments in next section show that OLDA often leads to better performance than ULDA in classification.

Data Set	Size (n)	Dimension (m)	# of classes (k)
tr41	210	7454	7
re0	320	2887	4
PIX	300	10000	30
AR	1638	8888	126
GCM	198	16063	14
ALL	248	12558	6

Table 2: Statistics for our test data sets

5. Experiments

We divide the experiments into three parts. Section 5.1 describes our test data sets. Section 5.2 evaluates ULDA and OLDA in terms of classification accuracy. We study the effect of the matrix M in Section 5.3. Recall that $G = X_q M$, for any nonsingular M maximizes the F_1 criterion.

Both ULDA and OLDA were implemented in MATLAB and the source codes may be accessed at *http://www.cs.umn.edu/∼jieping/UOLDA*.

5.1 Data Sets

We have three types of data for the evaluation: text documents, including **tr41** and **re0**; face images, including **PIX** and **AR**; and gene expression data, including **GCM** and **ALL**. The important statistics of these data sets are summarized as follows (see also Table 2):

- **tr41** is a text document data set, derived from the TREC-5, TREC-6, and TREC-7 collections (TREC, 1999). It includes 210 documents belonging to 7 different classes. The dimension of this data set is 7454.

- **re0** is another text document data set, derived from *Reuters-21578* text categorization test collection Distribution 1.0 (Lewis, 1999). It includes 320 documents belonging to 4 different classes. The dimension of this data set is 2887.

- **PIX**[1] is a face image data set, which contains 300 face images of 30 persons. The size of PIX images is 512×512. We subsample the images down to a size of $100 \times 100 = 10000$.

- **AR**[2] (Martinez and Benavente, 1998), is a large face image data set. The instance of each face may contain pretty large areas of occlusion, due to the presence of sun glasses and scarves. We use a subset of AR. This subset contains 1638 face images of 126 individuals. Its image size is 768×576. We first crop the image from row 100 to 500, and column 200 to 550, and then subsample the cropped images down to a size of $101 \times 88 = 8888$.

- **GCM** is a gene expression data set consisting of 198 human tumor samples spanning fourteen different cancer types. The data set was first studied in (Ramaswamy and et al., 2001; Yeang and et al., 2001). The breakdown of the sample classes is as follows: 12 *breast* samples, 14

1. http://peipa.essex.ac.uk/ipa/pix/faces/manchester/test-hard/
2. http://rvl1.ecn.purdue.edu/∼aleix/aleix_face_DB.html

prostate samples, 12 *lung* samples, 12 *colorectal* samples, 22 *lymphoma* samples, 11 *bladder* samples, 10 *melanoma* samples, 10 *uterus* samples, 30 *leukemia* samples, 11 *renal* samples, 11 *pancreas* samples, 12 *ovary* samples, 11 *mesothelioma* samples, and 20 *CNS* samples.

- **ALL**[3] is another gene expression data set consisting of six diagnostic groups (Yeoh and et al., 2002). The breakdown of the samples is: 15 samples for *BCR*, 27 samples for *E2A*, 64 samples for *Hyperdip*, 20 samples for *MLL*, 43 samples for *T*, and 79 samples for *TEL*.

5.2 Comparison on Classification Accuracy

In this experiment, we evaluate ULDA and OLDA in terms of classification accuracy. For the **GCM** and **ALL** gene expression data sets, the test sets were provided. In the absence of original test sets, such as the two document data sets and the two face image data sets, we perform our comparative study by repeated random splitting into training and test sets exactly as in (Dudoit et al., 2002). The data were randomly partitioned into a training set consisting of two-thirds of the whole set and a test set consisting of one-third of the whole set. To reduce the variability, the splitting was repeated 50 times and the resulting accuracies were averaged. Note that during each run, dimension reduction is applied to the training set only. For RLDA, the results depend on the choice of the parameter μ. We choose the best μ through cross-validation. The range for μ is between 0.001 and 10.

The results of the three algorithms on the six data sets are presented in Table 3. The main observation from Table 3 is that OLDA is competitive with ULDA and RLDA in all six data sets. We also observe that in most cases, RLDA outperforms ULDA and is competitive with OLDA.

It is interesting to note that OLDA achieves higher accuracies than ULDA for the two face image data sets and two gene expression data sets, while it achieves accuracies close to those of ULDA for the two text document data sets. For the **GCM** gene expression data set, OLDA achieved classification accuracy 3% higher than that of OLDA. This may be related to the effect of the noise removal inherent in OLDA as discussed in Section 4.3.

5.3 Effect of the Matrix M

In this experiment, we study the effect of the matrix M on classification using the **GCM** and **ALL** data sets. Recall that the solution to the proposed criterion is $G = X_qM$, for any nonsingular M. Two specific choices of M were studied, which correspond to ULDA and OLDA. In this experiment, we randomly generated 100 matrices for M and computed the accuracies using the corresponding transformation matrices. Figure 1 shows the histogram of the resulting accuracies on **GCM**, where the x-axis represents the range of resulting accuracies (divided into small intervals), and the y-axis represents the number (count) for each interval. The main observations are:

- None of the accuracies is higher than those of ULDA (73.91%) and OLDA (76.09%). ULDA and OLDA are probably two of the best ones among the family of solutions to the proposed criterion.

- In Figure 1, most of the accuracies are around 55%, which is much lower than those of ULDA and OLDA. Thus, the choice of M does make a big difference. Among the family of solutions to the proposed criterion, most of them perform quite poorly in comparison to ULDA and OLDA.

3. http://www.stjuderesearch.org/data/ALL1/

Data Set	Accuracy		
	ULDA	OLDA	RLDA
tr41	96.69 ± 1.90	96.34 ± 2.10	96.23 ± 2.17
re0	86.26 ± 2.46	86.13 ± 2.58	87.34 ± 2.37
PIX	96.16 ± 2.48	98.00 ± 1.66	96.31 ± 2.20
AR	90.94 ± 0.96	92.77 ± 1.04	91.11 ± 1.02
GCM	73.91	76.09	78.26
ALL	98.82	100.0	98.82

Table 3: Comparison of classification accuracy and standard deviation of three algorithms: ULDA (Uncorrelated LDA), OLDA (Orthogonal LDA), and RLDA (Regularized LDA), on the six data sets. The mean and standard deviation of accuracies from fifty runs are reported for **tr41**, **re0**, **PIX**, and **AR**. Note that for the two gene expression data sets: **GCM** and **ALL**, we use the original test sets. Thus the standard deviation for these two data sets are not reported.

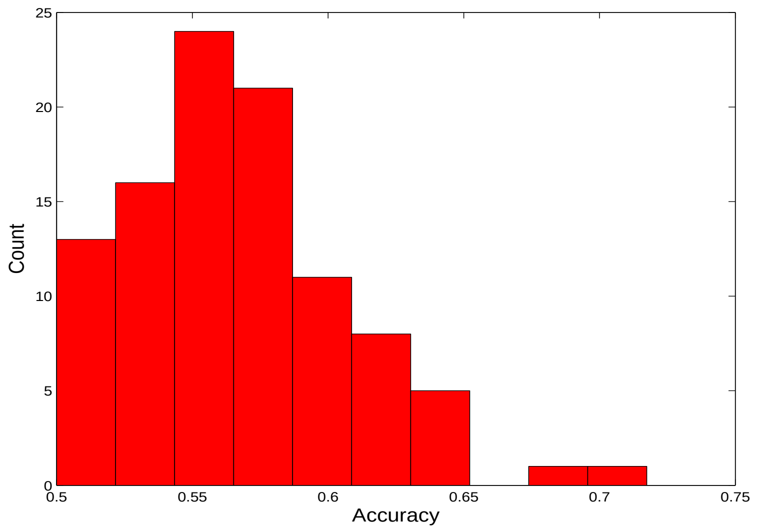


Figure 1: Effect of the matrix M using the **GCM** data set. The corresponding accuracies of ULDA and OLDA are 73.91% and 76.09%, respectively.

The result on **ALL** is shown in Figure 2. We can observe the same trend as in **GCM**, that is, most of the accuracies are much lower than those of ULDA and OLDA.

6. Conclusions and Future Directions

In this paper, a new optimization criterion for discriminant analysis is presented. The new criterion extends the optimization criteria of the classical LDA when the scatter matrices are singular. It

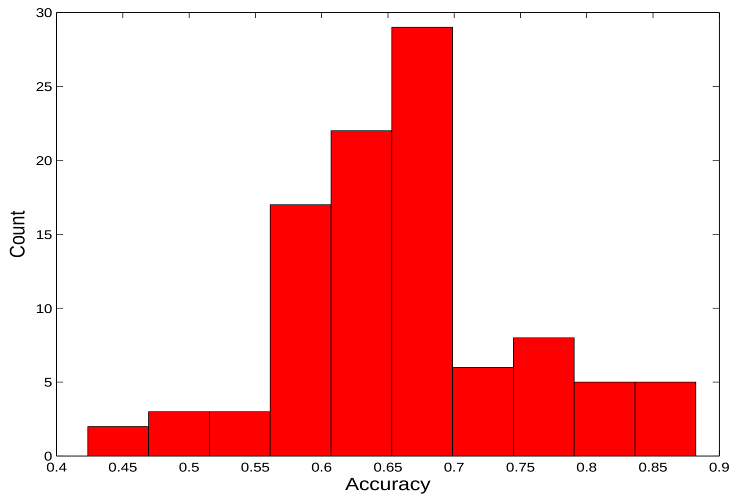


Figure 2: Effect of the matrix M using the **ALL** data set. The corresponding accuracies of ULDA and OLDA are 98.82% and 100.0%, respectively.

is applicable regardless of the relative sizes of the data dimension and sample size, overcoming a limitation of the classical LDA. A detailed mathematical derivation for the proposed optimization problem is presented. It is based on the simultaneous diagonalization of the three scatter matrices.

The solutions to the proposed criterion form a family of algorithms for generalized LDA, which can be characterized in a closed form. Among the family of solutions, we study two specific ones, namely ULDA and OLDA, where ULDA was previously proposed for feature extraction and dimension reduction and OLDA is a novel algorithm. ULDA has the property that the features in the reduced space are uncorrelated, while OLDA has the property that the discriminant vectors obtained are orthogonal to each other. Experiment on a variety of real-world data sets show that OLDA is competitive with ULDA and RLDA in terms of classification accuracy.

In this paper, we focus on two specific algorithms, ULDA and OLDA, for generalized LDA. A promising direction is to find algorithms with sparse transformation matrices. Sparsity has recently received much attention for extending Principal Component Analysis (d'Aspremont et al., 2004; Jolliffe and Uddin, 2003). One of our future work is to incorporate the sparsity criterion in discriminant analysis.

Acknowledgments

Research is sponsored, in part, by the Army High Performance Computing Research Center under the auspices of the Department of the Army, Army Research Laboratory cooperative agreement number DAAD19-01-2-0014, the content of which does not necessarily reflect the position or the policy of the government, and no official endorsement should be inferred. Support Fellowships from Guidant Corporation and from the Department of Computer Science & Engineering, at the University of Minnesota, Twin Cities is gratefully acknowledged.

Appendix A.

The pseudo-inverse of a matrix is defined as follows.

Definition 2 *The pseudo-inverse of a matrix A, denoted as A^+, refers to the unique matrix satisfying the following four conditions:*

$$(1) A^+AA^+ = A^+, \quad (2) AA^+A = A, \quad (3)\ (AA^+)^T = AA^+, \quad (4) (A^+A)^T = A^+A.$$

The pseudo-inverse is commonly computed by the SVD as follows (Golub and Loan, 1996). Let $A = U \begin{pmatrix} \Sigma & 0 \\ 0 & 0 \end{pmatrix} V^T$ be the SVD of A, where U and V are orthogonal and Σ is diagonal with positive diagonal entries. Then, $A^+ = V \begin{pmatrix} \Sigma^{-1} & 0 \\ 0 & 0 \end{pmatrix} U^T$.

References

G. Baudat and F. Anouar. Generalized discriminant analysis using a kernel approach. *Neural Computation*, 12(10):2385–2404, 2000.

P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, 1997.

M. W. Berry, S. T. Dumais, and G. W. O'Brie. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37:573–595, 1995.

D. Q. Dai and P. C. Yuen. Regularized discriminant analysis and its application to face recognition. *Pattern Recognition*, 36:845–847, 2003.

A. d'Aspremont, L. Ghaoui, M. I. Jordan, and G. R. G. Lanckriet. A direct formulation for sparse PCA using semidefinite programming. In *Proceedings of the Eighteenth Annual Conference on Advances in Neural Information Processing Systems*, 2004.

S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the Society for Information Scienc*, 41:391–407, 1990.

L. Duchene and S. Leclerq. An optimal transformation for discriminant and principal component analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6):978–983, 1988.

R. O. Duda, P. E. Hart, and D. Stork. *Pattern Classification*. Wiley, 2000.

S. Dudoit, J. Fridlyand, and T. P. Speed. Comparison of discrimination methods for the classification of tumors using gene expression data. *Journal of the American Statistical Association*, 97(457): 77–87, 2002.

A. Edelman, T. A. Arias, and S. T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, 20(2):303–353, 1998.

D. H. Foley and J. W. Sammon. An optimal set of discriminant vectors. *IEEE Transactions on Computers*, 24(3):281–289, 1975.

J. H. Friedman. Regularized discriminant analysis. *Journal of the American Statistical Association*, 84(405):165–175, 1989.

K. Fukunaga. *Introduction to Statistical Pattern Classification*. Academic Press, USA, 1990.

G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, MD, USA, third edition, 1996.

D. J. Hand. *Kernel discriminant analysis*. Research Studies Press/Wiley, 1982.

T. Hastie, A. Buja, and R. Tibshirani. Penalized discriminant analysis. *Annals of Statistics*, 23: 73–102, 1995.

T. Hastie and R. Tibshirani. Discriminant analysis by Gaussian mixtures. *Journal of the Royal Statistical Society series B*, 58:158–176, 1996.

T. Hastie, R. Tibshirani, and J. H. Friedman. *The elements of statistical learning : data mining, inference, and prediction*. Springer, 2001.

P. Howland, M. Jeon, and H. Park. Structure preserving dimension reduction for clustered text data based on the generalized singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 25(1):165–179, 2003.

Z. Jin, J. Y. Yang, Z. S. Hu, and Z. Lou. Face recognition based on the uncorrelated discriminant transformation. *Pattern Recognition*, 34:1405–1416, 2001a.

Z. Jin, J. Y. Yang, Z. M. Tang, and Z. S. Hu. A theorem on the uncorrelated optimal discriminant vectors. *Pattern Recognition*, 34(10):2041–2047, 2001b.

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

W. J. Krzanowski, P. Jonathan, W. V. McCarthy, and M. R. Thomas. Discriminant analysis with singular covariance matrices: methods and applications to spectroscopic data. *Applied Statistics*, 44:101–115, 1995.

D. D. Lewis. *Reuters-21578 text categorization test collection distribution 1.0*. http://www.research.att.com/∼lewis, 1999.

J. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos. Face recognition using kernel direct discriminant analysis algorithms. *IEEE Transactions on Neural Networks*, 14(1):117–126, 2003.

A. M. Martinez and R. Benavente. The AR face database. Technical Report No. 24, 1998.

S. Ramaswamy and et al. Multiclass cancer diagnosis using tumor gene expression signatures. *Proceedings of the National Academy of Science*, 98(26):15149–15154, 2001.

S. Raudys and R. P. W. Duin. On expected classification error of the fisher linear classifier with pseudo-inverse covariance matrix. *Pattern Recognition Letters*, 19(5-6):385–392, 1998.

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

M. Skurichina and R. P. W. Duin. Stabilizing classifiers for very small sample size. In *Proc. International Conference on Pattern Recognition*, pages 891–896, 1996.

M. Skurichina and R. P. W. Duin. Regularization of linear classifiers by adding redundant features. *Pattern Analysis and Applications*, 2(1):44–52, 1999.

D. L. Swets and J. Weng. Using discriminant eigenfeatures for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):831–836, 1996.

TREC. *Text Retrieval conference*. http://trec.nist.gov, 1999.

M. A. Turk and A. P. Pentland. Face recognition using Eigenfaces. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 586–591, 1991.

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

G. Wahba. *Spline Models for Observational Data*. Society for Industrial & Applied Mathematics, 1998.

J. Ye, R. Janardan, Q. Li, and H. Park. Feature extraction via generalized uncorrelated linear discriminant analysis. In *The Twenty-First International Conference on Machine Learning*, pages 895–902, 2004a.

J. Ye, R. Janardan, C. H. Park, and H. Park. An optimization criterion for generalized discriminant analysis on undersampled problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8):982–994, 2004b.

C. H. Yeang and et al. Molecular classification of multiple tumor types. *Bioinformatics*, 17(1):1–7, 2001.

E. J. Yeoh and et al. Classification, subtype diswcovery, and prediction of outcome in pediatric lymphoblastic leukemia by gene expression profiling. *Cancer Cell*, 1(2):133–143, 2002.

W. Zhao, R. Chellappa, and P. Phillips. Subspace linear discriminant analysis for face recognition. Technical Report CAR-TR-914. Center for Automation Research, University of Maryland, 1999.

Journal of Machine Learning Research 6 (2005) 503–556 Submitted 11/03; Revised 4/04; Published 4/05

Tree-Based Batch Mode Reinforcement Learning

Damien Ernst ERNST@MONTEFIORE.ULG.AC.BE
Pierre Geurts GEURTS@MONTEFIORE.ULG.AC.BE
Louis Wehenkel LWH@MONTEFIORE.ULG.AC.BE
Department of Electrical Engineering and Computer Science
Institut Montefiore, University of Liège
Sart-Tilman B28
B4000 Liège, Belgium

Editor: Michael L. Littman

Abstract

Reinforcement learning aims to determine an optimal control policy from interaction with a system or from observations gathered from a system. In batch mode, it can be achieved by approximating the so-called Q-function based on a set of four-tuples (x_t, u_t, r_t, x_{t+1}) where x_t denotes the system state at time t, u_t the control action taken, r_t the instantaneous reward obtained and x_{t+1} the successor state of the system, and by determining the control policy from this Q-function. The Q-function approximation may be obtained from the limit of a sequence of (batch mode) supervised learning problems. Within this framework we describe the use of several classical tree-based supervised learning methods (CART, Kd-tree, tree bagging) and two newly proposed ensemble algorithms, namely *extremely* and *totally* randomized trees. We study their performances on several examples and find that the ensemble methods based on regression trees perform well in extracting relevant information about the optimal control policy from sets of four-tuples. In particular, the totally randomized trees give good results while ensuring the convergence of the sequence, whereas by relaxing the convergence constraint even better accuracy results are provided by the extremely randomized trees.

Keywords: batch mode reinforcement learning, regression trees, ensemble methods, supervised learning, fitted value iteration, optimal control

1. Introduction

Research in reinforcement learning (RL) aims at designing algorithms by which autonomous agents can learn to behave in some appropriate fashion in some environment, from their interaction with this environment or from observations gathered from the environment (see e.g. Kaelbling et al. (1996) or Sutton and Barto (1998) for a broad overview). The standard RL protocol considers a performance agent operating in discrete time, observing at time t the environment state x_t, taking an action u_t, and receiving back information from the environment (the next state x_{t+1} and the instantaneous reward r_t). After some finite time, the experience the agent has gathered from interaction with the environment may thus be represented by a set of four-tuples (x_t, u_t, r_t, x_{t+1}).

In on-line learning the performance agent is also the learning agent which at each time step can revise its control policy with the objective of converging as quickly as possible to an optimal control policy. In this paper we consider batch mode learning, where the learning agent is in principle not directly interacting with the system but receives only a set of four-tuples and is asked to determine from this set a control policy which is as close as possible to an optimal policy. Inspired by the on-line Q-learning paradigm (Watkins, 1989), we will approach this batch mode learning problem by computing from the set of four-tuples an approximation of the so-called Q-function defined on the state-action space and by deriving from this latter function the control policy.

When the state and action spaces are finite and small enough, the Q-function can be represented in tabular form, and its approximation (in batch and in on-line mode) as well as the control policy derivation are straightforward. However, when dealing with continuous or very large discrete state and/or action spaces, the Q-function cannot be represented anymore by a table with one entry for each state-action pair. Moreover, in the context of reinforcement learning an approximation of the Q-function all over the state-action space must be determined from finite and generally very sparse sets of four-tuples.

To overcome this generalization problem, a particularly attractive framework is the one used by Ormoneit and Sen (2002) which applies the idea of fitted value iteration (Gordon, 1999) to kernel-based reinforcement learning, and reformulates the Q-function determination problem as a sequence of kernel-based regression problems. Actually, this framework makes it possible to take full advantage in the context of reinforcement learning of the generalization capabilities of *any* regression algorithm, and this contrary to stochastic approximation algorithms (Sutton, 1988; Tsitsiklis, 1994) which can only use parametric function approximators (for example, linear combinations of feature vectors or neural networks). In the rest of this paper we will call this framework the *fitted Q iteration algorithm* so as to stress the fact that it allows to fit (using a set of four-tuples) any (parametric or non-parametric) approximation architecture to the Q-function.

The fitted Q iteration algorithm is a batch mode reinforcement learning algorithm which yields an approximation of the Q-function corresponding to an infinite horizon optimal control problem with discounted rewards, by iteratively extending the optimization horizon (Ernst et al., 2003):

- At the first iteration it produces an approximation of a Q_1-function corresponding to a 1-step optimization. Since the true Q_1-function is the conditional expectation of the instantaneous reward given the state-action pair (i.e., $Q_1(x,u) = E[r_t|x_t = x, u_t = u]$), an approximation of it can be constructed by applying a (batch mode) regression algorithm to a training set whose inputs are the pairs (x_t, u_t) and whose target output values are the instantaneous rewards r_t (i.e., $q_{1,t} = r_t$).

- The Nth iteration derives (using a batch mode regression algorithm) an approximation of a Q_N-function corresponding to an N-step optimization horizon. The training set at this step is obtained by merely refreshing the output values of the training set of the previous step by using the "value iteration" based on the approximate Q_N-function returned at the previous step (i.e., $q_{N,t} = r_t + \gamma \max_u \hat{Q}_{N-1}(x_{t+1}, u)$, where $\gamma \in [0,1)$ is the discount factor).

Ormoneit and Sen (2002) have studied the theoretical convergence and consistency properties of this algorithm when combined with kernel-based regressors. In this paper, we study within this framework the empirical properties and performances of several tree-based regression algorithms on several applications. Just like kernel-based methods, tree-based methods are non-parametric and offer a great modeling flexibility, which is a paramount characteristic for the framework to be successful since the regression algorithm must be able to model any Q_N-function of the sequence, functions which are a priori totally unpredictable in shape. But, from a practical point of view these tree-based methods have a priori some additional advantages, such as their high computational efficiency and scalability to high-dimensional spaces, their fully autonomous character, and their recognized robustness to irrelevant variables, outliers, and noise.

In addition to good accuracy when trained with finite sets of four-tuples, one desirable feature of the regression method used in the context of the fitted Q iteration algorithm is to ensure convergence of the sequence. We will analyze under which conditions the tree-based methods share this property and also what is the relation between convergence and quality of approximation. In particular, we will see that ensembles of totally randomized trees (i.e., trees built by selecting their splits randomly) can be adapted to ensure the convergence of the sequence while leading to good approximation performances. On the other hand, another tree-based algorithm named extremely randomized trees (Geurts et al., 2004), will be found to perform consistently better than totally randomized trees even though it does not strictly ensure the convergence of the sequence of Q-function approximations.

The remainder of this paper is organized as follows. In Section 2, we formalize the reinforcement learning problem considered here and recall some classical results from optimal control theory upon which the approach is based. In Section 3 we present the *fitted Q iteration algorithm* and in Section 4 we describe the different tree-based regression methods considered in our empirical tests. Section 5 is dedicated to the experiments where we apply the fitted Q iteration algorithm used with tree-based methods to several control problems with continuous state spaces and evaluate its performances in a wide range of conditions. Section 6 concludes and also provides our main directions for further research. Three appendices collect relevant details about algorithms, mathematical proofs and benchmark control problems.

2. Problem Formulation and Dynamic Programming

We consider a time-invariant stochastic system in discrete time for which a closed loop stationary control policy[1] must be chosen in order to maximize an expected discounted return over an infinite time horizon. We formulate hereafter the batch mode reinforcement learning problem in this context and we restate some classical results stemming from Bellman's dynamic programming approach to optimal control theory (introduced in Bellman, 1957) and from which the fitted Q iteration algorithm takes its roots.

2.1 Batch Mode Reinforcement Learning Problem Formulation

Let us consider a system having a *discrete-time dynamics* described by

$$x_{t+1} = f(x_t, u_t, w_t) \quad t = 0, 1, \cdots, \tag{1}$$

where for all t, the state x_t is an element of the state space X, the action u_t is an element of the action space U and the random disturbance w_t an element of the disturbance space W. The disturbance w_t is generated by the time-invariant conditional probability distribution $P_w(w|x,u)$.[2]

To the transition from t to $t+1$ is associated an instantaneous *reward signal* $r_t = r(x_t, u_t, w_t)$ where $r(x,u,w)$ is the reward function supposed to be bounded by some constant B_r.

Let $\mu(\cdot) : X \to U$ denote a stationary control policy and J^{μ}_{∞} denote the expected return obtained over an infinite time horizon when the system is controlled using this policy (i.e., when

1. Indeed, in terms of optimality this restricted family of control policies is as good as the broader set of all non-anticipating (and possibly time-variant) control policies.
2. In other words, the probability $P(w_t = w|x_t = x, u_t = u)$ of occurrence of $w_t = w$ given that the current state x_t and the current control u_t are x and u respectively, is equal to $P_w(w|x,u), \forall t = 0, 1, \cdots$.

$u_t = \mu(x_t), \forall t$). For a given initial condition $x_0 = x$, J^{μ}_{∞} is defined by

$$J^{\mu}_{\infty}(x) = \lim_{N\to\infty} \underset{\substack{w_t \\ t=0,1,\cdots,N-1}}{E} [\sum_{t=0}^{N-1} \gamma^t r(x_t, \mu(x_t), w_t) | x_0 = x], \tag{2}$$

where γ is a discount factor ($0 \leq \gamma < 1$) that weights short-term rewards more than long-term ones, and where the conditional expectation is taken over all trajectories starting with the initial condition $x_0 = x$. Our objective is to find an optimal stationary policy μ^*, i.e. a stationary policy that maximizes J^{μ}_{∞} for all x.

The existence of an optimal stationary closed loop policy is a classical result from dynamic programming theory. It could be determined in principle by solving the Bellman equation (see below, Eqn (6)) given the knowledge of the system dynamics and reward function. However, the sole information that we assume available to solve the problem is the one obtained from the observation of a certain number of one-step system transitions (from t to $t+1$). Each system transition provides the knowledge of a new four-tuple (x_t, u_t, r_t, x_{t+1}) of information. Since, except for very special conditions, it is not possible to determine exactly an optimal control policy from a finite sample of such transitions, we aim at computing an approximation of such a μ^* from a set

$$\mathcal{F} = \{(x^l_t, u^l_t, r^l_t, x^l_{t+1}), l = 1, \cdots, \#\mathcal{F}\}$$

of such four-tuples.

We do not make any particular assumptions on the way the set of four-tuples is generated. It could be generated by gathering the four-tuples corresponding to one single trajectory (or episode) as well as by considering several independently generated one or multi-step episodes.

We call this problem the *batch mode* reinforcement learning problem because the algorithm is allowed to use a set of transitions of arbitrary size to produce its control policy in a single step. In contrast, an *on-line* algorithm would produce a sequence of policies corresponding to a sequence of four-tuples.

2.2 Results from Dynamic Programming Theory

For a temporal horizon of N steps, let us denote by

$$\pi_N(t, x) \in U, t \in \{0, \cdots, N-1\}; x \in X$$

a (possibly time-varying) N-step control policy (i.e., $u_t = \pi_N(t, x_t)$), and by

$$J^{\pi_N}_N(x) = \underset{\substack{w_t \\ t=0,1,\cdots,N-1}}{E} [\sum_{t=0}^{N-1} \gamma^t r(x_t, \pi_N(t, x_t), w_t) | x_0 = x] \tag{3}$$

its expected return over N steps. An N-step optimal policy π^*_N is a policy which among all possible such policies maximizes $J^{\pi_N}_N$ for any x. Notice that under mild conditions (see e.g. Hernández-Lerma and Lasserre, 1996, for the detailed conditions) such a policy always does indeed exist although it is not necessarily unique.

Our algorithm exploits the following classical results from dynamic programming theory (Bellman, 1957):

1. The sequence of Q_N-functions defined on $X \times U$ by

$$Q_0(x,u) \equiv 0 \tag{4}$$
$$Q_N(x,u) = (HQ_{N-1})(x,u), \quad \forall N > 0, \tag{5}$$

converges (in infinity norm) to the Q-function, defined as the (unique) solution of the Bellman equation:

$$Q(x,u) = (HQ)(x,u) \tag{6}$$

where H is an operator mapping any function $K : X \times U \to \mathbb{R}$ and defined as follows:[3]

$$(HK)(x,u) = \underset{w}{E}[r(x,u,w) + \gamma \max_{u' \in U} K(f(x,u,w),u')]. \tag{7}$$

Uniqueness of solution of Eqn (6) as well as convergence of the sequence of Q_N-functions to this solution are direct consequences of the fixed point theorem and of the fact that H is a contraction mapping.

2. The sequence of policies defined by the two conditions[4]

$$\pi^*_N(0,x) = \arg\max_{u' \in U} Q_N(x,u'), \forall N > 0 \tag{8}$$
$$\pi^*_N(t+1,x) = \pi^*_{N-1}(t,x), \forall N > 1, t \in \{0,\ldots,N-2\} \tag{9}$$

are N-step optimal policies, and their expected returns over N steps are given by

$$J_N^{\pi^*_N}(x) = \max_{u \in U} Q_N(x,u).$$

3. A policy μ^* that satisfies

$$\mu^*(x) = \arg\max_{u \in U} Q(x,u) \tag{10}$$

is an optimal stationary policy for the infinite horizon case and the expected return of $\mu^*_N(x) \doteq \pi^*_N(0,x)$ converges to the expected return of μ^*:

$$\lim_{N \to \infty} J_\infty^{\mu^*_N}(x) = J_\infty^{\mu^*}(x) \quad \forall x \in X. \tag{11}$$

We have also $\lim_{N\to\infty} J_N^{\pi^*_N}(x) = J_\infty^{\mu^*}(x) \quad \forall x \in X$.

Equation (5) defines the so-called *value iteration algorithm*[5] providing a way to determine iteratively a sequence of functions converging to the Q-function and hence of policies whose return converges to that of an optimal stationary policy, assuming that the system dynamics, the reward function and the noise distribution are known. As we will see in the next section, it suggests also a way to determine approximations of these Q_N-functions and policies from a sample $\mathcal{F}$.

3. The expectation is computed by using $P(w) = P_w(w|x,u)$.
4. Actually this definition does not necessarily yield a unique policy, but any policy which satisfies these conditions is appropriate.
5. Strictly, the term "value iteration" refers to the computation of the *value* function $J_\infty^{\mu^*}$ and corresponds to the iteration $J_N^{\pi^*_N} = \max_{u \in U} \underset{w}{E}[r(x,u,w) + \gamma J_{N-1}^{\pi^*_{N-1}}(f(x,u,w))], \forall N > 0$ rather than Eqn (5).

3. Fitted Q Iteration Algorithm

In this section, we introduce the fitted Q iteration algorithm which computes from a set of four-tuples an approximation of the optimal stationary policy.

3.1 The Algorithm

A tabular version of the fitted Q iteration algorithm is given in Figure 1. At each step this algorithm may use the full set of four-tuples gathered from observation of the system together with the function computed at the previous step to determine a new training set which is used by a supervised learning (regression) method to compute the next function of the sequence. It produces a sequence of $\hat{Q}_N$-functions, approximations of the Q_N-functions defined by Eqn (5).

Inputs: a set of four-tuples $\mathcal{F}$ and a regression algorithm.
Initialization:
Set N to 0 .
Let $\hat{Q}_N$ be a function equal to zero everywhere on $X \times U$.
Iterations:
Repeat until stopping conditions are reached

- $N \leftarrow N+1$.

- Build the training set $\mathcal{TS} = \{(i^l, o^l), l = 1, \cdots, \#\mathcal{F}\}$ based on the the function $\hat{Q}_{N-1}$ and on the full set of four-tuples $\mathcal{F}$:

$$i^l = (x_t^l, u_t^l), \tag{12}$$
$$o^l = r_t^l + \gamma \max_{u \in U} \hat{Q}_{N-1}(x_{t+1}^l, u). \tag{13}$$

- Use the regression algorithm to induce from $\mathcal{TS}$ the function $\hat{Q}_N(x,u)$.

Figure 1: Fitted Q iteration algorithm

Notice that at the first iteration the fitted Q iteration algorithm is used in order to produce an approximation of the expected reward $Q_1(x,u) = E_w[r(x,u,w)]$. Therefore, the considered training set uses input/output pairs (denoted (i^l, o^l)) where the inputs are the state-action pairs and the outputs the observed rewards. In the subsequent iterations, only the output values of these input/output pairs are updated using the value iteration based on the $\hat{Q}_N$-function produced at the preceding step and information about the reward and the successor state reached in each tuple.

It is important to realize that the successive calls to the supervised learning algorithm are totally independent. Hence, at each step it is possible to adapt the resolution (or complexity) of the learned model so as to reach the best bias/variance tradeoff at this step, given the available sample.

3.2 Algorithm Motivation

To motivate the algorithm, let us first consider the deterministic case. In this case the system dynamics and the reward signal depend only on the state and action at time t. In other words we have

$x_{t+1} = f(x_t, u_t)$ and $r_t = r(x_t, u_t)$ and Eqn (5) may be rewritten

$$Q_N(x,u) = r(x,u) + \gamma \max_{u' \in U} Q_{N-1}(f(x,u), u'). \tag{14}$$

If we suppose that the function Q_{N-1} is known, we can use this latter equation and the set of four-tuples $\mathcal{F}$ in order to determine the value of Q_N for the state-action pairs $(x_t^l, u_t^l), l = 1, 2, \cdots, \#\mathcal{F}$. We have indeed $Q_N(x_t^l, u_t^l) = r_t^l + \gamma \max_{u' \in U} Q_{N-1}(x_{t+1}^l, u')$, since $x_{t+1}^l = f(x_t^l, u_t^l)$ and $r_t^l = r(x_t^l, u_t^l)$.

We can thus build a training set $\mathcal{TS} = \{((x_t^l, u_t^l), Q_N(x_t^l, u_t^l)), l = 1, \cdots, \#\mathcal{F}\}$ and use a regression algorithm in order to generalize this information to any unseen state-action pair or, stated in another way, to *fit* a function approximator to this training set in order to get an approximation $\hat{Q}_N$ of Q_N over the whole state-action space. If we substitute $\hat{Q}_N$ for Q_N we can, by applying the same reasoning, determine iteratively $\hat{Q}_{N+1}$, $\hat{Q}_{N+2}$, etc.

In the stochastic case, the evaluation of the right hand side of Eqn (14) for some four-tuples (x_t, u_t, r_t, x_{t+1}) is no longer equal to $Q_N(x_t, u_t)$ but rather is the realization of a random variable whose expectation is $Q_N(x_t, u_t)$. Nevertheless, since a regression algorithm usually[6] seeks an approximation of the conditional expectation of the output variable given the inputs, its application to the training set $\mathcal{TS}$ will still provide an approximation of $Q_N(x,u)$ over the whole state-action space.

3.3 Stopping Conditions

The stopping conditions are required to decide at which iteration (i.e., for which value of N) the process can be stopped. A simple way to stop the process is to define a priori a maximum number of iterations. This can be done for example by noting that for a sequence of optimal policies μ_N^*, an error bound on the sub-optimality in terms of number of iterations is given by the following equation

$$\|J_\infty^{\mu_N^*} - J_\infty^{\mu^*}\|_\infty \leq 2\frac{\gamma^N B_r}{(1-\gamma)^2}. \tag{15}$$

Given the value of B_r and a desired level of accuracy, one can then fix the maximum number of iterations by computing the minimum value of N such that the right hand side of this equation is smaller than the tolerance fixed.[7]

Another possibility would be to stop the iterative process when the distance between $\hat{Q}_N$ and $\hat{Q}_{N-1}$ drops below a certain value. Unfortunately, for some supervised learning algorithms there is no guarantee that the sequence of $\hat{Q}_N$-functions actually converges and hence this kind of convergence criterion does not necessarily make sense in practice.

3.4 Control Policy Derivation

When the stopping conditions - whatever they are - are reached, the final control policy, seen as an approximation of the optimal stationary closed loop control policy is derived by

$$\hat{\mu}_N^*(x) = \arg\max_{u \in U} \hat{Q}_N(x,u). \tag{16}$$

6. This is true in the case of least squares regression, i.e. in the vast majority of regression methods.
7. Equation (15) gives an upper bound on the suboptimality of μ_N^* and not of $\hat{\mu}_N^*$. By exploiting this upper bound to determine a maximum number of iterations, we assume implicitly that $\hat{\mu}_N^*$ is a good approximation of μ_N^* (that $\|J_\infty^{\hat{\mu}_N^*} - J_\infty^{\mu_N^*}\|_\infty$ is small).

When the action space is discrete, it is possible to compute the value $\hat{Q}_N(x,u)$ for each value of u and then find the maximum. Nevertheless, in our experiments we have sometimes adopted a different approach to handle discrete action spaces. It consists of splitting the training samples according to the value of u and of building the approximation $\hat{Q}_N(x,u)$ by separately calling for each value of $u \in U$ the regression method on the corresponding subsample. In other words, each such model is induced from the subset of four-tuples whose value of the action is u, i.e.

$$\mathcal{F}_u = \{(x_t, u_t, r_t, x_{t+1}) \in \mathcal{F} | u_t = u\}.$$

At the end, the action at some point x of the state space is computed by applying to this state each model $\hat{Q}_N(x,u), u \in U$ and looking for the value of u yielding the highest value.

When the action space is continuous, it may be difficult to compute the maximum especially because we can not make any a priori assumption about the shape of the Q-function (e.g. convexity). However, taking into account particularities of the models learned by a particular supervised learning method, it may be more or less easy to compute this value (see Section 4.5 for the case of tree-based models).

3.5 Convergence of the Fitted Q Iteration Algorithm

The fitted Q iteration algorithm is said to converge if there exists a function $\hat{Q} : X \times U \to \mathbb{R}$ such that $\forall \varepsilon > 0$ there exists a $n \in \mathbb{N}$ such that:

$$\|\hat{Q}_N - \hat{Q}\|_\infty < \varepsilon \quad \forall N > n.$$

Convergence may be ensured if we use a supervised learning method which given a sample $\mathcal{TS} = \{(i^1, o^1), \ldots, (i^{\#\mathcal{TS}}, o^{\#\mathcal{TS}})\}$ produces at each call the model (proof in Appendix B):

$$f(i) = \sum_{l=1}^{\#\mathcal{TS}} k_{\mathcal{TS}}(i^l, i) * o^l, \tag{17}$$

with the kernel $k_{\mathcal{TS}}(i^l, i)$ being the same from one call to the other within the fitted Q iteration algorithm[8] and satisfying the normalizing condition:

$$\sum_{l=1}^{\#\mathcal{TS}} |k_{\mathcal{TS}}(i^l, i)| = 1, \forall i. \tag{18}$$

Supervised learning methods satisfying these conditions are for example the k-nearest-neighbors method, partition and multi-partition methods, locally weighted averaging, linear, and multi-linear interpolation. They are collectively referred to as kernel-based methods (see Gordon, 1999; Ormoneit and Sen, 2002).

3.6 Related Work

As stated in the Introduction, the idea of trying to approximate the Q-function from a set of four-tuples by solving a sequence of supervised learning problems may already be found in Ormoneit and

8. This is true when the kernel does not depend on the output values of the training sample and when the supervised learning method is deterministic.

Sen (2002). This work however focuses on kernel-based methods for which it provides convergence and consistency proofs, as well as a bias-variance characterization. While in our formulation state and action spaces are handled in a symmetric way and may both be continuous or discrete, in their work Ormoneit and Sen consider only discrete action spaces and use a separate kernel for each value of the action.

The work of Ormoneit and Sen is related to earlier work aimed to solve large-scale dynamic programming problems (see for example Bellman et al., 1973; Gordon, 1995b; Tsitsiklis and Van Roy, 1996; Rust, 1997). The main difference is that in these works the various elements that compose the optimal control problem are supposed to be known. We gave the name *fitted Q iteration* to our algorithm given in Figure 1 to emphasize that it is a reinforcement learning version of the *fitted value iteration* algorithm whose description may be found in Gordon (1999). Both algorithms are quite similar except that Gordon supposes that a complete generative model is available,[9] which is a rather strong restriction with respect to the assumptions of the present paper.

In his work, Gordon characterizes a class of supervised learning methods referred to as averagers that lead to convergence of his algorithm. These averagers are in fact a particular family of kernels as considered by Ormoneit and Sen. In Boyan and Moore (1995), serious convergence problems that may plague the fitted value iteration algorithm when used with polynomial regression, backpropagation, or locally weighted regression are shown and these also apply to the reinforcement learning context. In their paper, Boyan and Moore propose also a way to overcome this problem by relying on some kind of Monte-Carlo simulations. In Gordon (1995a) and Singh et al. (1995) on-line versions of the fitted value iteration algorithm used with averagers are presented.

In Moore and Atkeson (1993) and Ernst (2003), several reinforcement learning algorithms closely related to the fitted Q iteration algorithm are given. These algorithms, known as model-based algorithms, build explicitly from the set of observations a finite Markov Decision Process (MDP) whose solution is then used to adjust the parameters of the approximation architecture used to represent the Q-function. When the states of the MDP correspond to a finite partition of the original state space, it can be shown that these methods are strictly equivalent to using the fitted Q iteration algorithm with a regression method which consists of simply averaging the output values of the training samples belonging to a given cell of the partition.

In Boyan (2002), the Least-Squares Temporal-Difference (LSTD) algorithm is proposed. This algorithm uses linear approximation architectures and learns the expected return of a policy. It is similar to the fitted Q iteration algorithm combined with linear regression techniques on problems for which the action space is composed of a single element. Lagoudakis and Parr (2003a) introduce the Least-Squares Policy Iteration (LSPI) which is an extension of LSTD to control problems. The model-based algorithms in Ernst (2003) that consider representative states as approximation architecture may equally be seen as an extension of LSTD to control problems.

Finally, we would like to mention some recent works based on the idea of reductions of reinforcement learning to supervised learning (classification or regression) with various assumptions concerning the available a priori knowledge (see e.g. Kakade and Langford, 2002; Langford and Zadrozny, 2004, and the references therein). For example, assuming that a generative model is available,[10] an approach to solve the optimal control problem by reformulating it as a sequence of

9. Gordon supposes that the functions $f(\cdot,\cdot,\cdot)$, $r(\cdot,\cdot,\cdot)$, and $P_w(\cdot|\cdot,\cdot)$ are known and considers training sets composed of elements of the type $(x, \max_{u\in U} E_w[r(x,u,w)+\gamma \hat{J}_{N-1}^{\pi^*_{N-1}}(f(x,u,w))])$.

10. A generative model allows simulating the effect of any action on the system at any starting point; this is less restrictive than the *complete* generative model assumption of Gordon (footnote 9, page 511).

standard supervised classification problems has been developed (see Lagoudakis and Parr, 2003b; Bagnell et al., 2003), taking its roots from the policy iteration algorithm, another classical dynamic programming algorithm. Within this "reductionist" framework, the fitted Q iteration algorithm can be considered as a *reduction* of reinforcement learning to a sequence of regression tasks, inspired by the value iteration algorithm and usable in the rather broad context where the available information is given in the form of a set of four-tuples. This *batch mode* context incorporates indeed both the on-line context (since one can always store data gathered on-line, at least for a finite time interval) as well as the generative context (since one can always use the generative model to generate a sample of four-tuples) as particular cases.

4. Tree-Based Methods

We will consider in our experiments five different tree-based methods all based on the same top-down approach as in the classical tree induction algorithm. Some of these methods will produce from the training set a model composed of one *single* regression tree while the others build an *ensemble* of regression trees. We characterize first the models that will be produced by these tree-based methods and then explain how the different tree-based methods generate these models. Finally, we will consider some specific aspects related to the use of tree-based methods with the fitted Q iteration algorithm.

4.1 Characterization of the Models Produced

A regression tree partitions the input space into several regions and determines a constant prediction in each region of the partition by averaging the output values of the elements of the training set $\mathcal{TS}$ which belong to this region. Let $S(i)$ be the function that assigns to an input i (i.e., a state-action pair) the region of the partition it belongs to. A regression tree produces a model that can be described by Eqn (17) with the kernel defined by the expression:

$$k_{\mathcal{TS}}(i^l,i) = \frac{I_{S(i)}(i^l)}{\sum_{(a,b)\in\mathcal{TS}} I_{S(i)}(a)} \tag{19}$$

where $I_B(\cdot)$ denotes the characteristic function of the region B ($I_B(i) = 1$ if $i \in B$ and 0 otherwise).

When a tree-based method builds an ensemble of regression trees, the model it produces averages the predictions of the different regression trees to make a final prediction. Suppose that a tree-based ensemble method produces p regression trees and gets as input a training set $\mathcal{TS}$. Let $\mathcal{TS}_m$[11] be the training set used to build the mth regression tree (and therefore the mth partition) and $S_m(i)$ be the function that assigns to each i the region of the mth partition it belongs to. The model produced by the tree-based method may also be described by Eqn (17) with the kernel defined now by the expression:

$$k_{\mathcal{TS}}(i^l,i) = \frac{1}{p}\sum_{m=1}^{p} \frac{I_{S_m(i)}(i^l)}{\sum_{(a,b)\in\mathcal{TS}_m} I_{S_m(i)}(a)}. \tag{20}$$

It should also be noticed that kernels (19) and (20) satisfy the normalizing condition (18).

11. These subsets may be obtained in different ways from the original training set, e.g. by sampling with or without replacement, but we can assume that each element of $\mathcal{TS}_m$ is also an element of $\mathcal{TS}$.

4.2 The Different Tree-Based Algorithms

All the tree induction algorithms that we consider are top-down in the sense that they create their partition by starting with a single subset and progressively refining it by splitting its subsets into pieces. The tree-based algorithms that we consider differ by the number of regression trees they build (one or an ensemble), the way they grow a tree from a training set (i.e., the way the different tests inside the tree are chosen) and, in the case of methods that produce an ensemble of regression trees, also the way they derive from the original training set $\mathcal{TS}$ the training set $\mathcal{TS}_m$ they use to build a particular tree. They all consider binary splits of the type $[i_j < t]$, i.e. "if i_j smaller than t go left else go right" where i_j represents the jth input (or jth attribute) of the input vector i. In what follows the split variables t and i_j are referred to as the cut-point and the cut-direction (or attribute) of the split (or test) $[i_j < t]$.

We now describe the tree-based regression algorithms used in this paper.

4.2.1 KD-TREE

In this method the regression tree is built from the training set by choosing the cut-point at the local median of the cut-direction so that the tree partitions the local training set into two subsets of the same cardinality. The cut-directions alternate from one node to the other: if the direction of cut is i_j for the parent node, it is equal to i_{j+1} for the two children nodes if $j+1 < n$ with n the number of possible cut-directions and i_1 otherwise. A node is a leaf (i.e., is not partitioned) if the training sample corresponding to this node contains less than n_{min} tuples. In this method the tree structure is independent of the output values of the training sample, i.e. it does not change from one iteration to another of the fitted Q iteration algorithm.

4.2.2 PRUNED CART TREE

The classical CART algorithm is used to grow completely the tree from the training set (Breiman et al., 1984). This algorithm selects at a node the test (i.e., the cut-direction and cut-point) that maximizes the average variance reduction of the output variable (see Eqn (25) in Appendix A). The tree is pruned according to the cost-complexity pruning algorithm with error estimate by ten-fold cross validation. Because of the score maximization and the post-pruning, the tree structure depends on the output values of the training sample; hence, it may change from one iteration to another.

4.2.3 TREE BAGGING

We refer here to the standard algorithm published by Breiman (1996). An ensemble of M trees is built. Each tree of the ensemble is grown from a training set by first creating a bootstrap replica (random sampling with replacement of the same number of elements) of the training set and then building an unpruned CART tree using that replica. Compared to the Pruned CART Tree algorithm, Tree Bagging often improves dramatically the accuracy of the model produced by reducing its variance but increases the computing times significantly. Note that during the tree building we also stop splitting a node if the number of training samples in this node is less than n_{min}. This algorithm has therefore two parameters, the number M of trees to build and the value of n_{min}.

	One single regression tree is built	An ensemble of regression trees is built
Tests **do depend** on the output values (o) of the $(i,o) \in \mathcal{TS}$	CART	Tree Bagging Extra-Trees
Tests **do not depend** on the output values (o) of the $(i,o) \in \mathcal{TS}$	Kd-Tree	Totally Randomized Trees

Table 1: Main characteristics of the different tree-based algorithms used in the experiments.

4.2.4 EXTRA-TREES

Besides Tree Bagging, several other methods to build tree ensembles have been proposed that often improve the accuracy with respect to Tree Bagging (e.g. Random Forests, Breiman, 2001). In this paper, we evaluate our recently developed algorithm that we call "Extra-Trees", for extremely randomized trees (Geurts et al., 2004). Like Tree Bagging, this algorithm works by building several (M) trees. However, contrary to Tree Bagging which uses the standard CART algorithm to derive the trees from a bootstrap sample, in the case of Extra-Trees, each tree is built from the complete original training set. To determine a test at a node, this algorithm selects K cut-directions at random and for each cut-direction, a cut-point at random. It then computes a score for each of the K tests and chooses among these K tests the one that maximizes the score. Again, the algorithm stops splitting a node when the number of elements in this node is less than a parameter n_{min}. Three parameters are associated to this algorithm: the number M of trees to build, the number K of candidate tests at each node and the minimal leaf size n_{min}. The detailed tree building procedure is given in Appendix A.

4.2.5 TOTALLY RANDOMIZED TREES

Totally Randomized Trees corresponds to the case of Extra-Trees when the parameter K is chosen equal to one. Indeed, in this case the tests at the different nodes are chosen totally randomly and independently from the output values of the elements of the training set. Actually, this algorithm is equivalent to an algorithm that would build the tree structure totally at random without even looking at the training set and then use the training set only to remove the tests that lead to empty branches and decide when to stop the development of a branch (Geurts et al., 2004). This algorithm can therefore be degenerated in the context of the usage that we make of it in this paper by freezing the tree structure after the first iteration, just as the Kd-Trees.

4.2.6 DISCUSSION

Table 1 classifies the different tree-based algorithms considered according to two criteria: whether they build one single or an ensemble of regression trees and whether the tests computed in the trees depend on the output values of the elements of the training set. We will see in the experiments that these two criteria often characterize the results obtained.

Concerning the value of parameter M (the number of trees to be built) we will use the same value for Tree Bagging, Extra-Trees and Totally Randomized Trees and set it equal to 50 (except in Section 5.3.6 where we will assess its influence on the solution computed).

For the Extra-Trees, experiments in Geurts et al. (2004) have shown that a good default value for the parameter K in regression is actually the dimension of the input space. In all our experiments, K will be set to this default value.

While pruning generally improves significantly the accuracy of single regression trees, in the context of ensemble methods it is commonly admitted that unpruned trees are better. This is suggested from the bias/variance tradeoff, more specifically because pruning reduces variance but increases bias and since ensemble methods reduce very much the variance without increasing too much bias, there is often no need for pruning trees in the context of ensemble methods. However, in high-noise conditions, pruning may be useful even with ensemble methods. Therefore, we will use a cross-validation approach to automatically determine the value of n_{min} in the context of ensemble methods. In this case, pruning is carried out by selecting at random two thirds of the elements of $\mathcal{TS}$, using the particular ensemble method with this smaller training set and determining for which value of n_{min} the ensemble minimizes the square error over the last third of the elements. Then, the ensemble method is run again on the whole training set using this value of n_{min} to produce the final model. In our experiments, the resulting algorithm will have the same name as the original ensemble method preceded by the term *Pruned* (e.g. Pruned Tree Bagging). The same approach will also be used to prune Kd-Trees.

4.3 Convergence of the Fitted Q Iteration Algorithm

Since the models produced by the tree-based methods may be described by an expression of the type (17) with the kernel $k_{\mathcal{TS}}(i^l, i)$ satisfying the normalizing condition (18), convergence of the fitted Q iteration algorithm can be ensured if the kernel $k_{\mathcal{TS}}(i^l, i)$ remains the same from one iteration to the other. This latter condition is satisfied when the tree structures remain unchanged throughout the different iterations.

For the Kd-Tree algorithm which selects tests independently of the output values of the elements of the training set, it can be readily seen that it will produce at each iteration the same tree structure if the minimum number of elements to split a leaf (n_{min}) is kept constant. This also implies that the tree structure has just to be built at the first iteration and that in the subsequent iterations, only the values of the terminal leaves have to be refreshed. Refreshment may be done by propagating all the elements of the new training set in the tree structure and associating to a terminal leaf the average output value of the elements having reached this leaf.

For the totally randomized trees, the tests do not depend either on the output values of the elements of the training set but the algorithm being non-deterministic, it will not produce the same tree structures at each call even if the training set and the minimum number of elements (n_{min}) to split a leaf are kept constant. However, since the tree structures are independent from the output, it is not necessary to refresh them from one iteration to the other. Hence, in our experiments, we will build the set of totally randomized trees only at the first iteration and then only refresh predictions at terminal nodes at subsequent iterations. The tree structures are therefore kept constant from one iteration to the other and this will ensure convergence.

4.4 No Divergence to Infinity

We say that the sequence of functions $\hat{Q}_N$ diverges to infinity if $\lim_{N\to\infty} \|\hat{Q}_N\|_\infty \to \infty$.

With the tree-based methods considered in this paper, such divergence to infinity is impossible since we can guarantee that, even for the tree-based methods for which the tests chosen in the tree depend on the output values (o) of the input-output pairs ((i,o)), the sequence of $\hat{Q}_N$-functions remains bounded. Indeed, the prediction value of a leaf being the average value of the outputs of the elements of the training set that correspond to this leaf, we have $\|\hat{Q}_N(x,u)\|_\infty \leq B_r + \gamma\|\hat{Q}_{N-1}(x,u)\|_\infty$ where B_r is the bound of the rewards. And, since $\hat{Q}_0(x,u) = 0$ everywhere, we therefore have $\|\hat{Q}_N(x,u)\|_\infty \leq \frac{B_r}{1-\gamma}\ \forall N \in \mathbb{N}$.

However, we have observed in our experiments that for some other supervised learning methods, divergence to infinity problems were plaguing the fitted Q iteration algorithm (Section 5.3.3); such problems have already been highlighted in the context of approximate dynamic programming (Boyan and Moore, 1995).

4.5 Computation of $\max_{\mathbf{u}\in\mathbf{U}}\hat{\mathbf{Q}}_{\mathbf{N}}(\mathbf{x},\mathbf{u})$ when u Continuous

In the case of a single regression tree, $\hat{Q}_N(x,u)$ is a piecewise-constant function of its argument u, when fixing the state value x. Thus, to determine $\max\limits_{u\in U}\hat{Q}_N(x,u)$, it is sufficient to compute the value of $\hat{Q}_N(x,u)$ for a finite number of values of U, one in each hyperrectangle delimited by the values of discretization thresholds found in the tree.

The same argument can be extended to ensembles of regression trees. However, in this case, the number of discretization thresholds might be much higher and this resolution scheme might become computationally inefficient.

5. Experiments

Before discussing our simulation results, we first give an overview of our test problems, of the type of experiments carried out and of the different metrics used to assess the performances of the algorithms.

5.1 Overview

We consider five different problems, and for each of them we use the fitted Q iteration algorithm with the tree-based methods described in Section 4 and assess their ability to extract from different sets of four-tuples information about the optimal control policy.

5.1.1 TEST PROBLEMS

The first problem, referred to as the “Left or Right” control problem, has a one-dimensional state space and a stochastic dynamics. Performances of tree-based methods are illustrated and compared with grid-based methods.

Next we consider the “Car on the Hill” test problem. Here we compare our algorithms in depth with other methods (k-nearest-neighbors, grid-based methods, a gradient version of the on-line Q-learning algorithm) in terms of accuracy and convergence properties. We also discuss CPU considerations, analyze the influence of the number of trees built on the solution, and the effect of irrelevant state variables and continuous action spaces.

The third problem is the “Acrobot Swing Up” control problem. It is a four-dimensional and deterministic control problem. While in the first two problems the four-tuples are generated randomly prior to learning, here we consider the case where the estimate of μ^* deduced from the available four-tuples is used to generate new four-tuples.

The two last problems ("Bicycle Balancing" and "Bicycle Balancing and Riding") are treated together since they differ only in their reward function. They have a stochastic dynamics, a seven-dimensional state space and a two-dimensional control space. Here we look at the capability of our method to handle rather challenging problems.

5.1.2 METRICS TO ASSESS PERFORMANCES OF THE ALGORITHMS

In our experiments, we will use the fitted Q iteration algorithm with several types of supervised learning methods as well as other algorithms like Q-learning or Watkin's $Q(\lambda)$ with various approximation architectures. To rank performances of the various algorithms, we need to define some metrics to measure the quality of the solution they produce. Hereafter we review the different metrics considered in this paper.

Expected return of a policy. To measure the quality of a solution given by a RL algorithm, we can use the stationary policy it produces, compute the expected return of this stationary policy and say that the higher this expected return is, the better the RL algorithm performs. Rather than computing the expected return for one single initial state, we define in our examples a set of initial states named X^i, chosen independently from the set of four-tuples $\mathcal{F}$, and compute the average expected return of the stationary policy over this set of initial states. This metric is referred to as the *score* of a policy and is the most frequently used one in the examples. If μ is the policy, its score is defined by:

$$\text{score of } \mu = \frac{\sum_{x \in X^i} J^{\mu}_{\infty}(x)}{\# X^i} \tag{21}$$

To evaluate this expression, we estimate, for every initial state $x \in X^i$, $J^{\mu}_{\infty}(x)$ by Monte-Carlo simulations. If the control problem is deterministic, one simulation is enough to estimate $J^{\mu}_{\infty}(x)$. If the control problem is stochastic, several simulations are carried out. For the "Left or Right" control problem, $100,000$ simulations are considered. For the "Bicycle Balancing" and "Bicycle Balancing and Riding" problems, whose dynamics is less stochastic and Monte-Carlo simulations computationally more demanding, 10 simulations are done. For the sake of compactness, the *score of* μ is represented in the figures by J^{μ}_{∞}.

Fulfillment of a specific task. The score of a policy assesses the quality of a policy through its expected return. In the "Bicycle Balancing" control problem, we also assess the quality of a policy through its ability to avoid crashing the bicycle during a certain period of time. Similarly, for the "Bicycle Balancing and Riding" control problem, we consider a criterion of the type "How often does the policy manage to drive the bicycle, within a certain period of time, to a goal ?".

Bellman residual. While the two previous metrics were relying on the policy produced by the RL algorithm, the metric described here relies on the approximate Q-function computed by the RL algorithm. For a given function $\hat{Q}$ and a given state-action pair (x,u), the Bellman residual is defined to be the difference between the two sides of the Bellman equation (Baird, 1995), the Q-function being the only function leading to a zero Bellman residual for every state-action pair. In our simulation, to estimate the quality of a function $\hat{Q}$, we exploit the Bellman residual concept by associating to $\hat{Q}$ the mean square of the Bellman residual over the set $X^i \times U$, value that will be referred to as the *Bellman residual of* $\hat{Q}$. We have

$$\text{Bellman residual of } \hat{Q} = \frac{\sum_{(x,u) \in X^i \times U} (\hat{Q}(x,u) - (H\hat{Q})(x,u))^2}{\#(X^i \times U)}. \tag{22}$$

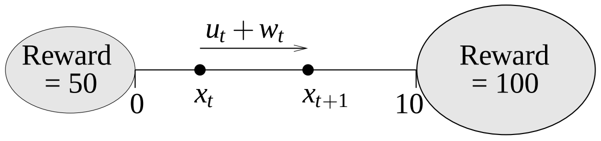


Figure 2: The "Left or Right" control problem.

This metric is only used in the "Left or Right" control problem to compare the quality of the solutions obtained. A metric relying on the score is not discriminant enough for this control problem, since all the algorithms considered can easily learn a good approximation of the optimal stationary policy. Furthermore, for this control problem, the term $(H\hat{Q})(x,u)$ in the right side of Eqn (22) is estimated by drawing independently and for each $(x,u) \in X^i \times U$, 100,000 values of w according to $P_w(.|x,u)$ (see Eqn (7)).
In the figures, the Bellman residual of $\hat{Q}$ is represented by $d(\hat{Q}, H\hat{Q})$.

5.2 The "Left or Right" Control Problem

We consider here the "Left or Right" optimal control problem whose precise definition is given in Appendix C.1.

The main characteristics of the control problem are represented on Figure 2. A point travels in the interval $[0,10]$. Two control actions are possible. One tends to drive the point to the right ($u = 2$) while the other to the left ($u = -2$). As long as the point stays inside the interval, only zero rewards are observed. When the point leaves the interval, a terminal state[12] is reached. If the point goes out on the right side then a reward of 100 is obtained while it is twice less if it goes out on the left.

Even if going out on the right may finally lead to a better reward, μ^* is not necessarily equal to 2 everywhere since the importance of the reward signal obtained after t steps is weighted by a factor $\gamma^{(t-1)} = 0.75^{(t-1)}$.

5.2.1 FOUR-TUPLES GENERATION

To collect the four-tuples we observe 300 episodes of the system. Each episode starts from an initial state chosen at random in $[0,10]$ and finishes when a terminal state is reached. During the episodes, the action u_t selected at time t is chosen at random with equal probability among its two possible values $u = -2$ and $u = 2$. The resulting set $\mathcal{F}$ is composed of 2010 four-tuples.

5.2.2 SOME BASIC RESULTS

To illustrate the fitted Q iteration algorithm behavior we first use "Pruned CART Tree" as supervised learning method. Elements of the sequence of functions $\hat{Q}_N$ obtained are represented on Figure 3. While the first functions of the sequence differ a lot, they gain in similarities when N increases which is confirmed by computing the distance on $\mathcal{F}$ between functions $\hat{Q}_N$ and $\hat{Q}_{N-1}$ (Figure 4a). We observe that the distance rapidly decreases but, due to the fact that the tree structure is refreshed at each iteration, never vanishes.

12. A terminal state can be seen as a regular state in which the system is stuck and for which all the future rewards obtained in the aftermath are zero. Note that the value of $Q_N(\textit{terminal state}, u)$ is equal to 0 $\forall N \in \mathbb{N}$ and $\forall u \in U$.

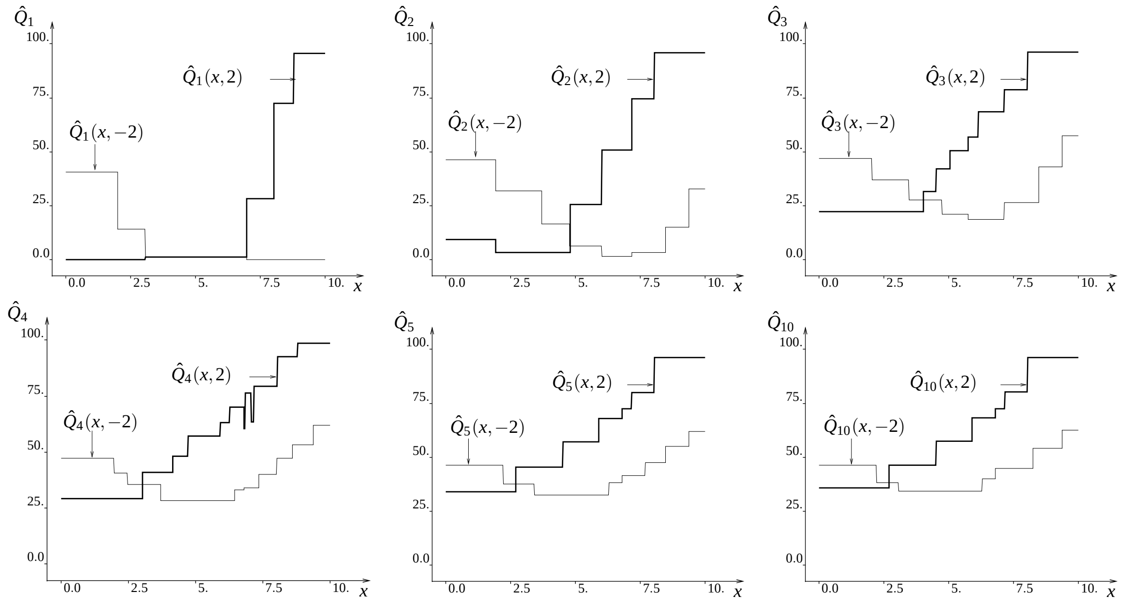
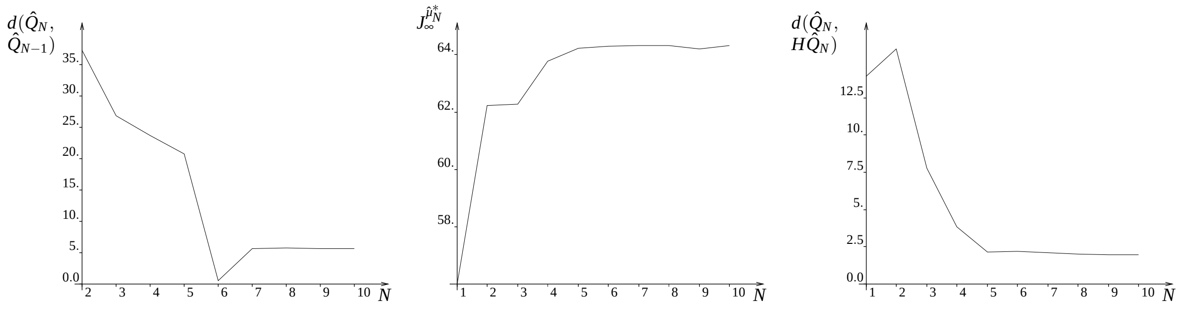


Figure 3: Representation of $\hat{Q}_N$ for different values of N. The set $\mathcal{F}$ is composed of 2010 elements and the supervised learning method used is Pruned CART Tree.



(a) $d(\hat{Q}_N, \hat{Q}_{N-1}) = \frac{\sum_{l=1}^{\#\mathcal{F}} (\hat{Q}_N(x_t^l, u_t^l) - \hat{Q}_{N-1}(x_t^l, u_t^l))^2}{\#\mathcal{F}}$

(b) $J_\infty^{\hat{\mu}_N^*} = \frac{\sum_{X^i} J_\infty^{\hat{\mu}_N^*}(x)}{\#X^i}$

(c) $d(\hat{Q}_N, H\hat{Q}_N) = \frac{\sum_{X^i \times U} (\hat{Q}_N(x,u) - (H\hat{Q}_N)(x,u))^2}{\#(X^i \times U)}$

Figure 4: Figure (a) represents the distance between $\hat{Q}_N$ and $\hat{Q}_{N-1}$. Figure (b) provides the average return obtained by the policy $\hat{\mu}_N^*$ while starting from an element of X^i. Figure (c) represents the Bellman residual of $\hat{Q}_N$.

From the function $\hat{Q}_N$ we can determine the policy $\hat{\mu}_N$. States x for which $\hat{Q}_N(x,2) \geq \hat{Q}_N(x,-2)$ correspond to a value of $\hat{\mu}_N(x) = 2$ while $\hat{\mu}_N(x) = -2$ if $\hat{Q}_N(x,2) < \hat{Q}_N(x,-2)$. For example, $\hat{\mu}^*_{10}$ consists of choosing $u = -2$ on the interval $[0,2.7[$ and $u = 2$ on $[2.7,10]$. To associate a score to each policy $\hat{\mu}^*_N$, we define a set of states $X^i = \{0,1,2,\cdots,10\}$, evaluate $J^{\hat{\mu}_N}_\infty(x)$ for each element of this set and average the values obtained. The evolution of the score of $\hat{\mu}^*_N$ with N is drawn on Figure 4b. We observe that the score first increases rapidly to become finally almost constant for values of N greater than 5.

In order to assess the quality of the functions $\hat{Q}_N$ computed, we have computed the Bellman residual of these $\hat{Q}_N$-functions. We observe in Figure 4c that even if the Bellman residual tends to decrease when N increases, it does not vanish even for large values of N. By observing Table 2, one can however see that by using 6251 four-tuples (1000 episodes) rather than 2010 (300 episodes), the Bellman residual further decreases.

5.2.3 INFLUENCE OF THE TREE-BASED METHOD

When dealing with such a system for which the dynamics is highly stochastic, pruning is necessary, even for tree-based methods producing an ensemble of regression trees. Figure 5 thus represents the $\hat{Q}_N$-functions for different values of N with the pruned version of the Extra-Trees. By comparing this figure with Figure 3, we observe that the averaging of several trees produces smoother functions than single regression trees.

By way of illustration, we have also used the Extra-Trees algorithm with fully developed trees (i.e., $n_{min} = 2$) and computed the $\hat{Q}_{10}$-function with the fitted Q iteration using the same set of four-tuples as in the previous section. This function is represented in Figure 6. As fully grown trees are able to match perfectly the output in the training set, they also catch the noise and this explains the chaotic nature of the resulting approximation.

Table 2 gathers the Bellman residuals of $\hat{Q}_{10}$ obtained when using different tree-based methods and this for different sets of four-tuples. Tree-based ensemble methods produce smaller Bellman residuals and among these methods, Extra-Trees behaves the best. We can also observe that for any of the tree-based methods used, the Bellman residual decreases with the size of $\mathcal{F}$.

Note that here, the policies produced by the different tree-based algorithms offer quite similar scores. For example, the score is 64.30 when Pruned CART Tree is applied to the 2010 four-tuple set and it does not differ from more than one percent with any of the other methods. We will see that the main reason behind this, is the simplicity of the optimal control problem considered and the small dimensionality of the state space.

5.2.4 FITTED Q ITERATION AND BASIS FUNCTION METHODS

We now assess performances of the fitted Q iteration algorithm when combined with basis function methods. Basis function methods suppose a relation of the type

$$o = \sum_{j=1}^{nbBasis} c_j \phi_j(i) \tag{23}$$

between the input and the output where $c_j \in \mathbb{R}$ and where the basis functions $\phi_j(i)$ are defined on the input space and take their values on $\mathbb{R}$. These basis functions form the approximation architec-

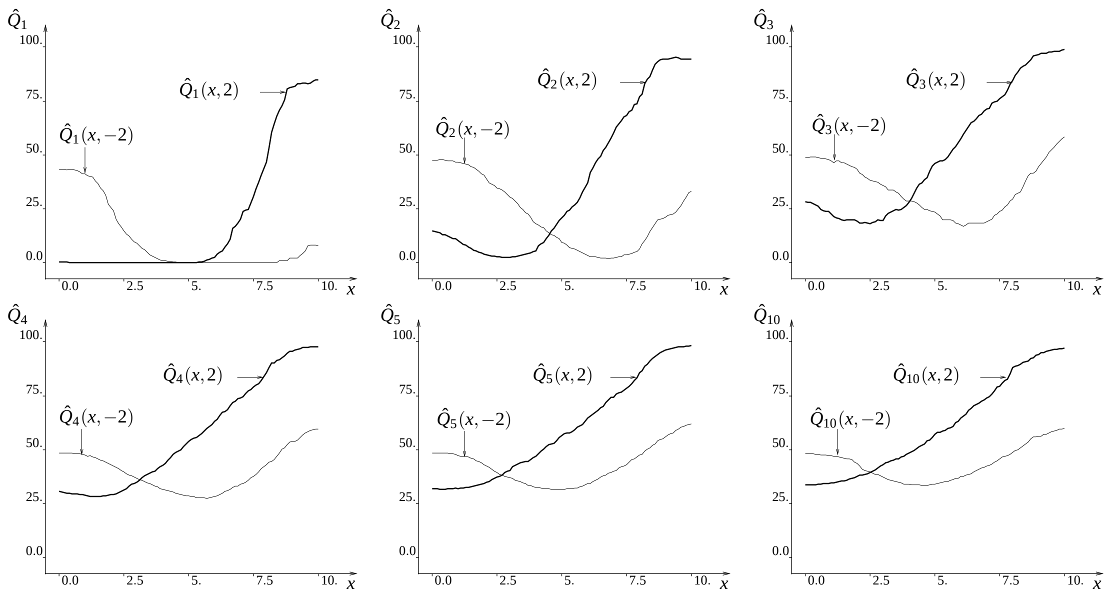


Figure 5: Representation of $\hat{Q}_N$ for different values of N. The set $\mathcal{F}$ is composed of 2010 elements and the supervised learning method used is the Pruned Extra-Trees.

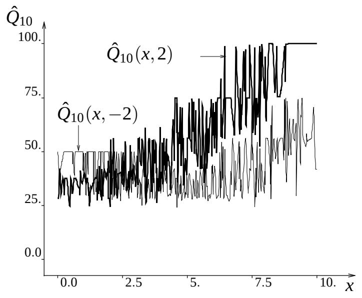


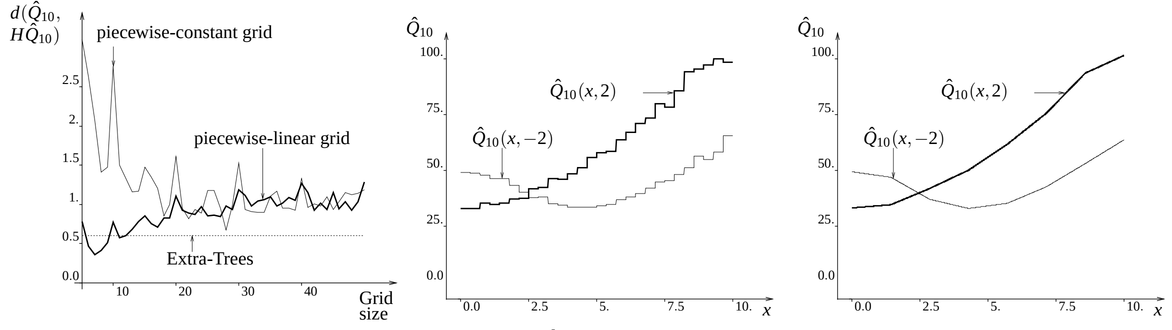
Figure 6: Representation of $\hat{Q}_{10}$ when Extra-Trees is used with no pruning

Tree-based	$\#\mathcal{F}$		
method	720	2010	6251
Pruned CART Tree	2.62	1.96	1.29
Pruned Kd-Tree	1.94	1.31	0.76
Pruned Tree Bagging	1.61	0.79	0.67
Pruned Extra-Trees	1.29	0.60	0.49
Pruned Tot. Rand. Trees	1.55	0.72	0.59

Table 2: Bellman residual of $\hat{Q}_{10}$. Three different sets of four-tuples are used. These sets have been generated by considering 100, 300 and 1000 episodes and are composed respectively of 720, 2010 and 6251 four-tuples.

ture. The training set is used to determine the values of the different c_j by solving the following minimization problem:[13]

13. This minimization problem can be solved by building the $(\#\mathcal{TS} \times nbBasis)$ Y matrix with $Y_{lj} = \phi_j(i^l)$. If Y^TY is invertible, then the minimization problem has a unique solution $c = (c_1, c_2, \cdots, c_{nbBasis})$ given by the following expression: $c = (Y^TY)^{-1}Y^Tb$ with $b \in \mathbb{R}^{\#\mathcal{TS}}$ such that $b_l = o^l$. In order to overcome the possible problem of non-invertibility of Y^TY that occurs when solution of (24) is not unique, we have added to Y^TY the strictly definite positive matrix δI, where δ is a small positive constant, before inverting it. The value of c used in our experiments as solution of (24) is therefore equal to $(Y^TY + \delta I)^{-1}Y^Tb$ where δ has been chosen equal to 0.001.



(a) Bellman residual of $\hat{Q}_{10}$

(b) $\hat{Q}_{10}$ computed when using a 28 piecewise-constant grid as approx. arch.

(c) $\hat{Q}_{10}$ computed when using a 7 piecewise-linear grid as approx. arch.

Figure 7: Fitted Q iteration with basis function methods. Two different types of approximation architectures are considered: piecewise-constant and piecewise-linear grids. 300 episodes are used to generate $\mathcal{F}$.

$$\operatorname*{arg\,min}_{(c_1,c_2,\cdots,c_{nbBasis})\in\mathbb{R}^{nbBasis}} \sum_{l=1}^{\#\mathcal{TS}} \Big(\sum_{j=1}^{nbBasis} c_j \phi_j(i^l) - o^l \Big)^2. \tag{24}$$

We consider two different sets of basis functions ϕ_j. The first set is defined by partitioning the state space into a grid and by considering one basis function for each grid cell, equal to the indicator function of this cell. This leads to piecewise constant $\hat{Q}$-functions. The other type is defined by partitioning the state space into a grid, triangulating every element of the grid and considering that $\hat{Q}(x,u) = \sum_{v\in Vertices(x)} W(x,v)\hat{Q}(v,u)$ where $Vertices(x)$ is the set of vertices of the hypertriangle x belongs to and $W(x,v)$ is the barycentric coordinate of x that corresponds to v. This leads to a set of overlapping piecewise linear basis functions, and yields a piecewise linear and continuous model. In this paper, these approximation architectures are respectively referred to as *piecewise-constant grid* and *piecewise-linear grid*. The reader can refer to Ernst (2003) for more information.

To assess performances of fitted Q iteration combined with piecewise-constant and piecewise-linear grids as approximation architectures, we have used several grid resolutions to partition the interval $[0,10]$ (a 5 grid, a 6 grid, $\cdots$, a 50 grid). For each grid, we have used fitted Q iteration with each of the two types of approximation architectures and computed $\hat{Q}_{10}$. The Bellman residuals obtained by the different $\hat{Q}_{10}$-functions are represented on Figure 7a. We can see that basis function methods with piecewise-constant grids perform systematically worse than Extra-Trees, the tree-based method that produces the lowest Bellman residual. This type of approximation architecture leads to the lowest Bellman residual for a 28 grid and the corresponding $\hat{Q}_{10}$-function is sketched in Figure 7b. Basis function methods with piecewise-linear grids reach their lowest Bellman residual for a 7 grid, Bellman residual that is smaller than the one obtained by Extra-Trees. The corresponding smoother $\hat{Q}_{10}$-function is drawn on Figure 7b.

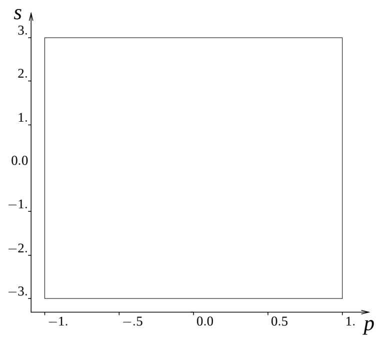
Even if piecewise-linear grids were able to produce on this example better results than the tree-based methods, it should however be noted that it has been achieved by tuning the grid resolution and that this resolution strongly influences the quality of the solution. We will see below that, as the state space dimensionality increases, piecewise-constant or piecewise-linear grids do not compete anymore with tree-based methods. Furthermore, we will also observe that piecewise-linear grids may lead to divergence to infinity of the fitted Q iteration algorithm (see Section 5.3.3).

5.3 The "Car on the Hill" Control Problem

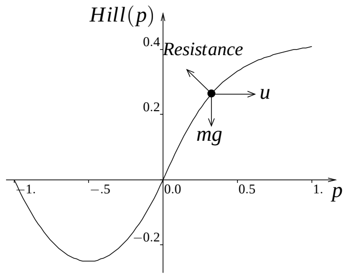
We consider here the "Car on the Hill" optimal control problem whose precise definition is given in Appendix C.2.

A car modeled by a point mass is traveling on a hill (the shape of which is given by the function $Hill(p)$ of Figure 8b). The action u acts directly on the acceleration of the car (Eqn (31), Appendix C) and can only assume two extreme values (full acceleration ($u = 4$) or full deceleration ($u = -4$)). The control problem objective is roughly to bring the car in a minimum time to the top of the hill ($p = 1$ in Figure 8b) while preventing the position p of the car to become smaller than -1 and its speed s to go outside the interval $[-3, 3]$. This problem has a (continuous) state space of dimension two (the position p and the speed s of the car) represented on Figure 8a.

Note that by exploiting the particular structure of the system dynamics and the reward function of this optimal control problem, it is possible to determine with a reasonable amount of computation the exact value of $J_\infty^{\mu^*}$ (Q) for any state x (state-action pair (x,u)).[14]



(a) $X \setminus \{terminal\ state\}$



(b) Representation of $Hill(p)$ (shape of the hill) and of the different forces applied to the car.

Figure 8: The "Car on the Hill" control problem.

5.3.1 SOME BASIC RESULTS

To generate the four-tuples we consider episodes starting from the same initial state corresponding to the car stopped at the bottom of the hill (i.e., $(p,s) = (-0.5, 0)$) and stopping when the car leaves the region represented on Figure 8a (i.e., when a terminal state is reached). In each episode, the action u_t at each time step is chosen with equal probability among its two possible values $u = -4$ and $u = 4$. We consider 1000 episodes. The corresponding set $\mathcal{F}$ is composed of 58090 four-tuples. Note that during these 1000 episodes the reward $r(x_t, u_t, w_t) = 1$ (corresponding to an arrival of the car at the top of the hill with a speed comprised in $[-3, 3]$) has been observed only 18 times.

14. To compute $J_\infty^{\mu^*}(x)$, we determine by successive trials the smallest value of k for which one of the two following conditions is satisfied (i) at least one sequence of actions of length k leads to a reward equal to 1 when $x_0 = x$ (ii) all

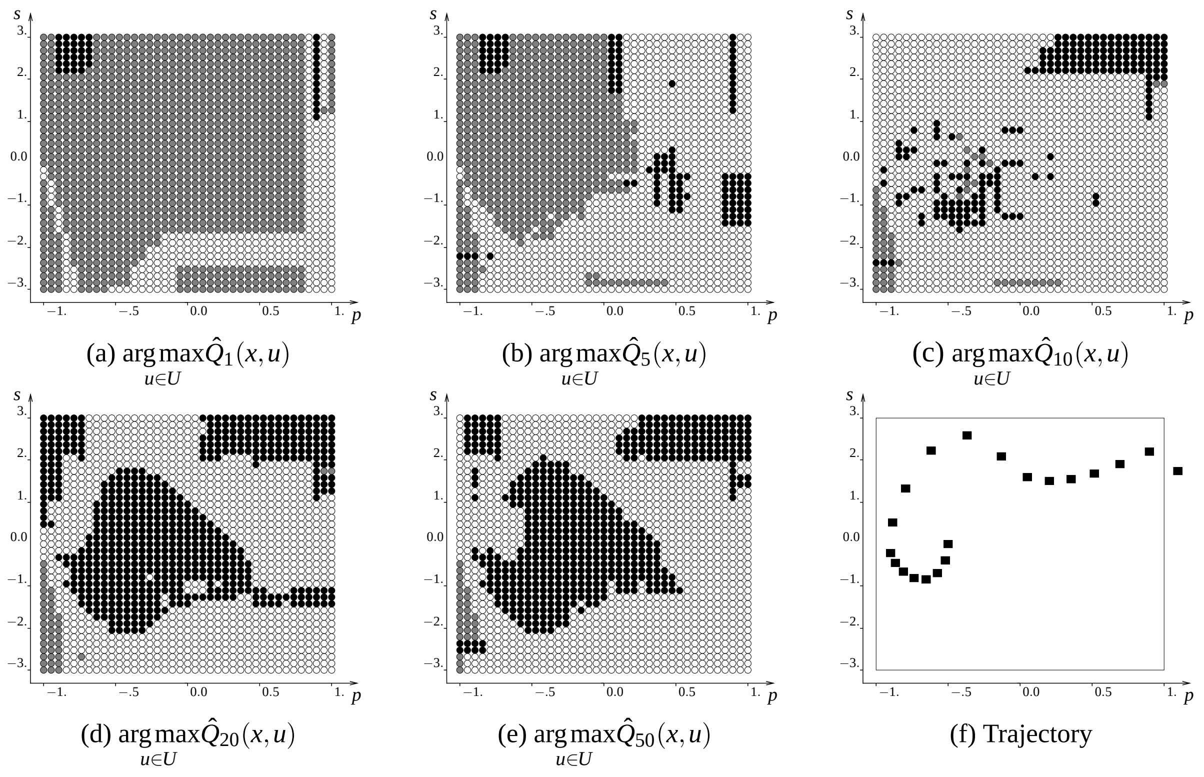
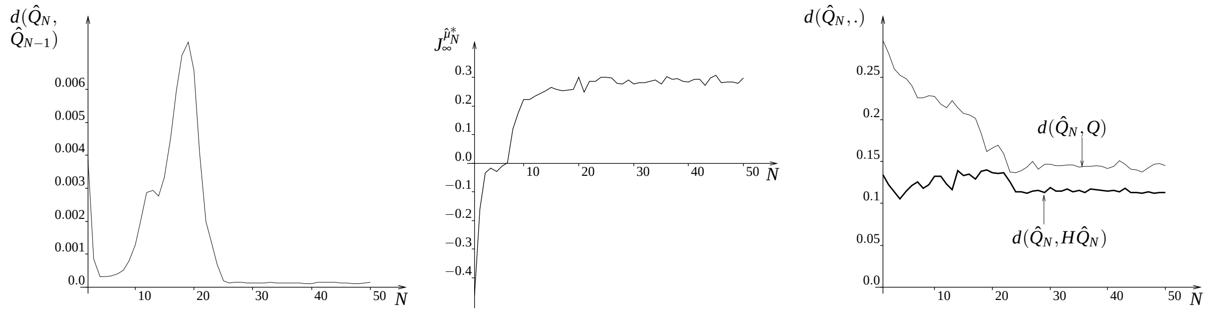


Figure 9: (a)-(e): Representation of $\hat{\mu}^*_N$ for different values of N. (f): Trajectory when $x_0 = (-0.5, 0)$ and when the policy $\hat{\mu}^*_{50}$ is used to control the system.



(a) $d(\hat{Q}_N, \hat{Q}_{N-1}) = \frac{\sum_{l=1}^{\#\mathcal{F}}(\hat{Q}_N(x_t^l,u_t^l)-\hat{Q}_{N-1}(x_t^l,u_t^l))^2}{\#\mathcal{F}}$

(b) $J_\infty^{\hat{\mu}_N^*} = \frac{\sum_{X^i} J_\infty^{\hat{\mu}_N^*}(x)}{\#X^i}$

(c) $d(\hat{Q}_N, F) = \frac{\sum_{X^i \times U}(\hat{Q}_N(x,u)-F(x,u))^2}{\#(X^i \times U)}$
$F = Q$ or $H\hat{Q}_N$

Figure 10: Figure (a) represents the distance between $\hat{Q}_N$ and $\hat{Q}_{N-1}$. Figure (b) provides the average return obtained by the policy $\hat{\mu}_N^*$ while starting from an element of X^i. Figure (c) represents the distance between $\hat{Q}_N$ and Q as well as the Bellman residual of $\hat{Q}_N$ as a function of N (distance between $\hat{Q}_N$ and $H\hat{Q}_N$).

We first use Tree Bagging as the supervised learning method. As the action space is binary, we again model the functions $\hat{Q}_N(x,-4)$ and $\hat{Q}_N(x,4)$ by two ensembles of 50 trees each, and $n_{min} = 2$. The policy $\hat{\mu}_1^*$ so obtained is represented on Figure 9a. Black bullets represent states for which $\hat{Q}_1(x,-4) > \hat{Q}_1(x,4)$, white bullets states for which $\hat{Q}_1(x,-4) < \hat{Q}_1(x,4)$ and grey bullets states for which $\hat{Q}_1(x,-4) = \hat{Q}_1(x,4)$. Successive policies $\hat{\mu}_N^*$ for increasing N are given on Figures 9b-9e.

On Figure 9f, we have represented the trajectory obtained when starting from $(s,p) = (-0.5,0)$ and using the policy $\hat{\mu}_{50}^*$ to control the system. Since, for this particular state the computation of $J_\infty^{\mu^*}$ gives the same value as $J_\infty^{\hat{\mu}_{50}^*}$, the trajectory drawn is actually an optimal one.

Figure 10a shows the evolution of distance between $\hat{Q}_N$ and $\hat{Q}_{N-1}$ with N. Notice that while a monotonic decrease of the distance was observed with the "Left or Right" control problem (Figure 4a), it is not the case anymore here. The distance first decreases and then from $N = 5$ suddenly increases to reach a maximum for $N = 19$ and to finally redecrease to an almost zero value. Actually, this apparently strange behavior is due to the way the distance is evaluated and to the nature of the control problem. Indeed, we have chosen to use in the distance computation the state-action pairs (x_t^l, u_t^l) $l = 1, \cdots, \#\mathcal{F}$ from the set of four-tuples. Since most of the states x_t^l are located around the initial state $(p,s) = (-0.5,0)$ (see Figure 11), the distance is mostly determined by variations between $\hat{Q}_N$ and $\hat{Q}_{N-1}$ in this latter region. This remark combined with the fact that the algorithm needs a certain number of iterations before obtaining values of $\hat{Q}_N$ around $(p,s) = (-0.5,0)$ different from zero explains this sudden increase of the distance.[15]

To compute policy scores, we consider the set $X^i : X^i = \{(p,s) \in X \setminus \{x^t\} | \exists i,j \in \mathbb{Z} | (p,s) = (0.125 * i, 0.375 * j)\}$ and evaluate the average value of $J_\infty^{\hat{\mu}_N^*}(x)$ over this set. The evolution of the

the sequences of actions of length k lead to a reward equal to -1 when $x_0 = x$. Let k_{min} be this smallest value of k. Then $J_\infty^{\mu^*}(x)$ is equal to $\gamma^{k_{min}-1}$ if condition (i) is satisfied when $k = k_{min}$ and $-\gamma^{k_{min}-1}$ otherwise.

15. The reason for $\hat{Q}_N$ being equal to zero around $(p,s) = (-0.5,0)$ for small values of N is that when the system starts from $(-0.5,0)$ several steps are needed to observe non zero rewards whatever the policy used.

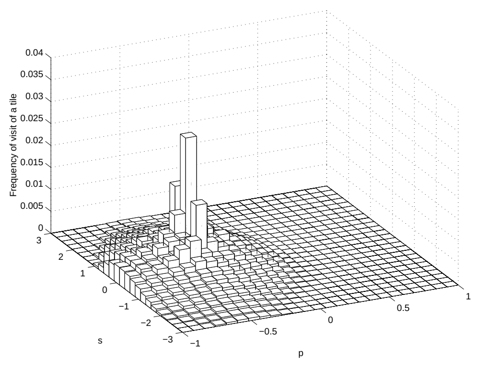
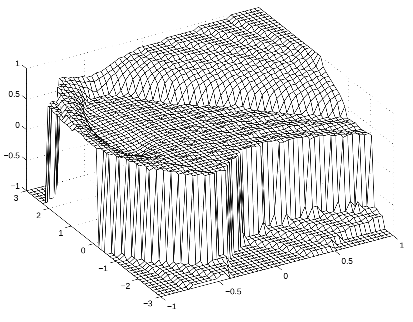
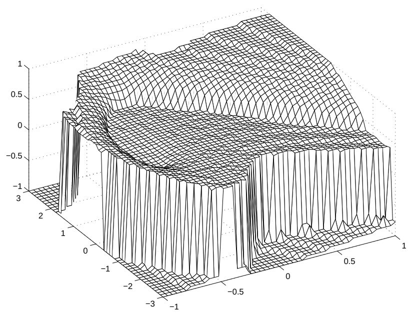
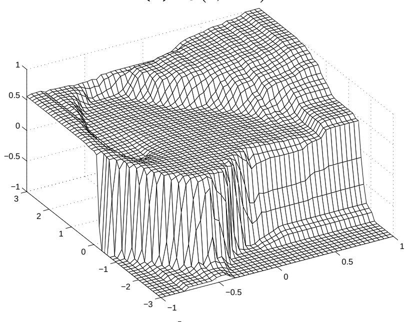


Figure 11: Estimation of the x_t distribution while using episodes starting from $(-0.5, 0)$ and choosing actions at random.


(a) $Q(., -4)$


(b) $Q(., 4)$


(c) $\hat{Q}_{50}(., -4)$

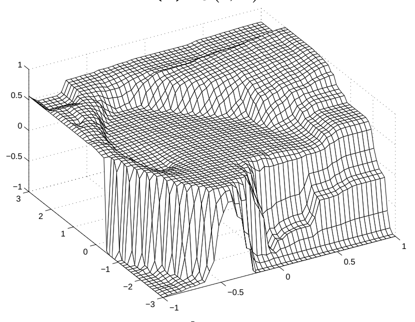
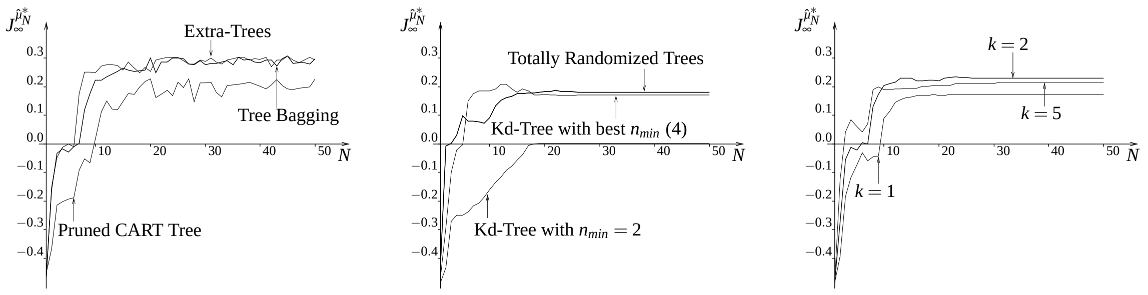

(d) $\hat{Q}_{50}(., 4)$

Figure 12: Representation of the Q-function and of $\hat{Q}_{50}$. $\hat{Q}_{50}$ is computed by using fitted Q iteration together with Tree Bagging.

score for increasing values of N is represented in Figure 10b. We see that the score rapidly increases to finally oscillate slightly around a value close to 0.295. The score of μ^* being equal to 0.360, we see that the policies $\hat{\mu}^*_N$ are suboptimal. To get an idea of how different is the $\hat{Q}_{50}$-function computed by fitted Q iteration from the true Q-function, we have represented both functions on Figure 12. As we may observe, some significant differences exist between them, especially in areas were very few information has been generated, like the state space area around $x = (-1, 3)$.



(a) Output values (o) used to choose tree tests

(b) Output values (o) not used to choose tree tests

(c) kNN

Figure 13: Influence of the supervised learning method on the solution. For each supervised learning method $\hat{Q}_N(x,-4)$ and $\hat{Q}_N(x,4)$ are modeled separately. For Tree Bagging, Extra-Trees, and Totally Randomized Trees, the trees are developed completely ($n_{min} = 2$). The distance used in the nearest neighbors computation is the Euclidean distance.

5.3.2 INFLUENCE OF THE TREE-BASED METHOD AND COMPARISON WITH kNN.

Figure 13a sketches the scores obtained by the policies $\hat{\mu}_N^*$ when using different tree-based methods which use the output values (o) of the input-output pair ((i,o)) of the training set to compute the tests. It is clear that Tree Bagging and Extra-Trees are significantly superior to Pruned CART Tree. Figure 13b compares the performances of tree-based methods for which the tests are chosen independently of the output values. We observe that even when using the value of n_{min} leading to the best score, Kd-Tree does not perform better than Totally Randomized Trees. On Figure 13c, we have drawn the scores obtained with a k-nearest-neighbors (kNN) technique.

Notice that the score curves corresponding to the k-nearest-neighbors, Totally Randomized Trees, and Kd-Tree methods stabilize indeed after a certain number of iterations.

To compare more systematically the performances of all these supervised learning algorithms, we have computed for each one of them and for several sets of four-tuples the score of $\hat{\mu}_{50}^*$. Results are gathered in Table 3. A first remark suggested by this table and which holds for all the supervised learning methods is that the more episodes are used to generate the four-tuples, the larger the score of the induced policy. Compared to the other methods, performances of Tree Bagging and Extra-Trees are excellent on the two largest sets. Extra-Trees still gives good results on the smallest set but this is not true for Tree Bagging. The strong deterioration of Tree Bagging performances is mainly due to the fact that when dealing with this set of four-tuples, information about the optimal solution is really scarce (only two four-tuples correspond to a reward of 1) and, since a training instance has 67% chance of being present in a bootstrap sample, Tree Bagging often discards some critical information. On the other hand, Extra-Trees and Totally Randomized Trees which use the whole training set to build each tree do not suffer from this problem. Hence, these two methods behave particularly well compared to Tree Bagging on the smallest set.

One should also observe from Table 3 that even when used with the value of k that produces the largest score, kNN is far from being able to reach for example the performances of the Extra-Trees.

Supervised learning method	Nb of episodes used to generate $\mathcal{F}$		
	1000	300	100
Kd-Tree (Best n_{min})	0.17	0.16	-0.06
Pruned CART Tree	0.23	0.13	-0.26
Tree Bagging	0.30	0.24	-0.09
Extra-Trees	0.29	0.25	0.12
Totally Randomized Trees	0.18	0.14	0.11
kNN (Best k)	0.23	0.18	0.02

Table 3: Score of $\hat{\mu}^*_{50}$ for different set of four-tuples and supervised learning methods.

5.3.3 FITTED Q ITERATION AND BASIS FUNCTION METHODS

In Section 5.2.4, when dealing with the "Left or Right" control problem, basis function methods with two types of approximation architectures, piecewise-constant or piecewise-linear grids, have been used in combination with the fitted Q iteration algorithm.

In this section, the same types of approximation architectures are also considered and, for each type of approximation architecture, the policy $\hat{\mu}^*_{50}$ has been computed for different grid resolutions (a 10×10 grid, a 11×11 grid, $\cdots$, a 50×50 grid). The score obtained by each policy is represented on Figure 14a. The horizontal line shows the score previously obtained on the same sample of four-tuples by Tree Bagging. As we may see, whatever the grid considered, both approximation architectures lead to worse results than Tree Bagging, the best performing tree-based method. The highest score is obtained by a 18×18 grid for the piecewise-constant approximation architecture and by a 14×14 grid for the piecewise-linear approximation architecture. These two highest scores are respectively 0.21 and 0.25, while Tree Bagging was producing a score of 0.30. The two corresponding policies are sketched in Figures 14b and 14c. Black polygons represent areas where $\hat{Q}(x,-4) > \hat{Q}(x,4)$, white polygons areas where $\hat{Q}(x,-4) < \hat{Q}(x,4)$ and grey polygons areas where $\hat{Q}(x,-4) = \hat{Q}(x,4)$.

When looking at the score curve corresponding to piecewise-linear grids as approximation architectures, one may be surprised to note its harsh aspect. For some grids, this type of approximation architecture leads to some good results while by varying slightly the grid size, the score may strongly deteriorate. This strong deterioration of the score is due to fact that for some grid sizes, the fitted Q iteration actually diverges to infinity while it is not the case for other grid sizes. Divergence to infinity of the algorithm is illustrated on Figures 15a and 15c where we have drawn for a 12×12 grid the distance between $\hat{Q}_N$ and $\hat{Q}_{N-1}$, $\hat{Q}_N$ and Q, and $\hat{Q}_N$ and $H\hat{Q}_N$. Remark that a logarithmic scale has been used for the y-axis. When using Tree Bagging in the inner loop of the fitted Q iteration, similar graphics have been drawn (Figure 10) and the reader may refer to them for comparison.

5.3.4 COMPARISON WITH Q-LEARNING

In this section we use a gradient descent version of the standard Q-learning algorithm to compute the c_j parameters of the approximation architectures of Section 5.3.3. The degree of correction α used inside this algorithm is chosen equal to 0.1 and the estimate of the Q-function is initialized to 0 everywhere. This latter being refreshed by this algorithm on a four-tuple by four-tuple basis, we have chosen to use each element of $\mathcal{F}$ only once to refresh the estimate of the Q-function. The

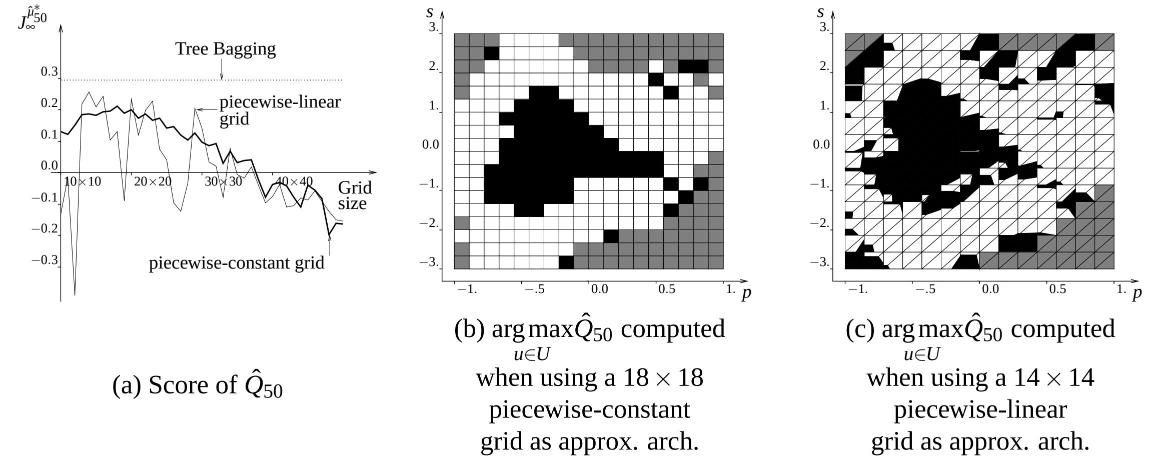


Figure 14: Fitted Q iteration with basis function methods. Two different types of approximation architecture are considered: piecewise-constant and piecewise-linear grids. $\mathcal{F}$ is composed of the four-tuples gathered during 1000 episodes.

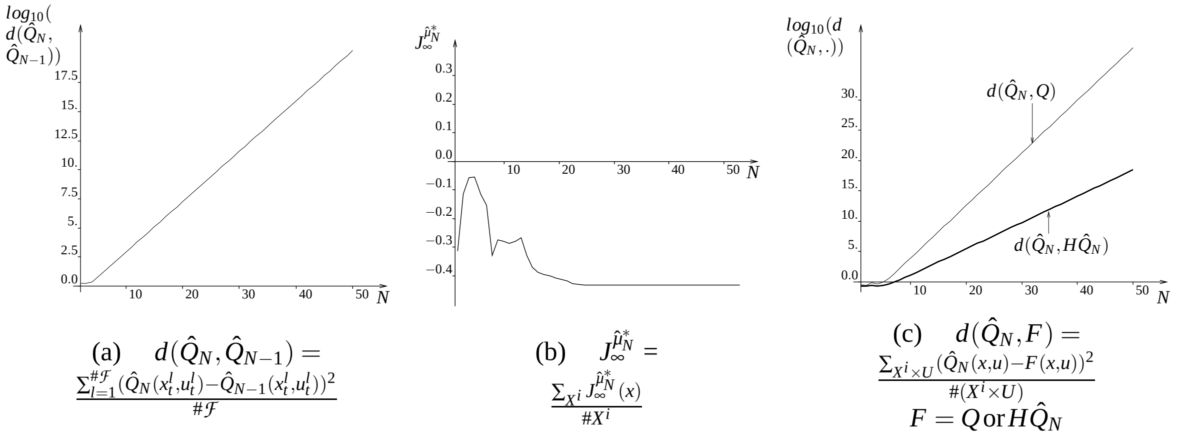
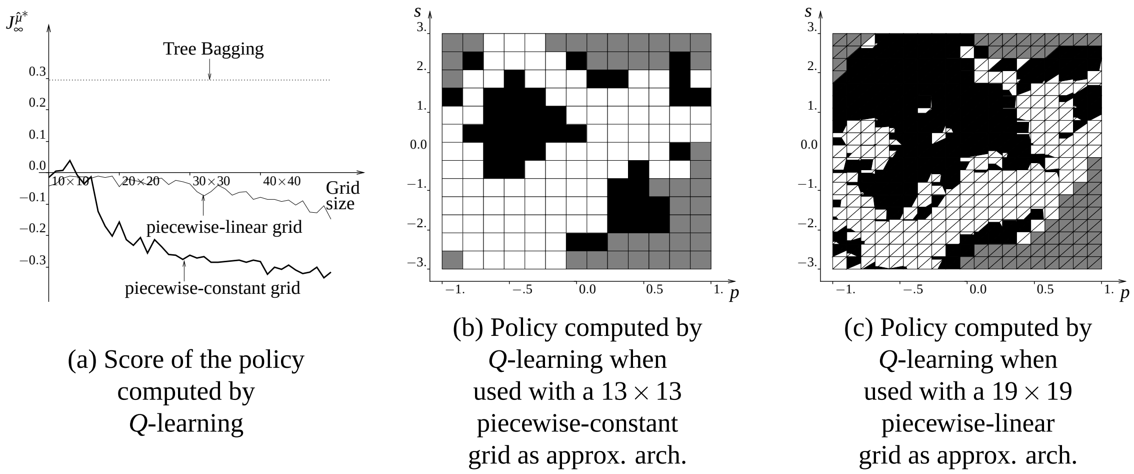


Figure 15: Fitted Q iteration algorithm with basis function methods. A 12×12 piecewise-linear grid is the approximation architecture considered. The sequence of $\hat{Q}_N$-functions *diverges to infinity*.



(a) Score of the policy computed by Q-learning

(b) Policy computed by Q-learning when used with a 13×13 piecewise-constant grid as approx. arch.

(c) Policy computed by Q-learning when used with a 19×19 piecewise-linear grid as approx. arch.

Figure 16: Q-learning with piecewise-constant and piecewise-linear grids as approximation architectures. Each element of $\mathcal{F}$ is used once to refresh the estimate of the Q-function. The set $\mathcal{F}$ is composed of the four-tuples gathered during 1000 episodes.

main motivation for this is to compare the performances of the fitted Q iteration algorithm with an algorithm that does not require to store the four-tuples.[16]

The scores obtained by Q-learning for the two types of approximation architectures and for different grid sizes are reported on Figure 16a. Figure 16b (16c) represents the policies that have led, by screening different grid sizes, to the highest score when piecewise-constant grids (piecewise-linear grids) are the approximation architectures considered. By comparing Figure 16a with Figure 14a, it is obvious that fitted Q iteration exploits more effectively the set of four-tuples than Q-learning. In particular, the highest score is 0.21 for fitted Q iteration while it is only of 0.04 for Q-learning. If we compare the score curves corresponding to piecewise-linear grids as approximation architectures, we observe also that the highest score produced by fitted Q iteration (over the different grids), is higher than the highest score produced by Q-learning. However, when fitted Q iteration is plagued with some divergence to infinity problems, as illustrated on Figure 15, it may lead to worse results than Q-learning.

Observe that even when considering $10,000$ episodes with Q-learning, we still obtain worse scores than the one produced by Tree Bagging with 1000 episodes. Indeed, the highest score pro-

16. Performances of the gradient descent version of the *Q-learning* algorithm could be improved by processing several times each four-tuple to refresh the estimate of the Q-function, for example by using the experience replay technique of Lin (1993). This however requires to store the four-tuples.

It should also be noted that if a piecewise-constant grid is the approximation architecture considered, if each element of $\mathcal{F}$ is used an infinite number of times to refresh the estimate of the Q-function and if the sequence of αs satisfies the stochastic approximation condition (i.e., $\sum_{k=1}^{\infty} \alpha_k \to \infty$ and $\sum_{k=1}^{\infty} \alpha_k^2 < \infty$, α_k being the value of α the kth times the estimate of the Q-function is refreshed), then the Q-function estimated by the Q-learning algorithm would be the same as the one estimated by fitted Q iteration using the same piecewise-constant grid as approximation architecture. This can be seen by noting that in such conditions, the Q-function estimated by Q-learning would be the same as the one estimated by a model-based algorithm using the same grid (see Ernst (2003), page 131 for the proof) which in turn can be shown to be equivalent to fitted Q iteration (see Ernst et al., 2005).

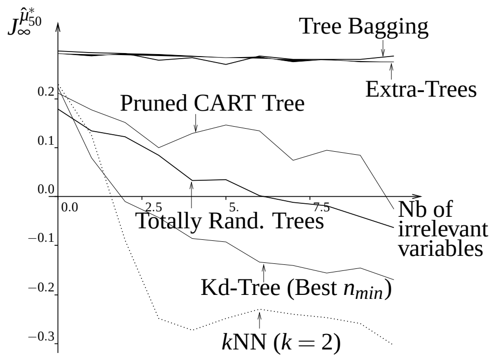


Figure 17: Score of $\hat{\mu}_{50}^*$ when four-tuples are gathered during 1000 episodes and some variables that do not contain any information about the position and the speed of the car are added to the state vector.

duced by Q-learning with $10,000$ episodes and over the different grid sizes, is 0.23 if piecewise-constant grids are considered as approximation architectures and 0.27 for piecewise-linear grids, compared to a score of 0.30 for Tree Bagging with 1000 episodes.

At this point, one may wonder whether the poor performances of Q-learning are due to the fact that it is used without eligibility traces. To answer this question, we have assessed the performances of Watkin's $Q(\lambda)$ algorithm (Watkins, 1989) that combines Q-learning with eligibility traces.[17] The degree of correction α is chosen, as previously, equal to 0.1 and the value of λ is set equal to 0.95. This algorithm has been combined with piecewise-constant grids and 1000 episodes have been considered. The best score obtained over the different grids is equal to -0.05 while it was slightly higher (0.04) for Q-learning.

5.3.5 ROBUSTNESS WITH RESPECT TO IRRELEVANT VARIABLES

In this section we compare the robustness of the tree-based regression methods and kNN with respect to the addition of irrelevant variables. Indeed, in many practical applications the elementary variables which compose the state vector are not necessarily all of the same importance in determining the optimal control action. Thus, some variables may be of paramount importance, while some others may influence only weakly or even sometimes not at all the optimal control.

On Figure 17, we have drawn the evolution of the score when using four-tuples gathered during 1000 episodes and adding progressively irrelevant variables to the state-vector.[18] It is clear that not all the methods are equally robust to the introduction of irrelevant variables. In particular, we observe that the three methods for which the approximation architecture is independent of the output variable are not robust: the kNN presents the fastest deterioration, followed by Kd-Tree and Totally Randomized Trees. The latter is more robust because it averages out several trees, which gives the relevant variables a better chance to be taken into account in the model.

17. In Watkin's $Q(\lambda)$, accumulating traces are considered and eligibility traces are cut when a non-greedy action is chosen. Remark that by not cutting the eligibility traces when a non-greedy action is selected, we have obtained worse results.
18. See Section C.2 for the description of the irrelevant variables dynamics.

On the other hand, the methods which take into account the output variables in their approximation architecture are all significantly more robust than the former ones. Among them, Tree Bagging and Extra-Trees which are based on the averaging of several trees are almost totally immune, since even with 10 irrelevant variables (leading to the a 12-dimensional input space) their score decrease is almost insignificant.

This experiment shows that the regression tree based ensemble methods which adapt their kernel to the output variable may have a strong advantage in terms of robustness over methods with a kernel which is independent of the output, even if these latter have nicer convergence properties.

5.3.6 Influence of the Number of Regression Trees in an Ensemble

In this paper, we have chosen to build ensembles of regression trees composed of 50 trees ($M = 50$, Section 4.2), a number of elements which, according to our simulations, is large enough to ensure that accuracy of the models produced could not be improved significantly by increasing it. In order to highlight the influence of M on the quality of the solution obtained, we have drawn on Figure 18, for the different regression tree based ensemble methods, the quality of the solution obtained as a function of M. We observe that the score grows rapidly with M, especially with Extra-Trees and Tree Bagging in which cases a value of $M = 10$ would have been sufficient to obtain a good solution.

Note that since the CPU times required to compute the solution grow linearly with the number of trees built, computational requirements of the regression tree based ensemble methods could be adjusted by choosing a value of M.

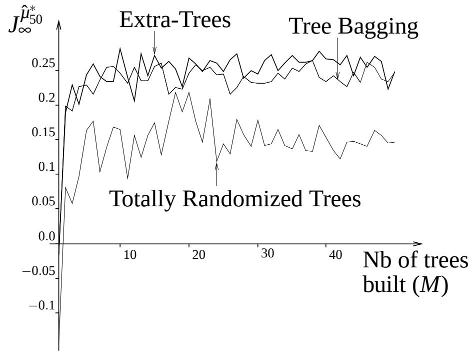
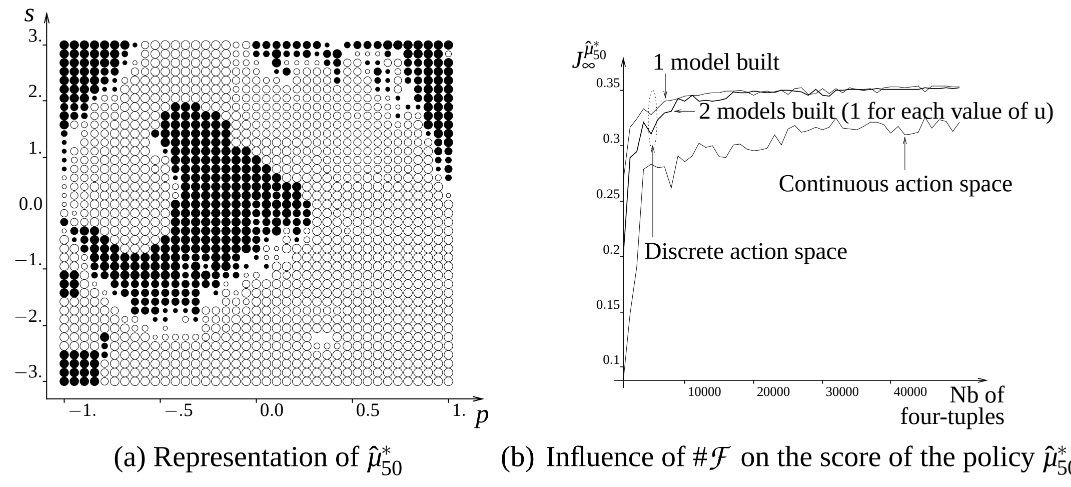


Figure 18: Evolution of the score of $\hat{\mu}_{50}^*$ with the number of trees built. $\mathcal{F}$ is composed of the four-tuples gathered during 300 episodes.

5.3.7 Car on the Hill with Continuous Action Space

To illustrate the use of the fitted Q iteration algorithm with continuous action spaces we consider here $U = [-4, 4]$ rather than $\{-4, 4\}$. We use one-step episodes with (x_0, u_0) drawn at random with uniform probability in $X \times U$ to generate a set $\mathcal{F}$ of $50,000$ four-tuples and use Tree Bagging with 50 trees as supervised learning method. We have approximated the maximization over the continuous action space needed during the training sample refreshment step (see Eqn (13), Figure 1) by an exhaustive search over $u \in \{-4, -3, \cdots, 3, 4\}$. The policy $\hat{\mu}_{50}^*$ thus obtained by our algorithm after



(a) Representation of $\hat{\mu}^*_{50}$ (b) Influence of $\#\mathcal{F}$ on the score of the policy $\hat{\mu}^*_{50}$

Figure 19: Car on the Hill with continuous action space. Tree Bagging is used on one-step episodes with (x_0, u_0) drawn at random in $X \times U$ are used to generate the four-tuples.

50 iterations is represented on Figure 19a, where black bullets are used to represent states x for which $\hat{\mu}^*_{50}(x)$ is negative, white ones when $\hat{\mu}^*_{50}(x)$ is positive. The size of a bullet is proportional to the absolute value of the control signal $|\hat{\mu}^*_{50}(x)|$. We see that the control policy obtained is not far from a "bang-bang" policy.

To compare these results with those obtained in similar conditions with a discrete action space, we have made two additional experiments, where the action space is restricted again to the extreme values, i.e. $u \in \{-4, 4\}$. The two variants differ in the way the Q_N-functions are modeled. Namely, in the first case one single model is learned where u is included in the input variables whereas in the second case one model is learned per possible value of u, i.e. one model for $Q_N(x, -4)$ and one for $Q_N(x, 4)$. All experiments are carried out for an increasing number of samples and a fixed number of iterations ($N = 50$) and bagged trees ($M = 50$). The three curves of Figure 19b show the resulting scores. The two upper curves correspond to the score of the policy $\hat{\mu}^*_{50}$ obtained when considering a discrete action space $U = \{-4, 4\}$. We observe that both curves are close to each other and dominate the "Continuous action space" scores. Obviously the discrete approach is favored because of the "bang-bang" nature of the problem; nevertheless, the continuous action space approach is able to provide results of comparable quality.[19]

5.3.8 COMPUTATIONAL COMPLEXITY AND CPU TIME CONSIDERATIONS

Table 4 gathers the CPU times required by the fitted Q iteration algorithm to carry out 50 iterations (i.e., to compute $\hat{Q}_{50}(x, u)$) for different types of supervised learning methods and different sets $\mathcal{F}$. We have also given in the same table the repartition of the CPU times between the two tasks the algorithm has to perform, namely the task which consists of building the training sets (evaluation of Eqns (12) and (13) for all $l \in \{1, 2, \cdots, \#\mathcal{F}\}$) and the task which consists of building the models from the training sets. These two tasks are referred to hereafter respectively as the "Training Set

19. The bang-bang nature was also observed in Smart and Kaelbling (2000), where continuous and a discrete action spaces are treated on the "Car on the Hill" problem, with qualitatively the same results.

Building" (TSB) task and the "Model Building" (MB) task. When Kd-Tree or Totally Randomized Trees are used, each tree structure is frozen after the first iteration and only the value of its terminal nodes are refreshed. The supervised learning technique referred to in the table as "*k*NN smart" is a smart implementation the fitted Q iteration algorithm when used with kNN in the sense that the k nearest neighbors of x^l_{t+1} are determined only once and not recomputed at each subsequent iteration of the algorithm.

Supervised learning algorithm	CPU times consumed by the Models Building (MB) and Training Sets Building (TSB) tasks								
	$\#\mathcal{F}=5000$			$\#\mathcal{F}=10000$			$\#\mathcal{F}=20000$		
	MB	TSB	Total	MB	TSB	Total	MB	TSB	Total
Kd-Tree ($n_{min}=4$)	0.01	0.39	0.40	0.04	0.91	0.95	0.06	2.05	2.11
Pruned CART Tree	16.6	0.3	16.9	42.4	0.8	43.2	95.7	1.6	97.3
Tree Bagging	97.8	54.0	151.8	219.7	142.3	362.0	474.4	333.7	808.1
Extra-Trees	24.6	55.5	80.1	51.0	145.8	196.8	105.72	337.48	443.2
Totally Rand. Trees	0.4	67.8	68.2	0.8	165.3	166.2	1.7	407.5	409.2
*k*NN	0.0	1032.2	1032.2	0.0	4096.2	4096.2	0.0	16537.7	16537.7
*k*NN smart	0.0	21.0	21.0	0.0	83.0	83.0	0.0	332.4	332.4

Table 4: CPU times (in seconds on a Pentium-IV, 2.4GHz, 1GB, Linux) required to compute $\hat{Q}_{50}$. For each of the supervised learning method $\hat{Q}_N(x,-4)$ and $\hat{Q}_N(x,4)$ have been modeled separately. 50 trees are used with Tree Bagging, Extra-Trees and Totally Randomized Trees and the value of k for kNN is 2.

By analyzing the table, the following remarks apply:

- CPU times required to build the training sets are non negligible with respect to CPU times for building the models (except for Pruned CART Tree which produces only one single regression tree). In the case of Extra-Trees, Totally Randomized Trees and *k*NN, training set update is even the dominant task in terms of CPU times.

- Kd-Tree is (by far) the fastest method, even faster than Pruned CART Tree which produces also one single tree. This is due to the fact that the MB task is really inexpensive. Indeed, it just requires building one single tree structure at the first iteration and refresh its terminal nodes in the aftermath.

- Concerning Pruned CART Tree, it may be noticed that tree pruning by ten-fold cross validation requires to build in total eleven trees which explains why the CPU times for building 50 trees with Tree Bagging is about five times greater than the CPU times required for Pruned CART Tree.

- The MB task is about four times faster with Extra-Trees than with Tree Bagging, because Extra-Trees only computes a small number (K) of test scores, while CART searches for an optimal threshold for each input variable. Note that the trees produced by the Extra-Trees algorithm are slightly more complex, which explains why the TSB task is slightly more time consuming. On the two largest training sets, Extra-Trees leads to almost 50 % less CPU times than Tree Bagging.

- The MB task for Totally Randomized Trees is much faster than the MB task for Extra-Trees, mainly because the totally randomized tree structures are built only at the first iteration. Note that, when totally randomized trees are built at the first iteration, branch development is not stopped when the elements of the local training set have the same value, because it can not be assumed that these elements would still have the same value in subsequent iterations. This also implies that totally randomized trees are more complex than trees built by Extra-Trees and explains why the TSB task with Totally Randomized Trees is more time consuming.

- Full kNN is the slowest method. However, its smart implementation is almost 50 times (the number of iterations realized by the algorithm) faster than the naive one. In the present case, it is even faster than the methods based on regression trees ensembles. However, as its computational complexity (in both implementations) is quadratic with respect to the size of the training set while it is only slightly super-linear for tree-based methods, its advantage quickly vanishes when the training set size increases.

5.4 The "Acrobot Swing Up" Control Problem

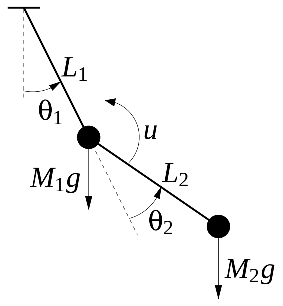


Figure 20: Representation of the Acrobot.

We consider here the "Acrobot Swing Up" control problem whose precise definition is given in Appendix C.3.

The Acrobot is a two-link underactuated robot, depicted in Figure 20. The second joint applies a torque (represented by u), while the first joint does not. The system has four continuous state variables: two joint positions (θ_1 and θ_2) and two joint velocities ($\dot{\theta}_1$ and $\dot{\theta}_2$). This system has been extensively studied by control engineers (e.g. Spong, 1994) as well as machine learning researchers (e.g. Yoshimoto et al., 1999).

We have stated this control problem so that the optimal stationary policy brings the Acrobot quickly into a specified neighborhood of its unstable inverted position, and ideally as close as possible to this latter position. Thus, the reward signal is equal to zero except when this neighborhood is reached, in which case it is positive (see Eqn (44) in Appendix C.3). The torque u can only take two values: -5 and 5.

5.4.1 FOUR-TUPLES GENERATION

To generate the four-tuples we have considered 2000 episodes starting from an initial state chosen at random in $\{(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2) \in \mathbb{R}^4 | \, \theta_1 \in [-\pi+1, \pi-1], \theta_2 = \dot{\theta}_1 = \dot{\theta}_2 = 0\}$ and finishing when $t = 100$ or earlier if the terminal state is reached before.[20]

Two types of strategies are used here to control the system, leading to two different sets of four-tuples. The first one is the same as in the previous examples: at each instant the system is controlled by using a policy that selects actions fully at random. The second strategy however interleaves the sequence of four-tuples generation with the computation of an approximate Q-function from the four-tuples already generated and uses a policy that exploits this $\hat{Q}$-function to control the system while generating additional four-tuples. More precisely, it generates the four-tuples according to the following procedure:[21]

- Initialize $\hat{Q}$ to zero everywhere and $\mathcal{F}$ to the empty set;
- Repeat 20 times:
  - use an ε-greedy policy from $\hat{Q}$ to generate 100 episodes and add the resulting four-tuples to $\mathcal{F}$;
  - use the fitted Q iteration algorithm to build a new approximation $\hat{Q}_N$ from $\mathcal{F}$ and set $\hat{Q}$ to $\hat{Q}_N$.

where $\varepsilon = 0.1$ and where the fitted Q iteration algorithm is combined with the Extra-Trees ($n_{min} = 2$, $K = 5$, $M = 50$) algorithm and iterates 100 times.

The random policy strategy produces a set of four-tuples composed of $193,237$ elements while $154,345$ four-tuples compose the set corresponding to the ε-greedy policy.

Note that since the states $(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2)$ and $(\theta_1 + 2k_1\pi, \theta_2 + 2k_2\pi, \dot{\theta}_1, \dot{\theta}_2)$ $k_1, k_2 \in \mathbb{Z}$ are equivalent from a physical point of view, we have, before using the four-tuples as input of the fitted Q iteration algorithm, added or subtracted to the values of θ_1 and θ_2 a multiple of 2π to guarantee that these values belong to the interval $[-\pi, \pi]$. A similar transformation is also carried out on each state $(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2)$ before it is used as input of a policy $\hat{\mu}^*_N(x)$.

5.4.2 SIMULATION RESULTS

First, we consider the set of four-tuples gathered when using the ε-greedy policy to control the system. We have represented on Figure 21 the evolution of the Acrobot starting with zero speed in a downward position and being controlled by the policy $\hat{\mu}_{100}$ when the fitted Q iteration algorithm is used with Extra-Trees. As we observe, the control policy computed manages to bring the Acrobot close to its unstable equilibrium position.

In order to attribute a score to a policy $\hat{\mu}_N$, we define a set

$$X^i = \{(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2) \in \mathbb{R}^4 | \theta_1 \in \{-2, -1.9, \cdots, 2\}, \theta_2 \dot{\theta}_1 = \dot{\theta}_2 = 0\},$$

evaluate $J^{\hat{\mu}_N}_\infty(x)$ for each element x of this set and average the values obtained. The evolution of the score of $\hat{\mu}^*_N$ with N for different tree-based methods is drawn on Figure 22. Extra-Trees gives

20. We say that a terminal state is reached when the Acrobot has reached the target neighborhood of the unstable equilibrium set.
21. The ε-greedy policy chooses with probability $1-\varepsilon$ the control action u_t at random in the set $\{u \in U | u = \arg\max_{u \in U} \hat{Q}(x_t, u)\}$) and with probability ε at random in U.

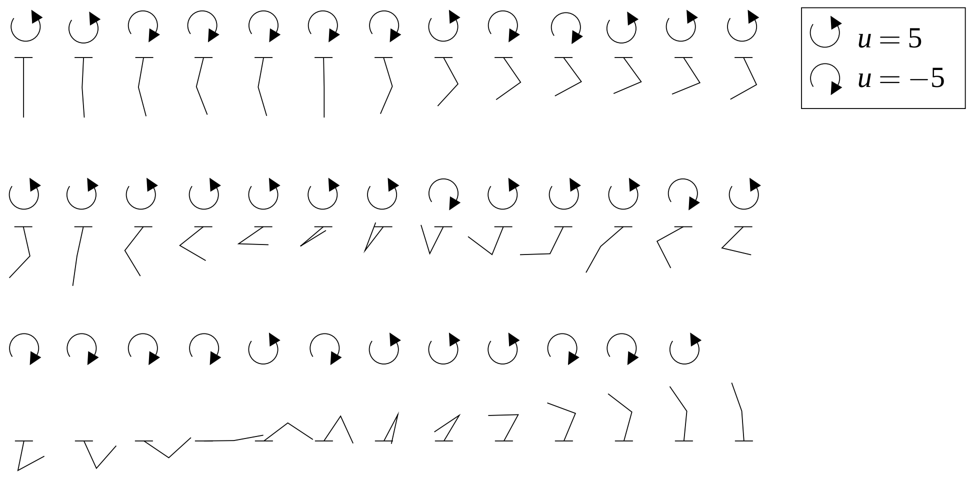


Figure 21: A typical control sequence with a learned policy. The Acrobot starts with zero speed in a downward position. Its position and the applied control action are represented at successive time steps. The last step corresponds to a terminal state.

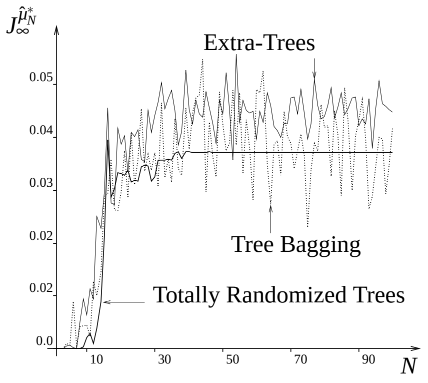


Figure 22: Score of $\hat{\mu}_N^*$ when using the set generated by the ε-greedy policy.

Tree-based method	Policy which generates $\mathcal{F}$	
	ε-greedy	Random
Pruned CART Tree	0.0006	0.
Kd-Tree (Best n_{min})	0.0004	0.
Tree Bagging	0.0417	0.0047
Extra-Trees	0.0447	0.0107
Totally Rand. Trees	0.0371	0.0071

Table 5: Score of $\hat{\mu}_{100}^*$ for the two sets of four-tuples and different tree-based methods.

the best score while the score of Tree Bagging seems to oscillate around the value of the score corresponding to Totally Randomized Trees.

The score obtained by $\hat{\mu}_{100}^*$ for the different tree-based methods and the different sets of four-tuples is represented in Table 5. One may observe, once again, that methods which build an ensemble of regression trees perform much better. Surprisingly, Totally Randomized Trees behaves well compared to Tree Bagging and to a lesser extent to Extra-Trees. On the other hand, the single tree methods offer rather poor performances. Note that for Kd-Tree, we have computed $\hat{\mu}_{100}^*$ and its associated score for each value of $n_{min} \in \{2,3,4,5,10,20,\cdots,100\}$ and reported in the Table 5 the highest score thus obtained.

We can also observe from this table that the scores obtained while using the set of four-tuples corresponding to the totally random policy are much worse than those obtained when using an ε-greedy policy. This is certainly because the use of a totally random policy leads to very little information along the optimal trajectories starting from elements of X^i. In particular, out of the 2000 episodes used to generate the set of four-tuples, only 21 manage to reach the goal region. Note also that while Extra-Trees remains superior, Tree Bagging offers this time poorer performances than Totally Randomized Trees.

5.5 The Bicycle

We consider two control problems related to a bicycle which moves at constant speed on a horizontal plane (Figure 23). For the first problem, the agent has to learn how to balance the bicycle. For the second problem, he has not only to learn how to balance the bicycle but also how to drive it to a specific goal. The exact definitions of the two optimal control problems related to these two tasks are given in Appendix C.4.[22]

These two optimal control problems have the same system dynamics and differ only by their reward function. The system dynamics is composed of seven variables. Four are related to the bicycle itself and three to the position of the bicycle on the plane. The state variables related to the bicycle are ω (the angle from vertical to the bicycle), $\dot{\omega}$, θ (the angle the handlebars are displaced from normal) and $\dot{\theta}$. If $|\omega|$ becomes larger than 12 degrees, then the bicycle is supposed to have fallen down and a terminal state is reached. The three state variables related to the position of the bicycle on the plane are the coordinates (x_b, y_b) of the contact point of the back tire with the horizontal plane and the angle ψ formed by the bicycle frame and the x-axis. The actions are the torque T applied to the handlebars (discretized to $\{-2, 0, 2\}$) and the displacement d of the rider (discretized to $\{-0.02, 0, 0.02\}$). The noise in the system is a uniformly distributed term in $[-0.02, 0.02]$ added to the displacement component action d.

As is usually the case when dealing with these bicycle control problems, we suppose that the state variables x_b and y_b cannot be observed. Since these two state variables do not intervene in the dynamics of the other state variables nor in the reward functions considered, they may be taken as irrelevant variables for the optimal control problems and, therefore, their lack of observability does not make the control problem partially observable.

The reward function for the "Bicycle Balancing" control problem (Eqn (56), page 553) is such that zero rewards are always observed, except when the bicycle has fallen down, in which case the reward is equal to -1. For the "Bicycle Balancing and Riding" control problem, a reward of -1 is also observed when the bicycle has fallen down. However, this time, non-zero rewards are also observed when the bicycle is riding (Eqn (57), page 553). Indeed, the reward r_t when the bicycle is supposed not to have fallen down, is now equal to $c_{reward}(d_{angle}(\psi_t) - d_{angle}(\psi_{t+1}))$ with

22. Several other papers treat the problems of balancing and/or balancing and riding a bicycle (e.g. Randløv and Alstrøm, 1998; Ng and Jordan, 1999; Lagoudakis and Parr, 2003b,a). The reader can refer to them in order to put the performances of fitted Q iteration in comparison with some other RL algorithms. In particular, he could refer to Randløv and Alstrøm (1998) to get an idea of the performances of SARSA(λ), an on-line algorithm, on these bicycle control problems and to Lagoudakis and Parr (2003b) to see how the Least-Square Policy Iteration (LSPI), a batch mode RL algorithm, performs. If his reading of these papers and of the simulation results reported in Sections 5.5.1 and 5.5.2 is similar to ours, he will conclude that fitted Q iteration combined with Extra-Trees performs much better than SARSA(λ) in terms of ability to extract from the information acquired from interaction with the system, a good control policy. He will also conclude that LSPI and fitted Q iteration combined with Extra-Trees are both able to produce good policies with approximately the same number of episodes. Moreover, the reader will certainly notice the obvious strong dependence of performances of LSPI and SARSA(λ) on the choice of the parametric approximation architecture these algorithms use to approximate the Q-function, which makes extremely difficult any strict comparison with them.

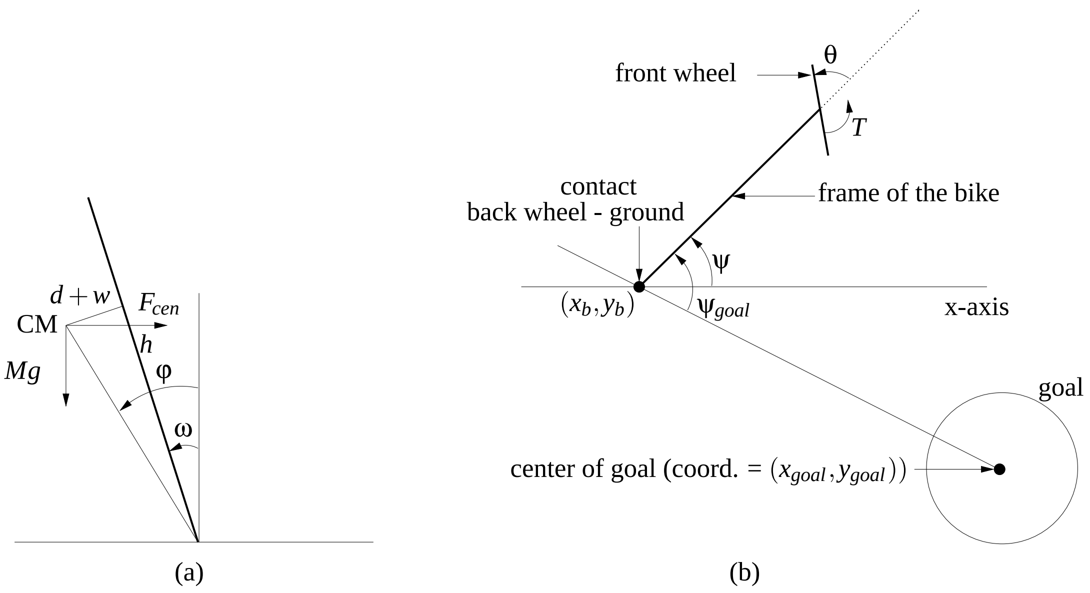


Figure 23: Figure (a) represents the bicycle seen from behind. The thick line represents the bicycle. CM is the center of mass of the bicycle and the cyclist. h represents the height of the CM over the ground. ω represents the angle from vertical to bicycle. The angle φ represents the total angle of tilt of the center of mass. Action d represents how much the agent decides to displace the center of mass from the bicycle's plan and w is the noise laid on the choice of displacement, to simulate imperfect balance. Figure (b) represents the bicycle seen from above. θ is the angle the handlebars are displaced from normal, ψ the angle formed by the bicycle frame and the x-axis and ψ_{goal} the angle between the bicycle frame and the line joining the back - wheel ground contact and the center of the goal. T is the torque applied by the cyclist to the handlebars. (x_b,y_b) is the contact point of the backwheel with the ground.

$c_{reward} = 0.1$ and $d_{angle}(\psi) = \min_{k\in\mathbb{Z}}|\psi + 2k\pi|$ ($d_{angle}(\psi)$ represents the "distance" between an angle ψ and the angle 0). Positive rewards are therefore observed when the bicycle frame gets closer to the position $\psi = 0$ and negative rewards otherwise. With such a choice for the reward function, the optimal policy μ^* tends to control the bicycle so that it moves to the right with its frame parallel to the x-axis. Such an optimal policy or a good approximate $\hat{\mu}^*$ of it can then be used to drive the bicycle to a specific goal. If ψ_{goal_t} represents the angle between the bicycle frame and a line joining the point (x_b,y_b) to the center of the goal (x_{goal},y_{goal}) (Figure 23b), this is achieved by selecting at time t the action $\hat{\mu}^*(\omega_t,\dot{\omega}_t,\theta_t,\dot{\theta}_t,\psi_{goal_t})$, rather than $\hat{\mu}^*(\omega_t,\dot{\omega}_t,\theta_t,\dot{\theta}_t,\psi_t)$. In this way, we proceed as if the line joining (x_b,y_b) to (x_{goal},y_{goal}) were the x-axis when selecting control actions, which makes the bicycle moving towards the goal.[23] Note that in our simulations, $(x_{goal},y_{goal}) = (1000,0)$ and the goal is a ten meter radius circle centered on this point. Concerning the value of the decay

23. The reader may wonder why, contrary to the approach taken by other authors (Lagoudakis, Parr, Randløv, Alstrøm, Ng, Jordan),

- we did not consider in the state signal available during the four-tuples generation phase ψ_{goal} rather than ψ (which would have amounted here to consider $(\omega,\dot{\omega},\theta,\dot{\theta},\psi_{goal})$ as state signal when generating the four-tuples) factor γ, it has been chosen for both problems equal to 0.98. The influence of γ and c_{reward} on the trajectories for the "Bicycle Balancing and Riding" control problem will be discussed later.

All episodes used to generate the four-tuples start from a state selected at random in

$$\{(\omega,\dot{\omega},\theta,\dot{\theta},x_b,y_b,\psi) \in \mathbb{R}^7 | \psi \in [-\pi,\pi] \text{ and } \omega=\dot{\omega}=\theta=\dot{\theta}=x_b=y_b=0\},$$

and end when a terminal state is reached, i.e. when the bicycle is supposed to have fallen down. The policy considered during the four-tuples generation phase is a policy that selects at each instant an action at random in U.

For both optimal control problems, the set X^i considered for the score computation (Section 5.1.2) is:

$$X^i = \{(\omega,\dot{\omega},\theta,\dot{\theta},x_b,y_b,\psi) \in \mathbb{R}^7 | \psi \in \{-\pi, -\frac{3\pi}{4}, \cdots, \pi\} \text{ and } \omega=\dot{\omega}=\theta=\dot{\theta}=x_b=y_b=0\}.$$

Since ψ and $\psi+2k\pi$ ($k \in \mathbb{Z}$) are equivalent from a physical point of view, in our simulations we have modified each value of ψ observed by a factor $2k\pi$ in order to guarantee that it always belongs to $[-\pi,\pi]$.

5.5.1 THE "BICYCLE BALANCING" CONTROL PROBLEM

To generate the four-tuples, we have considered 1000 episodes. The corresponding set $\mathcal{F}$ is composed of 97,969 four-tuples. First, we discuss the results obtained by Extra-Trees ($n_{min}=4$, $K=7$, $M=50$)[24] and then we assess the performances of the other tree-based methods.

Figure 24a represents the evolution of the score of the policies $\hat{\mu}^*_N$ with N when using Extra-Trees. To assess the quality of a policy μ, we use also another criterion than the score. For this criterion, we simulate for each $x_0 \in X^i$ ten times the system with the policy μ, leading to a total of 90 trajectories. If no terminal state has been reached before $t=50,000$, that is if the policy was able to avoid crashing the bicycle during 500 seconds (the discretization time step is 0.01 second), we say that the trajectory has been successful. On Figure 24c we have represented for the different policies $\hat{\mu}^*_N$ the number of successful trajectories among the 90 simulated. Remark that if from $N=60$ the score remains really close to zero, polices $\hat{\mu}^*_{60}$ and $\hat{\mu}^*_{70}$ do not produce as yet any successful trajectories, meaning that the bicycle crashes for large values of t even if these are smaller than

- the reward function for the bicycle balancing and riding control problem does not give directly information about the direction to the goal (which would have led here to observe at $t+1$ the reward $c_{reward}(d_{angle}(\psi_{goal_t}) - d_{angle}(\psi_{goal_{t+1}})))$.

We did not choose to proceed like this because $\psi_{goal_{t+1}}$ depends not only on ψ_{goal_t} and θ_t but also on x_{b_t} and y_{b_t}. Therefore, since we suppose that the coordinates of the back tire cannot be observed, the optimal control problems would have been partially observable if we had replaced ψ by ψ_{goal} in the state signal and the reward function. Although in our simulations this does not make much difference since $\psi \simeq \psi_{goal}$ during the four-tuples generation phases, we prefer to stick with fully observable systems in this paper.

24. When considering ensemble methods (Extra-Trees, Totally Randomized Trees, Tree Bagging) we always keep constant the value of these parameters. Since we are not dealing with a highly stochastic system, as for the case of the "Left or Right" control problem, we decided not to rely on the pruned version of these algorithms. However, we found out that by developing the trees fully ($n_{min}=2$), variance was still high. Therefore, we decided to use a larger value for n_{min}. This value is equal to 4 and was leading to a good bias-variance tradeoff. Concerning the value of $K=7$, it is equal to the dimension of the input space, that is the dimension of the state signal $(\omega,\dot{\omega},\theta,\dot{\theta},\psi)$ plus the dimension of the action space.

Tree-based method	Control problem	
	Bicycle Balancing	Bicycle Balancing and Riding
Pruned CART Tree	-0.02736	-0.03022
Kd-Tree (Best $n_{min} \in \{2,3,4,5,10,20,\cdots,100\}$)	-0.02729	-0.10865
Tree Bagging ($n_{min} = 4$, $M = 50$)	0	0.00062
Extra-Trees ($n_{min} = 4$, $K = 7$, $M = 50$)	0	0.00157
Totally Randomized Trees ($n_{min} = 4$, $M = 50$)	-0.00537	-0.01628

Table 6: Score of $\hat{\mu}^*_{300}$ for the "Balancing" and "Balancing and Riding" problems. Different tree-based methods are considered, with a 1000 episode based set of four-tuples.

$50,000$. Figure 24b gives an idea of the trajectories of the bicycle on the horizontal plane when starting from the different elements of X^i and being controlled by $\hat{\mu}^*_{300}$.

To assess the influence of the number of four-tuples on the quality of the policy computed, we have drawn on Figure 24d the number of successful trajectories when different number of episodes are used to generate $\mathcal{F}$. As one may see, by using Extra-Trees, from 300 episodes ($\simeq 10,000$ four-tuples) only successful trajectories are observed. Tree Bagging and Totally Randomized Trees perform less well. It should be noted that Kd-Tree and Pruned CART Tree were not able to produce any successful trajectories, even for the largest set of four-tuples. Furthermore, the fact that we obtained some successful trajectories with Totally Randomized Trees is only because we have modified the algorithm to avoid selection of tests according to ψ, a state variable that plays for this "Bicycle Balancing" control problem the role of an irrelevant variable (ψ does not intervene in the reward function and does not influence the dynamics of ω, $\dot{\omega}$, θ, $\dot{\theta}$) (see also Section 5.3.5). Note that the scores obtained by $\hat{\mu}^*_{300}$ for the different tree-based methods when considering the $97,969$ four-tuples are reported in the second column of Table 6.

5.5.2 THE "BICYCLE BALANCING AND RIDING" CONTROL PROBLEM

To generate the four-tuples, we considered 1000 episodes that led to a set $\mathcal{F}$ composed of 97,241 elements. First, we study the performances of Extra-Trees ($n_{min} = 4$, $K = 7$, $M = 50$). Figure 25a represents the evolution of the score of $\hat{\mu}^*_N$ with N. The final value of the score (score of $\hat{\mu}^*_{300}$) is equal to 0.00157.

As mentioned earlier, with the reward function chosen, the policy computed by our algorithm should be able to drive the bicycle to the right, parallel to the x-axis, provided that the policy is a good approximation of the optimal policy. To assess this ability, we have simulated, for each $x_0 \in X^i$, the system with the policy $\hat{\mu}^*_{300}$ and have represented on Figure 25b the different trajectories of the back tire. As one may see, the policy tends indeed to drive the bicycle to the right, parallel to the x-axis. The slight shift that exists between the trajectories and the x-axis (the shift is less than 10 degrees) could be reduced if more four-tuples were used as input of the fitted Q iteration algorithm.

Now, if rather than using the policy $\hat{\mu}^*_{300}$ with the state signal $(\omega, \dot{\omega}, \theta, \dot{\theta}, \psi)$ we consider the state signal $(\omega, \dot{\omega}, \theta, \dot{\theta}, \psi_{goal})$, where ψ_{goal} is the angle between the bicycle frame and the line joining (x_b, y_b) with (x_{goal}, y_{goal}), we indeed observe that the trajectories converge to the goal (see Figure 25c). Under such conditions, by simulating from each $x_0 \in X^i$ ten times the system over 50,000 time steps, leading to a total of 90 trajectories, we observed that every trajectory managed to reach

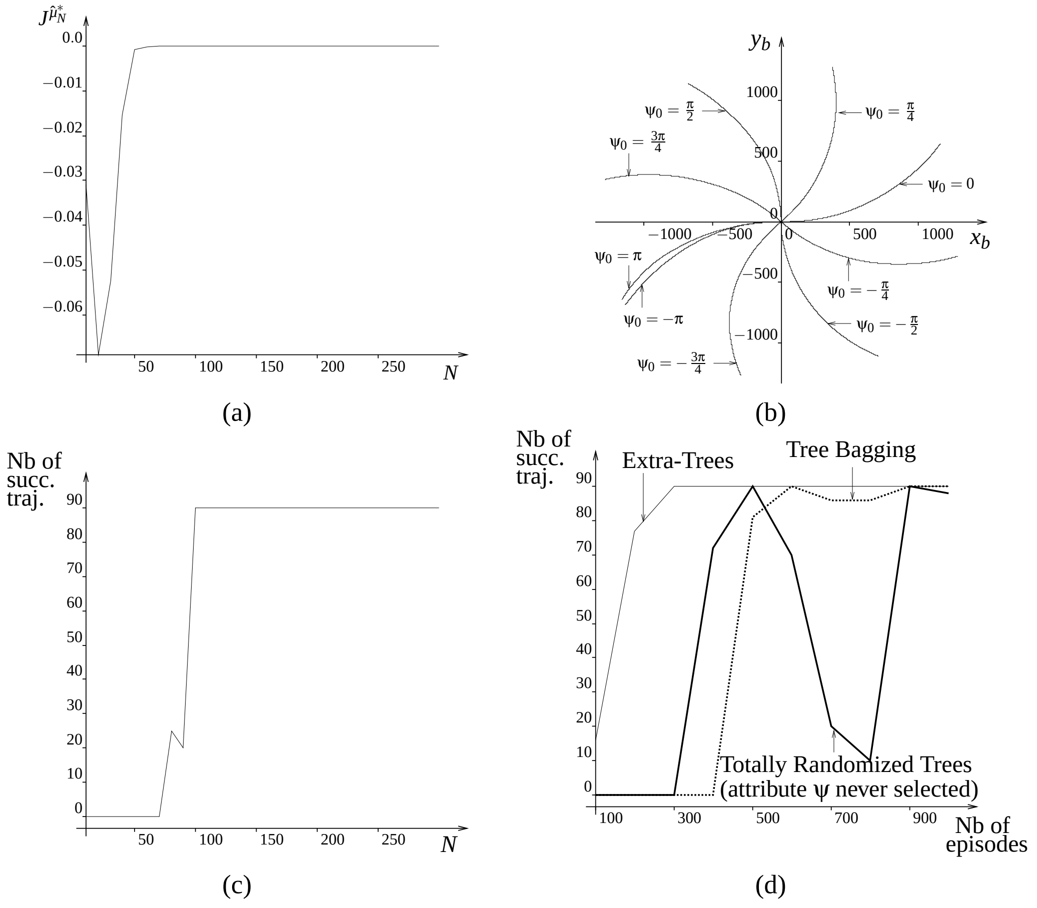


Figure 24: The "Bicycle Balancing" control problem. Figure (a) represents the score of $\hat{\mu}^*_N$ with Extra-Trees and 1000 episodes used to generate $\mathcal{F}$. Figure (b) sketches trajectories of the bicycle on the $x_b - y_b$ plane when controlled by $\hat{\mu}^*_{300}$. Trajectories are drawn from $t = 0$ till $t = 50,000$. Figure (c) represents the number of times (out of 90 trials) the policy $\hat{\mu}^*_N$ (Extra-Trees, 1000 episodes) manages to balance the bicycle during 50,000 time steps, i.e. 500 s. Figure (d) gives for different numbers of episodes and for different tree-based methods the number of times $\hat{\mu}^*_{300}$ leads to successful trajectories.

a 2 meter neighborhood of (x_{goal}, y_{goal}). Furthermore, every trajectory was able to reach the goal in less that $47,000$ time steps. Note that, since the bicycle rides at constant speed $v = \frac{10}{3.6} \simeq 2.77ms^{-1}$ and since the time discretization step is $0.01\,s$, the bicycle does not have to cover a distance of more that $1278\,m$ before reaching the goal while starting from any element of X^i.

It is clear that these good performances in terms of the policy ability to drive the bicycle to the goal depend on the choice of the reward function. For example, if the same experience is repeated with c_{reward} chosen equal to 1 rather than 0.1 in the reward function, the trajectories lead rapidly to a terminal state. This can be explained by the fact that, in this case, the large positive rewards obtained for moving the frame of the bicycle parallel to the x-axis lead to a control policy that modifies too rapidly the bicycle riding direction which tends to destabilize it. If now, the coefficient c_{reward} is taken smaller than 0.1, the bicycle tends to turn more slowly and to take more time to reach the goal. This is illustrated on Figure 25d where a trajectory corresponding to $c_{reward} = 0.01$ is drawn together with a trajectory corresponding to $c_{reward} = 0.1$. On this figure, we may also clearly observe that after leaving the goal, the control policies tend to drive again the bicycle to it. It should be noticed that the policy corresponding to $c_{reward} = 0.1$ manages at each loop to bring the bicycle back to the goal while it is not the case with the policy corresponding to $c_{reward} = 0.01$. Note that the coefficient γ influences also the trajectories obtained. For example, by taking $\gamma = 0.95$ instead of $\gamma = 0.98$, the bicycle crashes rapidly. This is due to the fact that a smaller value of γ tends to increase the importance of short-term rewards over long-term ones, which favors actions that turn rapidly the bicycle frame, even if they may eventually lead to a fall of the bicycle.

Rather than relying only on the score to assess the performances of a policy, let us now associate to a policy a value that depends on its ability to drive the bicycle to the goal within a certain time interval, when $(\omega, \dot{\omega}, \theta, \dot{\theta}, \psi_{goal})$ is the state signal considered. To do so, we simulate from each $x_0 \in X^i$ ten times the system over $50,000$ time steps and count the number of times the goal has been reached. Figure 25e represents the "number of successful trajectories" obtained by $\hat{\mu}_N^*$ for different values of N. Observe that 150 iterations of fitted Q iteration are needed before starting to observe some successful trajectories. Observe also that the "number of successful trajectories" sometimes drops when N increases, contrary to intuition. These drops are however not observed on the score values (e.g. for $N = 230$, all 90 trajectories are successful and the score is equal to 0.00156, while for $N = 240$, the number of successful trajectories drops to 62 but the score increases to 0.00179). Additional simulations have shown that these sudden drops tend to disappear when using more four-tuples.

Figure 25f illustrates the influence of the size of $\mathcal{F}$ on the number of successful trajectories when fitted Q iteration is combined with Extra-Trees. As expected, the number of successful trajectories tends to increase with the number of episodes considered in the four-tuples generation process. It should be noted that the other tree-based methods considered in this paper did not manage to produce successful trajectories when only 1000 episodes are used to generate the four-tuples. The different scores obtained by $\hat{\mu}_{300}^*$ when 1000 episodes are considered and for the different tree-based methods are gathered in Table 6, page 541. Using this score metric, Extra-Trees is the method performing the best, which is in agreement with the "number of successful trajectories" metric, followed successively by Tree Bagging, Totally Randomized Trees, Pruned CART Tree and Kd-Tree.

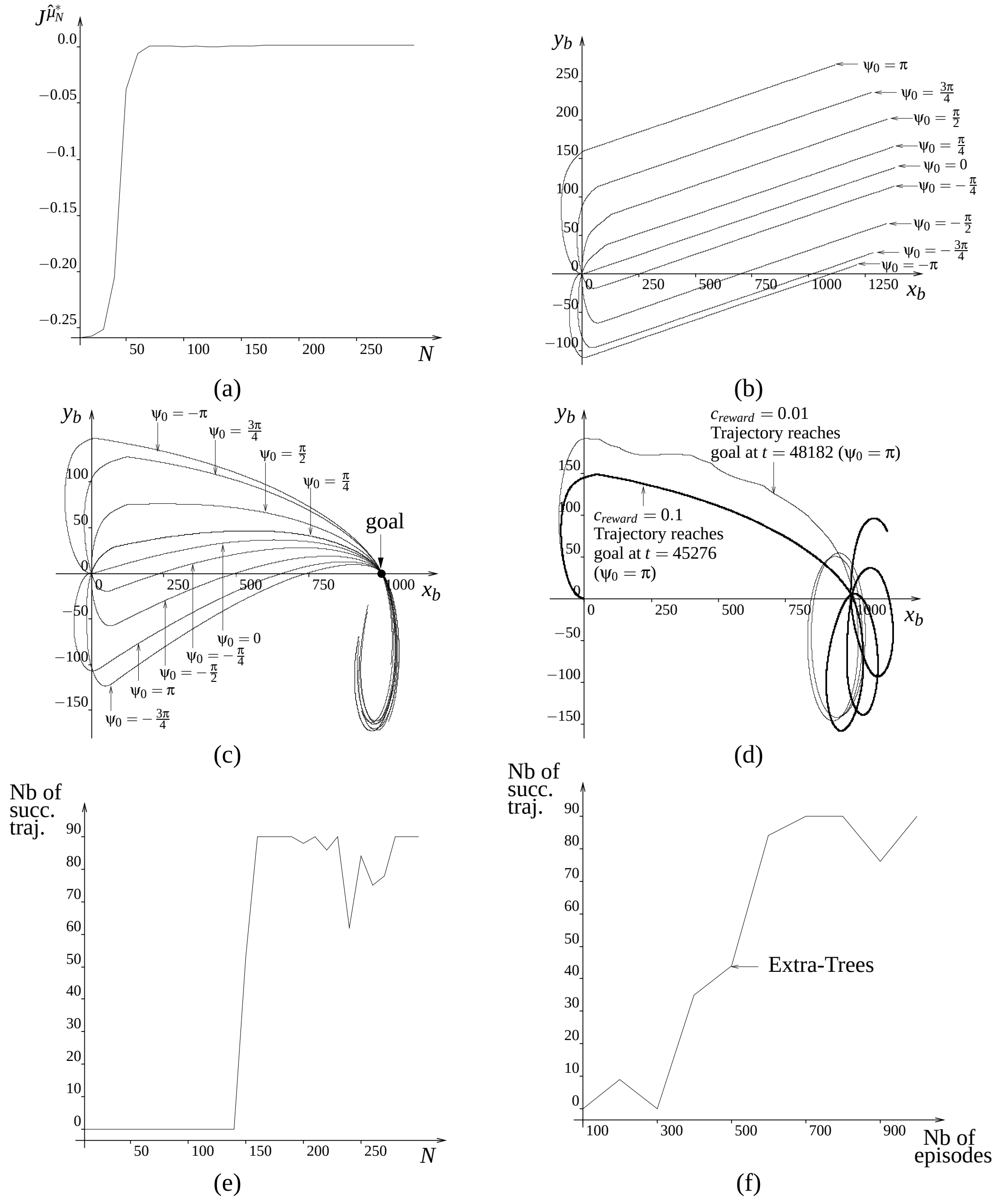


Figure 25: The "Bicycle Balancing and Riding" control problem. Figure (a) represents the score of $\hat{\mu}^*_N$ (Extra-Trees, 1000 episodes). Figure (b) sketches trajectories when $\hat{\mu}^*_{300}$ (Extra-Trees, 1000 episodes) controls the bicycle (trajectories drawn from $t = 0$ till $t = 50,000$). Figure (c) represents trajectories when $\hat{\mu}^*_{300}$ (Extra-Trees, 1000 episodes) controls the bicycle with $(\omega_t, \dot{\omega}_t, \theta_t, \dot{\theta}_t, \psi_{goal_t})$ used as input signal for the policy (trajectories drawn from $t = 0$ till $t = 50,000$). Figure (d) represents the influence on c_{reward} on the trajectories ($\hat{\mu}^*_{300}$, Extra-Trees, 1000 episodes and trajectories drawn from $t = 0$ till $t = 100,000$). Figure (e) lists the number of times the policy $\hat{\mu}^*_N$ manages to bring the bicycle to the goal in less than $50,000$ time steps (highest possible value for "Number of successful trajectories" is 90). Figure (f) gives for different number of episodes the number of times $\hat{\mu}^*_{300}$ leads to successful trajectories.

5.6 Conclusion of the Experiments

We discuss in this section the main conclusions that may be drawn from the simulation results previously reported.

5.6.1 INFLUENCE OF THE TREE-BASED METHODS

Let us analyze the results of our experiments in the light of the classification given in Table 1.

Single trees vs ensembles of trees. Whatever the set of four-tuples used, the top score has always been reached by a method building an ensemble of regression trees, and furthermore, the larger the state space, the better these regression tree ensemble methods behave compared to methods building only one single tree. These results are in agreement with previous work in reinforcement learning which suggests that multi-partitioning of the state space is leading to better function approximators than single partitioning.[25] They are also in agreement with the evaluation of these ensemble algorithms on many standard supervised learning problems (classification and regression), where tree-based ensemble methods typically significantly outperform single trees (Geurts et al., 2004).

However, from the viewpoint of computational requirements, we found that ensemble methods are clearly more demanding, both in terms of computing times and memory requirements for the storage of models.

Kernel-based vs non kernel-based methods. Among the single tree methods, Pruned CART Tree, which adapts the tree structure to the output variable, offers typically the same performances as Kd-Tree, except in the case of irrelevant variables where it is significantly more robust. Among the tree-based ensemble methods, Extra-Trees outperforms Totally Randomized Trees in all cases. On the other hand, Tree Bagging is generally better than the Totally Randomized Trees, except when dealing with very small numbers of samples, where the bootstrap resampling appears to be penalizing. These experiments thus show that tree-based methods that adapt their structure to the new output at each iteration usually provide better results than methods that do not (that we name kernel-based). Furthermore, the non kernel-based tree-based algorithms are much more robust to the presence of irrelevant variables thanks to their ability to filter out tests involving these variables.

A drawback of non kernel-based methods is that they do not guarantee convergence. However, with the Extra-Trees algorithm, even if the sequence was not converging, the policy quality was oscillating only moderately around a stable value and even when at its lowest, it was still superior to the one obtained by the kernel-based methods ensuring the convergence of the algorithm. Furthermore, if really required, convergence to a stable approximation may always be provided in an ad hoc fashion, for example by freezing the tree structures after a certain number of iterations and then only refreshing predictions at terminal nodes.

5.6.2 PARAMETRIC VERSUS NON-PARAMETRIC SUPERVISED LEARNING METHOD

Fitted Q iteration has been used in our experiments with non-parametric supervised learning methods (kNN, tree-based methods) and parametric supervised learning methods (basis function methods with piecewise-constant or piecewise-linear grids as approximation architectures).

25. See e.g. Sutton (1996); Sutton and Barto (1998), where the authors show that by overlaying several shifted tilings of the state space (type of approximation architecture known as CMACs), good function approximators could be obtained.

It has been shown that the parametric supervised learning methods, compared to the non-parametric ones, were not performing well. The main reason is the difficulty to select a priori the shape of the parametric approximation architecture that may lead to some good results. It should also be stressed that divergence to infinity of the fitted Q iteration has sometimes been observed when piecewise-linear grids were the approximation architectures considered.

5.6.3 FITTED Q ITERATION VERSUS ON-LINE ALGORITHMS

An advantage of fitted Q iteration over on-line algorithms is that it can be combined with some non-parametric function approximators, shown to be really efficient to generalize the information. We have also compared the performances of fitted Q iteration and Q-learning for some a priori given parametric approximation architectures. In this context, we found out that when the approximation architecture used was chosen so as to avoid serious convergence problems of the fitted Q iteration algorithm, then this latter was also performing much better than Q-learning on the same architecture.

6. Conclusions and Future Work

In this paper, we have considered a batch mode approach to reinforcement learning, which consists of reformulating the reinforcement learaning problem as a sequence of standard supervised learning problems. After introducing the *fitted Q iteration algorithm* which formalizes this framework, we have studied the properties and performances of the algorithm when combined with three classical tree-based methods (Kd-Trees, CART Trees, Tree Bagging) and two newly proposed tree-based ensemble methods namely Extra-Trees and Totally Randomized Trees.

Compared with grid-based methods on low-dimensional problems, as well as with kNN and single tree-based methods in higher dimensions, we found out that the fitted Q iteration algorithm was giving excellent results when combined with any one of the considered tree-based ensemble methods (Extra-Trees, Tree Bagging and Totally Randomized Trees). On the different cases studied, *Extra-Trees* was the supervised learning method able to extract at best information from a set of four-tuples. It is also faster than Tree Bagging and was performing significantly better than this latter algorithm, especially on the higher dimensional problems and on low-dimensional problems with small sample sizes. We also found out that fitted Q iteration combined with tree-based methods was performing much better than Q-learning combined with piecewise-constant or piecewise-linear grids.

Since Extra-Trees and Tree Bagging, the two best performing supervised learning algorithms, readjust their approximation architecture to the output variable at each iteration, they do not ensure the convergence of the fitted Q iteration algorithm. However, and contrary to many parametric approximation schemes, they do not lead to divergence to infinity problems. The convergence property is satisfied by the *Totally Randomized Trees* because their set of trees is frozen at the beginning of the iteration. They perform however less well than Extra-Trees and Tree Bagging, especially in the presence of irrelevant variables. They are nevertheless better than some other methods that also ensure the convergence of the sequence, like kNN kernel methods and piecewise-constant grids, in terms of performances as well as scalability to large numbers of variables and four-tuples. Within this context, it would be worth to study versions of Extra-Trees and Tree Bagging which would freeze their trees at some stage of the iteration process, and thus recover the convergence property.

From a theoretical point of view, it would certainly be very interesting to further study the consistency of the fitted Q iteration algorithm, in order to determine general conditions under which the algorithm converges to an optimal policy when the number of four-tuples collected grows to infinity. With this in mind, one could possibly seek inspiration in the work of Ormoneit and Sen (2002) and Ormoneit and Glynn (2002), who provide consistency conditions for kernel-based supervised learning methods within the context of fitted Q iteration, and also in some of the material published in the supervised learning literature (e.g. Lin and Jeon, 2002; Breiman, 2000). More specifically, further investigation in order to characterize ensembles of regression trees with respect to consistency is particularly wishful, because of their good practical performances.

In this paper, the score associated to a test node of a tree was the relative variance reduction. Several authors who adapted regression trees in other ways to reinforcement learning have suggested the use of other score criteria for example based on the violation of the Markov assumption (McCallum, 1996; Uther and Veloso, 1998) or on the combination of several error terms like the supervised, the Bellman, and the advantage error terms (Wang and Diettrich, 1999). Investigating the effect of such score measures within the fitted Q iteration framework is another interesting topic of research.

While the fitted Q iteration algorithm used with tree-based ensemble methods reveals itself to be very effective to extract relevant information from a set of four-tuples, it has nevertheless one drawback: with increasing number of four-tuples, it involves a superlinear increase in computing time and a linear increase in memory requirements. Although our algorithms offer a very good accuracy/efficiency tradeoff, we believe that further research should explore different ways to try to improve the computational efficiency and the memory usage, by introducing algorithm modifications specific to the reinforcement learning context.

Acknowledgments

Damien Ernst and Pierre Geurts gratefully acknowledge the financial support of the Belgian National Fund of Scientific Research (FNRS). The authors thank the action editor Michael Littman and the three anonymous reviewers for their many suggestions for improving the quality of the manuscript. They are also very grateful to Bruno Scherrer for reviewing an earlier version of this work, to Mania Pavella for her helpful comments and to Michail Lagoudakis for pointing out relevant information about the bicycle control problems.

Appendix A. Extra-Trees Induction Algorithm

The procedure used by the Extra-Trees algorithm to build a tree from a training set is described in Figure 26. This algorithm has two parameters: n_{min}, the minimum number of elements required to split a node and K, the maximum number of cut-directions evaluated at each node. If $K = 1$ then at each test node the cut-direction and the cut-point are chosen totally at random. If in addition the condition (iii) is dropped, then the tree structure is completely independent of the output values found in the $\mathcal{TS}$, and the algorithm generates *Totally Randomized Trees*.

The score measure used is the relative variance reduction. In other words, if $\mathcal{TS}_l$ (resp. $\mathcal{TS}_r$) denotes the subset of cases from $\mathcal{TS}$ such that $[i_j < t]$ (resp. $[i_j \geq t]$), then the Score is defined as follows:

$$\text{Score}([i_j < t], \mathcal{TS}) = \frac{var(o|\mathcal{TS}) - \frac{\#\mathcal{TS}_l}{\#\mathcal{TS}} var(o|\mathcal{TS}_l) - \frac{\#\mathcal{TS}_r}{\#\mathcal{TS}} var(o|\mathcal{TS}_r)}{var(o|\mathcal{TS})}, \tag{25}$$

```
**Build_a_tree**($\mathcal{TS}$)
Input: a training set $\mathcal{TS}$
Output: a tree $T$;

• If
      (i) $\#\mathcal{TS} < n_{min}$, or
      (ii) all input variables are constant in $\mathcal{TS}$, or
      (iii) the output variable is constant over the $\mathcal{TS}$,
  return a leaf labeled by the average value $\frac{1}{\#\mathcal{TS}} \sum_l o^l$.

• Otherwise:
      1. Let $[i_j < t_j] = \text{Find\_a\_test}(\mathcal{TS})$.
      2. Split $\mathcal{TS}$ into $\mathcal{TS}_l$ and $\mathcal{TS}_r$ according to the test $[i_j < t]$.
      3. Build $T_l = \text{Build\_a\_tree}(\mathcal{TS}_l)$ and $T_r = \text{Build\_a\_tree}(\mathcal{TS}_r)$ from these subsets;
      4. Create a node with the test $[i_j < t_j]$, attach $T_l$ and $T_r$ as left and right subtrees of this
         node and return the resulting tree.

**Find_a_test**($\mathcal{TS}$)
Input: a training set $\mathcal{TS}$
Output: a test $[i_j < t_j]$:

  1. Select $K$ inputs,$\{i_1, ..., i_K\}$, at random, without replacement, among all (non constant) input
     variables.

  2. For $k$ going from 1 to $K$:
      (a) Compute the maximal and minimal value of $i_k$ in $\mathcal{TS}$, denoted respectively $i^{\mathcal{TS}}_{k,min}$ and
          $i^{\mathcal{TS}}_{k,max}$.
      (b) Draw a discretization threshold $t_k$ uniformly in $]i^{\mathcal{TS}}_{k,min}, i^{\mathcal{TS}}_{k,max}]$
      (c) Compute the score $S_k = \text{Score}([i_k < t_k], \mathcal{TS})$

  3. Return a test $[i_j < t_j]$ such that $S_j = \max_{k=1,...,K} S_k$.
```

Figure 26: Procedure used by the Extra-Trees algorithm to build a tree. The *Totally Randomized Trees* algorithm is obtained from this algorithm by setting $K = 1$ and by dropping the stopping condition (iii).

where $var(o|\mathcal{X})$ is the variance of the output o in the training set $\mathcal{X}$.

Appendix B. Convergence of the Sequence of $\hat{Q}_N$-Functions

Theorem 1 *If the fitted Q iteration algorithm is combined with a supervised learning method which produces a model of the type (17) with the kernel $k_{\mathcal{TS}}$ being the same from one iteration to the other and satisfying the normalizing condition (18), then the sequence of $\hat{Q}_N$-functions converges.*

Proof The proof is adapted in a straightforward way from Ormoneit and Sen (2002) to the fact that the kernel $k_{\mathcal{TS}}((x_t^l,u_t^l),(x,u))$ may not be decomposed here into the product $k'(x_t^l,x)\delta(u_t^l,u)$.

Let us first observe that in such conditions, the sequence of functions computed by the fitted Q iteration algorithm is determined by the recursive equation:

$$\hat{Q}_N(x,u) = \sum_{l=1}^{\#\mathcal{F}} k_{\mathcal{TS}}((x_t^l,u_t^l),(x,u))[r_t^l + \gamma \max_{u'\in U} \hat{Q}_{N-1}(x_{t+1}^l,u')], \quad \forall N > 0 \tag{26}$$

with $\hat{Q}_0(x,u) = 0\ \forall (x,u) \in X \times U$. Equation (26) may be rewritten:

$$\hat{Q}_N = \hat{H}\hat{Q}_{N-1} \tag{27}$$

where $\hat{H}$ is an operator mapping any function $K : X \times U \to \mathbb{R}$ and defined as follows:

$$(\hat{H}K)(x,u) = \sum_{l=1}^{\#\mathcal{F}} k_{\mathcal{TS}}((x_t^l,u_t^l),(x,u))[r_t^l + \gamma \max_{u'\in U} K(x_{t+1}^l,u')]. \tag{28}$$

This operator is a contraction on the Banach space of functions defined over $X \times U$ and the supremum norm. Indeed, we have:

$$\begin{aligned}
\|\hat{H}K - \hat{H}\overline{K}\|_\infty &= \gamma \max_{(x,u)\in X\times U} \Big|\sum_{l=1}^{\#\mathcal{F}} k_{\mathcal{TS}}((x_t^l,u_t^l),(x,u))[\max_{u'\in U} K(x_{t+1}^l,u') - \max_{u'\in U}\overline{K}(x_{t+1}^l,u')]\Big| \\
&\leq \gamma \max_{(x,u)\in X\times U} \Big|[\sum_{l=1}^{\#\mathcal{F}} k_{\mathcal{TS}}((x_t^l,u_t^l),(x,u)) \max_{u'\in U}[K(x_{t+1}^l,u') - \overline{K}(x_{t+1}^l,u')]\Big| \\
&\leq \gamma \max_{(x,u)\in X\times U} |K(x,u) - \overline{K}(x,u)| \\
&= \gamma \|K - \overline{K}\|_\infty \\
&< \|K - \overline{K}\|_\infty.
\end{aligned}$$

By virtue of the fixed-point theorem (Luenberger, 1969) the sequence converges, independently of the initial conditions, to the function $\hat{Q} : X \times U \to \mathbb{R}$ which is unique solution of the equation $\hat{Q} = \hat{H}\hat{Q}$. ■

Appendix C. Definition of the Benchmark Optimal Control Problems

We define in this section the different optimal control problems used in our experiments. Simulators, additional documentation and sets of four-tuples are available upon request.

C.1 The “Left or Right” Control Problem

System dynamics:

$$x_{t+1} = x_t + u_t + w_t$$

where w is drawn according the standard (zero mean, unit variance) Gaussian distribution.
If x_{t+1} is such that $|x_{t+1}| > 10$ or $|x_{t+1}| < 0$ then a terminal state is reached.
State space: The state space X is composed of $\{x \in \mathbb{R} | x \in [0,10]\}$ and of a terminal state.
Action space: The action space $U = \{-2, 2\}$.
Reward function: The reward function $r(x,u,w)$ is defined through the following expression:

$$r(x_t, u_t, w_t) = \begin{cases} 0 & \text{if} \quad x_{t+1} \in [0,10] \\ 50 & \text{if} \quad x_{t+1} < 0 \\ 100 & \text{if} \quad x_{t+1} > 10. \end{cases} \tag{29}$$

Decay factor: The decay factor γ is equal to 0.75.

C.2 The “Car on the Hill” Control Problem

System dynamics: The system has a continuous-time dynamics described by these two differential equations:

$$\dot{p} = s \tag{30}$$

$$\dot{s} = \frac{u}{m(1+Hill'(p)^2)} - \frac{gHill'(p)}{1+Hill'(p)^2} - \frac{s^2 Hill'(p)Hill''(p)}{1+Hill'(p)^2} \tag{31}$$

where m and g are parameters equal respectively to 1 and 9.81 and where $Hill(p)$ is a function of p defined by the following expression:

$$Hill(p) = \begin{cases} p^2 + p & \text{if} \quad p < 0 \\ \frac{p}{\sqrt{1+5p^2}} & \text{if} \quad p \geq 0. \end{cases} \tag{32}$$

The discrete-time dynamics is obtained by discretizing the time with the time between t and $t+1$ chosen equal to $0.100\,s$.
If p_{t+1} and s_{t+1} are such that $|p_{t+1}| > 1$ or $|s_{t+1}| > 3$ then a terminal state is reached.
State space: The state space X is composed of $\{(p,s) \in \mathbb{R}^2 | \, |p| \leq 1 \,\text{and}\, |s| \leq 3\}$ and of a terminal state. $X \setminus \{\textit{terminal state}\}$ is represented on Figure 8a.
Action space: The action space $U = \{-4, 4\}$.
Reward function: The reward function $r(x,u)$ is defined through the following expression:

$$r(x_t, u_t) = \begin{cases} -1 & \text{if} \quad p_{t+1} < -1 \quad \text{or} \quad |s_{t+1}| > 3 \\ 1 & \text{if} \quad p_{t+1} > 1 \quad \text{and} \quad |s_{t+1}| \leq 3 \\ 0 & \text{otherwise.} \end{cases} \tag{33}$$

Decay factor: The decay factor γ has been chosen equal to 0.95.
Integration: The dynamical system is integrated by using an Euler method with a $0.001\,s$ integration time step.

Remark: This "Car on the Hill" problem is similar to the one found in Moore and Atkeson (1995) except that the term $-\frac{s^2 Hill'(p)Hill''(p)}{1+Hill'(p)^2}$ is not neglected here in the system dynamics.
Variants of the control problem: In our experiments, we have also considered two other variants of this problem:

- *The "Car on the Hill" with irrelevant variables:* some irrelevant variables are added to the state vector. The value of an irrelevant variable at time t is determined by drawing at random a number in $[-2,2]$ with a uniform probability (used in Section 5.3.5).

- *The "Car on the Hill" with continuous action space:* the action space is not yet discrete anymore. It is continuous and equal to $[-4,4]$ (used in Section 5.3.7).

C.3 The "Acrobot Swing Up" Control Problem

System dynamics: The system has a continuous-time dynamics described by these two second-order differential equations (taken from Yoshimoto et al., 1999):

$$\begin{aligned} d_{11}\ddot{\theta}_1 + d_{12}\ddot{\theta}_2 + c_1 + \phi_1 &= -\mu_1\dot{\theta}_1 && (34)\\ d_{12}\ddot{\theta}_1 + d_{22}\ddot{\theta}_2 + c_2 + \phi_2 &= u - \mu_2\dot{\theta}_2 && (35)\\ & && (36)\end{aligned}$$

where

$$\begin{aligned} d_{11} &= M_1L_1^2 + M_2(L_1^2 + L_2^2 + 2L_1L_2\cos(\theta_2)) && (37)\\ d_{22} &= M_2L_2^2 && (38)\\ d_{12} &= M_2(L_2^2 + L_1L_2\cos(\theta_2)) && (39)\\ c_1 &= -M_2L_1L_2\dot{\theta}_2(2\dot{\theta}_1 + \dot{\theta}_2\sin(\theta_2)) && (40)\\ c_2 &= M_2L_1L_2\dot{\theta}_1^2\sin(\theta_2) && (41)\\ \phi_1 &= (M_1L_1 + M_2L_1)g\sin(\theta_1) + M_2L_2g\sin(\theta_1+\theta_2) && (42)\\ \phi_2 &= M_2L_2g\sin(\theta_1+\theta_2). && (43)\end{aligned}$$

M_1 (M_2), L_1 (L_2) and μ_1 (μ_2) are the mass, length, and friction, respectively, of the first (second) link. θ_1 is the angle of the first link from a downward position and θ_2 is the angle of the second link from the direction of the first link (Figure 20). $\dot{\theta}_1$ and $\dot{\theta}_2$ are the angular velocities of the first and second links, respectively. The system has four continuous state variables $x = (\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2)$. The physical parameters have been chosen equal to $M_1 = M_2 = 1.0$, $L_1 = L_2 = 1.0$, $\mu_1 = \mu_2 = 0.01$, $g = 9.81$.
The discrete-time dynamics is obtained by discretizing the time with the time between t and $t+1$ chosen equal to $0.100\,s$.
Let us denote by O the set composed of the states $x = ((2*k+1)*\pi, 0, 0, 0)$ $k \in \mathbb{Z}$ and by $d(x,O)$ the value $\min\limits_{o\in O}\|x-o\|$.
If x_{t+1} is such that $d(x_{t+1},O) < 1$ then a terminal state is reached.
State space: The state space is composed of $\{x \in \mathbb{R}^4 | d(x,O) \geq 1\}$ and of a terminal state.
Action space: The action space $U = \{-5, 5\}$.

Reward function: The reward function $r(x,u)$ is defined through the following expression:

$$r(x_t,u_t) = \begin{cases} 0 & \text{if} \quad d(x_{t+1},O) \geq 1 \\ 1-d(x_{t+1},O) & \text{if} \quad d(x_{t+1},O) < 1. \end{cases} \tag{44}$$

Decay factor: The decay factor γ has been chosen equal to 0.95.
Integration: The dynamical system is integrated by using an Euler method with a $0.001s$ integration time step.

C.4 The "Bicycle Balancing" and "Bicycle Balancing and Riding" Control Problems

We define hereafter the "Bicycle Balancing" and the "Bicycle Balancing and Riding" control problems. These optimal control problems differ only by their reward functions.
System dynamics: The system studied has the following dynamics:

$$\begin{aligned}
\omega_{t+1} &= \omega_t + \Delta t \dot{\omega}_t & (45)\\
\dot{\omega}_{t+1} &= \dot{\omega}_t + \Delta t \Big(\frac{1}{I_{bicycle\ and\ cyclist}}(Mhg\sin(\varphi_t) - \cos(\varphi_t) & (46)\\
&\quad (I_{dc}\dot{\sigma}\dot{\theta}_t + \text{sign}(\theta_t)v^2(M_d r(invr_{f_t} + invr_{b_t}) + Mh\, invr_{CM_t})))\Big) & \\
\theta_{t+1} &= \begin{cases} \theta_t + \Delta t\dot{\theta}_t & \text{if} \quad |\theta_t + \Delta t\dot{\theta}_t| \leq \frac{80}{180}\pi \\ \text{sign}(\theta_t + \Delta t\dot{\theta}_t)\frac{80}{180}\pi & \text{if} \quad |\theta_t + \Delta t\dot{\theta}_t| > \frac{80}{180}\pi \end{cases} & (47)\\
\dot{\theta}_{t+1} &= \begin{cases} \dot{\theta}_t + \Delta t \frac{T - I_{dv}\dot{\sigma}\dot{\omega}_t}{I_{dl}} & \text{if} \quad |\theta_t + \Delta t\dot{\theta}_t| \leq \frac{80}{180}\pi \\ 0 & \text{if} \quad |\theta_t + \Delta t\dot{\theta}_t| > \frac{80}{180}\pi \end{cases} & (48)\\
x_{b_{t+1}} &= x_{b_t} + \Delta t\, v\cos(\psi_t) & (49)\\
y_{b_{t+1}} &= y_{b_t} + \Delta t\, v\sin(\psi_t) & (50)\\
\psi_{t+1} &= \psi_t + \Delta t\, \text{sign}(\theta_t) v\, invr_{b_t} & (51)
\end{aligned}$$

with

$$\begin{aligned}
\varphi_t &= \omega_t + \frac{\arctan(d_t + w_t)}{h} & (52)\\
invr_{f_t} &= \frac{|\sin(\theta_t)|}{l} & (53)\\
invr_{b_t} &= \frac{|\tan(\theta_t)|}{l} & (54)\\
invr_{CM_t} &= \begin{cases} \frac{1}{\sqrt{((l-c)^2 + (\frac{1}{invr_{b_t}})^2)}} & \text{if} \quad \theta_t \neq 0 \\ 0 & \text{otherwise} \end{cases} & (55)
\end{aligned}$$

where w_t is drawn according to a uniform distribution in the interval $[-0.02, 0.02]$. The different parameters are equal to the following values: $\Delta t = 0.01$, $v = \frac{10}{3.6}$, $g = 9.82$, $d_{CM} = 0.3$, $c = 0.66$, $h = 0.94$, $M_c = 15$, $M_d = 1.7$, $M_p = 60.0$, $M = (M_c + M_p)$, $r = 0.34$, $\dot{\sigma} = \frac{v}{r}$, $I_{bicycle\ and\ cyclist} = (\frac{13}{3}M_c h^2 + M_p(h + d_{CM})^2)$, $I_{dc} = (M_d r^2)$, $I_{dv} = (\frac{3}{2}M_d r^2)$, $I_{dl} = (\frac{1}{2}M_d r^2)$ and $l = 1.11$. This dynamics holds valid if $|\omega_{t+1}| \leq \frac{12}{180}\pi$. When $|\omega_{t+1}| > \frac{12}{180}\pi$, the bicycle is supposed to have fallen down and a *terminal state* is reached.
State space: The state space for this control problem is $\{(\omega, \dot{\omega}, \theta, \dot{\theta}, x_b, y_b, \psi) \in \mathbb{R}^7 | \theta \in [-\frac{80}{180}\pi, \frac{80}{180}\pi]$ and $\omega \in$

$[-\frac{12}{180}\pi, \frac{12}{180}\pi]\}$ plus a *terminal state*.
Action space: The action space $U = \{(d,T) \in \{-0.02, 0, 0.02\} \times \{-2, 0, 2\}\}$. U is composed of 9 elements.
Reward functions: The reward function for the "Bicycle Balancing" control problem is defined hereafter:

$$r(x_t, u_t, w_t) \quad = \quad \begin{cases} -1 & \text{if} \quad |\omega_{t+1}| > \frac{12}{180}\pi \\ 0 & \text{otherwise.} \end{cases} \tag{56}$$

The reward function for "Bicycle Balancing and Riding" control problem is:

$$r(x_t, u_t, w_t) \quad = \quad \begin{cases} -1 & \text{if} \quad |\omega_{t+1}| > \frac{12}{180}\pi \\ c_{reward}(d_{angle}(\psi_t) - d_{angle}(\psi_{t+1})) & \text{otherwise} \end{cases} \tag{57}$$

where $c_{reward} = 0.1$ and $d_{angle} : \mathbb{R} \to \mathbb{R}$ such that $d_{angle}(\psi) = \min_{k \in \mathbb{Z}} |\psi + 2k\pi|$.
Decay factor: The decay factor γ is equal to 0.98.
Remark: The bicycle dynamics is based on the one found in Randløv and Alstrøm (1998) and in their corresponding simulator available at *http://www.nbi.dk/~randlov/bike.html*.

References

D. Bagnell, S. Kakade, A. Y. Ng, and J. Schneider. Policy search by dynamic programming. In *Proceedings of Neural Information Processing Systems*, 2003.

L. C. Baird. Residual algorithms: reinforcement learning with function approximation. In Armand Prieditis and Stuart Russell, editors, *Machine Learning: Proceedings of the Twelfth International Conference*, pages 9–12, San Francisco, CA, July 1995. Morgan Kaufman.

R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

R. Bellman, R. Kalaba, and B. Kotkin. Polynomial approximation - a new computational technique in dynamic programming: allocation processes. *Mathematical Computation*, 17:155–161, 1973.

J. A. Boyan. Technical update: least-squares temporal difference learning. *Machine Learning*, 49 (2-3):233–246, 2002.

J. A. Boyan and A. W. Moore. Generalization in reinforcement learning: safely approximating the value function. *Advances in Neural Information Processing Systems*, 7:369–376, 1995.

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

L. Breiman. Some infinity theory for predictor ensembles. Technical Report 577, University of California, Department of Statistics, 2000.

L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

L. Breiman, J. H. Friedman, R. A. Olsen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth International (California), 1984.

D. Ernst. *Near Optimal Closed-Loop Control. Application to Electric Power Systems*. PhD thesis, University of Liège, Belgium, March 2003.

D. Ernst, P. Geurts, and L. Wehenkel. Iteratively extending time horizon reinforcement learning. In N. Lavra, L. Gamberger, and L. Todorovski, editors, *Proceedings of the 14th European Conference on Machine Learning*, pages 96–107, Dubrovnik, Croatia, September 2003. Springer-Verlag Heidelberg.

D. Ernst, M. Glavic, P. Geurts, and L. Wehenkel. Approximate value iteration in the reinforcement learning context. Application to electrical power system control. *To appear in Intelligent Automation and Soft Computing*, 2005.

P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. Submitted, 2004.

G. J. Gordon. Online fitted reinforcement learning. In *VFA workshop at ML-95*, 1995a.

G. J. Gordon. Stable function approximation in dynamic programming. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 261–268, San Francisco, CA, 1995b. Morgan Kaufmann.

G. J. Gordon. *Approximate Solutions to Markov Decision Processes*. PhD thesis, Carnegie Mellon University, June 1999.

O. Hernández-Lerma and B. Lasserre. *Discrete-Time Markov Control Processes*. Springer, New-York, 1996.

L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

S. Kakade and J. Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 267–274, 2002.

M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003a.

M. G. Lagoudakis and R. Parr. Reinforcement learning as classification: leveraging modern classifiers. In *Proceedings of ICML 2003*, pages 424–431, 2003b.

J. Langford and B. Zadrozny. Reducing T-step reinforcement learning to classification. Submitted, 2004.

L. J. Lin. *Reinforcement Learning for Robots Using Neural Networks*. PhD thesis, Carnegie Mellon University, Pittsburgh, USA, 1993.

Y. Lin and Y. Jeon. Random forests and adaptive nearest neighbors. Technical Report 1005, Department of Statistics, University of Wisconsin, 2002.

D. G. Luenberger. *Optimization by Vector Space Methods*. Wiley, N.Y., 1969.

A. K. McCallum. *Reinforcement Learning with Selective Perception and Hidden State*. PhD thesis, University of Rochester, Rochester, New-York, 1996.

A. W. Moore and C. G. Atkeson. Prioritized sweeping: reinforcement learning with less data and less real time. *Machine Learning*, 13:103–130, 1993.

A. W. Moore and C. G. Atkeson. The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces. *Machine Learning*, 21(3):199–233, 1995.

A. Y. Ng and M. Jordan. PEGASUS: a policy search method for large MDPs and POMDPs. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 406–415, 1999.

D. Ormoneit and P. Glynn. Kernel-based reinforcement learning in average-cost problems. *IEEE Transactions on Automatic Control*, 47(10):1624–1636, 2002.

D. Ormoneit and S. Sen. Kernel-based reinforcement learning. *Machine Learning*, 49(2-3):161–178, 2002.

J. Randløv and P. Alstrøm. Learning to drive a bicycle using reinforcement learning and shaping. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 463–471, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

J. Rust. Using randomization to break the curse of dimensionality. *Econometrica*, 65(3):487–516, 1997.

S. P. Singh, T. Jaakkola, and M. I. Jordan. Reinforcement learning with soft state aggregation. In G. Tesauro, D. S. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems : Proceedings of the 1994 Conference*, pages 359–368, Cambridge, MA, 1995. MIT press.

W. D. Smart and L. P. Kaelbling. Practical reinforcement learning in continuous spaces. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 903–910, 2000.

M. W. Spong. Swing up control of the Acrobot. In *1994 IEEE International Conference on Robotics and Automation*, pages 2356–2361, San Diego, CA, May 1994.

R. S. Sutton. Learning to predict by the method of temporal differences. *Machine Learning*, 3(1): 9–44, 1988.

R. S. Sutton. Generalization in reinforcement learning: successful examples using sparse coarse coding. *Advances in Neural Information Processing Systems*, 8:1038–1044, 1996.

R. S. Sutton and A. G. Barto. *Reinforcement Learning, an Introduction*. MIT Press, 1998.

J. N. Tsitsiklis. Asynchronous stochastic approximation and Q-learning. *Machine Learning*, 16(3): 185–202, 1994.

J. N. Tsitsiklis and B. Van Roy. Feature-based methods for large-scale dynamic programming. *Machine Learning*, 22:59–94, 1996.

W. T. B. Uther and M. M. Veloso. Tree based discretization for continuous state space reinforcement learning. In *Proceedings of AAAI-98*, pages 769–774, 1998.

X. Wang and T. G. Diettrich. Efficient value function approximation using regression trees. In *Proceedings of IJCAI-99 Workshop on Statistical Machine Learning for Large-Scale Optimization*, Stockholm, Sweden, 1999.

C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, Cambridge University, Cambridge, England, 1989.

J. Yoshimoto, S. Ishii, and M. Sato. Application of reinforcement learning to balancing Acrobot. In *Proceedings of the 1999 IEEE International Conference on Systems, Man and Cybernetics*, pages 516–521, 1999.

Journal of Machine Learning Research 6 (2005) 557–588 Submitted 3/04; Revised 1/05; Published 4/05

Learning Module Networks

Eran Segal ERAN@CS.STANFORD.EDU
Computer Science Department
Stanford University
Stanford, CA 94305-9010, USA

Dana Pe'er DPEER@GENETICS.MED.HARVARD.EDU
Genetics Department
Harvard Medical School
Boston, MA 02115, USA

Aviv Regev AREGEV@CGR.HARVARD.EDU
Bauer Center for Genomic Research
Harvard University
Cambridge, MA 02138, USA

Daphne Koller KOLLER@CS.STANFORD.EDU
Computer Science Department
Stanford University
Stanford, CA 94305-9010, USA

Nir Friedman NIR@CS.HUJI.AC.IL
Computer Science & Engineering
Hebrew University
Jerusalem, 91904, Israel

Editor: Tommi Jaakkola

Abstract

Methods for learning Bayesian networks can discover dependency structure between observed variables. Although these methods are useful in many applications, they run into computational and statistical problems in domains that involve a large number of variables. In this paper,[1] we consider a solution that is applicable when many variables have similar behavior. We introduce a new class of models, *module networks*, that explicitly partition the variables into modules, so that the variables in each module share the same parents in the network and the same conditional probability distribution. We define the semantics of module networks, and describe an algorithm that learns the modules' composition and their dependency structure from data. Evaluation on real data in the domains of gene expression and the stock market shows that module networks generalize better than Bayesian networks, and that the learned module network structure reveals regularities that are obscured in learned Bayesian networks.

1. A preliminary version of this paper appeared in the Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence, 2003 (UAI '03).

1. Introduction

Over the last decade, there has been much research on the problem of learning Bayesian networks from data (Heckerman, 1998), and successfully applying it both to density estimation, and to discovering dependency structures among variables. Many real-world domains, however, are very complex, involving thousands of relevant variables. Examples include modeling the dependencies among expression levels (a rough indicator of activity) of all the genes in a cell (Friedman *et al.*, 2000a; Lander, 1999) or among changes in stock prices. Unfortunately, in complex domains, the amount of data is rarely enough to robustly learn a model of the underlying distribution. In the gene expression domain, a typical data set describes thousands of variables, but at most a few hundred instances. In such situations, statistical noise is likely to lead to spurious dependencies, resulting in models that significantly overfit the data.

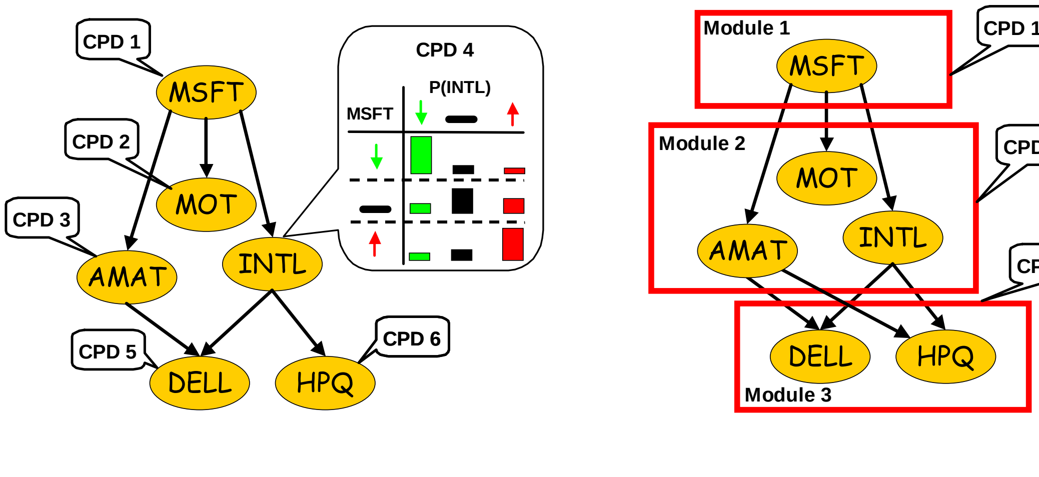
Moreover, if our goal is structure discovery, such domains pose additional challenges. First, due to the small number of instances, we are unlikely to have much confidence in the learned structure (Pe'er *et al.*, 2001). Second, a Bayesian network structure over thousands of variables is typically highly unstructured, and therefore very hard to interpret.

In this paper, we propose an approach to address these issues. We start by observing that, in many large domains, the variables can be partitioned into sets so that, to a first approximation, the variables within each set have a similar set of dependencies and therefore exhibit a similar behavior. For example, many genes in a cell are organized into *modules*, in which sets of genes required for the same biological function or response are co-regulated by the same inputs in order to coordinate their joint activity. As another example, when reasoning about thousands of NASDAQ stocks, entire sectors of stocks often respond together to sector-influencing factors (e.g., oil stocks tend to respond similarly to a war in Iraq).

We define a new representation called a *module network*, which explicitly partitions the variables into *modules*. Each module represents a set of variables that have the same statistical behavior, i.e., they share the same set of parents and local probabilistic model. By enforcing this constraint on the learned network, we significantly reduce the complexity of our model space as well as the number of parameters. These reductions lead to more robust estimation and better generalization on unseen data. Moreover, even if a modular structure exists in the domain, it can be obscured by a general Bayesian network learning algorithm which does not have an explicit representation for modules. By making the modular structure explicit, the module network representation provides insight about the domain that are often be obscured by the intricate details of a large Bayesian network structure.

A module network can be viewed simply as a Bayesian network in which variables in the same module share parents and parameters. Indeed, probabilistic models with shared parameters are common in a variety of applications, and are also used in other general representation languages, such as *dynamic Bayesian networks* (Dean and Kanazawa, 1989), *object-oriented Bayesian Networks* (Koller and Pfeffer, 1997), and *probabilistic relational models* (Koller and Pfeffer, 1998; Friedman *et al.*, 1999a). (See Section 8 for further discussion of the relationship between module networks and these formalisms.) In most cases, the shared structure is imposed by the designer of the model, using prior knowledge about the domain. A key contribution of this paper is the design of a learning algorithm that directly searches for and finds sets of variables with similar behavior, which are then defined to be a module.

We describe the basic semantics of the module network framework, present a Bayesian scoring function for module networks, and provide an algorithm that learns both the assignment of variables



(a) Bayesian network (b) Module network

Figure 1: (a) A simple Bayesian network over stock price variables; the stock price of Intel (*INTL*) is annotated with a visualization of its CPD, described as a different multinomial distribution for each value of its influencing stock price Microsoft (*MSFT*). (b) A simple module network; the boxes illustrate modules, where stock price variables share CPDs and parameters. Note that in a module network, variables in the same module have the same CPDs but may have different descendants.

to modules and the probabilistic model for each module. We evaluate the performance of our algorithm on two real data sets, in the domains of gene expression and the stock market. Our results show that our learned module network generalizes to unseen test data much better than a Bayesian network. They also illustrate the ability of the learned module network to reveal high-level structure that provides important insights.

2. The Module Network Framework

We start with an example that introduces the main idea of module networks and then provide a formal definition. For concreteness, consider a simple toy example of modeling changes in stock prices. The Bayesian network of Figure 1(a) describes dependencies between different stocks. In this network, each random variable corresponds to the change in price of a single stock. For illustration purposes, we assume that these random variables take one of three values: ‘down’, ‘same’ or ‘up’, denoting the change during a particular trading day. In our example, the stock price of Intel (*INTL*) depends on that of Microsoft (*MSFT*). The *conditional probability distribution (CPD)* shown in the figure indicates that the behavior of Intel’s stock is similar to that of Microsoft. That is, if Microsoft’s stock goes up, there is a high probability that Intel’s stock will also go up and vice versa. Overall, the Bayesian network specifies a CPD for each stock price as a stochastic function of its parents. Thus, in our example, the network specifies a separate behavior for each stock.

The stock domain, however, has higher order structural features that are not explicitly modeled by the Bayesian network. For instance, we can see that the stock price of Microsoft (*MSFT*) influences the stock price of all of the major chip manufacturers — Intel (*INTL*), Applied Materials (*AMAT*), and Motorola (*MOT*). In turn, the stock price of computer manufacturers Dell (*DELL*) and Hewlett Packard (*HPQ*), are influenced by the stock prices of their chip suppliers — Intel and Applied Materials. An examination of the CPDs might also reveal that, to a first approximation, the stock price of all chip making companies depends on that of Microsoft and in much the same way. Similarly, the stock price of computer manufacturers that buy their chips from Intel and Applied Materials depends on these chip manufacturers' stock and in much the same way.

To model this type of situation, we might divide stock price variables into groups, which we call *modules*, and require that variables in the same module have the same probabilistic model; that is, all variables in the module have the same set of parents and the same CPD. Our example contains three modules: one containing only Microsoft, a second containing chip manufacturers Intel, Applied Materials, and Motorola, and a third containing computer manufacturers Dell and HP (see Figure 1(b)). In this model, we need only specify three CPDs, one for each module, since all the variables in each module share the same CPD. By comparison, six different CPDs are required for a Bayesian network representation. This notion of a module is the key idea underlying the module network formalism.

We now provide a formal definition of a module network. Throughout this paper, we assume that we are given a domain of random variables $\mathcal{X} = \{X_1, \ldots, X_n\}$. We use $Val(X_i)$ to denote the domain of values of the variable X_i.

As described above, a module represents a set of variables that share the same set of parents and the same CPD. As a notation, we represent each module by a *formal variable* that we use as a placeholder for the variables in the module. A *module set* $\mathcal{C}$ is a set of such formal variables $\mathbf{M}_1, \ldots, \mathbf{M}_K$. As all the variables in a module share the same CPD, they must have the same domain of values. We represent by $Val(\mathbf{M}_j)$ the set of possible values of the formal variable of the j'th module.

A module network relative to $\mathcal{C}$ consists of two components. The first defines a template probabilistic model for each module in $\mathcal{C}$; all of the variables assigned to the module will share this probabilistic model.

Definition 1 *A* module network template $\mathcal{T} = (\mathcal{S}, \theta)$ *for* $\mathcal{C}$ *defines, for each module* $\mathbf{M}_j \in \mathcal{C}$*:*

- *a set of parents* $\mathbf{Pa}_{\mathbf{M}_j} \subset \mathcal{X}$*;*
- *a* conditional probability distribution template $P(\mathbf{M}_j \mid \mathbf{Pa}_{\mathbf{M}_j})$ *which specifies a distribution over* $Val(\mathbf{M}_j)$ *for each assignment in* $Val(\mathbf{Pa}_{\mathbf{M}_j})$*.*

We use $\mathcal{S}$ *to denote the dependency structure encoded by* $\{\mathbf{Pa}_{\mathbf{M}_j} : \mathbf{M}_j \in \mathcal{C}\}$ *and* θ *to denote the parameters required for the CPD templates* $\{P(\mathbf{M}_j \mid \mathbf{Pa}_{\mathbf{M}_j}) : \mathbf{M}_j \in \mathcal{C}\}$. ▮

In our example, we have three modules M_1, M_2, and M_3, with $\mathbf{Pa}_{\mathbf{M}_1} = \emptyset$, $\mathbf{Pa}_{\mathbf{M}_2} = \{MSFT\}$, and $\mathbf{Pa}_{\mathbf{M}_3} = \{AMAT, INTL\}$.

The second component is a module assignment function that assigns each variable $X_i \in \mathcal{X}$ to one of the K modules, $\mathbf{M}_1, \ldots, \mathbf{M}_K$. Clearly, we can only assign a variable to a module that has the same domain.

Definition 2 *A* module assignment function *for* $\mathcal{C}$ *is a function* $\mathcal{A} : \mathcal{X} \to \{1, \ldots, K\}$ *such that* $\mathcal{A}(X_i) = j$ *only if* $Val(X_i) = Val(\mathbf{M}_j)$. ▮

In our example, we have that $\mathcal{A}(MSFT) = 1$, $\mathcal{A}(MOT) = 2$, $\mathcal{A}(INTL) = 2$, and so on.

A module network defines a probabilistic model by using the formal random variables $\mathbf{M}_j$ and their associated CPDs as templates that encode the behavior of all of the variables assigned to that module. Specifically, we define the semantics of a module network by "unrolling" a Bayesian network where all of the variables assigned to module $\mathbf{M}_j$ share the parents and conditional probability template assigned to $\mathbf{M}_j$ in $\mathcal{T}$. For this unrolling process to produce a well-defined distribution, the resulting network must be acyclic. Acyclicity can be guaranteed by the following simple condition on the module network:

Definition 3 *Let $\mathcal{M}$ be a triple $(\mathcal{C}, \mathcal{T}, \mathcal{A})$, where $\mathcal{C}$ is a module set, $\mathcal{T}$ is a module network template for $\mathcal{C}$, and $\mathcal{A}$ is a module assignment function for $\mathcal{C}$. $\mathcal{M}$ defines a directed* module graph *$\mathcal{G}_{\mathcal{M}}$ as follows:*

- *the nodes in $\mathcal{G}_{\mathcal{M}}$ correspond to the modules in $\mathcal{C}$;*
- *$\mathcal{G}_{\mathcal{M}}$ contains an edge $\mathbf{M}_j \to \mathbf{M}_k$ if and only if there is a variable $X \in \mathcal{X}$ so that $\mathcal{A}(X) = j$ and $X \in \mathbf{Pa}_{\mathbf{M}_k}$.*

We say that $\mathcal{M}$ is a module network *if the module graph $\mathcal{G}_{\mathcal{M}}$ is acyclic.* ∎

For example, for the module network of Figure 1(b), the module graph has the structure $\mathbf{M}_1 \to \mathbf{M}_2 \to \mathbf{M}_3$.

We can now define the semantics of a module network:

Definition 4 *A module network $\mathcal{M} = (\mathcal{C}, \mathcal{T}, \mathcal{A})$ defines a* ground Bayesian network *$\mathcal{B}_{\mathcal{M}}$ over $\mathcal{X}$ as follows: For each variable $X_i \in \mathcal{X}$, where $\mathcal{A}(X_i) = j$, we define the parents of X_i in $\mathcal{B}_{\mathcal{M}}$ to be $\mathbf{Pa}_{\mathbf{M}_j}$, and its conditional probability distribution to be $P(\mathbf{M}_j \mid \mathbf{Pa}_{\mathbf{M}_j})$, as specified in $\mathcal{T}$. The distribution associated with $\mathcal{M}$ is the one represented by the Bayesian network $\mathcal{B}_{\mathcal{M}}$.* ∎

Returning to our example, the Bayesian network of Figure 1(a) is the ground Bayesian network of the module network of Figure 1(b).

Using the acyclicity of the module graph, we can now show that the semantics for a module network is well-defined.

Proposition 5 *The graph $\mathcal{G}_{\mathcal{M}}$ is acyclic if and only if the dependency graph of $\mathcal{B}_{\mathcal{M}}$ is acyclic.*

Proof: The proof follows from the direct correspondence between edges in the module graph and edges in the ground Bayesian network. Consider some edge $X_i \to X_j$ in $\mathcal{B}_{\mathcal{M}}$. By definition of the module graph, we must have an edge $\mathbf{M}_{\mathcal{A}(X_i)} \to \mathbf{M}_{\mathcal{A}(X_j)}$ in the module graph. Thus, any cyclic path in $\mathcal{B}_{\mathcal{M}}$ corresponds directly to a cyclic path in the module graph, proving one direction of the theorem. The proof in the other direction is slightly more subtle. Assume that there exists a cyclic path $\mathbf{p} = (\mathbf{M}_1 \to \mathbf{M}_2 \ldots \mathbf{M}_l \to \mathbf{M}_1)$ in the module graph. By definition of the module graph, if $\mathbf{M}_i \to \mathbf{M}_{i+1}$ there is a variable X_i with $\mathcal{A}(X_i) = \mathbf{M}_i$ that is a parent of X_{i+1}, for each $i = 1, \ldots, l-1$. By construction, it follows that there is an arc $X_i \to X_{i+1}$ in $\mathcal{B}_{\mathcal{M}}$. Similarly, there is a variable X_l with $\mathcal{A}(X_l) = \mathbf{M}_l$ that is a parent of $\mathbf{M}_1$. And so, we conclude that $\mathcal{B}_{ModNet}$ contains a cycle $X_1 \to X_2 \to \ldots X_l \to X_1$, proving the other direction of the theorem ∎

Corollary 6 *For any module network $\mathcal{M}$, $\mathcal{B}_{\mathcal{M}}$ defines a coherent probability distribution over $\mathcal{X}$.*

As we can see, a module network provides a succinct representation of the ground Bayesian network. In a realistic version of our stock example, we might have several thousand stocks. A Bayesian network in this domain needs to represent thousands of CPDs. On the other hand, a module network can often represent a good approximation of the domain using a model with only few dozen CPDs.

3. Data Likelihood and Bayesian Scoring

We now turn to the task of learning module networks from data. Recall that a module network is specified by a set of modules $\mathcal{C}$, an assignment function $\mathcal{A}$ of nodes to modules, the parent structure $\mathcal{S}$ specified in $\mathcal{T}$, and the parameters θ for the local probability distributions $P(\mathbf{M}_j \mid \mathbf{Pa}_{\mathbf{M}_j})$. We assume in this paper that the set of modules $\mathcal{C}$ is given, and omit reference to it from now on. We note that, in the models we consider in this paper, we do not associate properties with specific modules and thus only the number of modules is of relevance to us. However, in other settings (e.g., in cases with different types of random variables) we may wish to distinguish between different module types. Such distinctions can be made within the module network framework through more elaborate prior probability functions that take the module type into account.

One can consider several learning tasks for module networks, depending on which of the remaining aspects of the module network specification are known. In this paper, we focus on the most general task of learning the network structure and the assignment function, as well as a Bayesian posterior over the network parameters. The other tasks are special cases that can be derived as a by-product of our algorithm.

Thus, we are given a training set $\mathcal{D} = \{\mathbf{x}[1], \ldots, \mathbf{x}[M]\}$, consisting of M instances drawn independently from an unknown distribution $P(\mathcal{X})$. Our primary goal is to learn a module network structure and assignment function for this distribution. We take a *score-based approach* to this learning task. In this section, we define a scoring function that measures how well each candidate model fits the observed data. We adopt the Bayesian paradigm and derive a Bayesian scoring function similar to the Bayesian score for Bayesian networks (Cooper and Herskovits, 1992; Heckerman *et al.*, 1995). In the next section, we consider the algorithmic problem of finding a high scoring model.

3.1 Likelihood Function

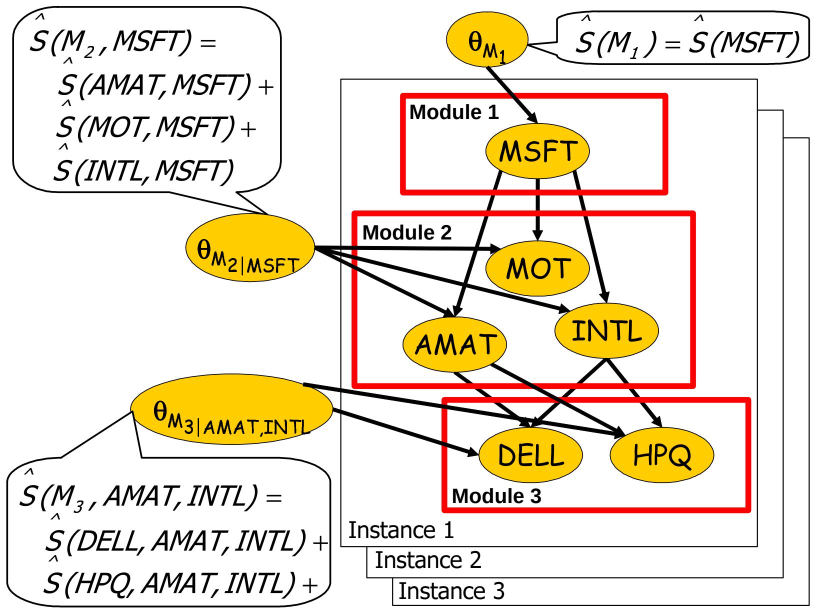
We begin by examining the *data likelihood* function

$$L(\mathcal{M} : \mathcal{D}) = P(\mathcal{D} \mid \mathcal{M}) = \prod_{m=1}^{M} P(\mathbf{x}[m] \mid \mathcal{T}, \mathcal{A}).$$

This function plays a key role both in the parameter estimation task and in the definition of the structure score.

As the semantics of a module network is defined via the ground Bayesian network, we have that, in the case of complete data, the likelihood decomposes into a product of *local likelihood functions*, one for each variable. In our setting, however, we have the additional property that the variables in a module share the same local probabilistic model. Hence, we can aggregate these local likelihoods, obtaining a decomposition according to modules.

More precisely, let $\mathbf{X}^j = \{X \in \mathcal{X} \mid \mathcal{A}(X) = j\}$, and let $\theta_{\mathbf{M}_j \mid \mathbf{Pa}_{\mathbf{M}_j}}$ be the parameters associated with the CPD template $P(\mathbf{M}_j \mid \mathbf{Pa}_{\mathbf{M}_j})$. We can decompose the likelihood function as a product of



$\hat{S}(M_2, MSFT) = \hat{S}(AMAT, MSFT) + \hat{S}(MOT, MSFT) + \hat{S}(INTL, MSFT)$
θ_{M_1}
$\hat{S}(M_1) = \hat{S}(MSFT)$
Module 1
MSFT
$\theta_{M2|MSFT}$
Module 2
MOT
AMAT
INTL
$\theta_{M3|AMAT,INTL}$
DELL
HPQ
Module 3
$\hat{S}(M_3, AMAT, INTL) = \hat{S}(DELL, AMAT, INTL) + \hat{S}(HPQ, AMAT, INTL) +$
Instance 1
Instance 2
Instance 3

Figure 2: Shown is a plate model for three instances of the module network example of Figure 1(b). The CPD template of each module is connected to all variables assigned to that module (e.g. $\theta_{\mathbf{M}_2|MSFT}$ is connected to *AMAT*, *MOT*, and *INTL*). The sufficient statistics of each CPD template are the sum of the sufficient statistics of each variable assigned to the module and the module parents.

module likelihoods, each of which can be calculated independently and depends only on the values of $\mathbf{X}^j$ and $\mathbf{Pa}_{\mathbf{M}_j}$, and on the parameters $\theta_{\mathbf{M}_j|\mathbf{Pa}_{\mathbf{M}_j}}$:

$$\begin{aligned} & L(\mathcal{M} : \mathcal{D}) \\ & = \prod_{j=1}^{K} \left[\prod_{m=1}^{M} \prod_{X_i \in \mathbf{X}^j} P(x_i[m] \mid \mathbf{pa}_{\mathbf{M}_j}[m], \theta_{\mathbf{M}_j|\mathbf{Pa}_{\mathbf{M}_j}}) \right] \\ & = \prod_{j=1}^{K} L_j(\mathbf{Pa}_{\mathbf{M}_j}, \mathbf{X}^j, \theta_{\mathbf{M}_j|\mathbf{Pa}_{\mathbf{M}_j}} : \mathcal{D}). \qquad (1) \end{aligned}$$

If we are learning conditional probability distributions from the exponential family (e.g., discrete distribution, Gaussian distributions, and many others), then the local likelihood functions can be reformulated in terms of *sufficient statistics* of the data. The sufficient statistics summarize the relevant aspects of the data. Their use here is similar to that in Bayesian networks (Heckerman, 1998), with one key difference. In a module network, all of the variables in the same module share the same parameters. Thus, we pool all of the data from the variables in $\mathbf{X}^j$, and calculate our statistics based on this pooled data. More precisely, let $S_j(M_j, \mathbf{Pa}_{\mathbf{M}_j})$ be a sufficient statistic function for the CPD $P(M_j \mid \mathbf{Pa}_{\mathbf{M}_j})$. Then the value of the statistic on the data set $\mathcal{D}$ is

$$\hat{S}_j = \sum_{m=1}^{M} \sum_{X_i \in \mathbf{X}^j} S_j(x_i[m], \mathbf{pa}_{\mathbf{M}_j}[m]). \qquad (2)$$

For example, in the case of networks that use only multinomial table CPDs, we have one sufficient statistic function for each joint assignment $x \in \mathit{Val}(\mathbf{M}_j), \mathbf{u} \in \mathit{Val}(\mathbf{Pa}_{\mathbf{M}_j})$, which is

$$\eta\{X_i[m] = x, \mathbf{pa}_{\mathbf{M}_j}[m] = \mathbf{u}\},$$

the indicator function that takes the value 1 if the event $(X_i[m] = x, \mathbf{Pa}_{\mathbf{M}_j}[m] = \mathbf{u})$ holds, and 0 otherwise. The statistic on the data is

$$\hat{S}_j[x, \mathbf{u}] \quad = \quad \sum_{m=1}^{M} \sum_{X_i \in \mathbf{X}^j} \eta\{X_i[m] = x, \mathbf{Pa}_{\mathbf{M}_j}[m] = \mathbf{u}\}.$$

Given these sufficient statistics, the formula for the module likelihood function is:

$$L_j(\mathbf{Pa}_{\mathbf{M}_j}, \mathbf{X}^j, \theta_{\mathbf{M}_j|\mathbf{Pa}_{\mathbf{M}_j}} : \mathcal{D}) = \prod_{x, \mathbf{u} \in \mathit{Val}(\mathbf{M}_j, \mathbf{Pa}_{\mathbf{M}_j})} \theta_{x|\mathbf{u}}^{\hat{S}_j[x,\mathbf{u}]}.$$

This term is precisely the one we would use in the likelihood of Bayesian networks with multinomial table CPDs. The only difference is that the vector of sufficient statistics for a local likelihood term is pooled over all the variables in the corresponding module.

For example, consider the likelihood function for the module network of Figure 1(b). In this network we have three modules. The first consists of a single variable and has no parents, and so the vector of statistics $\hat{S}[\mathbf{M}_1]$ is the same as the statistics of the single variable $\hat{S}[\mathit{MSFT}]$. The second module contains three variables; thus, the sufficient statistics for the module CPD is the sum of the statistics we would collect in the ground Bayesian network of Figure 1(a):

$$\hat{S}[\mathbf{M}_2, \mathit{MSFT}] = \hat{S}[\mathit{AMAT}, \mathit{MSFT}] + \hat{S}[\mathit{MOT}, \mathit{MSFT}] + \hat{S}[\mathit{INTL}, \mathit{MSFT}].$$

Finally,

$$\hat{S}[\mathbf{M}_3, \mathit{AMAT}, \mathit{INTL}] = \hat{S}[\mathit{DELL}, \mathit{AMAT}, \mathit{INTL}] + \hat{S}[\mathit{HPQ}, \mathit{AMAT}, \mathit{INTL}].$$

An illustration of the decomposition of the likelihood and the associated sufficient statistics using the plate model is shown in Figure 2.

As usual, the decomposition of the likelihood function allows us to perform maximum likelihood or MAP parameter estimation efficiently, optimizing the parameters for each module separately. The details are standard (Heckerman, 1998), and are thus omitted.

3.2 Priors and the Bayesian Score

As we discussed, our approach for learning module networks is based on the use of a Bayesian score. Specifically, we define a model score for a pair $(\mathcal{S}, \mathcal{A})$ as the posterior probability of the pair, integrating out the possible choices for the parameters θ. We define an assignment prior $P(\mathcal{A})$, a structure prior $P(\mathcal{S} \mid \mathcal{A})$ and a parameter prior $P(\theta \mid \mathcal{S}, \mathcal{A})$. These describe our preferences over different networks *before* seeing the data. By Bayes' rule, we then have

$$P(\mathcal{S}, \mathcal{A} \mid \mathcal{D}) \propto P(\mathcal{A}) P(\mathcal{S} \mid \mathcal{A}) P(\mathcal{D} \mid \mathcal{S}, \mathcal{A}),$$

where the last term is the *marginal likelihood*

$$P(\mathcal{D} \mid \mathcal{S}, \mathcal{A}) = \int P(\mathcal{D} \mid \mathcal{S}, \mathcal{A}, \theta) P(\theta \mid \mathcal{S}) d\theta.$$

We define the Bayesian score as the log of $P(\mathcal{S},\mathcal{A} \mid \mathcal{D})$, ignoring the normalization constant

$$\text{score}(\mathcal{S},\mathcal{A} : \mathcal{D}) = \log P(\mathcal{A}) + \log P(\mathcal{S} \mid \mathcal{A}) + \log P(\mathcal{D} \mid \mathcal{S},\mathcal{A}). \tag{3}$$

As with Bayesian networks, when the priors satisfy certain conditions, the Bayesian score decomposes. This decomposition allows to efficiently evaluate a large number of alternatives. The same general ideas carry over to module networks, but we also have to include assumptions that take the assignment function into account. Following is a list of conditions on the prior required for the decomposability of the Bayesian score in the case of module networks:

Definition 7 *Let $P(\theta,\mathcal{S},\mathcal{A})$ be a prior over assignments, structures, and parameters.*

- $P(\theta,\mathcal{S},\mathcal{A})$ *is* globally modular *if*

  $$P(\theta \mid \mathcal{S},\mathcal{A}) = P(\theta \mid \mathcal{S}),$$

  and

  $$P(\mathcal{S},\mathcal{A}) \propto \rho(\mathcal{S})\kappa(\mathcal{A})C(\mathcal{A},\mathcal{S}),$$

  where $\rho(\mathcal{S})$ and $\kappa(\mathcal{A})$ are non-negative measures over structures and assignments, and $C(\mathcal{A},\mathcal{S})$ is a constraint indicator function that is equal to 1 if the combination of structure and assignment is a legal one (i.e., the module graph induced by the assignment $\mathcal{A}$ and structure $\mathcal{S}$ is acyclic), and 0 otherwise.

- $P(\theta \mid \mathcal{S})$ *satisfies* parameter independence *if*

  $$P(\theta \mid \mathcal{S}) = \prod_{j=1}^{K} P(\theta_{\mathbf{M}_j|\mathbf{Pa}_{\mathbf{M}_j}} \mid \mathcal{S}).$$

- $P(\theta \mid \mathcal{S})$ *satisfies* parameter modularity *if*

  $$P(\theta_{\mathbf{M}_j|\mathbf{Pa}_{\mathbf{M}_j}} \mid \mathcal{S}_1) = P(\theta_{\mathbf{M}_j|\mathbf{Pa}_{\mathbf{M}_j}} \mid \mathcal{S}_2).$$

  for all structures $\mathcal{S}_1$ and $\mathcal{S}_2$ such that $\mathbf{Pa}^{\mathcal{S}_1}_{\mathbf{M}_j} = \mathbf{Pa}^{\mathcal{S}_2}_{\mathbf{M}_j}$.

- $\rho(\mathcal{S})$ *satisfies* structure modularity *if*

  $$\rho(\mathcal{S}) = \prod_j \rho_j(\mathcal{S}_j),$$

  where $\mathcal{S}_j$ denotes the choice of parents for module $\mathbf{M}_j$ and ρ_j is a non-negative measure over these choices.

- $\kappa(\mathcal{A})$ *satisfies* assignment modularity *if*

  $$\kappa(\mathcal{A}) = \prod_j \kappa_j(\mathcal{A}_j),$$

  where $\mathcal{A}_j$ denote is the choice of variables assigned to module $\mathbf{M}_j$ and κ_j is a non-negative measure over these choices. ▮

Global modularity implies that the prior can be thought of as a combination of three components — a parameter prior that depends on the network structure, a structure prior, and an assignment prior. Clearly the last two components cannot be independent, as the the assignment and the structure together must define a legal network. However, global modularity implies that these two priors are "as independent as possible". The legality requirement, which is encoded by the indicator function $C(\mathcal{A},\mathcal{S})$ ensures that only legal assignment/structure pairs have a non-zero probability. Other than this constraint, the preferences over structures and over assignments are specified separately.

Parameter independence and parameter modularity are the natural analogues of standard assumptions in Bayesian network learning (Heckerman *et al.*, 1995). Parameter independence implies that $P(\theta \mid \mathcal{S})$ is a product of terms that parallels the decomposition of the likelihood in Equation (1), with one prior term per local likelihood term L_j. Parameter modularity states that the prior for the parameters of a module M_j depends only on the choice of parents for M_j and not on other aspects of the structure.

Finally, structure modularity and assignment modularity imply that the structure an assignments priors are products of local terms that encode preferences over parents and variable assignments separately for each module.

As for the standard conditions on Bayesian network priors, the conditions we define are not universally justified, and one can easily construct examples where we would want to relax them. However, they simplify many of the computations significantly, and are therefore useful even if they are only a rough approximation. Moreover, the assumptions, although restrictive, still allow broad flexibility in our choice of priors. For example, we can encode preference (or restrictions) on the assignments of particular variables to specific modules. In addition, we can also encode preference for particular module sizes.

For priors satisfying the assumptions of Definition 7, we can prove the decomposability property of the Bayesian score for module networks:

Theorem 8 *Let $P(\theta,\mathcal{S},\mathcal{A})$ be a prior satisfying the assumptions of Definition 7. Then, the Bayesian score decomposes into local* module scores*:*

$$\text{score}(\mathcal{S},\mathcal{A} \,:\, \mathcal{D}) = \sum_{j=1}^{K} \text{score}_{\mathbf{M}_j}(\mathbf{Pa}_{\mathbf{M}_j}, \mathcal{A}(\mathbf{X}^j) \,:\, \mathcal{D}),$$

where

$$\begin{aligned}\text{score}_{\mathbf{M}_j}(\mathbf{U},\mathbf{X} \,:\, \mathcal{D}) \quad = \quad & \log \int L_j(\mathbf{U},\mathbf{X},\theta_{\mathbf{M}_j|\mathbf{U}} : \mathcal{D}) P(\theta_{\mathbf{M}_j} \mid \mathbf{U}) d\theta_{\mathbf{M}_j|\mathbf{U}} \\ & + \log \rho_j(\mathbf{U}) + \log \kappa_j(\mathbf{X}). \end{aligned} \qquad (4)$$

Proof Recall that we defined the Bayesian score of a module network as:

$$\text{score}(\mathcal{S},\mathcal{A} \,:\, \mathcal{D}) = \log P(\mathcal{D} \mid \mathcal{S},\mathcal{A}) + \log P(\mathcal{S},\mathcal{A}).$$

Using *global modularity*, *structure modularity* and *assignment modularity* assumptions of Definition 7, $\log P(\mathcal{S},\mathcal{A})$ decomposes by modules, resulting in the second and third terms Equation (4) that capture the preferences for the parents of module $\mathbf{M}_j$ and the variables assigned to it. Note that we can ignore the normalization constant of the prior $P(\mathcal{S},\mathcal{A})$. For the first term of Equation (4), we can write:

$$\begin{aligned}\log P(\mathcal{D} \mid \mathcal{S},\mathcal{A}) &= \log \int P(\mathcal{D} \mid \mathcal{S},\mathcal{A},\theta)P(\theta \mid \mathcal{S},\mathcal{A})d\theta \\ &= \log \prod_{i=1}^{K} \int L_j(\mathbf{U},\mathbf{X},\theta_{\mathbf{M}_j|\mathbf{U}} : \mathcal{D})P(\theta_{\mathbf{M}_j} \mid \mathbf{U})d\theta_{\mathbf{M}_j|\mathbf{U}} \\ &= \sum_{i=1}^{K} \log \int L_j(\mathbf{U},\mathbf{X},\theta_{\mathbf{M}_j|\mathbf{U}} : \mathcal{D})P(\theta_{\mathbf{M}_j} \mid \mathbf{U})d\theta_{\mathbf{M}_j|\mathbf{U}},\end{aligned}$$

where in the second step we used the likelihood decomposition of Equation (1) and the global modularity, parameter independence, and parameter modularity assumptions of Definition 7. ■

As we shall see below, the decomposition of the Bayesian score plays a crucial rule in our ability to devise an efficient learning algorithm that searches the space of module networks for one with high score. The only question is how to evaluate the integral over $\theta_{\mathbf{M}_j}$ in $\text{score}_{\mathbf{M}_j}(\mathbf{U},\mathbf{X} : \mathcal{D})$. This depends on the parametric forms of the CPD and the form of the prior $P(\theta_{\mathbf{M}_j} \mid \mathcal{S})$. Usually we choose priors that are *conjugate* to the parameter distributions. Such a choice leads to closed form analytic formula of the value of the integral as a function of the sufficient statistics of $L_j(\mathbf{Pa}_{\mathbf{M}_j},\mathbf{X}^j,\theta_{\mathbf{M}_j|\mathbf{Pa}_{\mathbf{M}_j}} : \mathcal{D})$. For example, using Dirichlet priors with multinomial table CPDs leads to the following formula for the integral over $\theta_{\mathbf{M}_j}$:

$$\log \int L_j(\mathbf{U},\mathbf{X},\theta_{\mathbf{M}_j|\mathbf{U}} : \mathcal{D})P(\theta_{\mathbf{M}_j} \mid \mathbf{U})d\theta_{\mathbf{M}_j|\mathbf{U}} = \\ \sum_{\mathbf{u}\in\mathbf{U}} \log \frac{\Gamma(\sum_{v\in Val(\mathbf{M}_j)} \alpha_j[v,\mathbf{u}])}{\Gamma(\sum_{v\in Val(\mathbf{M}_j)} \hat{S}_j[v,\mathbf{u}] + \alpha_j[v,\mathbf{u}])} \prod_{v\in Val(\mathbf{M}_j)} \frac{\Gamma(\hat{S}_j[v,\mathbf{u}] + \alpha_j[v,\mathbf{u}])}{\Gamma(\alpha_j[v,\mathbf{u}])},$$

where $\hat{S}_j[v,\mathbf{u}]$ is the sufficient statistics function as defined in Equation (2), and $\alpha_j[v,\mathbf{u}]$ is the hyperparameter of the Dirichlet distribution given the assignment $\mathbf{u}$ to the parents $\mathbf{U}$ of $\mathbf{M}_j$. We note that in the above formula we have also made use of the *local parameter independence* assumption on the form of the prior (Heckerman, 1998), which states that the prior distribution for the different values of the parents are independent:

$$P(\theta_{\mathbf{M}_j|\mathbf{Pa}_{\mathbf{M}_j}} \mid \mathcal{S}) = \prod_{\mathbf{u}\in Val(\mathbf{Pa}_{\mathbf{M}_j})} P(\theta_{\mathbf{M}_j|\mathbf{u}} \mid \mathcal{S}).$$

4. Learning Algorithm

Given a scoring function over networks, we now consider how to find a high scoring module network. This problem is a challenging one, as it involves searching over two combinatorial spaces simultaneously — the space of structures and the space of module assignments. We therefore simplify our task by using an iterative approach that repeats two steps: In one step, we optimize a dependency structure relative to our current assignment function, and in the other, we optimize an assignment function relative to our current dependency structure.

4.1 Structure Search Step

The first type of step in our iterative algorithm learns the structure $\mathcal{S}$, assuming that $\mathcal{A}$ is fixed. This step involves a search over the space of dependency structures, attempting to maximize the score defined in Equation (3). This problem is analogous to the problem of structure learning in Bayesian networks. We use a standard heuristic search over the combinatorial space of dependency structures (Heckerman *et al.*, 1995). We define a search space, where each state in the space is a legal parent structure, and a set of operators that take us from one state to another. We traverse this space looking for high scoring structures using a search algorithm such as greedy hill climbing.

In many cases, an obvious choice of local search operators involves steps of adding or removing a variable X_i from a parent set $\mathbf{Pa}_{\mathbf{M}_j}$. (Note that edge reversal is not a well-defined operator for module networks, as an edge from a variable to a module represents a one-to-many relation between the variable and all of the variables in the module.) When an operator causes a parent X_i to be added to the parent set of module $\mathbf{M}_j$, we need to verify that the resulting module graph remains acyclic, relative to the current assignment $\mathcal{A}$. Note that this step is quite efficient, as acyclicity is tested on the module graph, which contains only K nodes, rather than on the dependency graph of the ground Bayesian network, which contains n nodes (usually $n \gg K$).

Also note that, as in Bayesian networks, the decomposition of the score provides considerable computational savings. When updating the dependency structure for a module $\mathbf{M}_j$, the module score for another module $\mathbf{M}_k$ does not change, nor do the changes in score induced by various operators applied to the dependency structure of $\mathbf{M}_k$. Hence, after applying an operator to $\mathbf{Pa}_{\mathbf{M}_j}$, we need only update the change in score for those operators that involve $\mathbf{M}_j$. Moreover, only the delta score of operators that add or remove a parent from module $\mathbf{M}_j$ need to be recomputed after a change to the dependency structure of module $\mathbf{M}_j$, resulting in additional savings. This is analogous to the case of Bayesian network learning, where after applying a step that changes the parents of a variable X, we only recompute the delta score of operators that affect the parents of X.

Overall, if the maximum number of parents per module is d, the cost of evaluating each operator applied to the module is, as usual, at most $O(Md)$, for accumulating the necessary sufficient statistics. The total number of structure update operators is $O(Kn)$, so the cost of computing the delta-scores for all structure search operators requires $O(KnMd)$. This computation is done at the beginning of each structure learning phase. During the structure learning phase, each step to the parent set of module $\mathbf{M}_j$ requires that we re-evaluate at most n operators (one for each existing or potential parent of $\mathbf{M}_j$), at a total cost of $O(nMd)$.

4.2 Module Assignment Search Step

The second type of step in our iteration learns an assignment function $\mathcal{A}$ from data. This type of step occurs in two places in our algorithm: once at the very beginning of the algorithm, in order to initialize the modules, and once at each iteration, given a module network structure $\mathcal{S}$ learned in the previous structure learning step.

4.2.1 MODULE ASSIGNMENT AS CLUSTERING

In this step, our task is as follows: Given a fixed structure $\mathcal{S}$ we want to find

$$\mathcal{A} = \text{argmax}_{\mathcal{A}'} \text{score}_{\mathbf{M}}(\mathcal{S}, \mathcal{A}' : \mathcal{D}).$$

Interestingly, we can view this task as a clustering problem. A module consists of a set of variables that have the same probabilistic model. Thus, for a given instance, two different variables in the same module define the same probabilistic model, and therefore should have similar behavior. We can therefore view the module assignment task as the task of clustering variables into sets, so that variables in the same set have a similar behavior across all instances.

For example, in our stock market example, we would cluster stocks based on the similarity of their behavior over different trading days. Note that in a typical application of a clustering algorithm (e.g., k-means or the AutoClass algorithm of Cheeseman *et al.* (1988)) to our data set, we would cluster data instances (trading days) based on the similarity of the variables characterizing them. Here, we view instances as features of variables, and try to cluster variables. (See Figure 5.)

However, there are several key differences between this task and the typical formulation of clustering. First, in general, the probabilistic model associated with each cluster has structure, as defined by the CPD template associated with the cluster (module). Moreover, our setting places certain constraints on the clustering, so that the resulting assignment function will induce a legal (acyclic) module network.

4.2.2 MODULE ASSIGNMENT INITIALIZATION

In the initialization phase, we exploit the clustering perspective directly, using a form of hierarchical agglomerative clustering that is tailored to our application. Our clustering algorithm uses an objective function that evaluates a partition of variables into modules by measuring the extent to which the module model is a good fit to the features (instances) of the module variables. This algorithm can also be thought of as performing *model merging* (as in (Elidan and Friedman, 2001; Cheeseman *et al.*, 1988)) in a simple probabilistic model.

In the initialization phase, we do not yet have a learned structure for the different modules. Thus, from a clustering perspective, we consider a simple naive Bayes model for each cluster, where the distributions over the different features within each cluster are independent and have a separate parameterization. We begin by forming a cluster for each variable, and then merge two clusters whose probabilistic models over the features (instances) are similar.

¿From a module network perspective, the naive Bayes model can be obtained by introducing a dummy variable U that encodes training instance identity — $u[m] = m$ for all m. Throughout our clustering process, each module will have $\mathbf{Pa}_{\mathbf{M}_i} = \{U\}$, providing exactly the effect that, for each variable X_i, the different values $x_i[m]$ have separate probabilistic models. We then begin by creating n modules, with $\mathcal{A}(X_i) = i$. In this module network, each instance and each variable has its own local probabilistic model.

We then consider all possible legal module mergers (those corresponding to modules with the same domain), where we change the assignment function to replace two modules j_1 and j_2 by a new module $j_{1,2}$. This step corresponds to creating a cluster containing the variables X_{j_1} and X_{j_2}. Note that, following the merger, the two variables X_{j_1} and X_{j_2} now must share parameters, but each instance still has a different probabilistic model (enforced by the dependence on the instance ID U). We evaluate each such merger by computing the score of the resulting module network. Thus, the procedure will merge two modules that are similar to each other across the different instances. We continue to do these merge steps until we construct a module network with the desired number of modules, as specified in the original choice of $\mathcal{C}$.

```
Input:
  $D$ // Data set
  $\mathcal{A}_0$ // Initial assignment function
  $\mathcal{S}$ // Given dependency structure
Output:
  $\mathcal{A}$ // improved assignment function
Sequential-Update
  $\mathcal{A} = \mathcal{A}_0$
  Loop
    For $i = 1$ to $n$
      For $j = 1$ to $K$
        $\mathcal{A}' = \mathcal{A}$ except that $\mathcal{A}'(X_i) = j$
        If $\langle \mathcal{G}_{\mathcal{M}}, \mathcal{A}' \rangle$ is cyclic, continue
        If score$(\mathcal{S}, \mathcal{A}' : \mathcal{D}) >$ score$(\mathcal{S}, \mathcal{A} : \mathcal{D})$
          $\mathcal{A} = \mathcal{A}'$
  Until no reassignments to any of $X_1, \ldots X_n$
  Return $\mathcal{A}$
```

Figure 3: Outline of sequential algorithm for finding the module assignment function

4.2.3 MODULE REASSIGNMENT

In the module reassignment step, the task is more complex. We now have a given structure $\mathcal{S}$, and wish to find $\mathcal{A} = \text{argmax}_{\mathcal{A}'} \text{score}_{\mathbf{M}}(\mathcal{S}, \mathcal{A}' : \mathcal{D})$. We thus wish to take each variable X_i, and select the assignment $\mathcal{A}(X_i)$ that provides the highest score.

At first glance, we might think that we can decompose the score across variables, allowing us to determine independently the optimal assignment $\mathcal{A}(X_i)$ for each variable X_i. Unfortunately, this is not the case. Most obviously, the assignments to different variables must be constrained so that the module graph remains acyclic. For example, if $X_1 \in \mathbf{Pa}_{\mathbf{M}_i}$ and $X_2 \in \mathbf{Pa}_{\mathbf{M}_j}$, we cannot simultaneously assign $\mathcal{A}(X_1) = j$ and $\mathcal{A}(X_2) = i$. More subtly, the Bayesian score for each module depends non-additively on the sufficient statistics of all the variables assigned to the module. (The log-likelihood function is additive in the sufficient statistics of the different variables, but the log marginal likelihood is not.) Thus, we can only compute the delta score for moving a variable from one module to another given a *fixed* assignment of the other variables to these two modules.

We therefore use a sequential update algorithm that reassigns the variables to modules one by one. The idea is simple. We start with an initial assignment function $\mathcal{A}^0$, and in a "round-robin" fashion iterate over all of the variables one at a time, and consider changing their module assignment. When considering a reassignment for a variable X_i, we keep the assignments of all other variables fixed and find the optimal legal (acyclic) assignment for X_i relative to the fixed assignment. We continue reassigning variables until no single reassignment can improve the score. An outline of this algorithm appears in Figure 3

The key to the correctness of this algorithm is its sequential nature: Each time a variable assignment changes, the assignment function as well as the associated sufficient statistics are updated before evaluating another variable. Thus, each change made to the assignment function leads to a legal assignment which improves the score. Our algorithm terminates when it can no longer im-

```
Input:
    $D$ // Data set
    $K$ // Number of modules
Output:
    M // A module network
Learn-Module-Network
    $\mathcal{A}_0$ = cluster $\mathcal{X}$ into $K$ modules
    $\mathcal{S}_0$ = empty structure
    Loop $t = 1, 2, \ldots$ until convergence
        $\mathcal{S}_t$ = Greedy-Structure-Search($\mathcal{A}_{t-1}, \mathcal{S}_{t-1}$)
        $\mathcal{A}_t$ = Sequential-Update($\mathcal{A}_{t-1}, \mathcal{S}_t$);
    Return $\mathbf{M} = (\mathcal{A}_t, \mathcal{S}_t)$
```

Figure 4: Outline of the *module network* learning algorithm. Greedy-Structure-Search successively applies operators that change the structure as long as each such operator results in a legal structure and improves the module network score

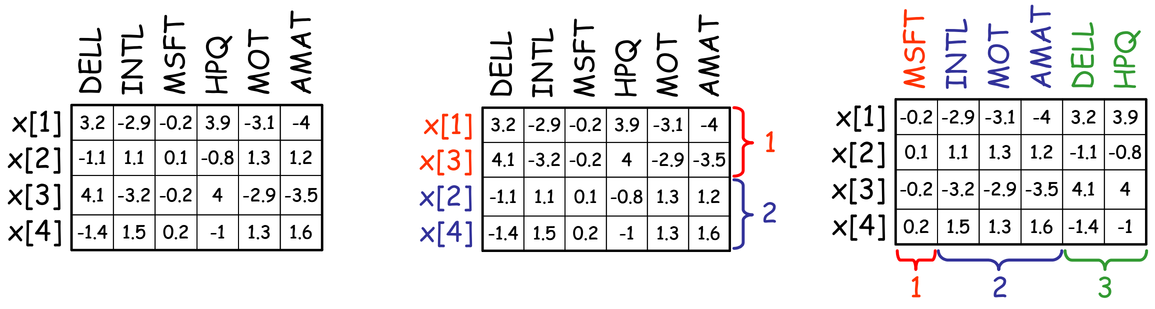
prove the score. Hence, it converges to a local maximum, in the sense that no single assignment change can improve the score.

The computation of the score is the most expensive step in the sequential algorithm. Once again, the decomposition of the score plays a key role in reducing the complexity of this computation: When reassigning a variable X_i from one module $\mathbf{M}_{old}$ to another $\mathbf{M}_{new}$, only the local scores of these modules change. The module score of all other modules remains unchanged. The rescoring of these two modules can be accomplished efficiently by subtracting X_i's statistics from the sufficient statistics of $\mathbf{M}_{old}$ and adding them to those of $\mathbf{M}_{new}$. Thus, assuming that we have precomputed the sufficient statistics associated with every pair of variable X_i and module $\mathbf{M}_j$, the cost of recomputing the delta-score for an operator is $O(s)$, where s is the size of the table of sufficient statistics for a module. The only operators whose delta-scores change are those involving reassignment of variables to/from these two modules. Assuming that each module has approximately $O(n/K)$ variables, and we have at most K possible destinations for reassigning each variable, the total number of such operators is generally linear in n. Thus, the cost of each reassignment step is approximately $O(ns)$. In addition, at the beginning of the module reassignment step, we must initialize all of the sufficient statistics at a cost of $O(Mnd)$, and compute all of the delta-scores at a cost of $O(nK)$.

4.3 Algorithm Summary

To summarize, our algorithm starts with an initial assignment of variables to modules. In general, this initial assignment can come from anywhere, and may even be a random guess. We choose to construct it using the clustering-based idea described in the previous section. The algorithm then iteratively applies the two steps described above: learning the module dependency structures, and reassigning variables to modules. These two steps are repeated until convergence, where convergence is defined by a score improvement of less than some fixed threshold Δ between two consecutive learned models. An outline of the module network learning algorithm is shown in Figure 4.

Each of these two steps — structure update and assignment update — is guaranteed to either improve the score or leave it unchanged. The following result therefore follows immediately:



	DELL	INTL	MSFT	HPQ	MOT	AMAT
x[1]	3.2	-2.9	-0.2	3.9	-3.1	-4
x[2]	-1.1	1.1	0.1	-0.8	1.3	1.2
x[3]	4.1	-3.2	-0.2	4	-2.9	-3.5
x[4]	-1.4	1.5	0.2	-1	1.3	1.6

(a) Data

	DELL	INTL	MSFT	HPQ	MOT	AMAT	
x[1]	3.2	-2.9	-0.2	3.9	-3.1	-4	1
x[3]	4.1	-3.2	-0.2	4	-2.9	-3.5	1
x[2]	-1.1	1.1	0.1	-0.8	1.3	1.2	2
x[4]	-1.4	1.5	0.2	-1	1.3	1.6	2

(b) Standard clustering

	MSFT	INTL	MOT	AMAT	DELL	HPQ
x[1]	-0.2	-2.9	-3.1	-4	3.2	3.9
x[2]	0.1	1.1	1.3	1.2	-1.1	-0.8
x[3]	-0.2	-3.2	-2.9	-3.5	4.1	4
x[4]	0.2	1.5	1.3	1.6	-1.4	-1
	1	2	2	2	3	3

(c) Initialization

Figure 5: Relationship between the module network procedure and clustering. Finding an assignment function can be viewed as a clustering of the variables whereas clustering typically clusters instances. Shown is sample data for the example domain of Figure 1, where the rows correspond to instances and the columns correspond to variables. (a) Data. (b) Standard clustering of the data in (a). Note that $x[2]$ and $x[3]$ were swapped to form the clusters. (c) Initialization of the assignment function for the module network procedure for the data in (a). Note that variables were swapped in their location to reflect the initial assignment into three modules.

Theorem 4.1: The iterative module network learning algorithm converges to a local maximum of $\text{score}(\mathcal{S},\mathcal{A} : \mathcal{D})$.

We note that both the structure search step and the module reassignment step are done using simple greedy hill-climbing operations. As in other settings, this approach is liable to get stuck in local maxima. We attempt to somewhat compensate for this limitation by initializing the search at a reasonable starting point, but local maxima are clearly still an issue. An additional strategy that would help circumvent some maxima is the introduction of some randomness into the search (e.g., by random restarts or simulated annealing), as is often done when searching complex spaces with multi-modal target functions.

5. Learning with Regression Trees

We now briefly review the family of conditional distributions we use in the experiments below. Many of the domains suited for module network models contain continuous valued variables, such as gene expression or price changes in the stock market. For these domains, we often use a conditional probability model represented as a *regression tree* (Breiman *et al.*, 1984). For our purposes, a regression tree T for $P(X \mid \mathbf{U})$ is defined via a rooted binary tree, where each *node* in the tree is either a *leaf* or an *interior node*. Each interior node is labeled with a test $U < u$ on some variable $U \in \mathbf{U}$ and $u \in \mathbb{R}$. Such an interior node has two outgoing *arcs* to its children, corresponding to the outcomes of the test (true or false). The tree structure T captures the *local* dependency structure of the conditional distribution. The parameters of T are the distributions associated with each leaf. In our implementation, each leaf ℓ is associated with a univariate Gaussian distribution over values of X, parameterized by a mean μ_ℓ and variance σ^2_ℓ. An example of a regression tree CPD is shown in

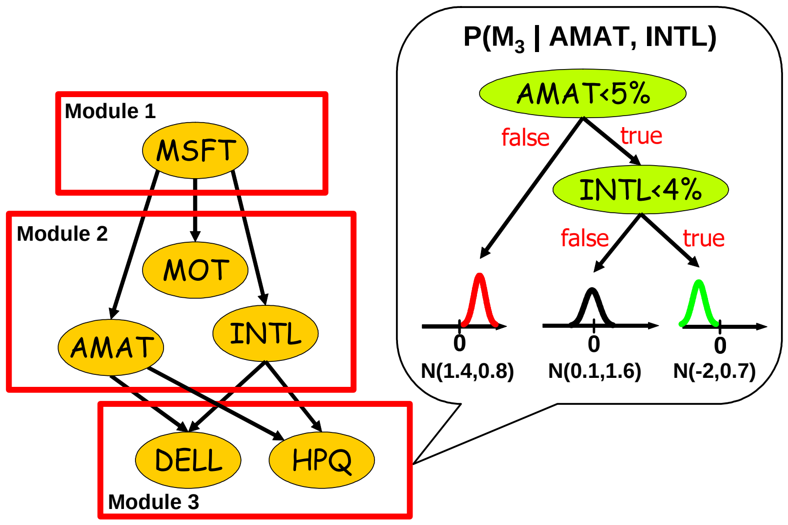


Figure 6: Example of a regression tree with univariate Gaussian distributions at the leaves for representing the CPD $P(\mathbf{M}_3 \mid AMAT, INTL)$, associated with $\mathbf{M}_3$. The tree has internal nodes labeled with a test on the variable (e.g. $AMAT < 5\%$). Each univariate Gaussian distribution at a leaf is parameterized by a mean and a variance. The tree structure captures the local dependency structure of the conditional distributions. In the example shown, when $AMAT \geq 5\%$, then the distribution over values of variables assigned to $\mathbf{M}_3$ will be Gaussian with mean 1.4 and standard deviation 0.8 regardless of the value of *INTL*.

Figure 6. We note that, in some domains, Gaussian distributions may not be the appropriate choice of models to assign at the leaves of the regression tree. In such cases, we can apply transformations to the data to make it more appropriate for modeling by Gaussian distributions, or use other continuous or discrete distributions at the leaves.

To learn module networks with regression-tree CPDs, we must extend our previous discussion by adding another component to $\mathcal{S}$ that represents the trees $T_1, \ldots, T_K$ associated with the different modules. Once we specify these components, the above discussion applies with several small differences. These issues are similar to those encountered when introducing decision trees to Bayesian networks (Chickering *et al.*, 1997; Friedman and Goldszmidt, 1998), so we discuss them only briefly.

Given a regression tree T_j for $P(\mathbf{M}_j \mid \mathbf{Pa}_{\mathbf{M}_j})$, the corresponding sufficient statistics are the statistics of the distributions at the leaves of the tree. For each leaf ℓ in the tree, and for each data instance $\mathbf{x}[m]$, we let $\ell_j[m]$ denote the leaf reached in the tree given the assignment to $\mathbf{Pa}_{\mathbf{M}_j}$ in $\mathbf{x}[m]$. The module likelihood decomposes as a product of terms, one for each leaf ℓ. Each term is the likelihood for the Gaussian distribution $\mathcal{N}\left(\mu_\ell; \sigma_\ell^2\right)$, with the usual sufficient statistics for a Gaussian distribution.

Given a regression tree T_j for $P(\mathbf{M}_j \mid \mathbf{Pa}_{\mathbf{M}_j})$, the corresponding sufficient statistics are the statistics of the distributions at the leaves of the tree. For each leaf ℓ in the tree, and for each data instance $\mathbf{x}[m]$, we let $\ell_j[m]$ denote the leaf reached in the tree given the assignment to $\mathbf{Pa}_{\mathbf{M}_j}$ in $\mathbf{x}[m]$. The module likelihood decomposes as a product of terms, one for each leaf ℓ. Each term is the likelihood for

the Gaussian distribution $\mathcal{N}\left(\mu_\ell;\sigma_\ell^2\right)$, with the sufficient statistics for a Gaussian distribution.

$$\begin{aligned}
\hat{S}^0_{j,\ell} &= \sum_m \sum_{X_i \in \mathbf{X}^j} \eta\{\ell_j[m]=\ell\}, \\
\hat{S}^1_{j,\ell} &= \sum_m \sum_{X_i \in \mathbf{X}^j} \eta\{\ell_j[m]=\ell\} x_i, \\
\hat{S}^2_{j,\ell} &= \sum_m \sum_{X_i \in \mathbf{X}^j} \eta\{\ell_j[m]=\ell\} x_i^2.
\end{aligned} \tag{5}$$

The local module score further decomposes into independent components, one for each leaf ℓ. Here, we use a Normal-Gamma prior (DeGroot, 1970) for the distribution at each leaf: Letting $\tau_\ell = 1/\sigma_\ell^2$ stand for the precision at leaf ℓ, we define: $P(\mu_\ell, \tau_\ell) = P(\mu_\ell \mid \tau_\ell)P(\tau_\ell)$, where $P(\tau_\ell) \sim \Gamma(\alpha_0, \beta_0)$ and $P(\mu_\ell \mid \tau_\ell) \sim \mathcal{N}\left(\mu_0; (\lambda_0\tau_\ell)^{-1}\right)$, where we assume that all leaves are associated with the same prior. Letting $\hat{S}^i_{j,\ell}$ be defined as in Equation (5), we have that the component of the log marginal likelihood associated with a leaf ℓ of module j is given by:

$$\begin{aligned}
&-\frac{1}{2}\hat{S}^0_{j,\ell}\log(2\pi) + \frac{1}{2}\log\left(\frac{\lambda_0}{\lambda_0 + \hat{S}^0_{j,\ell}}\right) + \log\left(\Gamma(\alpha_0 + \frac{1}{2}\hat{S}^0_{j,\ell})\right) \\
&- \log\left(\Gamma(\alpha_0)\right) + \alpha_0 \log\left(\beta_0\right) - \left(\alpha_0 + \frac{1}{2}\hat{S}^0_{j,\ell}\right)\log\left(\beta\right),
\end{aligned}$$

where

$$\beta = \beta_0 + \frac{1}{2}\left(\hat{S}^2_{j,\ell} - \frac{(\hat{S}^1_{j,\ell})^2}{\hat{S}^0_{j,\ell}}\right) + \frac{\hat{S}^0_{j,\ell}\lambda_0\left(\frac{\hat{S}^1_{j,\ell}}{\hat{S}^0_{j,\ell}} - \mu_0\right)^2}{2(\lambda_0 + \hat{S}^0_{j,\ell})}.$$

When performing structure search for module networks with regression-tree CPDs, in addition to choosing the parents of each module, we must also choose the associated tree structure. We use the search strategy proposed by Chickering *et al.* (1997), where the search operators are leaf splits. Such a *split* operator replaces a leaf in a tree T_j with an internal node with some test on a variable U. The two branches below the newly created internal node point to two new leaves, each with its associated Gaussian. This operator must check for acyclicity, as it implicitly adds U as a parent of $\mathbf{M}_j$.

When performing the search, we consider splitting each possible leaf on each possible parent U and each value u. As always in regression-tree learning, we do not have to consider all real values u as possible split points; it suffices to consider values that arise in the data set. Moreover, under an appropriate choice of prior (i.e., an independent prior for each leaf), regression-tree learning provides another level of score decomposition: The score of a particular tree is a sum of scores for the leaves in the tree. Thus, a split operation on one leaf in the tree does not affect the score component of another leaf, so that operators applied to other leaves do not need to re-evaluated.

6. Experimental Results

We evaluated our module network learning procedure on synthetic data and on two real data sets — gene expression data, and stock market data. In all cases, our data consisted solely of continuous values. As all of the variables have the same domain, the definition of the module set reduces simply

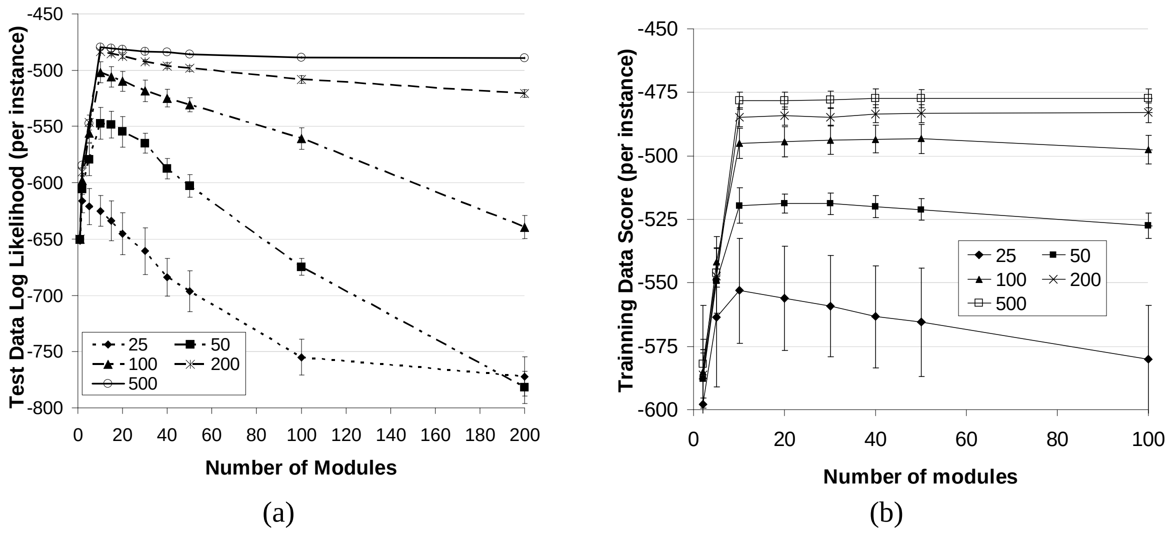


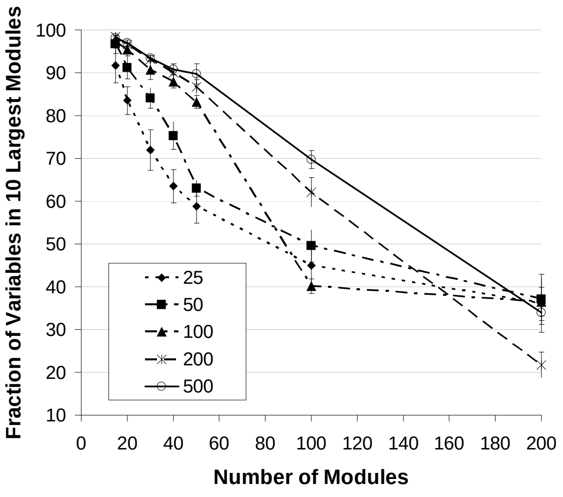
Figure 7: Performance of learning from synthetic data as a function of the number of modules and training set size. The *x*-axis corresponds to the number of modules, each curve corresponds to a different number of training instances, and each point shows the mean and standard deviations from the 10 sampled data sets. (a) Log-likelihood per instance assigned to held-out data. (b) Average score per instance on the training data.

to a specification of the total number of modules. We used regression trees as the local probability model for all modules, and uniform priors for $\rho(\mathcal{S})$ and $\kappa(\mathcal{A})$. For structure search, we used beam search, using a lookahead of three splits to evaluate each operator. When learning Bayesian networks, as a comparison, we used precisely the same structure learning algorithm, simply treating each variable as its own module.

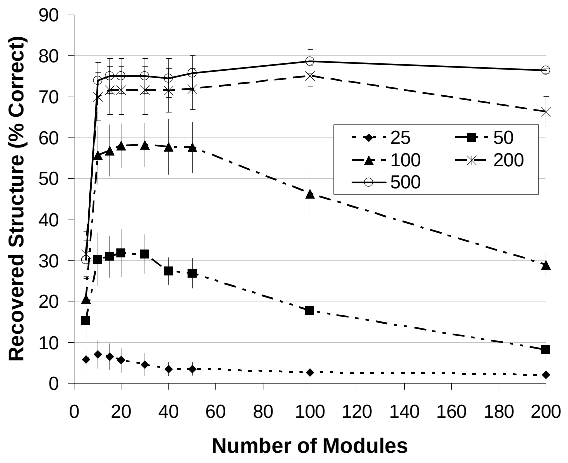
6.1 Synthetic Data

As a basic test of our procedure in a controlled setting, we used synthetic data generated by a known module network. This gives a known ground truth to which we can compare the learned models. To make the data realistic, we generated synthetic data from a model that was learned from the gene expression data set described below. The generating model had 10 modules and a total of 35 variables that were a parent of some module. From the learned module network, we selected 500 variables, including the 35 parents. We tested our algorithm's ability to reconstruct the network using different numbers of modules; this procedure was run for training sets of various sizes ranging from 25 instances to 500 instances, each repeated 10 times for different training sets.

We first evaluated the generalization to unseen test data, measuring the likelihood ascribed by the learned model to 4500 unseen instances. The results, summarized in Figure 7(a), show that, for all training set sizes, except the smallest one with 25 instances, the model with 10 modules performs the best. As expected, models learned with larger training sets do better; but, when run using the correct number of 10 modules, the gain of increasing the number of data instances beyond 100 samples is small and beyond 200 samples is negligible.



(a)



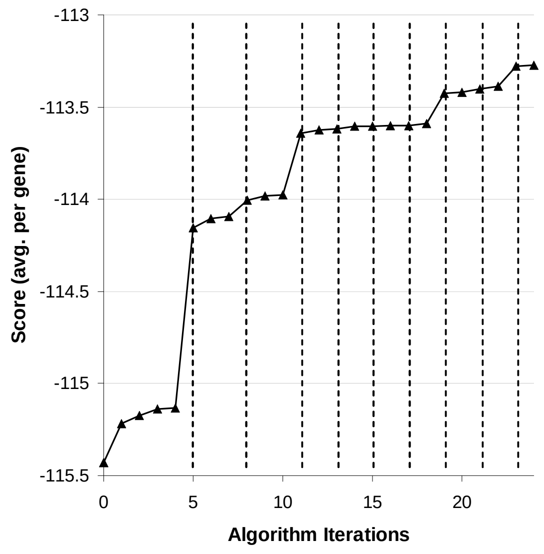
(b)

Figure 8: (a) Fraction of variables assigned to the 10 largest modules. (b) Average percentage of correct parent-child relationships recovered (fraction of parent-child relationships in the true model recovered in the learned model) when learning from synthetic data for models with various number of modules and different training set sizes. The *x*-axis corresponds to the number of modules, each curve corresponds to a different number of training instances, and each point shows the mean and standard deviations from the 10 sampled data sets.

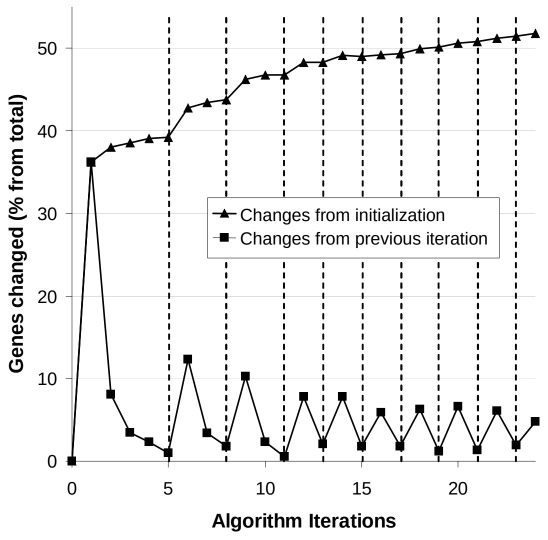
To test whether we can use the score of the model to select the number of modules, we also plotted the score of the learned model on the training data (Figure 7(b)). As can be seen, when the number of instances is small (25 or 50), the model with 10 modules achieves the highest score and for a larger number of instances, the score does not improve when increasing the number of modules beyond 10. Thus, these results suggest that we can select the number of modules by choosing the model with the smallest number of modules from among the highest scoring models.

A closer examination of the learned models reveals that, in many cases, they are almost a 10-module network. As shown in Figure 8(a), models learned using 100, 200, or 500 instances and up to 50 modules assigned $\geq$ 80% of the variables to 10 modules. Indeed, these models achieved high performance in Figure 7(a). However, models learned with a larger number of modules had a wider spread for the assignments of variables to modules and consequently achieved poor performance.

Finally, we evaluated the model's ability to recover the correct dependencies. The total number of parent-child relationships in the generating model was 2250. For each model learned, we report the fraction of correct parent-child relationships it contains. As shown in Figure 8(b), our procedure recovers 74% of the true relationships when learning from a data set with 500 instances. Once again, we see that, as the variables begin fragmenting over a large number of modules, the learned structure contains many spurious relationships. Thus, our results suggest that, in domains with a modular structure, statistical noise is likely to prevent overly detailed learned models such as Bayesian networks from extracting the commonality between different variables with a shared behavior.



(a)



(b)

Figure 9: (a) Score of the model (normalized by the number of variables/genes) across the iterations of the algorithm for a module network learned with 50 modules on the gene expression data. Iterations in which the structure was changed are indicated by dashed vertical lines. (b) Changes in the assignment of genes to modules for the module network learned in (a) across the iterations of the algorithm. Shown are both the total changes compared to the initial assignment (triangles) and the changes compared to the previous iteration (squares).

6.2 Gene Expression Data

We next evaluated the performance of our method on a real world data set of gene expression measurements. A *microarray* measures the activity level (mRNA expression level) of thousands of genes in the cell in a particular condition. We view each experiment as an instance, and the expression level of each measured gene as a variable (Friedman *et al.*, 2000a). In many cases, the coordinated activity of a group of genes is controlled by a small set of *regulators*, that are themselves encoded by genes. Thus, the activity level of a regulator gene can often predict the activity of the genes in the group. Our goal is to discover these modules of co-regulated genes, and their regulators.

We used the expression data of Gasch *et al.* (et al., 2000), which measured the response of yeast to different stress conditions. The data consists of 6157 genes and 173 experiments. In this domain, we have prior knowledge of which genes are likely to play a regulatory role (e.g., based on properties of their protein sequence). Consequently, we restricted the possible parents to 466 yeast genes that may play such a role. We then selected 2355 genes that varied significantly in the data and learned a module network over these genes. We also learned a Bayesian network over this data set.

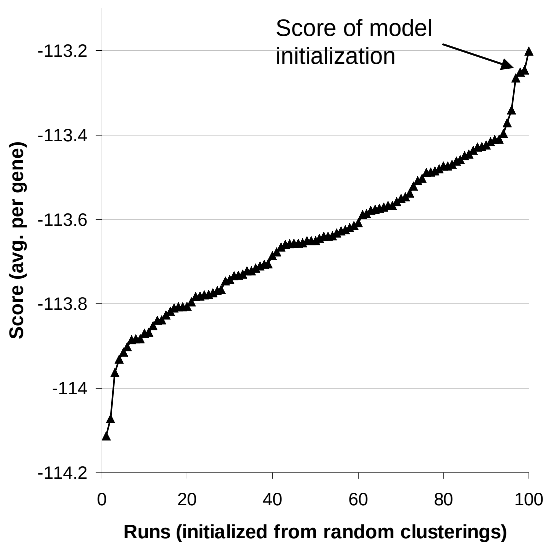


Figure 10: Score of 100 module networks (normalized by the number of variables/genes) each learned with 50 modules from a random clustering initialization, where the runs are sorted according to their score. The score of a module network learned using the deterministic clustering initialization described in Section 4.2 is indicated by a pointed arrow.

6.2.1 STATISTICAL EVALUATION

We first examined the behavior of the learning algorithm on the training data when learning a module network with 50 modules. This network converged after 24 iterations (of which nine were iterations in which the structure of the network changed). To characterize the trajectory of the algorithm, we plot in Figure 9 its improvement across the iterations, measured as the score on the training data, normalized by the number of genes (variables). To obtain a finer-grained picture, we explicitly show structure learning steps, as well as each pass over the variables in the module reassignment step. As can be seen in Figure 9(a), the model score improves nicely across these steps, with the largest gains in score occurring in iterations in which the structure was changed (dotted lines in Figure 9(a)). Figure 9(b) demonstrates how the algorithm changes the assignments of genes to modules, with 1221 of the 2355 (51.8%) genes changing their assignment upon convergence, and the largest assignment changes occurring immediately after structure modification steps.

As for most local search algorithms, initialization is an key component: A bad initialization can cause the algorithm to get trapped in a poor local maximum. As we discussed in Section 4.2, we initialize the assignment function using a clustering program. The advantage of a simple deterministic initialization procedure is that it is computationally efficient, and results in reproducible behavior. We evaluated this proposed initialization by comparing the results to module networks initialized randomly. We generated 100 random assignments of variables to modules, and learned a module network starting from each initialization. We compared the model score of the network learned using our deterministic initialization, and the 100 networks initialized randomly. A plot of these sorted scores is shown in Figure 10. Encouragingly, the score for the network initialized using our procedure was better than $97/100$ of the runs initialized from random clusters, and the $3/100$ runs that did better are only incrementally better.

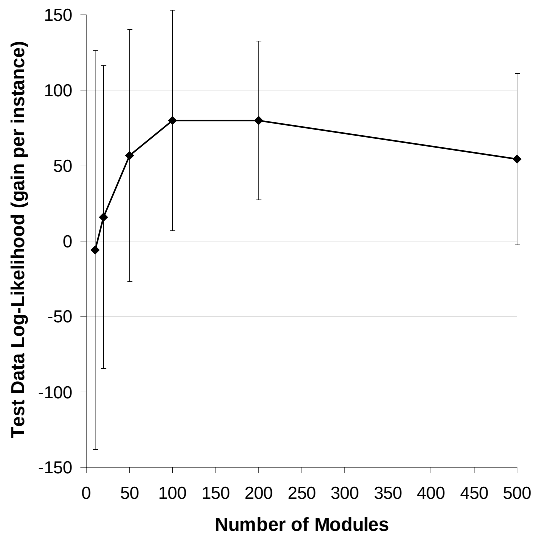
We evaluated the generalization ability of different models, in terms of log-likelihood of test data, using 10-fold cross validation. In Figure 11(a), we show the difference between module networks of different size and the baseline Bayesian network, demonstrating that module networks generalize much better to unseen data for almost all choices of number of modules.

6.2.2 BIOLOGICAL EVALUATION

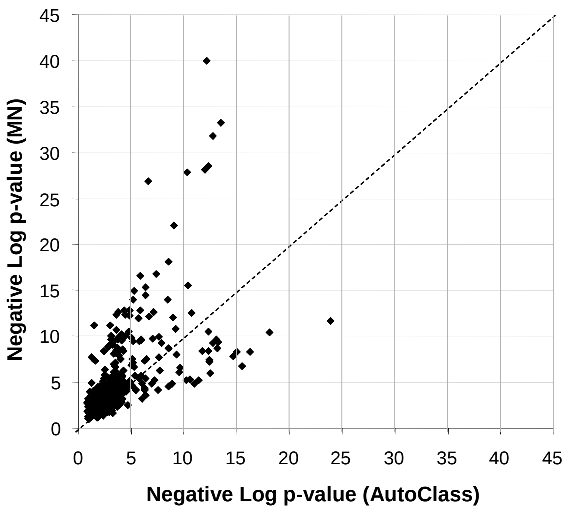
As we discussed in the introduction, a common goal in learning a network structure is to reveal structural properties of the underlying distribution. This goal is definitely an important one in the biological domain, where we want to discover both sets of co-regulated genes, and the regulatory mechanism governing their behavior. We therefore evaluated the ability of our module network learning procedure to reveal known biological properties of this domain.

We evaluated a learned module network with 50 modules, where we selected 50 modules due to the biological plausibility of having, on average, 40–50 genes per module. First, we examined whether genes in the same module have shared functional characteristics. To this end, we used annotations of the genes' biological functions from the Saccharomyces Genome Database (Cherry *et al.*, 1998). We systematically evaluated each module's gene set by testing for significantly enriched annotations. Suppose we find l genes with a certain annotation in a module of size N. To check for enrichment, we calculate the *hypergeometric p-value* of these numbers — the probability of finding that many genes of that annotation in a random subset of N genes. For example, the "protein folding" module contains 10 genes, 7 of which are annotated as protein folding genes. In the whole data set, there are only 26 genes with this annotation. The p-value of this annotation, that is, the probability of choosing 7 or more genes in this category by choosing 10 random genes, is less than 10^{-12}. As there are a large number of possible annotations, there is a nontrivial probability that some will be enriched simply by chance. We therefore corrected these p-values using the standard Bonferroni correction for independent multiple hypotheses (Savin, 1980). Our evaluation showed that, of the 50 modules, 42 (resp. 20) modules had at least one significantly enriched annotation with a p-value less than 0.005 (resp. less than 10^{-6}). Furthermore, the enriched annotations reflect the key biological processes expected in our data set. We used these annotations to label the modules with meaningful biological names. A comparison of the overall enrichments of the modules learned by module networks to the enrichments obtained for clusters using AutoClass is shown in Figure 11(b), indicating that there are many annotations that are much more significantly enriched in module networks.

We can use these annotations to reason about the dependencies between different biological processes at the module level. For example, we find that the *cell cycle* module, regulates the *histone* module. The cell cycle is the process in which the cell replicates its DNA and divides, and it is indeed known to regulate histones — key proteins in charge of maintaining and controlling the DNA structure. Another module regulated by the cell cycle module is the *nitrogen catabolite repression (NCR)* module, a cellular response activated when nitrogen sources are scarce. We find that the *NCR* module regulates the *amino acid metabolism*, *purine metabolism* and *protein synthesis* modules, all representing nitrogen-requiring processes, and hence likely to be regulated by the *NCR* module.



(a) Test-data generalization (Expression)



(b) Annotation enrichment (Expression)

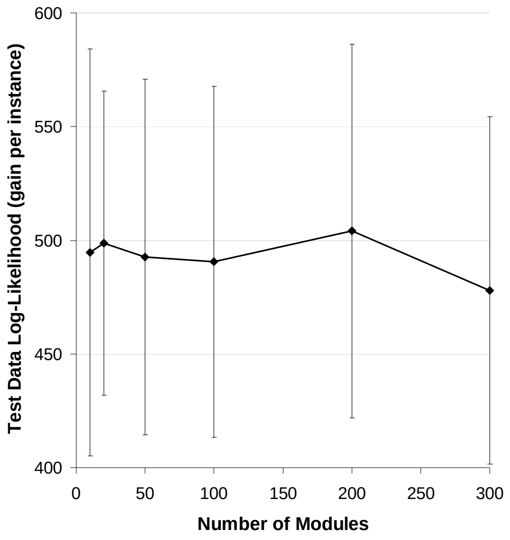
Figure 11: (a) Comparison of generalization ability of module networks learning with different numbers of modules on the gene expression data set. The *x*-axis denotes the number of modules. The *y*-axis denotes the difference in log-likelihood on held out data between the learned module network and the learned Bayesian network, averaged over 10 folds; the error bars show the standard deviation. (b) Comparison of the enrichment for annotations of functional annotations between the modules learned using the module network procedure and the clusters learned by the AutoClass clustering algorithm (Cheeseman *et al.*, 1988) applied to the variables. Each point corresponds to an annotation, and the *x* and *y* axes are the negative log *p*-values of its enrichment for the two models.

These examples demonstrate the insights that can be gleaned from a higher order model, and which would have been obscured in the unrolled Bayesian network over 2355 genes.

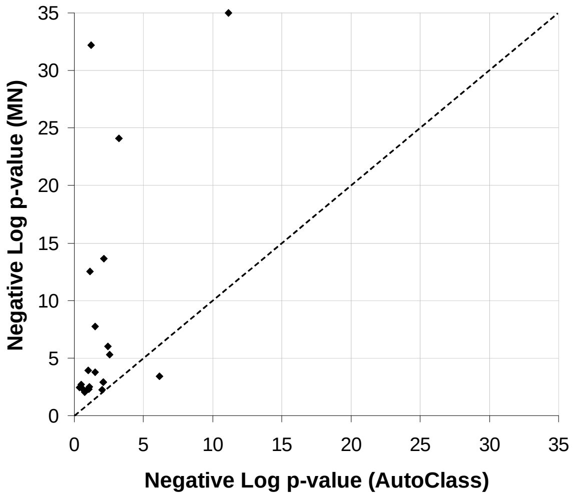
6.3 Stock Market Data

In a very different application, we examined a data set of NASDAQ stock prices. We collected stock prices for 2143 companies, in the period 1/1/2002–2/3/2003, covering 273 trading days (data was obtained from `http://finance.yahoo.com`). We took each stock to be a variable, and each instance to correspond to a trading day, where the value of the variable is the log of the ratio between that day's and the previous day's closing stock price. This choice of data representation focuses on the relative changes to the stock price, and eliminates the magnitude of the price itself (which depends on such irrelevant factors as the number of outstanding shares). As potential controllers, we selected 250 of the 2143 stocks, whose average trading volume was the largest across the data set.

As with gene expression data, we used cross validation to evaluate the generalization ability of different models. As we can see in Figure 12(a), module networks perform significantly better than Bayesian networks in this domain.



(a) Test-data generalization (Stock)



(b) Annotation enrichment (Stock)

Figure 12: (a) Comparison of generalization ability of module networks learning with different numbers of modules on the stock data set. The *x*-axis denotes the number of modules. The *y*-axis denotes the difference in log-likelihood on held out data between the learned module network and the learned Bayesian network, averaged over 10 folds; the error bars show the standard deviation. (b) Comparison of the enrichment for annotations of sectors between the modules learned using the module network procedure and the clusters learned by the AutoClass clustering algorithm (Cheeseman *et al.*, 1988) applied to the variables. Each point corresponds to an annotation, and the *x* and *y* axes are the negative log *p*-values of its enrichment for the two models.

To test the quality of our modules, we measured the enrichment of the modules in the network with 50 modules for annotations representing various sectors to which each stock belongs (based on sector classifications from `http://finance.yahoo.com`). We found significant enrichment for 21 such annotations, covering a wide variety of sectors. We also compared these results to the clusters of stocks obtained from applying the popular probabilistic clustering algorithm AutoClass (Cheeseman *et al.*, 1988) to the data. Here, as we described above, each instance corresponds to a stock and is described by 273 random variables, each representing a trading day. In 20 of the 21 cases, the enrichment was far more significant in the modules learned using module networks compared to the one learned by AutoClass, as can be seen in Figure 12(b).

Finally, we also looked at the structure of the module network, and found several cases where the structure fit our (limited) understanding of the stock domain. Several modules corresponded primarily to high tech stocks. One of these, consisting mostly of software, semi-conductor, communication, and broadcasting services, had as its two main predictors Molex, a large manufacturer of electronic, electrical and fiber optic interconnection products and systems, and Atmel, specializing in design, manufacturing and marketing of advanced semiconductors. Molex was also the parent for another module, consisting primarily of software, semi-conductor, and medical equipment companies; this module had as additional parents Maxim, which develop integrated circuits, and Affymetrix, which designs and develops gene microarray chips. In this, as in many other cases, the parents of a module are from similar sectors as the stocks in the module.

7. Related Work

Module networks are related to several other approaches, including plates Buntine (1994), hierarchical Bayesian models DeGroot (1970), *object-oriented Bayesian networks* (OOBNs) (Koller and Pfeffer, 1997) and to the framework of *probabilistic relational models* (PRMs) (Koller and Pfeffer, 1998; Friedman *et al.*, 1999a).

Both plates and hierarchical Bayesian approaches allow us to represent models where objects in the same class share parameters. Plate models also allow objects to share the same parent set. In many ways, they allow a more expressive dependency structure than module networks, as they allow a richly structured hierarchical set of variables, determined by the nested plate structure. However, variables in one plate can only depend on variables in an enclosing plate. Thus, plate models are not sufficiently expressive to encode the inter-module dependencies in a module-network. Hierarchical Bayesian models are more expressive than module networks in that they allow parameters of different variables to be statistically related but not necessarily equal. However, hierarchical Bayesian approaches are not a language that includes structure as well as parameters, so that an additional representation layer would have to be added to provide a framework similar to module networks. One can easily extend module networks with ideas from the hierarchical Bayesian framework, allowing the parameters of different variables in the same module to be correlated but not necessarily equal. Most importantly, neither plates nor the hierarchical Bayesian framework have provided a method that allows us to learn automatically which subsets of variables share parameters.

OOBNs and PRMs extend Bayesian Networks to a setting involving multiple related objects, and allow the attributes of objects of the same class to share parameters and dependency structure. One can view the module network framework as a restriction of these frameworks, where we have one object for every variable X_i, with a single attribute corresponding to the value of X_i. Each module can be viewed as a class, so that the variables in a single module share the same probabilistic model. As the module assignments are not known in advance, module networks correspond most closely to the variant of these frameworks where there is *type uncertainty* — uncertainty about the class assignment of objects. However, despite this high-level similarity, the module network framework differs in certain key points from both OOBNs and PRMs, with significant impact on the learning task.

In OOBNs, objects in the same class must have the same internal structure and parameterization, but can depend on different sets of variables (as specified in the mapping of variables in an object's interface to its actual inputs). By contrast, in a module network, all of the variables in a module (class) must have the same specific parents. This assumption greatly reduces the size and complexity of the hypothesis space, leading to a more robust learning algorithm. On the other hand, this assumption requires that we be careful in making certain steps in the structure search, as they have more global effects than on just one or two variables. Due to these differences, we cannot simply apply an OOBN structure-learning algorithm, such as the one proposed by Langseth and Nielsen (2003), to such complex, high-dimensional domains.

In PRMs, the probabilistic dependency structure of the objects in a class is determined by the relational structure of the domain (e.g., the *Cost* attribute of a particular car object might depend on the *Income* attribute of the object representing this particular car's owner). In the case of module networks, there is no known relational structure to which probabilistic dependencies can be attached. Without such a relational structure, PRMs only allow dependency models specified at the class level. Thus, we can assert that the objects in one class depend on some aggregate quantity of the objects in another. We cannot, however, state a dependence on a particular object in the other class (without some relationship specified in the model). Getoor *et al.* (2000) attempt to address this issue using a class hierarchy. Their approach is very different from ours, requiring some fairly complex search steps, and is not easily applied to the types of domains considered in this paper.

To better relate the PRM approach to module networks, recall the relationship between module networks and clustering, as described in Section 4.2. As we discussed, we can view the module network learning procedure as grouping variables into clusters that share the same probabilistic dependency model. As shown in Figure 5, we are taking the data points in the (variables x instances) matrix, and grouping rows. As we discussed, in other settings, we often group columns (instances). In fact, in many cases, the notion of "variables" and "instances" is somewhat arbitrary. PRMs allow us to define a probabilistic model where the value of a data point depends both on properties of the rows and properties of the column. In particular, we can define a hidden attribute for either rows, columns, or both; the values of this hidden attribute would correspond to a clustering of rows, or columns, or a two-sided clustering of both rows and columns simultaneously (see Segal *et al.* (2001)).

From this perspective, the module network framework can be viewed as being closely related to a PRM where the module assignment is a hidden attribute of a row. For example, in the gene expression domain, the expression value of gene g_i in microarray a_j depends on attributes both of g_i and of a_j. The gene g_i only has one attribute, representing its module assignment. The array a_j has attributes representing the expression levels of the different regulators in the array. The expression level of gene g_i in experiment a_j then depends on all of these attributes, i.e., on the gene's module assignment and on the values of the regulators. A key difference between the PRM-based approach and our module network framework is that, in the PRM, the regulators cannot themselves participate in the probabilistic model without leading to cycles. This restriction forces us to select a relatively small set of candidate regulators in advance. Moreover, as no probabilistic dependency model is learned for regulators, this approach cannot discover compound regulatory pathways, which are often of great interest.

Overall, the module network framework places strong restrictions on the richness of the set of objects and on the dependency structures that can be represented. However, these restrictions allow us to formulate a reasonably effective algorithm for learning which variables share parameters. Although it is possible to define such algorithms for the rich representation frameworks such as plates, OOBNs, or PRMs, it remains to be seen whether such algorithms can perform effectively, given that the much larger search space can introduce both computational problems and problems related to overfitting.

8. Discussion and Conclusions

We have introduced the framework of *module networks*, an extension of Bayesian networks that includes an explicit representation of *modules* — subsets of variables that share a statistical model. We have presented a Bayesian learning framework for module networks, which learns both the partitioning of variables into modules and the dependency structure of each module. We showed experimental results on two complex real-world data sets, each including measurements of thousands of variables, in the domains of gene expression and stock market. Our results show that our learned module networks have much higher generalization performance than a Bayesian network learned from the same data.

There are several reasons why a learned module network is a better model than a learned Bayesian network. Most obviously, parameter sharing between variables in the same module allows each parameter to be estimated based on a much larger sample. Moreover, this allows us to learn dependencies that are considered too weak based on statistics of single variables. These are well-known advantages of parameter sharing; the interesting aspect of our method is that we determine automatically which variables share parameters.

More interestingly, the assumption of shared structure significantly restricts the space of possible dependency structures, allowing us to learn more robust models than those learned in a classical Bayesian network setting. While the variables in the same module might behave according to the same model in underlying distribution, this will often not be the case in the empirical distribution based on a finite number of samples. A Bayesian network learning algorithm will treat each variable separately, optimizing the parent set and CPD for each variable in an independent manner. In the high-dimensional domains in which we are interested, there are bound to be spurious correlations that arise from sampling noise, inducing the algorithm to choose parent sets that do not reflect real dependencies, and will not generalize to unseen data. Conversely, in a module network setting, a spurious correlation would have to arise between a possible parent and a large number of other variables before the algorithm would find it worthwhile to introduce the dependency.

The module network framework, as presented here, has several important limitations, both from a modeling perspective and from the perspective of the learning algorithm.

¿From a modeling perspective, it is important to recognize that a module network is not a universally appropriate model for all domains. In particular, many domains do not have a natural organization of variables into higher level modules with common characteristics. In such domains, a module network would force variables into sharing dependency structures and CPDs and may result in poor representations of the underlying domain properties.

Even in domains where the modularity assumption is warranted, the module network models we presented here may not be ideal. In particular, the module network models we presented here allow each variable to be assigned to only one module. For instance, in the gene expression domain, this means that each gene is allowed to participate in only a single module. This assumption is not realistic biologically, as biological processes often involve partially overlapping sets of genes, so that many genes participate in more than one process. The framework presented in this paper, by restricting each gene to only one module, cannot represent such overlapping processes with different regulatory mechanisms. Recently (Segal *et al.*, 2003a; Battle *et al.*, 2004), we presented one possible extension to the module network framework presented in this paper, which allows genes to be assigned to several modules. The expression of a gene in a particular array is then modeled as a sum of its expression in each of the modules in which it participates, and each module can potentially have a different set of regulators. Clearly, this approach for "allocating" a variable and its observed signal among different modules is only one possible model, and one which is not appropriate to all settings. Other domains will likely require the development of other approaches.

Turning to the learning algorithm, one important limitation is our assumption that the number of modules is determined in advance. For instance, in the biological domain, the number of regulatory modules of an organism in an expression data set is obviously not known and thus determining the number of modules should be part of the regulatory module discovery task. In Section 6.1 we showed that, at least in synthetic data, where the number of modules is known, we can use the score of the model to select the correct number of modules by choosing the model with the smallest number of modules from among the highest scoring models. This observation is encouraging, as it suggests that we can extend our approach to select the number of modules automatically by adding search steps that modify the number of modules and use the model score to compare models that differ in their number of modules. However, much remains to be done on the problem of proposing new modules and initializing them.

Another important limitation of the learning algorithm is the use of heuristic search to select a single module network model. As other models may have comparable (or even better) scores to that of the final model selected, a critical issue is to provide confidence estimates for the structural relationships reported by the model. This problem is common to many learning algorithms, including standard methods for Bayesian network learning, but is particularly acute when we are trying to use the learned structure for knowledge discovery, as we do in the biology domain. In this paper, we addressed this issue only indirectly, through statistical generalization tests on held out data and through the evaluation of our results relative to the to existing annotations (e.g., of stock categories in the stock market domain).

As a more direct approach, in some cases we can make use of well known methods for confidence estimation such as *bootstrap* (Efron and Tibshirani, 1993), which repeatedly learns models from resamples of the original input data and then estimates the confidence of different features of the model based on the number of times they appear in all models learned. Such an approach was adopted for estimating the confidence in features of a Bayesian network by Friedman *et al.* (1999b) and consequently applied by Friedman *et al.* (2000b) for learning fragments of regulatory networks from expression data. An alternative approach is to use Markov Chain Monte Carlo methods to sample models from the posterior given the data. It is fairly straightforward to use the Bayesian score we devised here within a Metropolis-Hastings sampling procedure Doucet *et al.* (2001) to perform model averaging Hoeting *et al.* (1999). The challenge is to design sampling strategies that lead to rapid mixing of the Markov Chain sampler. In the context of Bayesian networks, recent results (e.g., (Friedman and Koller, 2003)) use the decomposable structure of the posterior for efficient sampling. In the context of module networks, we also need to construct efficient sampling strategies over assignment functions. Recall that the space of possible assignment functions is huge, and so *a priori* it is not clear that a simple sampling procedure (e.g., mirroring our search strategy and moving one variable at each step) will mix in reasonable time. Clearly, adapting such confidence estimation approaches for our models can greatly enhance the reliability of our results but require additional development and validation.

In this paper, we focused on the statistical properties of our method. In a companion biological paper (Segal *et al.*, 2003b), we use the module network learned from the gene expression data described above to predict gene regulation relationships. There, we performed a comprehensive evaluation of the validity of the biological structures reconstructed by our method. By analyzing biological databases and previous experimental results in the literature, we confirmed that many of the regulatory relations that our method automatically inferred are indeed correct. Furthermore, our model provided focused predictions for genes of previously uncharacterized function. We performed wet lab biological experiments that confirmed the three novel predictions we tested. Thus, we have demonstrated that the module network model is robust enough to learn a good approximation of the dependency structure between 2355 genes using only 173 instances. These results show that, by learning a structured probabilistic representation, we identify regulation networks from gene expression data and successfully address one of the central problems in analysis of gene expression data.

Acknowledgments

E. Segal, D. Koller, and N. Friedman were supported in part by NSF grant ACI-0082554 under the ITR Program. E. Segal was also supported by a Stanford Graduate Fellowship (SGF). A. Regev was supported by the Colton Foundation. D. Pe'er was supported by an Eshkol Fellowship. N. Friedman was also supported by an Alon Fellowship, by the Harry & Abe Sherman Senior Lectureship in Computer Science, and by the Israeli Ministry of Science.

References

A. Battle, E. Segal, and D. Koller. Probabilistic discovery of overlapping cellular processes and their regulation using gene expression data. In *Proceedings Eighth Annual International Conference on Research in Computational Molecular Biology (RECOMB)*, 2004.

L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth & Brooks, Monterey, CA, 1984.

W. Buntine. Operations for learning with graphical models. *Journal of Artificial Intelligence Research*, 2:159–225, 1994.

P. Cheeseman, J. Kelly, M. Self, J. Stutz, W. Taylor, and D. Freeman. Autoclass: a Bayesian classification system. In *Proceedings Fifth International Conference on Machine Learning (ML)*, pages 54–64, 1988.

J. M. Cherry, C. Ball, K. Dolinski, S. Dwight, M. Harris, J. C. Matese, G. Sherlock, G. Binkley, H. Jin, S. Weng, and D. Botstein. Saccharomyces genome database. *Nucleic Acid Research*, 26:73–79, 1998. http://genome-www.stanford.edu/Saccharomyces/.

D. M. Chickering, D. Heckerman, and C. Meek. A Bayesian approach to learning Bayesian networks with local structure. In *Proceedings Thirteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 80–89, 1997.

G. F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5:142–150, 1989.

M. H. DeGroot. *Optimal Statistical Decisions*. McGraw-Hill, New York, 1970.

A. Doucet, N. de Freitas, and N. Gordon (eds). *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, 2001.

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

G. Elidan and N. Friedman. Learning the dimensionality of hidden variables. In *Proceedings Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 144–151, 2001.

A. P. Gasch et al. Genomic expression program in the response of yeast cells to environmental changes. *Mol. Bio. Cell*, 11:4241–4257, 2000.

N. Friedman and M. Goldszmidt. Learning Bayesian networks with local structure. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 421–460. Kluwer, Dordrecht, Netherlands, 1998.

N. Friedman and D. Koller. Being Bayesian about Bayesian network structure: A Bayesian approach to structure discovery in Bayesian networks. *Machine Learning*, 50:95–126, 2003.

N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Proceedings Sixteenth International Conference on Artificial Intelligence (IJCAI)*, pages 1300–1309, 1999.

N. Friedman, M. Goldszmidt, and A. Wyner. Data analysis with Bayesian networks: A bootstrap approach. In *Proc. UAI*, pages 206–215, 1999.

N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using Bayesian networks to analyze expression data. *Journal of Computational Biology*, 7:601–620, 2000.

N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using Bayesian networks to analyze expression data. *Computational Biology*, 7:601–620, 2000.

L. Getoor, D. Koller, and N. Friedman. From instances to classes in probabilistic relational models. In *Proceedings of the ICML Workshop on Attribute-Value and Relational Learning*, 2000.

D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995.

D. Heckerman. A tutorial on learning with Bayesian networks. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer, Dordrecht, Netherlands, 1998.

J. A. Hoeting, D. Madigan, A. Raftery, and C. T. Volinsky. Bayesian model averaging: A tutorial. *Statistical Science*, 14(4), 1999.

D. Koller and A. Pfeffer. Object-oriented Bayesian networks. In *Proceedings Thirteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 302–313, 1997.

D. Koller and A. Pfeffer. Probabilistic frame-based systems. In *Proceedings National Conference on Artificial Intelligence (AAAI)*, pages 580–587, 1998.

E. Lander. Array of hope. *Nature Genetics*, 21:3–4, 1999.

H. Langseth and T. D. Nielsen. Fusion of domain knowledge with data for structural learning in object oriented domains. *Machine Learning Research*, 4:339–368, 2003.

D. Pe'er, A. Regev, G. Elidan, and N. Friedman. Inferring subnetworks from perturbed expression profiles. *Bioinformatics*, 17(Suppl 1):S215–24, 2001.

N. E. Savin. The Bonferroni and the Scheffe multiple comparison procedures. *Review of Economic Studies*, 47(1):255–73, 1980.

E. Segal, B. Taskar, A. Gasch, N. Friedman, and D. Koller. Rich probabilistic models for gene expression. *Bioinformatics*, 17(Suppl 1):S243–52, 2001.

E. Segal, A. Battle, and D. Koller. Decomposing gene expression into cellular processes. In *Proceedings Eighth Pacific Symposium on Biocomputing (PSB)*, 2003.

E. Segal, M. Shapira, A. Regev, D. Pe'er, D. Botstein, D. Koller, and N. Friedman. Module networks: Discovering regulatory modules and their condition specific regulators from gene expression data. *Nature Genetics*, 34(2):166–176, 2003.

Journal of Machine Learning Research 6 (2005) 589–613 Submitted 11/04; Revised 3/05; Published 4/05

Active Learning to Recognize Multiple Types of Plankton

Tong Luo TLUO2@CSEE.USF.EDU
Kurt Kramer KKRAMER@CSEE.USF.EDU
Dmitry B. Goldgof GOLDGOF@CSEE.USF.EDU
Lawrence O. Hall HALL@CSEE.USF.EDU
Department of Computer Science and Engineering
University of South Florida
Tampa, FL 33620, USA

Scott Samson SAMSON@MARINE.USF.EDU
Andrew Remsen AREMSEN@MARINE.USF.EDU
Thomas Hopkins THOPKINS@MARINE.USF.EDU
College of Marine Science
University of South Florida
St. Petersburg, FL 33701, USA

Editor: David Cohn

Abstract

This paper presents an active learning method which reduces the labeling effort of domain experts in multi-class classification problems. Active learning is applied in conjunction with support vector machines to recognize underwater zooplankton from higher-resolution, new generation SIPPER II images. Most previous work on active learning with support vector machines only deals with two class problems. In this paper, we propose an active learning approach "breaking ties" for multi-class support vector machines using the one-vs-one approach with a probability approximation. Experimental results indicate that our approach often requires significantly less labeled images to reach a given accuracy than the approach of labeling the least certain test example and random sampling. It can also be applied in batch mode resulting in an accuracy comparable to labeling one image at a time and retraining.

Keywords: active learning, support vector machine, plankton recognition, probabilistic output, multi-class support vector machine

1. Introduction

Recently, an advanced shadow image particle profiling evaluation recorder (SIPPER II) was developed to produce 3-bit grayscale images at 25 μm resolution. SIPPER II uses high-speed digital line-scan cameras to continuously sample plankton and suspended particles in the ocean. The high sampling rate of SIPPER II requires the development of an automated plankton recognition system. For example, in a previous study using approximately 150,000 SIPPER images from a two hour sampling deployment it took over one month to manually classify the images (Remsen et al., 2004). Also, this automated system is expected to continuously evolve from an existing model to a more accurate model created by training after adding some new labeled images into the training set. Since it is impossible to manually label all images during the time they are acquired on the ship, active learning to label the *most important images* seems attractive.

For plankton recognition, Luo et al. (2004b) developed an automated system to recognize 1-bit binary SIPPER (SIPPER I) (Samson et al., 2001) images at 50 μm resolution. Due to the instability of contour features, Luo et al. (2004b) designed several image features which did not depend heavily on contours, and applied a support vector machine (SVM) (Vapnik, 2000) to classify the feature vectors. The wrapper approach was used to do feature selection effectively reducing the feature vector from 29 to 15 features. Also, a new way of computing probabilistic output in a multi-class support vector machine was developed. Tang et al. (forthcoming) proposed several new features for the SIPPER I images and applied multilevel dominant eigenvector methods to select the best subset of features. A Gaussian classifier was employed to recognize the image features and validate the feature selection methods on selected identifiable plankton.

Recently, active learning with SVMs has been developed and applied in a variety of applications (Tong and Koller, 2000; Schohn and Cohn, 2000; Campbell et al., 2000; Sassano, 2002; Warmuth et al., 2003; Brinker, 2003; Wang et al., 2003; Onoda et al., 2003; Baram et al., 2004; Luo et al., 2004a; Nguyen and Smeulders, 2004; Park, 2004; Mitra et al., 2004a,b). The most representative and relevant work is reviewed in the following.

A similar active learning method for support vector machines (SVMs) in two class problems was independently developed by several researchers Tong and Koller (2000), Schohn and Cohn (2000), and Campbell et al. (2000). These approaches, which we term “simple”, caused the new examples closest to the decision boundary to be labeled. Tong and Koller (2000) used version spaces to analyze the hypotheses space of SVMs. It was shown that “simple” approximately found the examples which most dramatically reduced the version space. Compared to random sampling, “simple” reduced the required number of labeled images in experiments on text classification. Mitra et al. (2004a) argued that the greedy search method employed in “simple” is not robust and a confidence factor was proposed to measure the closeness of the current SVM to the optimal SVM. A random sampling factor was introduced when the confidence factor was low. Their proposed method performed better than “simple” in a set of experiments.

Roy and McCallum (2001) used a probability model to label examples which could maximize the posterior entropy on the unlabeled data set. We call this method “conf” in this paper. The “conf” method amounts to improving the current classifier’s classification confidence on the unlabeled data set. Although it initially was applied with naive bayes classifiers, it could be easily extended to any classifier with probability outputs. For example, the probability outputs of SVMs can be roughly approximated by a sigmoid function (Platt, 2000).

Baram et al. (2004) observed that there was no single winner from different active learning strategies on several data sets. They proposed dynamically selecting from four learning algorithms: “simple”, “conf”, random sampling and sampling examples furthest from the current labeled data set. The automatic selection was done by solving a multi-armed bandit problem through online learning.

Similar selection methods to label several examples at a time for two-class problems were developed by Brinker (2003) and Park (2004) and named “combined” by Brinker (2003). Based on the “simple” method, they chose to label examples which are close to the decision boundary and have the largest angles to previously selected candidates. A parameter λ was introduced to control the trade-off between the two criteria. Although Brinker (2003) did not provide a method to set the optimal value of λ, “combined” performed better than “simple” in batch mode on several data sets (when labeling several images at a time).

There are two elements of our work which differentiate it from previous approaches. The images sampled from first generation SIPPER (SIPPER I) did not have clear contours. The low image quality resulted in many unidentifiable particles, which made it important to create robust image features and handle unidentifiable particles (Luo et al., 2004b). Higher resolution SIPPER (SIPPER II) images provide relatively better quality images with clear contours. Also, 3-bit graylevel images have more texture information than binary images. As a result, handling many unidentifiable particles is no longer an issue. The higher resolution required new contour and texture features to improve recognition. Moreover, little previous work in active learning has been done with multiple class SVMs, which is required in plankton recognition. SVMs solve multiple class problems by building several two-class SVMs and a new example usually has different distances to the decision boundaries in those two-class SVMs. It is hard to use the "simple" approach because we do not know which distance to choose. In a very recent paper Mitra et al. (2004b) applied "simple" to each binary SVM in a multi-class SVM. For a multi-class problem with N binary SVMs, N examples were labeled at a time. However, this method is far from elegant. They did not suggest how to choose which example was best for all binary SVMs. It is not unusual that an "informative" example for one binary SVM is useless for other binary SVMs. The "combined" method suffers from the same problem. It is not clear which distance to minimize and which angle to maximize. The "conf" approach seems to be a natural solution for multi-class problems as long as there is a probability estimation for the output from a multi-class SVM. However, applying the "conf" approach involves estimating the decision boundary after adding each unlabeled example into the training data in each round. Suppose m is the number of unlabeled examples and c is the number of classes, "conf" needs to train a SVM cm times to decide which example to label next. Although there are several heuristics to speedup such a procedure, it remains quite computationally expensive.

A new image feature set was developed Luo et al. (2004a) which added some contour features and texture features into a previous feature set (Luo et al., 2004b). A least certainty active learning approach was proposed and evaluated for multiple class SVMs. In this paper we expand the work reported by Luo et al. (2004a) and propose a new active learning strategy for one-versus-one multi-class SVMs. After developing a probability model for multiple class SVMs, we label the example which has the smallest difference in probability between its most likely class and second most likely class. We compare our approach with other methods like random sampling and least certainty for the plankton recognition problem. To obtain the same classification accuracy, we show our approach required many fewer labeled examples than random sampling. It also outperformed the least certainty approach in terms of needed examples to reach a given accuracy level. Our proposed method can run in batch mode, labeling up to 20 images at a time, with an accuracy comparable to labeling one image at a time and retraining. In a simulation where plankton images come as a stream, active learning resulted in higher classification accuracy than random sampling.

This paper is organized as follows. Section 2 introduces our active learning approach for support vector machines and our approach to assigning a classification probability for a multi-class support vector machine. Experimental results are presented in Section 3. Finally we summarize our work and propose some ideas for future work in Section 4.

2. Active Learning Approach with Multi-Class Support Vector Machines

A soft margin support vector machine was used in this work. A probability model has been added to the support vector machine to help evaluate multi-class decision problems. The probability model was used in the development of an active learning model.

2.1 Support Vector Machines

Support vector machines (SVMs) (Vapnik, 2000) have received increasing attention recently and have been shown to have very good accuracy for pattern recognition, text classification, etc. (Cristianini and Shawe-Taylor, 2000).

SVMs first map the data into a higher dimension feature space with $\phi(x)$, then use a hyperplane in that feature space to separate the data into two classes. In the feature mapping stage, the kernel $k(x,y) = \langle\phi(x)\cdot\phi(y)\rangle$ is used to avoid explicit inner product calculation in the high-dimensional feature space. C-SVM (Vapnik, 2000), a typical example of soft margin SVMs, is described in the following.

Given m examples: $x_1, x_2, ..., x_m$ with class label $y_i \in$\{-1,1\}.

C-SVM:

$$\text{minimize } (\frac{1}{2}\langle w,w\rangle + C\sum_{i=1}^{m}\xi_i) \tag{1}$$

$$\text{subject to: } y_i(\langle w,\phi(x_i)\rangle + b) \geq 1-\xi_i, \tag{2}$$

where w is normal to the class separating hyperplane, C is a scalar value that controls the trade off between the empirical risk and the margin ($\frac{2}{|w|}$) , ξ_i is the slack variable to handle non-separable examples, b is a scalar value, and $C,\xi_i > 0$.

With Lagrange multipliers, the constraint optimization problem in Eq. (1) and (2) can be solved. The decision function is

$$f(x) = \sum_i \alpha_i y_i k(x_i,x) + b,$$

where α_i is a Lagrange multiplier. Both α_i and b are scalar values.

The Karush-Kuhn-Tucker condition of the optimal solution to Eq. (1) and (2) is

$$\alpha_i(y_i(\langle w,\phi(x_i)\rangle + b) - 1 + \xi_i) = 0.$$

α_i is nonzero only when

$$y_i(\langle w,\phi(x_i)\rangle + b) = 1-\xi_i. \tag{3}$$

In this case the x_i contributes to the decision function and is called a support vector (SV).

We applied the one-vs-one approach to extend SVMs to multiple class problems. All possible groups of 2 classes were used in building binary SVMs. In the N class case, we will build $\frac{N(N-1)}{2}$ binary SVMs. We chose the one-vs-one method because it showed superior accuracy in several experiments (see Hsu and Lin, 2002) over other multi-class methods–one-vs-all (Vapnik, 2000) and the decision directed acyclic graph (Platt et al., 2000).

2.2 Assigning Probability Values in Support Vector Machines

A probability associated with a classifier is often very useful and it provides an indication of how much to believe the classification result. The classification probability can be used to develop an active learning strategy for a multi-class SVM.

In Platt (2000) the sigmoid function was introduced as a probability model to fit $P(y=1|f)$ directly, where f is the decision function of the binary SVM. The parametric model is shown in Eq. (4).

$$P(y=1|f)=\frac{1}{1+exp(Af+B)}, \tag{4}$$

where A and B are scalar values, which are fit by maximum likelihood estimation.

A method to estimate classification probability for a series of pairwise classifiers was proposed by Hastie and Tibshirani (1998). Given the estimated probability for each binary classifier(P_{pq}), the probability of being class p in a binary classifier (class p vs. class q), they minimized the average Kullback-Leibler distance between P_{pq} and $\frac{P(p)}{P(p)+P(q)}$, where $P(p)$ and $P(q)$ were the probabilities of a given example belonging to classes p and q, respectively. An iterated algorithm was given to search for $P(p)$. Following this line of the work, Wu et al. (2004) developed two new criteria for the goodness of the estimated probabilities and applied their method to multi-class SVMs. Their approach has three steps to get the probability estimation. First, a grid-search is used to determine the best SVM parameters (C, g) based on a k-fold cross validation accuracy, where C is the regularization constant in Eq. (1) and g is the kernel parameter in the kernel function k. Second, with the optimal (C, g) found in the first step, A and B were fit individually for each binary SVM. Third, a constrained quadratic programming method was used to optimize the criteria they proposed.

However, this approach is time consuming. The second step involves estimating $N(N-1)$ parameters for SVMs using a one-vs-one approach. The third step requires quadratic programming to solve N variables for each example. On a data set with m examples, this step needs to run m times. Another issue was that the SVM parameters (C, g) were estimated based on accuracy and thus might not be good for probability estimation in the following two steps.

In real-time plankton recognition, the probability computation needs to be fast since retraining the probability model will be frequently needed as more plankton images are acquired on a cruise.

We (Luo et al., 2004b) developed a practical approximation method to compute the probability value, while avoiding expensive parameter fitting. By normalizing the real valued output $f(x)$ from each binary SVM, the probability model assumes the same A for all binary SVMs. Also, our approach can optimize SVM parameters (C, g) together with the probability parameter A simultaneously using a log-likelihood criterion.

1. We assume $P(y=1|f=0)=P(y=-1|f=0)=0.5$. This means that a point lying on the decision boundary will have a 0.5 probability of belonging to each class. This allows the elimination of B.

2. Since each binary SVM has a different margin, a crucial criterion in assigning the probability, it is not fair to assign a probability without considering the margin. Therefore, the decision function $f(x)$ is normalized by its margin in each binary SVM. The probability model of SVMs is shown in (5) and (6). P_{pq} represents the probability output for the binary SVM on class p vs. class q, class p is +1 and class q is -1. We added the negative sign before A to ensure A is positive:

$$P_{pq}(y=1|f) = \frac{1}{1+exp(\frac{-Af}{||w||})}, \tag{5}$$

$$P_{pq}(y=-1|f) = 1 - P_{pq}(y=1|f) = P_{qp}(y=1|f). \tag{6}$$

3. Assuming $P_{pq}, q = 1,2,...$ are independent, the final probability for class p is computed as follows:

$$P(p) = \prod_{q}^{q \neq p} P_{pq}(y=1|f). \tag{7}$$

   Normalize $P(p)$ to make $\sum_p P(p) = 1$.

4. Output $\hat{y} = arg\ max_p P(p)$ as the prediction.

(A, C, g) are determined through numeric search based on the cost function L from (8), where t_i is the true class label of x_i:

$$L = -\sum_i logP(t_i). \tag{8}$$

Although it is arguable whether P_{pq} and P_{pk} are really independent since P_{pq} and P_{pk} are both estimated using data from class p, the one-vs-one approach does not suffer much from any dependence. Consider differentiating examples from three classes (p, q and k). If a classifier is built for classes p and q with another built for p and k, there is clearly a relationship but one class is different. So, following this type of argument the classifiers will have a weak dependence. Knowing there is only a weak dependence between P_{pq} and P_{pk}, Eq. (7) provides a reasonable approximation.

Grid-search can be used to find the optimal (C, g, A) on the initial small labeled data set. It has the potential to be run in parallel to significantly reduce the computation time. If we want to update the probability model after adding more labeled images, we can fix C and g, and only search for A. As a result, it is very fast to update the probability model. Moreover, we directly optimize (C, g, A) together by minimizing the negative log-likelihood function in Eq. (8). Normalizing f by its margin and assuming the same A for each binary SVM trades off some flexibility to gain a regularization effect and speedup since it restricts the otherwise big ($N(N+1)$) parameter space. Experiments for this probability model were done on SIPPER images by Luo et al. (2004b).

2.3 Active Learning for Multi-Class SVMs

The least certainty active learning approach for SVMs (Luo et al., 2004a), which makes use of the estimated probability described in Section 2.2, provides good performance in multi-class SVM classification. The idea for it can be traced back to Lewis and Gale (1994), who used “uncertainty sampling” to label the examples with the least classification certainty. We call the least certainty approach for SVMs by Luo et al. (2004a) “LC”. In this paper, we propose another active learning approach–“breaking ties” (BT). The idea of “BT” is to improve the confidence of the multi-class classification. Recall in a multi-class SVM with probability outputs, we assign the class label of x to $argmax_p P(p)$. Suppose $P(a)$ is the largest and $P(b)$ is the second largest probability for example x, where a, b are class labels. “BT” tries to improve the $P(a) - P(b)$. Intuitively, improving the value of $P(a) - P(b)$ amounts to breaking the tie between $P(a)$ and $P(b)$, thus improving the classification confidence. The difference between “LC” and “BT” is that “LC” tries to improve the value of $P(a)$ rather than $P(a) - P(b)$.

The two algorithms work as follows:

1. Start with an initial training set and an unclassified set of images.
2. A multi-class support vector machine is built using the current training set.
3. Compute the probabilistic outputs of the classification results for each image on the unclassified set. Suppose the class with highest probability is a and the class with second highest probability is b. Record the value of $P(a)$ and $P(b)$ for each unclassified image.
4. If LC: Remove the image(s) from the unclassified set that have the smallest classification confidence, obtain the correct label(s) from human experts and add the labeled image(s) to the current training set.
5. If BT: Remove the image(s) from the unclassified set that have the smallest difference in probabilities between them ($P(a) - P(b)$) for the two highest probability classes, obtain the correct label from human experts and add the labeled image(s) to the current training set.
6. Go to 2.

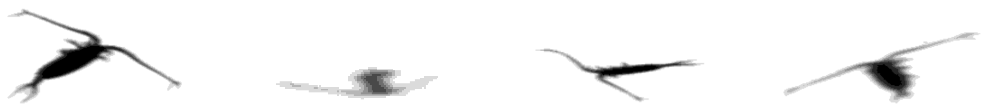
3. Experiments

The experimental data set consisted of 8440 SIPPER II images selected from the five most abundant types of plankton: 1688 images from each type of plankton. There were 1000 images (200 each type of plankton) randomly selected as the validation set used in the active learning experiments. The remaining 7440 image were used as the training set and to simulate the unlabeled image pool. Figures 1(a) to 1(e) are typical examples of the images produced by SIPPER II for the five most abundant plankton classes.

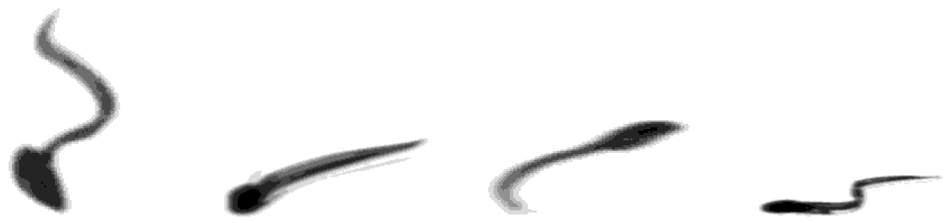
Given this new higher resolution data, 49 image features were developed (Luo et al., 2004b; Luo, forthcoming) consisting of: moment invariants, weighted moment invariants, granulometric features, Fourier descriptor, texture features and several domain specific features.

The Libsvm (Chang and Lin, 2001) support vector machine software was modified to produce probabilistic outputs. Rifkin and Klautau (2004) argued the one-vs-all approach was essentially as good as other voting algorithms, however, without postprocessing binary SVMs, we observed the one-vs-one approach provided better accuracy and required less training time than the one-vs-all approach in our previous experiments (Luo et al., 2004b). Also, when updating models with several more labeled examples in N class problems, the one-vs-one approach only requires the update of N binary SVMs built with a portion of the data, while the one-vs-all approach requires the update of N binary SVMs built with all the labeled data. Therefore, the one-vs-one approach was used in our experiments. In all experiments the Gaussian radial basis function (RBF) was used as the kernel: $k(x,y) = exp(-g\|x-y\|^2)$ where g is a scalar value.

The optimal feature subset was determined beforehand by our wrapper based feature selection method (Luo et al., 2004b) after the best (g, C) parameters were found by 5-fold cross validation.

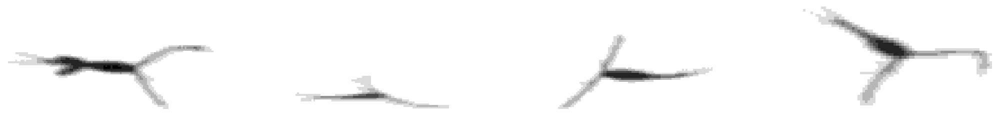


(a) Calanoid copepod



(b) Larvacean

(c) Marine snow



(d) Oithona copepod

(e) Trichodesmium

Figure 1: Five most abundant types of plankton from SIPPER II

We started off with all 49 image features and systematically eliminated features using best first search and later beam search. Five-fold cross validation on 80% of the training data was used to select the best feature subset for each feature set size. Then the best feature subsets were tested on the remaining 20% of the training set. This feature selection method is described in detail by Luo et al. (2004b). As a result, 17 out of 49 features were selected. In all the active learning experiments, we used the best 17 feature subset instead of the 49 feature set.

In the kind of problem embodied by plankton recognition, there is only a small amount of initial training data available. Therefore, the best parameter set for the probability model would be estimated from a small data set. The parameters (g, C, A) were optimized by performing a grid-search across a randomly selected 1000 images consisting of 200 images per class. We believe the parameters were obtained from a relatively small set of data and were reasonably stable. A five-fold cross validation was used to evaluate each combination of parameters based on the loss function L from (8). The parameters (g, C, A) were varied with a certain interval in the grid space. Since the parameters are independent, the grid-search ran very fast in a parallel implementation. The values of $g = 0.04096$, $C = 16$, and $A = 100$ were found to produce the best results.

We began with N randomly selected images per class as the initial training set. A series of retrainings were done for the two active learning methods and with random sampling. Each experiment was performed 30 times and the average statistics were recorded. Instead of exhausting all of the unlabeled data set, we only labeled 750 more images in each experiment because exhausting all unlabeled data was not a fair criterion for comparing between different sample selection algorithms. For example, active learning labeled the most "informative" new examples, which were available in the beginning of the experiment. As more "informative" examples were labeled, only "garbage" examples were left unlabeled in the late stages of the experiment. The term "garbage" examples means the examples correctly classified by the current classifier and far from the decision boundary. Therefore, "garbage" examples have no contribution to improving the current classifier. In contrast to active learning, random sampling labeled average "informative" examples throughout the whole experiment. It surely would catch up with active learning in the later stages when active learning only had "garbage" examples to label. Moreover, when the plankton recognition system is employed on a cruise, the unlabeled images come like a stream. The nature of such an application prevents one from exhausting all the unlabeled images because the time required to label them is prohibitive. Therefore, it made more sense to compare different algorithms in the early stage of the experiment when the unlabeled data set is not exhausted. To get an idea of the upper limit on the classification accuracy, we built a SVM using all 7400 training images. Its prediction accuracy was 88.3% on the 1000 held-out data set.

Several variations of the procedure described above were performed. We varied both the number of initial labeled images per class (IIPC) and the number of images selected for labeling at each retraining step (IPR).

3.1 Experiments with IPR=1, IIPC Varied

Figures 2–5 show the experimental results of active learning methods using different IIPC values. A paired-t test was used to test if there exists a statistically significantly difference. We used standard error for the error bars in the figures because the denominator of t test is in the form of standard error.

As shown in Figure 2, with only 10 images per class in the initial training sets we started off with rather poor accuracy (64.6%). At p=0.05, “BT” is statistically significantly more accurate than “LC” and both active learning methods are statistically significantly more accurate than random sampling. At 81% accuracy, random selection required approximately 1.7 times the number of newly labeled images compared to “BT”.

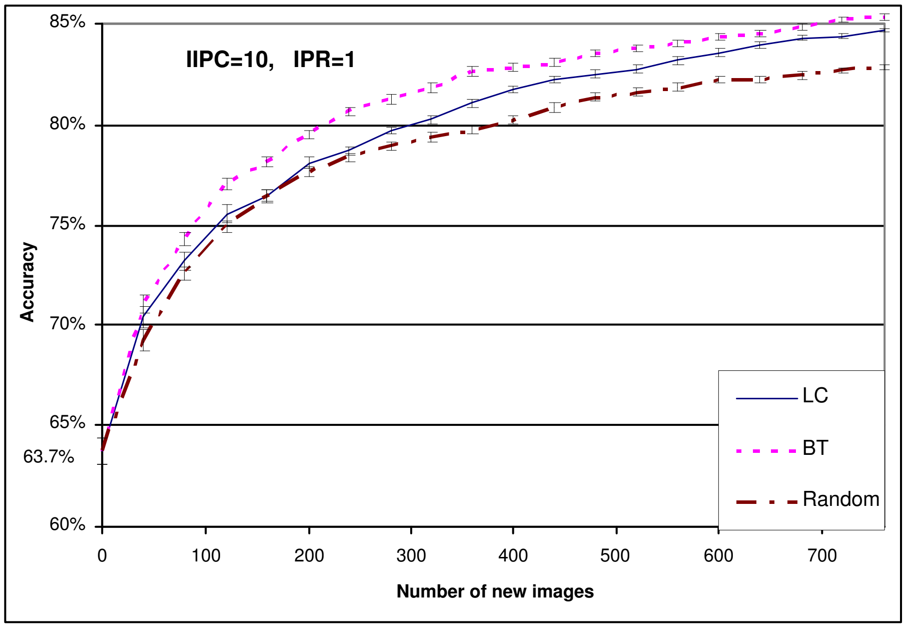
Active learning is designed to label the most “informative” new images, thus improving a newly trained classifier. In SVMs, the decision boundary is represented by support vectors (SVs). In general, an effective active learning approach finds more SVs than random sampling. Figure 2 also shows the average number of SVs versus the number of images added into the initial training set from the 30 runs. Active learning resulted in many more SVs than random sampling. Also, the slope of both active learning curves is about 0.9, which means that 90% of the labeled images turn out to be SVs. Our active learning approach efficiently captured support vectors. We note that a high slope of the support vector curve is not a sufficient condition for effective active learning because there are many SVs to be added into the current model and different SVs lead to different improvements. Ideally, a very effective active learning method discovers the SVs which provide the most improvement to the current model. In contrast, an active learning method, which always finds the SVs misclassified by the current classifier and far from its decision boundary, may result in poor performance because such SVs are very likely to be noise. Therefore, we cannot compare active learning methods based only on slight differences in the support vector curves.

With 50 IIPC in the initial training set as shown in Figure 3, the initial accuracy was 77%. Compared to 10 IIPC, the accuracy for both active learning approaches improved faster than random sampling. At the 81% accuracy level, random sampling required about 2.5 times and 1.7 times the number of images compared with using “BT” and “LC”, respectively. The slopes of support vector curves for active learning are higher than those of random sampling. Also, “BT” again outperformed “LC”, however, it is not as obvious as with IIPC=10.

In Figures 4 and 5, the initial accuracy was greater than 80% when using 100 and 200 initial images from each class, and active learning was very effective. Random sampling required more than 3 times the number of images to reach the same level of accuracy as both active learning approaches. The two active learning methods effectively capture many more SVs than random sampling. Also, our newly proposed active learning approach, “BT”, requires less images to reach a given accuracy than “LC” after adding 450 labeled images. Before adding 450 labeled images, however, “BT” performs similarly to “LC”.

It seems reasonable that the accuracy of the initial classifier affects the performance of active learning and random sampling. Active learning greedily chooses the most “informative” examples based on the previous model. So an un-informed model tends to be provide less important examples for labeling. Hence, their addition may not help to improve the classifier accuracy much. While random sampling provides the classifier with average “informative” examples whatever the initial classifier accuracy. Therefore, if the initial classifier helps active learning to choose examples more informative than average (random sampling), active learning will result in a more accurate classifier with fewer labeled examples. The better the initial classifier, the more labeling effort is saved.

When comparing the two active learning methods, “BT” outperformed “LC” under all four starting conditions. However, the difference in accuracy between them was insignificant as the initial classifier became more accurate. The justification is that an accurate initial classifier allows for less error reduction using active learning. “BT” improved the accuracy by more than 20% when IIPC=10 while it boosted the accuracy by less than 4% when IIPC=200. Therefore, as the amount



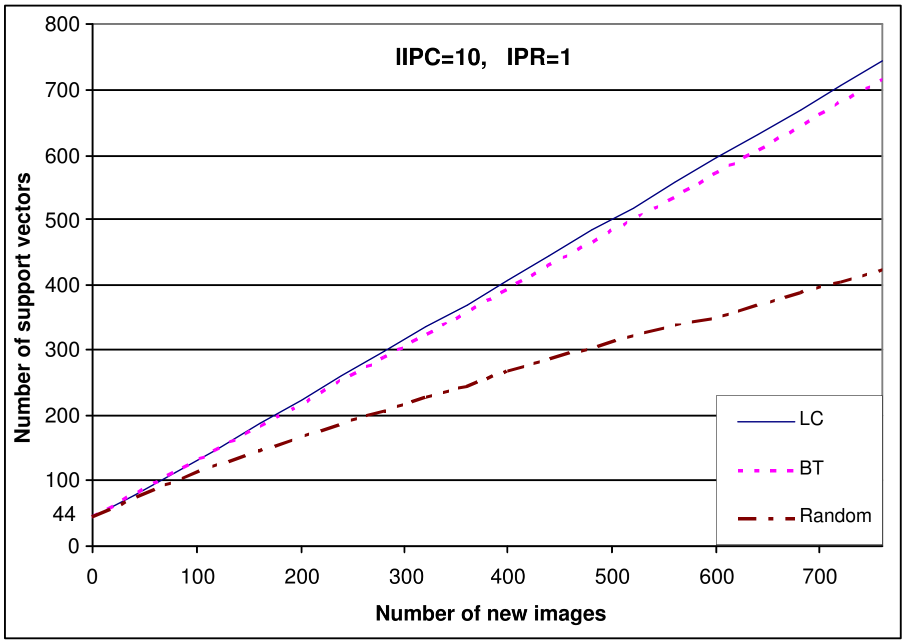
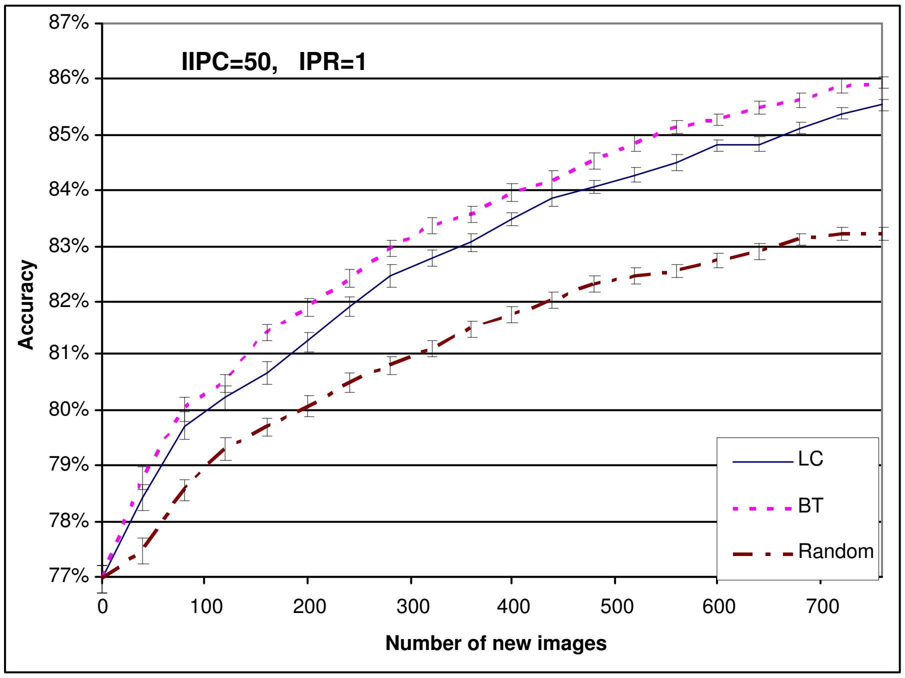


Figure 2: Comparison of active learning and random sampling in terms of accuracy and number of support vectors: initial training images per class are 10, one new labeled image added at a time. The error bars represent the standard errors.



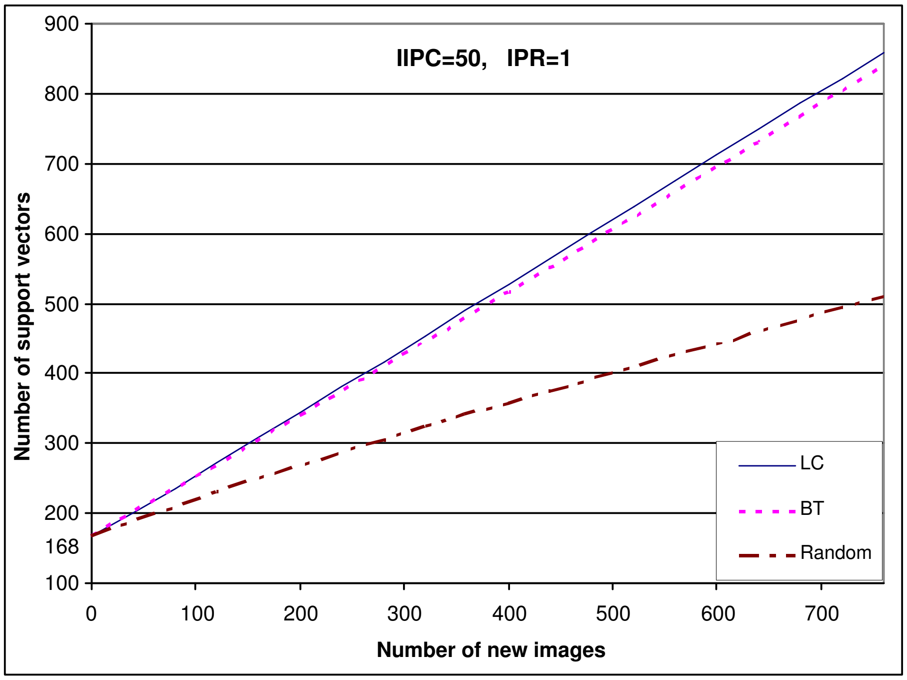
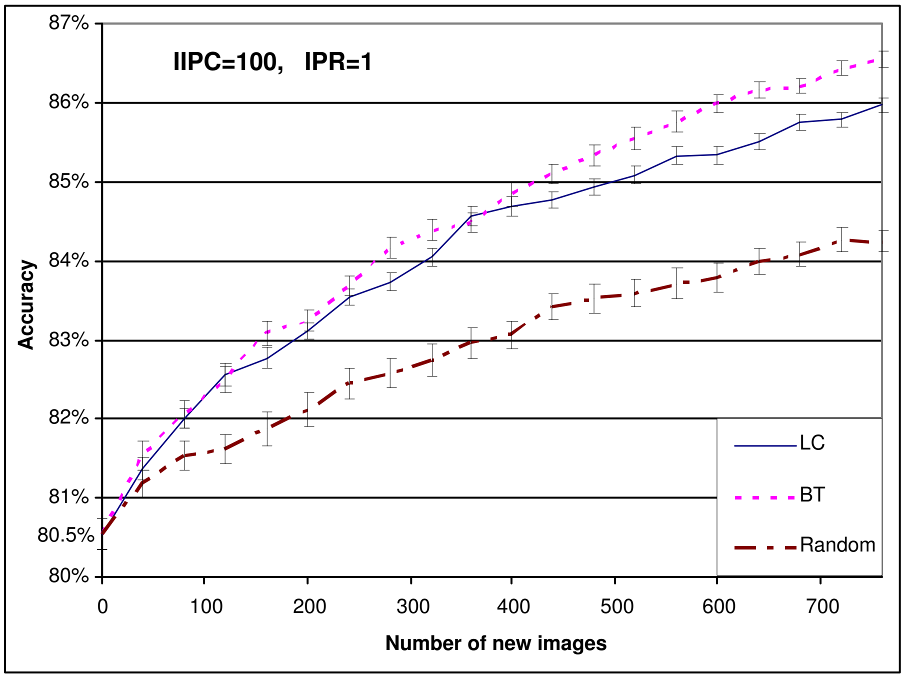


Figure 3: Comparison of active learning and random sampling in terms of accuracy and number of support vectors: initial training images per class are 50, one new labeled image added at a time. The error bars represent the standard error.



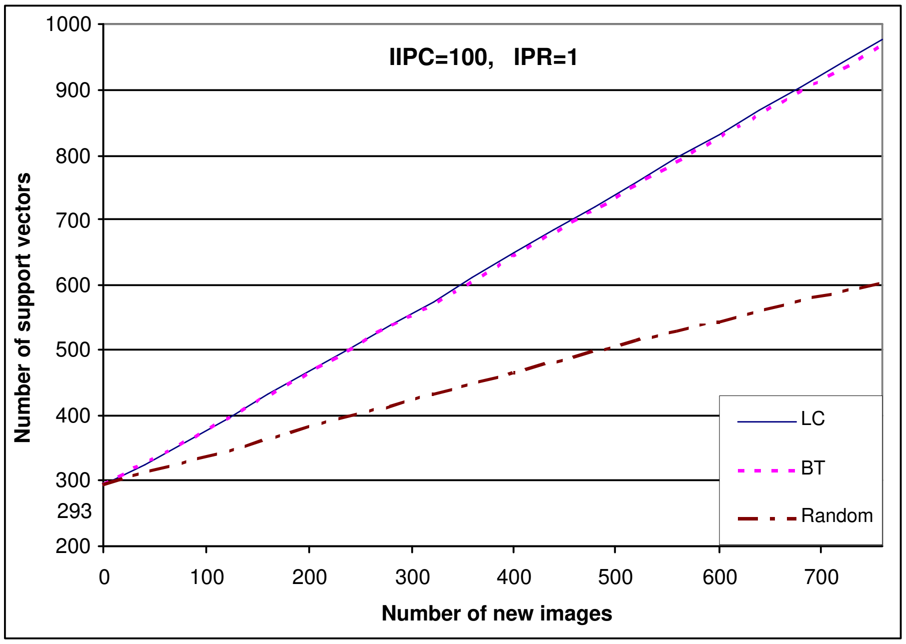
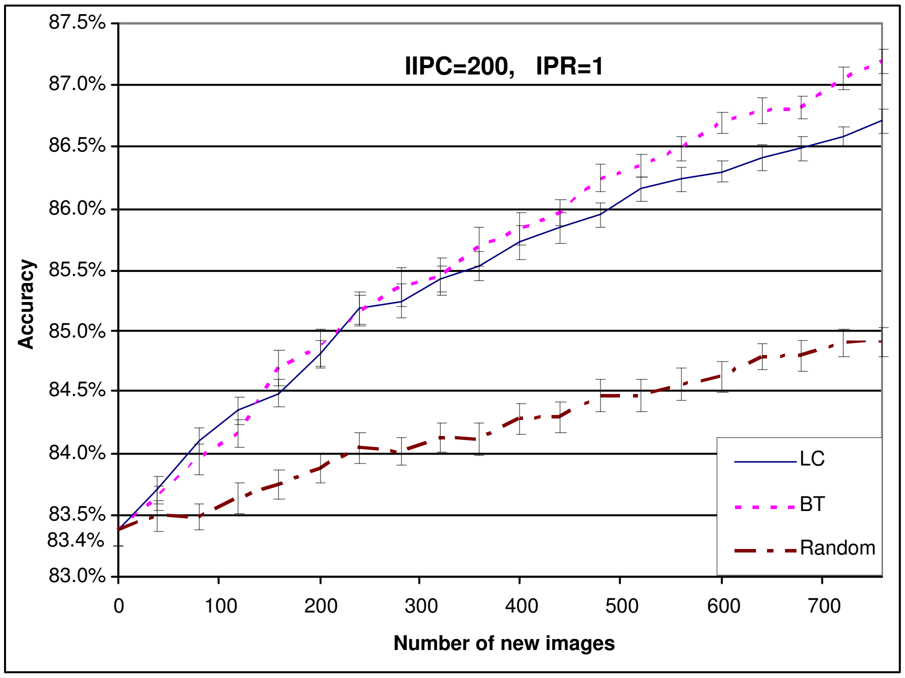


Figure 4: Comparison of active learning and random sampling in terms of accuracy and number of support vectors: initial training images per class are 100, one new labeled image added at a time. The error bars represent the standard error.



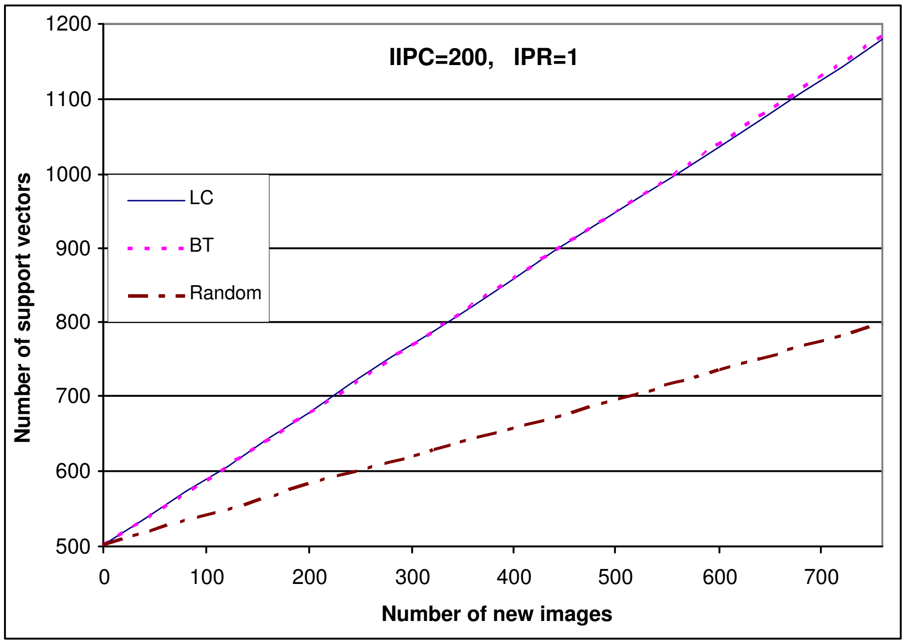


Figure 5: Comparison of active learning and random sampling in terms of accuracy and number of support vectors: initial training images per class are 200, one new labeled image added at a time. The error bars represent the standard error.

of available accuracy improvement was small, the difference in accuracy between the two active learning methods became insignificant.

3.2 Varying the IPR

One might expect that, in actual practice, more than one image would typically be labeled and added to the training set for retraining. It is convenient for an expert to label several images instead of one at a time. Also, given the total number of newly labeled images is U, it is approximately k times faster if we label k images at a time because it only requires a new model be learned $\frac{U}{k}$ times. Although an incremental SVM training algorithm was proposed by Cauwenberghs and Poggio (2000) to reduce the retraining time, model updating by labeling one image at a time was still quite time consuming, especially when many images are to be labeled. Therefore, we expected active learning to be effective even when adding several labeled images at a time.

The active learning method "BT" was good for adding only one "informative" example at a time, however there was no guarantee that adding several examples at a time would still favor "BT". The reason is that adding one "informative" example will update the model, which in turn changes the criterion for the next "informative" example. Therefore, the most "informative" example set is different from simply grouping several most "informative" examples together. However, such an optimal example set is very hard to compute. Therefore, we expect grouping several most "informative" examples together is a reasonable approximation of the optimal example set, or at least is superior to randomly selecting several examples.

Figures 6 to 9 show the experimental results using "BT" by varying IPR for each IIPC. In all the experiments, the IPR was varied from 1 to 50. We only show the error bars for random sampling because adding error bars to "BT" will make the graph too busy. We again used a paired-t test to compare "BT" with random sampling. Somewhat surprisingly, classification accuracy with large IPRs is almost as good as with small IPRs although a very large IPR (IPR=50) resulted in slightly less accurate classifiers than a small IPR in many cases. In all situations, even a large IPR (up to 50) enabled "BT" to result in a statistically significantly more accurate classifier than random sampling at p=0.05. These results indicate that our active learning approach "BT" can run in batch mode, labeling tens of examples at a time, to achieve speedup with at most a little compromise in accuracy.

3.3 Other Experiments

We experimented with "BT" in a streaming data simulation. In this experiment, unlabeled data was treated as a stream with only a block of unlabeled data available at a given time. The algorithm selectively labeled data within this block. Then all unlabeled data in the block was discarded and a new data block was pulled from the stream. In our experiment, we started off with 10 labeled images randomly selected from each class (IIPC=10). The size of data block was 100. From each data block, 10 images were selected to label at a time (IPR=10). Figure 10 shows a comparison of "BT" with random sampling in the stream setup. We also show "BT" without streaming for comparison. All the curves were averaged over 30 runs in which the order of the data blocks was randomized.

"BT" in a streaming simulation performed very well. At p=0.05, it was more accurate than random sampling and was as accurate as "BT" in a non-streaming setup. This experiment indicates that "BT" works well in "data streaming" situations.

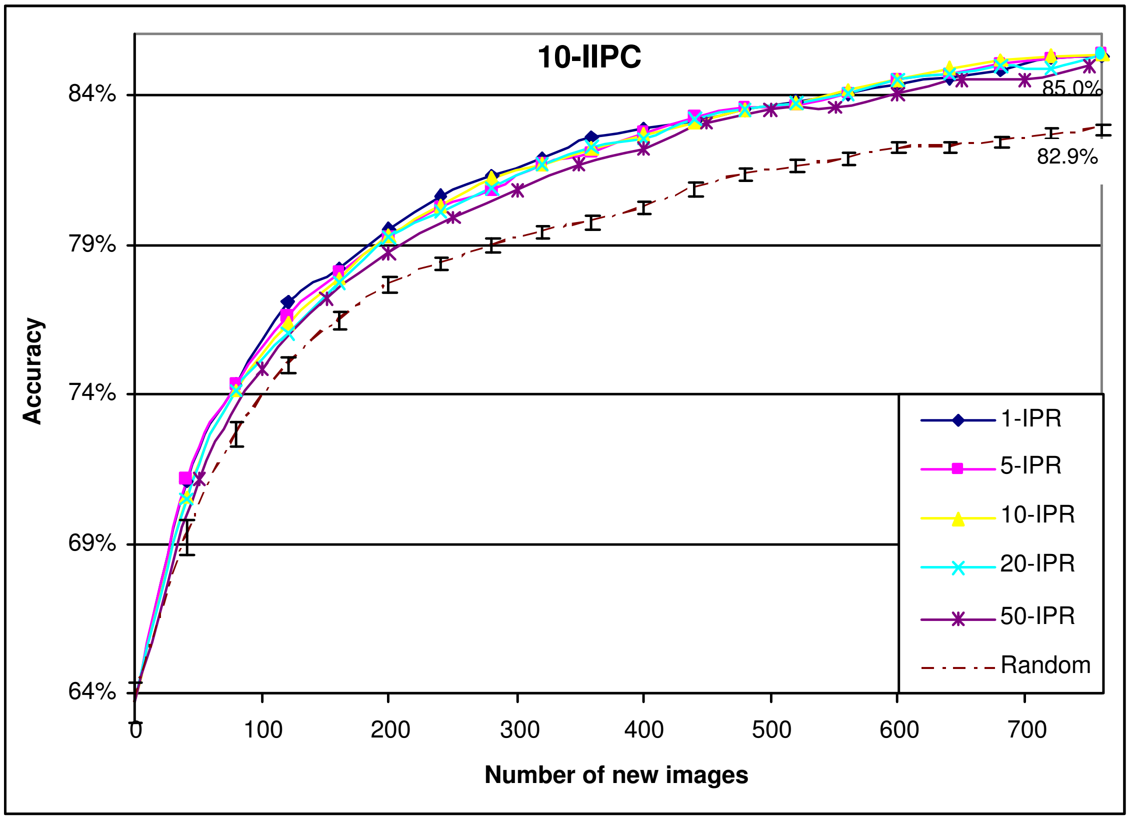


Figure 6: Comparison of active learning and random sampling in terms of accuracy with different IPR: initial training images per class are 10. Standard error bars are on the random sampling curve.

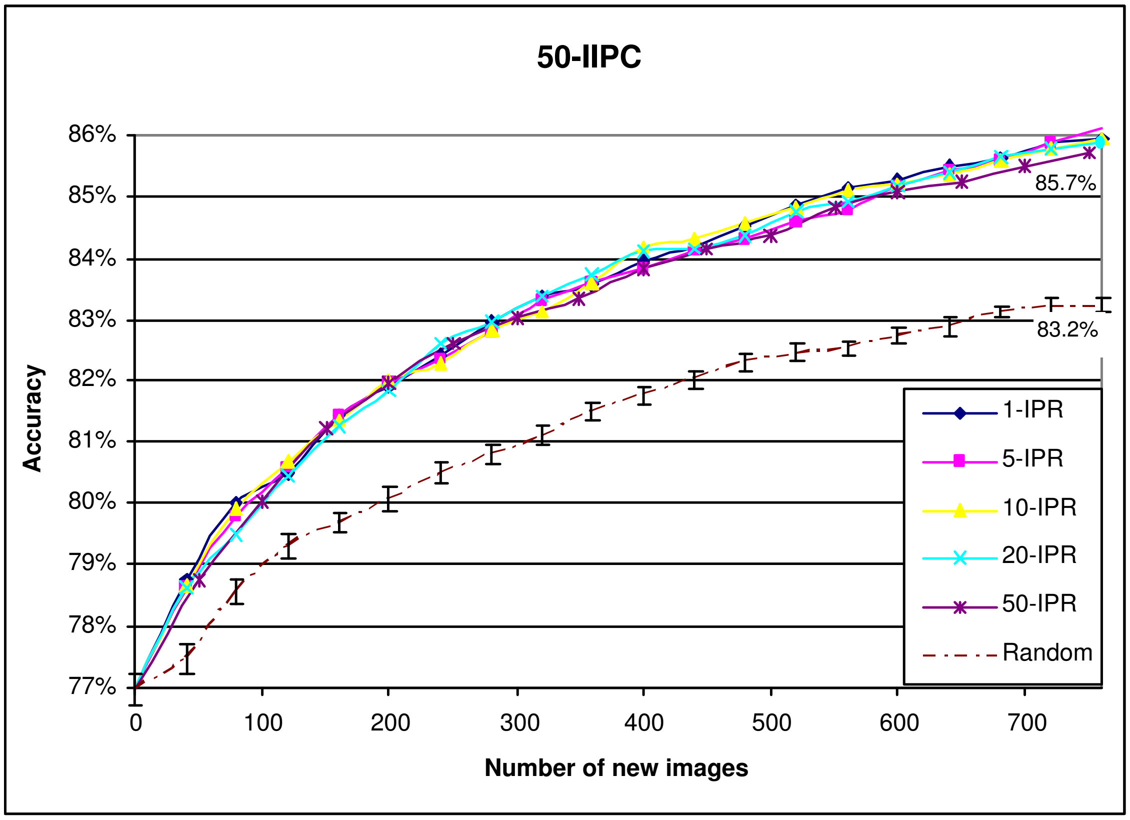


Figure 7: Comparison of active learning and random sampling in terms of accuracy with different IPR: initial training images per class are 50. Standard error bars are on the random sampling curve.

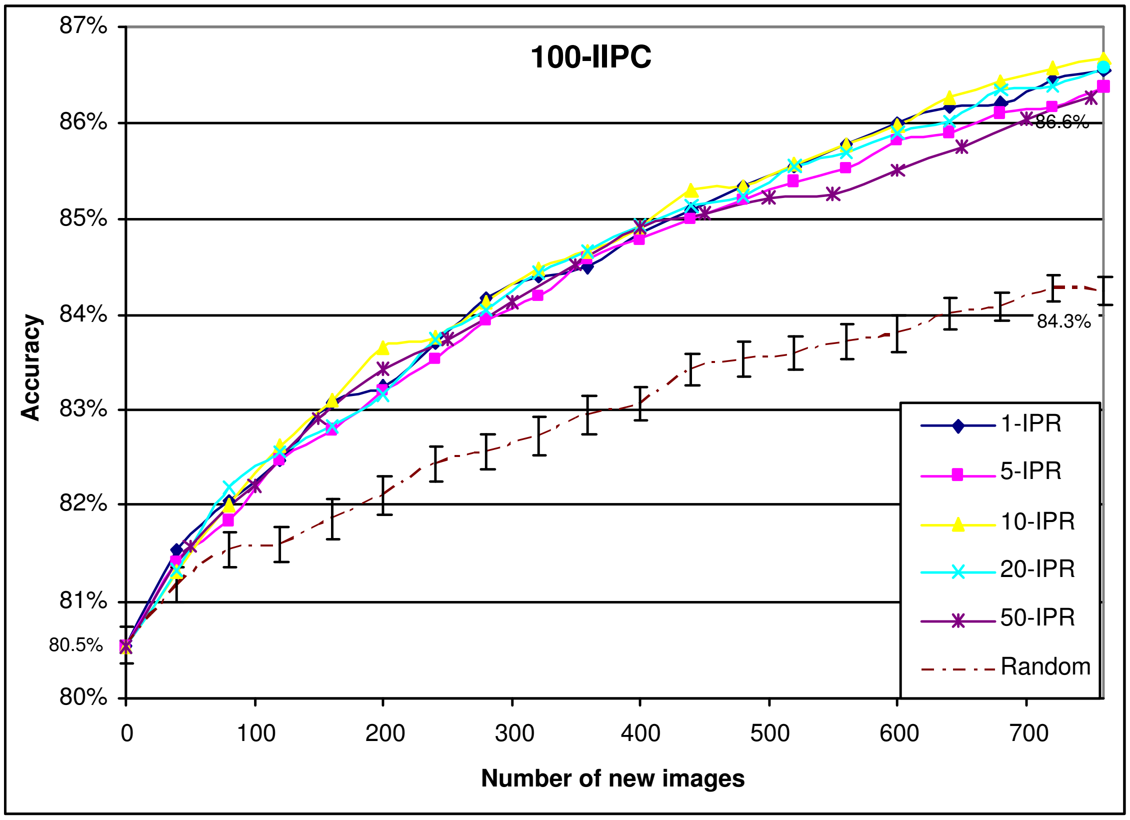


Figure 8: Comparison of active learning and random sampling in terms of accuracy with different IPR: initial training images per class are 100. Standard error bars are on the random sampling curve.

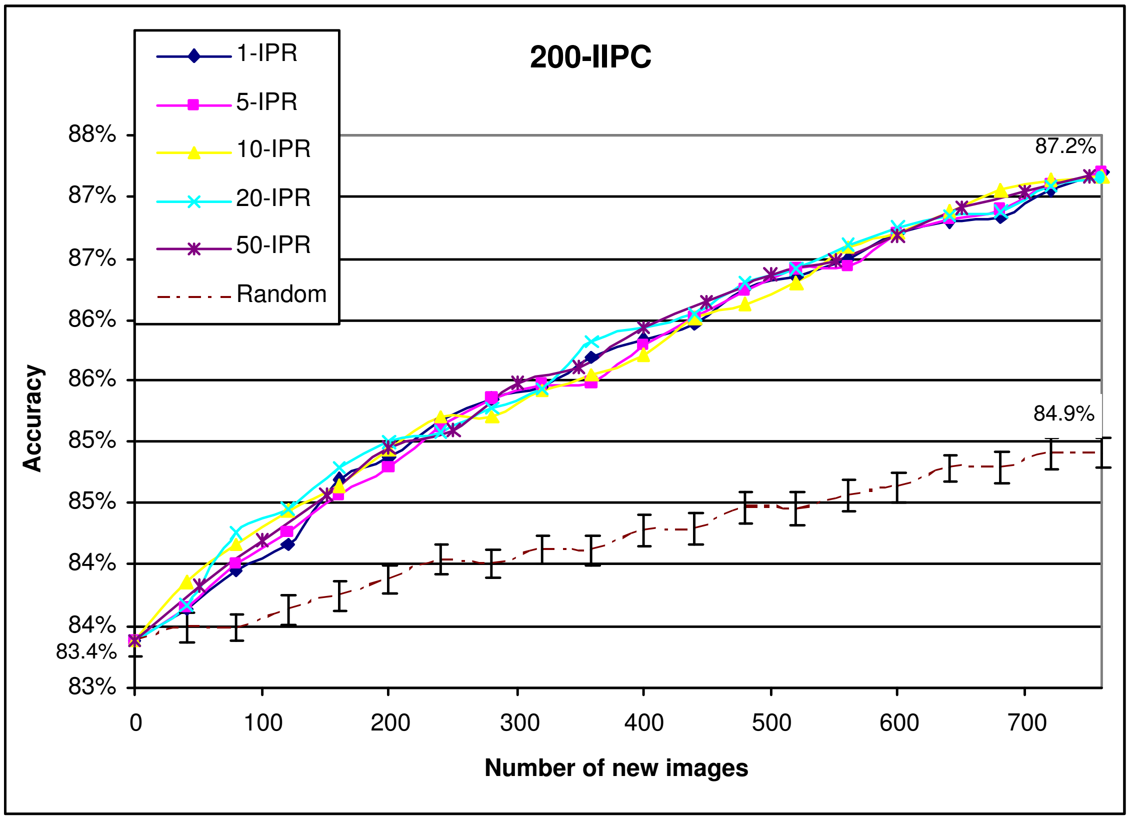


Figure 9: Comparison of active learning and random sampling in terms of accuracy with different IPR: initial training images per class are 200. Standard error bars are on the random sampling curve.

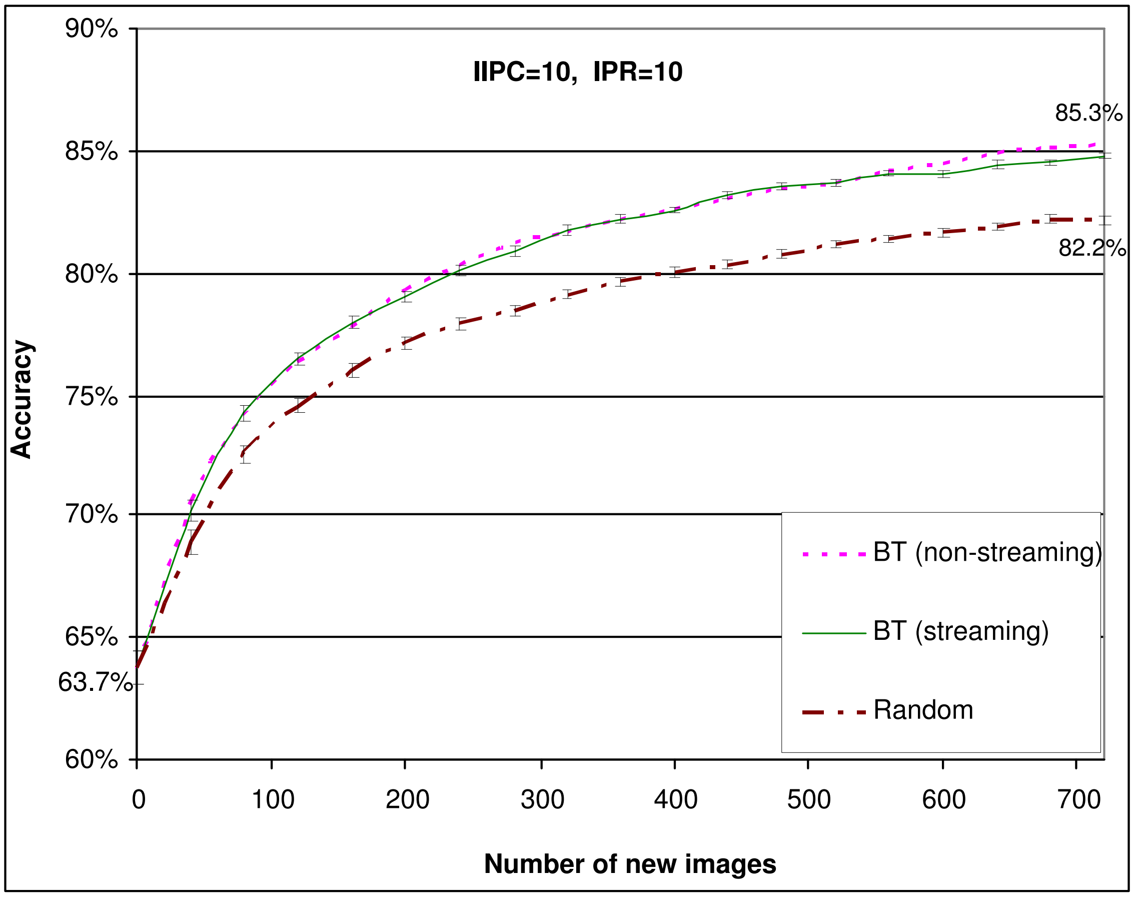


Figure 10: Comparison of active learning and random sampling in a data streaming simulation: initial training images per class are 10, 10 newly labeled images added at a time. The error bars represent the standard error.

In the previous experiments, all the classes have equal priors. We investigated how “BT” performed when the priors for each type of plankton were different. It should be noted that under the unequal prior condition, our probabilistic interpretation is different from traditional probability model in two aspects. First, the sigmoid function directly estimates $P(y|f)$ rather than computing the probability density function $P(f|y)$. Therefore, we can not simply apply Bayes rule to include the priors. Second, our probability model is built on top of a trained SVM. When the class distribution becomes skewed, the decision function of a SVM will vary accordingly. As a result, our probability model (built on the SVM) implicitly incorporates the unequal prior information. In our experiment, we selected three types of plankton with unequal priors. The ratio among the three types of plankton was 1:2:4. Both the unlabeled pool and the held-out data set had 200 larvacean, 400 oithona and 800 copepod images. We started off with a total of 30 initial labeled images randomly taken from the above distribution and labeled one image at a time per retraining (IPR=1). The initial 30 labeled image set consisted of 4 larvacean, 9 oithona, and 17 copepod images. The initial labeled images used unequal priors because they would typically be randomly sampled from the unlabeled pool and therefore would likely have the same class distribution.

Figure 11 shows the experimental results for the unequal prior experiment. When the distribution of different plankton was skewed, “BT” still outperformed random sampling. “BT” was statistically significantly more accurate than random sampling at p=0.05.

4. Discussion and Conclusions

This paper presents an active learning approach to reduce domain experts’ labeling efforts in recognizing plankton from higher-resolution, new generation SIPPER II images. It can be applied to any data set where the examples will be labeled over time and one wants to use the learned model as early as possible. The “breaking ties” active learning method was proposed and applied to a multi-class SVM using the one-vs-one approach on newly developed, image features extracted from gray-scale SIPPER images. The experimental results showed that our proposed active learning approach successfully reduced the number of labeled images required to reach a given accuracy level when compared to random sampling. It also outperforms the least certainty approach previously proposed by us. The new approach was also effective in batch mode, allowing for labeling up to 20 images at a time with classification accuracy which was similar to that achieved when labeling one image at a time and retraining. This results in a significant speedup in the training phase. In the following, we address and discuss several active learning in SVM issues which deserve further exploration.

One critique of active learning is the overhead related to searching for the next candidate to label. Random sampling just selects an example to label at random, but active learning needs to evaluate every unlabeled example. This overhead becomes significant when the unlabeled data set is very large. A simple solution would be random subset evaluation. Each time one searches for the next candidate example to label, instead of evaluating the entire unlabeled data set, can only evaluate a randomly drawn subset. We indicate without proof here that for IPR=1, we needed to sample 59 examples, which provided 95% probability confidence that the best candidate from the 59 example subset is superior to 95% data from the total unlabeled set (see Schölkopf and Smola, 2002, chap. 6.5). Also, the experiment with a “data streaming” simulation indicated “BT” worked well when the evaluation and active learning was performed on a small subset of the data.

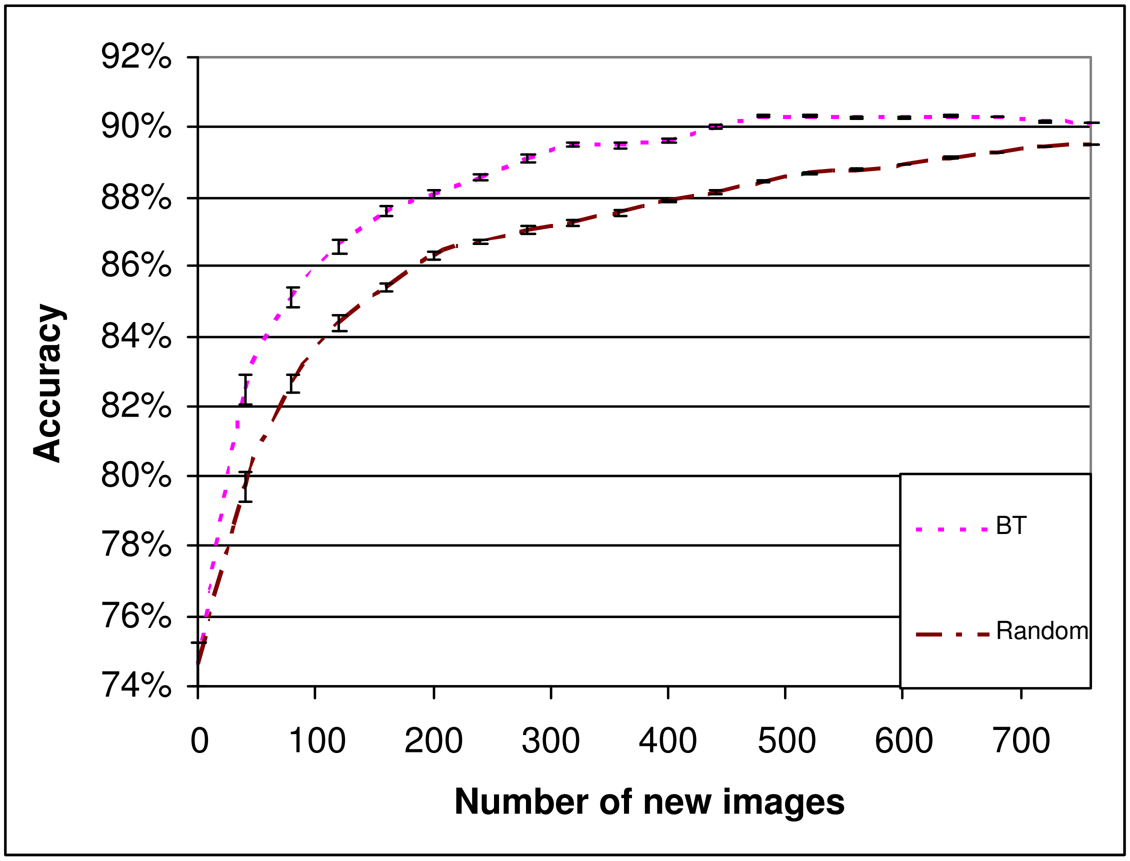


Figure 11: Unequal prior experiment. The ratio among the three classes are 1:2:4. The error bars represent the standard errors.

Another important issue is the change of optimal kernel parameters. We can find the optimal kernel parameters from the initial labeled data set. As more labeled data are added, however, such kernel parameters may no longer be optimal. Unless we can afford a held-out, labeled data set, it is difficult to tune the kernel parameters online. The key reason is we do not have a good method to evaluate different kernel parameters as active learning proceeds. The standard methods like cross-validation and leave-one-out tend to fail because active learning chooses biased data samples. Such failures were observed and discussed by Baram et al. (2004). An important future direction is to find a good online performance evaluation method for active learning. Otherwise, one could take it as one of the biggest bottlenecks for using a SVM as the classifier in active learning because a SVM depends heavily on good kernel parameters. An effort towards solving this problem was proposed by Baram et al. (2004) who used the classification entropy maximization (CEM) criterion to evaluate the performance of different active learners. Their work shows CEM can help select the best active learner on several data sets.

An important thing omitted in most active learning+SVMs literature is to try active learning in batch mode. Unless labeling an example is extremely expensive, it is always convenient and practical to use active learning in batch mode, namely labeling several examples at a time and then retraining. As indicated in this paper, the best candidate set to label might be found in a different way from a single best candidate point. The “combined” approach only works for two-class problems. A criterion for the best set of data to label in multi-class SVMs needs to be addressed in future active learning work. At the very least, existing active learning methods need to be shown to work well in batch mode. Fortunately, our proposed active learning method did work well in batch mode without requiring a new criterion for selecting a set of data to label.

In this paper, we do not deal with a significant amount of label noise, which means one assigns incorrect class labels to the examples. In general, active learning tries to minimize the redundancy of labeled examples to reach a given accuracy. Therefore, noisy labels will hurt its performance. In our case, we only selectively label a few images and we expect small labeling noise due to the relatively small labeling effort. Please see Kearns (1998) for more detail about handling noisy labels in statistical queries.

Acknowledgments

This research was partially supported by the United States Navy, Office of Naval Research, under grant number N00014-02-1-0266 and the NSF under grant EIA-0130768.

References

Y. Baram, R. Yaniv, and K. Luz. Online choice of active learning algorithms. *Journal of Machine Learning Research*, 5:255–291, 2004.

K. Brinker. Incorporating diversity in active learning with support vector machines. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 59–66, 2003.

C. Campbell, N. Cristianini, and A. Smola. Query learning with large margin classifiers. In *Proceedings of 17th International Conference on Machine Learning*, pages 111–118, 2000.

G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In *Advances in Neural Information Processing Systems*, volume 13, pages 409–415, 2000.

C. Chang and C. Lin. LIBSVM: a library for support vector machines (version 2.3). 2001. URL `http://www.csie.ntu.edu.tw/ cjlin/papers/libsvm.pdf`.

N. Cristianini and J. Shawe-Taylor. *Introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press, 2000.

T. Hastie and R. Tibshirani. Classification by pairwise coupling. In *Advances in Neural Information Processing Systems*, volume 10, pages 507–513, 1998.

C. W. Hsu and C. J. Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.

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

D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval*, pages 3–12, 1994.

T. Luo. *Scaling up support vector machines with applications to plankton recognition*. PhD thesis, University of South Florida, forthcoming.

T. Luo, K. Kramer, D. Goldgof, L.O. Hall, S. Samson, A. Remsen, and T. Hopkins. Active learning to recognize multiple types of plankton. In *17th conference of the International Association for Pattern Recognition*, volume 3, pages 478–481, 2004a.

T. Luo, K. Kramer, D. Goldgof, L.O. Hall, S. Samson, A. Remson, and T. Hopkins. Recognizing plankton images from the shadow image particle profiling evaluation recorder. *IEEE Transactions on System, Man, and Cybernetics–Part B: Cybernetics*, 34(4):1753–1762, August 2004b.

P. Mitra, C. A. Murthy, and S. K. Pal. A probabilistic active support vector learning algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(3):413–418, 2004a.

P. Mitra, B. U. Shankar, and S. K. Pal. Segmentation of multispectral remote sensing images using active support vector machines. *Pattern Recognition Letters*, 25(9):1067–1074, 2004b.

H. T. Nguyen and A. Smeulders. Active learning using pre-clustering. In *Twenty-first International Conference on Machine learning*, 2004.

T. Onoda, H. Murata, and S. Yamada. Relevance feedback with active learning for document retrieval. In *Proceedings of the International Joint Conference on Neural Networks 2003*, volume 3, pages 1757–1762, 2003.

J. M. Park. Convergence and application of online active sampling using orthogonal pillar vectors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1197–1207, 2004.

J. Platt. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In *Advances in Large Margin Classiers*, pages 61–74, 2000.

J. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin DAGs for multiclass classification. In *Advances in Neural Information Processing Systems 12*, pages 547–553, 2000.

A. Remsen, T. L. Hopkins, and S. Samson. What you see is not what you catch: a comparison of concurrently collected net, optical plankton counter, and shadowed image particle profiling evaluation recorder data from the northeast gulf of mexico. *Deep Sea Research Part I: Oceanographic Research Papers*, 51:129–151, 2004.

R. Rifkin and A. Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

N. Roy and A. McCallum. Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of 18th International Conference on Machine Learning*, pages 441–448, 2001.

S. Samson, T. Hopkins, A. Remsen, L. Langebrake, T. Sutton, and J. Patten. A system for high resolution zooplankton imaging. *IEEE journal of ocean engineering*, pages 671–676, 2001.

M. Sassano. An empirical study of active learning with support vector machines for japanese word segmentation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 505–512, 2002.

G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. In *Proceedings of 17th International Conference on Machine Learning*, pages 839–846, 2000.

B. Schölkopf and A. J. Smola. *Learning with kernels*. The MIT Press, 2002.

X. Tang, F. Lin, S. Samson, and A. Remsen. Feature extraction for binary plankton image classification. *accepted by IEEE Journal of oceanic engineering*, forthcoming.

S. Tong and D. Koller. Support vector machine active learning with applications to text classification. In *Proceedings of 17th International Conference on Machine Learning*, pages 999–1006, 2000.

V. N. Vapnik. *The nature of statistical learning theory*. Springer, 2000.

L. Wang, K. L. Chan, and Z. h. Zhang. Bootstrapping SVM active learning by incorporating unlabelled images for image retrieval. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 629–634, 2003.

M. K. Warmuth, G. Rätsch, M. Mathieson, J. Liao, and C. Lemmen. Support vector machines for active learning in the drug discovery process. *Journal of Chemical Information Sciences*, 43(2): 667–673, 2003.

T. F. Wu, C. J. Lin, and R. C. Weng. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, 5:975–1005, 2004.

Journal of Machine Learning Research 6 (2005) 615–637 Submitted 2/05; Published 4/05

Learning Multiple Tasks with Kernel Methods

Theodoros Evgeniou THEODOROS.EVGENIOU@INSEAD.EDU
Technology Management
INSEAD
77300 Fontainebleau, France

Charles A. Micchelli CAM@MATH.ALBANY.EDU
Department of Mathematics and Statistics
State University of New York
The University at Albany
1400 Washington Avenue
Albany, NY 12222, USA

Massimiliano Pontil M.PONTIL@CS.UCL.AC.UK
Department of Computer Science
University College London
Gower Street, London WC1E, UK

Editor: John Shawe-Taylor

Abstract

We study the problem of learning many related tasks simultaneously using kernel methods and regularization. The standard single-task kernel methods, such as support vector machines and regularization networks, are extended to the case of multi-task learning. Our analysis shows that the problem of estimating many task functions with regularization can be cast as a single task learning problem if a family of multi-task kernel functions we define is used. These kernels model relations among the tasks and are derived from a novel form of regularizers. Specific kernels that can be used for multi-task learning are provided and experimentally tested on two real data sets. In agreement with past empirical work on multi-task learning, the experiments show that learning multiple related tasks simultaneously using the proposed approach can significantly outperform standard single-task learning particularly when there are many related tasks but few data per task.

Keywords: multi-task learning, kernels, vector-valued functions, regularization, learning algorithms

1. Introduction

Past empirical work has shown that, when there are multiple related learning tasks it is beneficial to learn them simultaneously instead of independently as typically done in practice (Bakker and Heskes, 2003; Caruana, 1997; Heskes, 2000; Thrun and Pratt, 1997). However, there has been insufficient research on the theory of multi-task learning and on developing multi-task learning methods. A key *goal* of this paper is to extend the single-task kernel learning methods which have been successfully used in recent years to multi-task learning. Our analysis establishes that the problem of estimating many task functions with regularization can be linked to a single task learning problem provided a family of multi-task kernel functions we define is used. For this purpose, we use kernels for vector-valued functions recently developed by Micchelli and Pontil (2005). We elaborate on these ideas within a practical context and present experiments of the proposed kernel-based multi-task learning methods on two real data sets.

Multi-task learning is important in a variety of practical situations. For example, in finance and economics forecasting predicting the value of many possibly related indicators simultaneously is often required (Greene, 2002); in marketing modeling the preferences of many individuals, for example with similar demographics, simultaneously is common practice (Allenby and Rossi, 1999; Arora, Allenby, and Ginter, 1998); in bioinformatics, we may want to study tumor prediction from multiple micro–array data sets or analyze data from mutliple related diseases.

It is therefore important to extend the existing kernel-based learning methods, such as SVM and RN, that have been widely used in recent years, to the case of multi-task learning. In this paper we shall demonstrate experimentally that the proposed multi-task kernel-based methods lead to significant performance gains.

The paper is organized as follows. In Section 2 we briefly review the standard framework for single-task learning using kernel methods. We then extend this framework to multi-task learning for the case of learning linear functions in Section 3. Within this framework we develop a general multi-task learning formulation, in the spirit of SVM and RN type methods, and propose some specific multi-task learning methods as special cases. We describe experiments comparing two of the proposed multi-task learning methods to their standard single-task counterparts in Section 4. Finally, in Section 5 we discuss extensions of the results of Section 3 to non-linear models for multi-task learning, summarize our findings, and suggest future research directions.

1.1 Past Related Work

The empirical evidence that multi-task learning can lead to significant performance improvement (Bakker and Heskes, 2003; Caruana, 1997; Heskes, 2000; Thrun and Pratt, 1997) suggests that this area of machine learning should receive more development. The simultaneous estimation of multiple statistical models was considered within the econometrics and statistics literature (Greene, 2002; Zellner, 1962; Srivastava and Dwivedi, 1971) prior to the interests in multi-task learning in the machine learning community.

Task relationships have been typically modeled through the assumption that the error terms (noise) for the regressions estimated simultaneously—often called "Seemingly Unrelated Regressions"—are correlated (Greene, 2002). Alternatively, extensions of regularization type methods, such as ridge regression, to the case of multi-task learning have also been proposed. For example, Brown and Zidek (1980) consider the case of regression and propose an extension of the standard ridge regression to multivariate ridge regression. Breiman and Friedman (1998) propose the curds&whey method, where the relations between the various tasks are modeled in a post–processing fashion.

The problem of multi-task learning has been considered within the statistical learning and machine learning communities under the name "learning to learn" (see Baxter, 1997; Caruana, 1997; Thrun and Pratt, 1997). An extension of the VC-dimension notion and of the basic generalization bounds of SLT for single-task learning (Vapnik, 1998) to the case of multi-task learning has been developed in (Baxter, 1997, 2000) and (Ben-David and Schuller, 2003). In (Baxter, 2000) the problem of bias learning is considered, where the goal is to choose an optimal hypothesis space from a family of hypothesis spaces. In (Baxter, 2000) the notion of the "extended VC dimension" (for a family of hypothesis spaces) is defined and it is used to derive generalization bounds on the average error of T tasks learned which is shown to decrease at best as $\frac{1}{T}$. In (Baxter, 1997) the same setup was used to answer the question "how much information is needed per task in order to learn T tasks" instead of "how many examples are needed for each task in order to learn T tasks", and the theory is developed using Bayesian and information theory arguments instead of VC dimension ones. In (Ben-David and Schuller, 2003) the extended VC dimension was used to derive tighter bounds that hold for each task (not just the average error among tasks as considered in (Baxter, 2000)) in the case that the learning tasks are related in a particular way defined. More recent work considers learning multiple tasks in a semi-supervised setting (Ando and Zhang, 2004) and the problem of feature selection with SVM across the tasks (Jebara, 2004).

Finally, a number of approaches for learning multiple tasks are Bayesian, where a probability model capturing the relations between the different tasks is estimated simultaneously with the models' parameters for each of the individual tasks. In (Allenby and Rossi, 1999; Arora, Allenby, and Ginter, 1998) a hierarchical Bayes model is estimated. First, it is assumed a priori that the parameters of the T functions to be learned are all sampled from an unknown Gaussian distribution. Then, an iterative Gibbs sampling based approach is used to simultaneously estimate both the individual functions and the parameters of the Gaussian distribution. In this model relatedness between the tasks is captured by this Gaussian distribution: the smaller the variance of the Gaussian the more related the tasks are. Finally, (Bakker and Heskes, 2003; Heskes, 2000) suggest a similar hierarchical model. In (Bakker and Heskes, 2003) a mixture of Gaussians for the "upper level" distribution instead of a single Gaussian is used. This leads to clustering the tasks, one cluster for each Gaussian in the mixture.

In this paper we will not follow a Bayesian or a statistical approach. Instead, our goal is to develop multi-task learning methods and theory as an extension of widely used kernel learning methods developed within SLT or Regularization Theory, such as SVM and RN. We show that using a particular type of kernels, the regularized multi-task learning method we propose is equivalent to a single-task learning one when such a multi-task kernel is used. The work here improves upon the ideas discussed in (Evgeniou and Pontil, 2004; Micchelli and Pontil, 2005b).

One of our aims is to show experimentally that the multi-task learning methods we develop here significantly improve upon their single-task counterpart, for example SVM. Therefore, to emphasize and clarify this point we only compare the standard (single-task) SVM with a proposed multi-task version of SVM. Our experiments show the benefits of multi-task learning and indicate that multi-task kernel learning methods are superior to their single-task counterpart. An exhaustive comparison of *any* single-task kernel methods with their multi-task version is beyond the scope of this work.

2. Background and Notation

In this section, we briefly review the basic setup for single-task learning using regularization in a reproducing kernel Hilbert space (RHKS) $\mathcal{H}_K$ with kernel K. For more detailed accounts (see Evgeniou, Pontil, and Poggio, 2000; Shawe-Taylor and Cristianini, 2004; Schölkopf and Smola, 2002; Vapnik, 1998; Wahba, 1990) and references therein.

2.1 Single-Task Learning: A Brief Review

In the standard single-task learning setup we are given m examples $\{(x_i, y_i) : i \in \mathbb{N}_m\} \subset \mathcal{X} \times \mathcal{Y}$ (we use the notation $\mathbb{N}_m$ for the set $\{1, \ldots, m\}$) sampled *i.i.d.* from an unknown probability distribution P on $\mathcal{X} \times \mathcal{Y}$. The input space $\mathcal{X}$ is typically a subset of $\mathbb{R}^d$, the d dimensional Euclidean space, and the output space $\mathcal{Y}$ is a subset of $\mathbb{R}$. For example, in binary classification $\mathcal{Y}$ is chosen to be $\{-1, 1\}$.

The goal is to learn a function f with small expected error $E[L(y, f(x))]$, where the expectation is taken with respect to P and L is a prescribed loss function such as the square error $(y - f(x))^2$. To this end, a common approach within SLT and regularization theory is to learn f as the minimizer in $\mathcal{H}_K$ of the functional

$$\frac{1}{m} \sum_{j \in \mathbb{N}_m} L(y_j, f(x_j)) + \gamma \|f\|_K^2 \tag{1}$$

where $\|f\|_K^2$ is the norm of f in $\mathcal{H}_K$. When $\mathcal{H}_K$ consists of linear functions $f(x) = w'x$, with $w, x \in \mathbb{R}^d$, we minimize

$$\frac{1}{m} \sum_{j \in \mathbb{N}_m} L(y_j, w'x_j) + \gamma w'w \tag{2}$$

where all vectors are column vectors and we use the notation A' for the transpose of matrix A, and w is a $d \times 1$ matrix.

The positive constant γ is called the regularization parameter and controls the trade off between the error we make on the m examples (the training error) and the complexity (smoothness) of the solution as measured by the norm in the RKHS. Machines of this form have been motivated in the framework of statistical learning theory (Vapnik, 1998). Learning methods such as RN and SVM are particular cases of these machines for certain choices of the loss function L (Evgeniou, Pontil, and Poggio, 2000).

Under rather general conditions (Evgeniou, Pontil, and Poggio, 2000; Micchelli and Pontil, 2005b; Wahba, 1990) the solution of Equation (1) is of the form

$$f(x) = \sum_{j \in \mathbb{N}_m} c_j K(x_j, x) \tag{3}$$

where $\{c_j : j \in \mathbb{N}_m\}$ is a set of real parameters and K is a kernel such as an homogeneous polynomial kernel of degree r, $K(x,t) = (x't)^r$, $x, t \in \mathbb{R}^d$. The kernel K has the property that, for $x \in \mathcal{X}$, $K(x, \cdot) \in \mathcal{H}_K$ and, for $f \in \mathcal{H}_K$ $\langle f, K(x, \cdot)\rangle_K = f(x)$, where $\langle \cdot, \cdot \rangle_K$ is the inner product in $\mathcal{H}_K$ (Aronszajn, 1950). In particular, for $x, t \in \mathcal{X}$, $K(x,t) = \langle K(x, \cdot), K(t, \cdot)\rangle_K$ implying that the $m \times m$ matrix $(K(x_i, x_j) : i, j \in \mathbb{N}_m)$ is symmetric and positive semi-definite for *any* set of inputs $\{x_j : j \in \mathbb{N}_m\} \subseteq \mathcal{X}$.

The result in Equation (3) is known as the *representer theorem*. Although it is simple to prove, it is remarkable as it makes the variational problem (1) amenable for computations. In particular, if L is convex, the unique minimizer of functional (1) can be found by replacing f by the right hand side of Equation (3) in Equation (1) and then optimizing with respect to the parameters $\{c_j : j \in N_m\}$.

A popular way to define the space $\mathcal{H}_K$ is based on the notion of a *feature map* $\Phi : \mathcal{X} \to \mathcal{W}$, where $\mathcal{W}$ is a Hilbert space with inner product denoted by $\langle \cdot, \cdot \rangle_{\mathcal{W}}$. Such a feature map gives rise to the linear space of all functions $f : \mathcal{X} \to \mathbb{R}$ defined for $x \in \mathcal{X}$ and $w \in \mathcal{W}$ as $f(x) = \langle w, \Phi(x)\rangle_{\mathcal{W}}$ with norm $\langle w, w\rangle_{\mathcal{W}}$. It can be shown that this space is (modulo an isometry) the RKHS $\mathcal{H}_K$ with kernel K defined, for $x, t \in \mathcal{X}$, as $K(x,t) = \langle \Phi(x), \Phi(t)\rangle_{\mathcal{W}}$. Therefore, the regularization functional (1) becomes

$$\frac{1}{m} \sum_{j \in \mathbb{N}_m} L(y_j, \langle w, \Phi(x_j)\rangle_{\mathcal{W}}) + \gamma \langle w, w\rangle_{\mathcal{W}}. \tag{4}$$

Again, any minimizer of this functional has the form

$$w = \sum_{j \in \mathbb{N}_m} c_j \Phi(x_j) \tag{5}$$

which is consistent with Equation (3).

2.2 Multi-Task Learning: Notation

For multi-task learning we have n tasks and corresponding to the $\ell-$th task there are available m examples $\{(x_{i\ell}, y_{i\ell}) : i \in \mathbb{N}_m\}$ sampled from a distribution P_ℓ on $\mathcal{X}_\ell \times \mathcal{Y}_\ell$. Thus, the total data available is $\{(x_{i\ell}, y_{i\ell}) : i \in \mathbb{N}_m, \ell \in \mathbb{N}_n\}$. The goal it to learn all n functions $f_\ell : \mathcal{X}_\ell \to \mathcal{Y}_\ell$ from the available examples. In this paper we mainly discuss the case that the tasks have a common input space, that is $X_\ell = X$ for all ℓ and briefly comment on the more general case in Section 5.1.

There are various special cases of this setup which occur in practice. Typically, the input space $\mathcal{X}_\ell$ is independent of ℓ. Even more so, the input data $x_{i\ell}$ may be independent of ℓ for every sample i. This happens in marketing applications of preference modeling (Allenby and Rossi, 1999; Arora, Allenby, and Ginter, 1998) where the same choice panel questions are given to many individual consumers, each individual provides his/her own preferences, and we assume that there is some commonality among the preferences of the individuals. On the other hand, there are practical circumstances where the output data $y_{i\ell}$ is independent of ℓ. For example, this occurs in the problem of integrating information from heterogeneous databases (Ben-David, Gehrke, and Schuller, 2002).

In other cases one does not have either possibilities, that is, the spaces $\mathcal{X}_\ell \times \mathcal{Y}_\ell$ are different. This is for example the machine vision case of learning to recognize a face by first learning to recognize parts of the face, such as eyes, mouth, and nose (Heisele et al., 2002). Each of these tasks can be learned using images of different size (or different representations).

We now turn to the extension of the theory and methods for single-task learning using the regularization based kernel methods briefly reviewed above to the case of multi-task learning. In the following section we will consider the case that functions f_ℓ are all linear functions and postpone the discussion of non-linear multi-task learning to Section 5.

3. A Framework for Multi-Task Learning: The Linear Case

Throughout this section we assume that $\mathcal{X}_\ell = \mathbb{R}^d$, $\mathcal{Y}_\ell = \mathbb{R}$ and that the functions f_ℓ are linear, that is, $f_\ell(x) = u'_\ell x$ with $u_\ell \in \mathbb{R}^d$. We propose to estimate the vector of parameters $u = (u_\ell : \ell \in \mathbb{N}_n) \in \mathbb{R}^{nd}$ as the minimizer of a regularization function

$$R(u) := \frac{1}{nm} \sum_{\ell \in \mathbb{N}_n} \sum_{j \in \mathbb{N}_m} L(y_{j\ell}, u'_\ell x_{j\ell}) + \gamma J(u) \tag{6}$$

where γ is a positive parameter, J is a homogeneous quadratic function of u, that is,

$$J(u) = u' E u \tag{7}$$

and E a $dn \times dn$ matrix which captures the relations between the tasks. From now on we assume that matrix E is symmetric and *positive definite*, unless otherwise stated. We briefly comment on the case that E is positive semidefinite below.

For a certain choice of J (or, equivalently, matrix E), the regularization function (6) learns the tasks independently using the regularization method (1). In particular, for $J(u) = \sum_{\ell \in \mathbb{N}_n} \|u_\ell\|^2$ the function (6) decouples, that is, $R(u) = \frac{1}{n} \sum_{\ell \in \mathbb{N}_n} r_\ell(u_\ell)$ where $r_\ell(u_\ell) = \frac{1}{m} \sum_{j \in \mathbb{N}_m} L(y_{j\ell}, u'_\ell x_{j\ell}) + \gamma \|u_\ell\|^2$, meaning that the task parameters are learned *independently*. On the other hand, if we choose $J(u) =$

$\sum_{\ell,q\in\mathbb{N}_n}\|u_\ell - u_q\|^2$, we can “force” the task parameters to be close to each other: task parameters u_ℓ are learned *jointly* by minimizing (6).

Note that function (6) depends on dn parameters whose number can be very large if the number of tasks n is large. Our analysis below establishes that the multi-task learning method (6) is equivalent to a single-task learning method as in (2) for an appropriate choice of a multi-task kernel in Equation (10) below. As we shall see, the input space of this kernel depends is the original $d-$dimensional space of the data plus an additional dimension which records the task the data belongs to. For this purpose, we take the feature space point of view and write all functions f_ℓ in terms of the *same* feature vector $w \in \mathbb{R}^p$ for some $p \in \mathbb{N}$, $p \geq dn$. That is, for each f_ℓ we write

$$f_\ell(x) = w'B_\ell x, \;\; x \in \mathbb{R}^d, \;\; \ell \in \mathbb{N}_n \tag{8}$$

or, equivalently,

$$u_\ell = B'_\ell w, \;\; \ell \in \mathbb{N}_n \tag{9}$$

for some $p \times d$ matrix B_ℓ yet to be specified. We also define the $p \times dn$ *feature matrix* $B := [B_\ell : \ell \in \mathbb{N}_n]$ formed by concatenating the n matrices B_ℓ, $\ell \in \mathbb{N}_n$.

Note that, since the vector u_ℓ in Equation (9) is arbitrary, to ensure that there exists a solution w to this equation it is necessary that the matrix B_ℓ is of full rank d for each $\ell \in \mathbb{N}_n$. Moreover, we assume that the feature matrix B is of full rank dn as well. If this is not the case, the functions f_ℓ are linearly related. For example, if we choose $B_\ell = B_0$ for every $\ell \in \mathbb{N}_n$, where B_0 is a prescribed $p \times d$ matrix, Equation (8) tells us that all tasks are the same task, that is, $f_1 = f_2 = \cdots = f_n$. In particular if $p = d$ and $B_0 = I_d$ the function (11) (see below) implements a single-task learning problem, as in Equation (2) with all the mn data from the n tasks as if they all come from the same task.

Said in other words, we view the vector-valued function $f = (f_\ell : \ell \in \mathbb{N}_n)$ as the real-valued function

$$(x,\ell) \mapsto w'B_\ell x$$

on the input space $\mathbb{R}^d \times \mathbb{N}_n$ whose squared norm is $w'w$. The Hilbert space of all such real-valued functions has the reproducing kernel given by the formula

$$K((x,\ell),(t,q)) = x'B'_\ell B_q t, \;\; x,t \in \mathbb{R}^d, \; \ell,q \in \mathbb{N}_n. \tag{10}$$

We call this kernel a *linear multi-task kernel* since it is bilinear in x and t for fixed ℓ and q.

Using this linear feature map representation, we wish to convert the regularization function (6) to a function of the form (2), namely,

$$S(w) := \frac{1}{nm}\sum_{\ell\in N_n}\sum_{j\in N_m} L(y_{j\ell}, w'B_\ell x_{j\ell}) + \gamma w'w, \;\; w \in \mathbb{R}^p. \tag{11}$$

This transformation relates matrix E defining the homogeneous quadratic function of u we used in (6), $J(u)$, and the feature matrix B. We describe this relationship in the proposition below.

Proposition 1 *If the feature matrix B is full rank and we define the matrix E in Equation (7) as to be $E = (B'B)^{-1}$ then we have that*

$$S(w) = R(B'w), \;\; w \in \mathbb{R}^d. \tag{12}$$

Conversely, if we choose a symmetric and positive definite matrix E in Equation (7) and T is a squared root of E then for the choice of $B = T'E^{-1}$ Equation (12) holds true.

PROOF. We first prove the first part of the proposition. Since Equation (9) requires that the feature vector w is *common* to all vectors u_ℓ and those are arbitrary, the feature matrix B must be of full rank dn and, so, the matrix E above is well defined. This matrix has the property that $BEB' = I_p$, this being the $p \times p$ identity matrix. Consequently, we have that

$$w'w = J(B'w), \;\; w \in \mathbb{R}^p \tag{13}$$

and Equation (12) follows.

On the other direction, we have to find a matrix B such that $BEB' = I_p$. To this end, we express E in the form

$$E = TT'$$

where T is a $dn \times p$ matrix, $p \geq dn$. This maybe done in various ways since E is positive definite. For example, with $p = dn$ we can find a $dn \times dn$ matrix T by using the eigenvalues and eigenvectors of E. With this representation of E we can choose our features to be

$$B = VT'E^{-1}$$

where V is an *arbitrary* $p \times p$ orthogonal matrix. This fact follows because $BEB' = I_p$. In particular, if we choose $V = I_p$ the result follows.

□

Note that this proposition requires that B is of full rank because E is positive definite. As an example, consider the case that B_ℓ is a $dn \times d$ matrix all of whose $d \times d$ blocks are zero except for the $\ell-$th block which is equal to I_d. This choice means that we are learning all tasks independently, that is, $J(u) = \sum_{\ell \in \mathbb{N}_n} \|u_\ell\|^2$ and proposition (1) confirms that $E = I_{dn}$.

We conjecture that if the matrix B is not full rank, the equivalence between function (11) and (6) stated in proposition 1 still holds true provided matrix E is given by the pseudoinverse of matrix $(B'B)$ *and* we minimize the latter function on the linear subspace $\mathcal{S}$ spanned by the eigenvectors of E which have a positive eigenvalue. For example, in the above case where $B_\ell = B_0$ for all $\ell \in \mathbb{N}_n$ we have that $\mathcal{S} = \{(u_\ell : \ell \in \mathbb{N}_n) : u_1 = u_2 = \cdots = u_n\}$. This observation would also extend to the circumstance where there are arbitrary linear relations amongst the task functions. Indeed, we can impose such linear relations on the features directly to achieve this relation amongst the task functions. We discuss a specific example of this set up in Section 3.1.3. However, we leave a complete analysis of the positive semidefinite case to a future occasion.

The main implication of proposition 1 is the equivalence between function (6) and (11) when E is positive definite. In particular, this proposition implies that when matrix B and E are linked as stated in the proposition, the unique minimizers w^* of (11) and u^* of (6) are related by the equations $u^* = B'w^*$.

Since functional (11) is like a single task regularization functional (2), by the representer theorem—see Equation (5)—its minimizer has the form

$$w^* = \sum_{j \in \mathbb{N}_m} \sum_{\ell \in \mathbb{N}_n} c_{j\ell} B_\ell x_{j\ell}.$$

This implies that the optimal task functions are

$$f_q^*(x) = \sum_{j \in \mathbb{N}_m} \sum_{\ell \in \mathbb{N}_n} c_{j\ell} K((x_{j\ell}, \ell), (x, q)), \;\; x \in \mathbb{R}^d, q \in \mathbb{N}_n \tag{14}$$

where the kernel is defined in Equation (10). Note that these equations hold for *any* choice of the matrices B_ℓ, $\ell \in \mathbb{N}_n$.

Having defined the kernel for (10), we can now use standard single-task learning methods to learn multiple tasks simultaneously (we only need to define the appropriate kernel for the input data (x,ℓ)). Specific choices of the loss function L in Equation (11) lead to different learning methods.

Example 1 In regularization networks (RN) we choose the square loss $L(y,z) = (y-z)^2$, $y,z \in \mathbb{R}$ (see, for example, Evgeniou, Pontil, and Poggio, 2000). In this case the parameters $c_{j\ell}$ in Equation (14) are obtained by solving the system of linear equations

$$\sum_{q\in\mathbb{N}_n} \sum_{j\in\mathbb{N}_m} K((x_{jq},q),(x_{i\ell},\ell))c_{jq} = y_{i\ell}, \quad i \in \mathbb{N}_m, \ell \in \mathbb{N}_n. \tag{15}$$

When the kernel K is defined by Equation (10) this is a form of multi-task ridge regression.

□

Example 2 In support vector machines (SVM) for binary classification (Vapnik, 1998) we choose the hinge loss, namely $L(y,z) = (1-yz)_+$ where $(x)_+ = \max(0,x)$ and $y \in \{-1,1\}$. In this case, the minimization of function (11) can be rewritten in the usual form

Problem 3.1

$$\min \left\{ \sum_{\ell\in\mathbb{N}_n} \sum_{i\in\mathbb{N}_m} \xi_{i\ell} + \gamma\|w\|^2 \right\} \tag{16}$$

subject, for all $i \in \mathbb{N}_m$ and $\ell \in \mathbb{N}_n$, to the constraints that

$$\begin{aligned} y_{i\ell} w' B_\ell x_{i\ell} &\geq 1 - \xi_{i\ell} \\ \xi_{i\ell} &\geq 0. \end{aligned} \tag{17}$$

Following the derivation in Vapnik (1998) the dual of this problem is given by

Problem 3.2

$$\max_{c_{i\in\ell}} \left\{ \sum_{\ell\in\mathbb{N}_n} \sum_{i\in\mathbb{N}_m} c_{i\ell} - \frac{1}{2} \sum_{\ell,q\in\mathbb{N}_n} \sum_{i,j\in\mathbb{N}_m} c_{i\ell} y_{i\ell} c_{jq} y_{jq} K((x_{i\ell},\ell),(x_{jq},q)) \right\} \tag{18}$$

subject, for all $i \in \mathbb{N}_n$ and $\ell \in \mathbb{N}_n$, to the constrains that

$$0 \leq c_{i\ell} \leq \frac{1}{2\gamma}.$$

□

We now study particular examples some of which we also test experimentally in Section 5.

3.1 Examples of Linear Multi-Task Kernels

We discuss some examples of the above framework which are valuable for applications. These cases arise from different choices of matrices B_ℓ that we used above to model task relatedness or, equivalently, by directly choosing the function J in Equation (6).

Notice that a particular case of the regularizer J in Equation (7) is given by

$$J(u) = \sum_{\ell,q \in \mathbb{N}_n} u'_\ell u_q G_{\ell q} \tag{19}$$

where $G = (G_{\ell,q} : \ell, q \in \mathbb{N}_n)$ is a positive definite matrix. Proposition (1) implies that the kernel has the form

$$K((x,\ell),(t,q)) = x't\, G^{-1}_{\ell q}. \tag{20}$$

Indeed, J can be written as (u, Eu) where E is the $n \times n$ block matrix whose ℓ, q block is the $d \times d$ matrix $G_{\ell q} I_d$ and the result follows. The examples we discuss are with kernels of the form (20).

3.1.1 A USEFUL EXAMPLE

In our first example we choose B_ℓ to be the $(n+1)d \times d$ matrix whose $d \times d$ blocks are all zero except for the $1-$st and $(\ell+1)-$th block which are equal to $\sqrt{1-\lambda} I_d$ and $\sqrt{\lambda n} I_d$ respectively, where $\lambda \in (0,1)$ and I_d is the $d-$dimensional identity matrix. That is,

$$B'_\ell = [\sqrt{1-\lambda} I_d, \underbrace{0, \ldots, 0}_{\ell-1}, \sqrt{\lambda n} I_d, \underbrace{0, \ldots, 0}_{n-\ell}] \tag{21}$$

where here 0 stands for the $d \times d$ matrix all of whose entries are zero. Using Equation (10) the kernel is given by

$$K((x,\ell),(t,q)) = (1 - \lambda + \lambda n \delta_{\ell q}) x't, \;\; \ell, q \in \mathbb{N}_n, \; x, t \in \mathbb{R}^n. \tag{22}$$

A direct computation shows that

$$E_{\ell q} = ((B'B)^{-1})_{\ell q} = \frac{1}{n} \left(\frac{\delta_{\ell q}}{\lambda} - \frac{1-\lambda}{n\lambda} \right) I_d$$

where $E_{\ell q}$ is the $(\ell, q)-$th $d \times d$ block of matrix E. By proposition 1 we have that

$$J(u) = \frac{1}{n} \left(\sum_{\ell \in \mathbb{N}_n} \|u_\ell\|^2 + \frac{1-\lambda}{\lambda} \sum_{\ell \in \mathbb{N}_n} \|u_\ell - \frac{1}{n} \sum_{q \in \mathbb{N}_n} u_q\|^2 \right). \tag{23}$$

This regularizer enforces a trade–off between a desirable small size for per–task parameters and closeness of each of these parameters to their average. This trade-off is controlled by the coupling parameter λ. If λ is small the tasks parameters are *related* (closed to their average) whereas if $\lambda = 1$ the task are learned independently.

The model of minimizing (11) with the regularizer (24) was proposed by Evgeniou and Pontil (2004) in the context of support vector machines (SVM's). In this case the above regularizer trades off large margin of each per–task SVM with closeness of each SVM to the average SVM. In Section 4 we will present numerical experiments showing the good performance of this multi–task SVM compared to both independent per–task SVM's (that is, $\lambda = 1$ in Equation (22)) and previous multi–task learning methods.

We note in passing that an alternate form for the function J is

$$J(u) = \min\left\{ \frac{1}{\lambda n} \sum_{\ell \in \mathbb{N}_n} \|u_\ell - u_0\|^2 + \frac{1}{1-\lambda}\|u_0\|^2 : u_0 \in \mathbb{R}^d \right\}. \tag{24}$$

It was this formula which originated our interest in multi-task learning in the context of regularization, see (Evgeniou and Pontil, 2004) for a discussion. Moreover, if we replace the identity matrix I_d in Equation (21) by a (any) $d \times d$ matrix A we obtain the kernel

$$K((x,\ell),(t,q)) = (1 - \lambda + \lambda n \delta_{\ell q}) x' Q t, \quad \ell, q \in \mathbb{N}_n, \ x, t \in \mathbb{R}^n \tag{25}$$

where $Q = A'A$. In this case the norm in Equation (23) and (24) is replaced by $\|\cdot\|_{Q^{-1}}$.

3.1.2 TASK CLUSTERING REGULARIZATION

The regularizer in Equation (24) implements the idea that the task parameters u_ℓ are all related to each other in the sense that each u_ℓ is close to an "average parameter" u_0. Our second example extends this idea to different groups of tasks, that is, we assume that the task parameters can be put together in different groups so that the parameters in the $k-$th group are all close to an average parameter u_{0k}. More precisely, we consider the regularizer

$$J(u) = \min\left\{ \sum_{k \in \mathbb{N}_c} \left(\sum_{\ell \in \mathbb{N}_n} \rho_k^{(\ell)} \|u_\ell - u_{0k}\|^2 + \rho \|u_{0k}\|^2 \right) : u_{0k} \in \mathbb{R}^d, k \in \mathbb{N}_c \right\} \tag{26}$$

where $\rho_k^{(\ell)} \geq 0$, $\rho > 0$, and c is the number of clusters. Our previous example corresponds to $c = 1$, $\rho = \frac{1}{1-\lambda}$ and $\rho_1^{(\ell)} = \frac{1}{\lambda n}$. A direct computation shows that

$$J(u) = \sum_{\ell, q \in \mathbb{N}_n} u_\ell' u_q G_{\ell q}$$

where the elements of the matrix $G = (G_{\ell q} : \ell, q \in \mathbb{N}_n)$ are given by

$$G_{\ell q} = \sum_{k \in \mathbb{N}_c} \left(\rho_k^{(\ell)} \delta_{\ell q} - \frac{\rho_k^{(\ell)} \rho_k^{(q)}}{\rho + \sum_{r \in \mathbb{N}_n} \rho_k^{(r)}} \right).$$

If $\rho_k^{(\ell)}$ has the property that given any ℓ there is a cluster k such that $\rho_k^{(\ell)} > 0$ then G is positive definite. Then J is positive definite and by Equation (20) the kernel is given by $K((x,\ell),(t,q)) = G_{\ell q}^{-1} x' t$. In particular, if $\rho_h^{(\ell)} = \delta_{hk(\ell)}$ with $k(\ell)$ the cluster task ℓ belongs to, matrix G is invertible and takes the simple form

$$G_{\ell q}^{-1} = \delta_{\ell q} + \frac{1}{\rho} \theta_{\ell q} \tag{27}$$

where $\theta_{\ell q} = 1$ if tasks ℓ and q belong to the same cluster and zero otherwise. In particular, if $c = 1$ and we set $\rho = \frac{1-\lambda}{\lambda n}$ the kernel $K((x,\ell),(t,q)) = (\delta_{\ell q} + \frac{1}{\rho}) x' t$ is the same (modulo a constant) as the kernel in Equation (22).

3.1.3 GRAPH REGULARIZATION

In our third example we choose an $n \times n$ symmetric matrix A all of whose entries are in the unit interval, and consider the regularizer

$$J(u) := \frac{1}{2} \sum_{\ell, q \in \mathbb{N}_n} \|u_\ell - u_q\|^2 A_{\ell q} = \sum_{\ell, q \in \mathbb{N}_n} u'_\ell u_q L_{\ell q} \tag{28}$$

where $L = D - A$ with $D_{\ell q} = \delta_{\ell q} \sum_{h \in \mathbb{N}_n} A_{\ell h}$. The matrix A could be the weight matrix of a graph with n vertices and L the graph Laplacian (Chung, 1997). The equation $A_{\ell q} = 0$ means that tasks ℓ and q are not related, whereas $A_{\ell q} = 1$ means strong relation.

The quadratic function (28) is only positive semidefinite since $J(u) = 0$ whenever all the components of u_ℓ are independent of ℓ. To identify those vectors u for which $J(u) = 0$ we express the Laplacian L in terms of its eigenvalues and eigenvectors. Thus, we have that

$$L_{\ell q} = \sum_{k \in \mathbb{N}_n} \sigma_k v_{k\ell} v_{kq} \tag{29}$$

where the matrix $V = (v_{k\ell})$ is orthogonal, $\sigma_1 = \cdots = \sigma_r < \sigma_{r+1} \leq \cdots \leq \sigma_n$ are the eigenvalues of L and $r \geq 1$ is the multiplicity of the zero eigenvalue. The number r can be expressed in terms of the number of connected components of the graph, see, for example, (Chung, 1997). Substituting the expression (29) for L in the right hand side of (28) we obtain that

$$J(u) = \sum_{k \in \mathbb{N}_n} \sigma_k \left\| \sum_{\ell \in \mathbb{N}_n} u_\ell v_{k\ell} \right\|^2.$$

Therefore, we conclude that J is positive definite on the space

$$\mathcal{S} = \left\{ u : u \in \mathbb{R}^{dn}, \sum_{\ell \in \mathbb{N}_n} u_\ell v_{k\ell} = 0, k \in \mathbb{N}_r \right\}.$$

Clearly, the dimension of $\mathcal{S}$ is $d(n-r)$. $\mathcal{S}$ gives us a Hilbert space of vector-valued linear functions

$$\mathcal{H} = \left\{ f_u(x) = (u'_\ell x : \ell \in \mathbb{N}_n) : u \in \mathcal{S} \right\}$$

and the reproducing kernel of $\mathcal{H}$ is given by

$$K((x,\ell),(t,q)) = L^+_{\ell q} x' t. \tag{30}$$

where L^+ is the pseudoinverse of L, that is,

$$L^+_{\ell q} = \sum_{k=r+1}^{n} \sigma_k^{-1} v_{k\ell} v_{kq}.$$

The verification of these facts is straightforward and we do not elaborate on the details. We can use this observation to assert that on the space $\mathcal{S}$ the regularization function (6) corresponding to the Laplacian has a *unique* minimum and it is given in the form of a representer theorem for kernel (30).

4. Experiments

As discussed in the introduction, we conducted experiments to compare the (standard) single-task version of a kernel machine, in this case SVM, to a multi-task version developed above. We tested two multi-task versions of SVM: a) we considered the simple case that the matrix Q in Equation (25) is the identity matrix, that is, we use the multi-task kernel (22), and b) we estimate the matrix Q in (25) by running PCA on the previously learned task parameters. Specifically, we first initialize Q to be the identity matrix. We then iterate as follows:

1. We estimate parameters u_ℓ using (25) and the current estimate of matrix Q (which, for the first iteration is the identity matrix).

2. We run PCA on these estimates, and select only the top principal components (corresponding to the largest eigenvalues of the empirical correlation matrix of the estimated u_ℓ). In particular, we only select the eigenvectors so that the sum of the corresponding eigenvalues (total "energy" kept) is at least 90% of the sum of all the eigenvalues (not using the remaining eigenvalues once we reach this 90% threshold). We then use the covariance of these principal components as our estimate of matrix Q in (25) for the next iteration.

We can repeat steps (1) and (2) until all eigenvalues are needed to reach the 90% energy threshold – typically in 4-5 iterations for the experiments below. We can then pick the estimated u_ℓ after the iteration that lead to the best validation error. We emphasize, that this is simply a heuristic. We do not have a theoretical justification for this heuristic. Developing a theory as well as other methods for estimating matrix Q is an open question. Notice that instead of using PCA we could directly use for matrix Q simply the covariance of the estimated u_ℓ of the previous iteration. However doing so is sensitive to estimation errors of u_ℓ and leads (as we also observed experimentally – we don't show the results here for simplicity) to poorer performance.

One of the key questions we considered is: *how does multi-task learning perform relative to single-task as the number of data per task and as the number of tasks change?* This question is also motivated by a typical situation in practice, where it may be easy to have data from many related tasks, but it may be difficult to have many data per task. This could often be for example the case in analyzing customer data for marketing, where we may have data about many customers (tens of thousands) but only a few samples per customer (only tens) (Allenby and Rossi, 1999; Arora, Allenby, and Ginter, 1998). It can also be the case for biological data, where we may have data about many related diseases (for example, types of cancer), but only a few samples per disease (Rifkin et al., 2003). As noted by other researchers in (Baxter, 1997, 2000; Ben-David, Gehrke, and Schuller, 2002; Ben-David and Schuller, 2003), one should expect that multi-task learning helps more, relative to single task, when we have many tasks but only few data per task – while when we have many data per task then single-task learning may be as good.

We performed experiments with two real data sets. One was on customer choice data, and the other was on school exams used by (Bakker and Heskes, 2003; Heskes, 2000) which we use here also for comparison with (Bakker and Heskes, 2003; Heskes, 2000). We discuss these experiments next.

4.1 Customer Data Experiments

We tested the proposed methods using a real data set capturing choices among products made by many individuals.[1] The goal is to estimate a function for each individual modeling the preferences of the individual based on the choices he/she has made. This function is used in practice to predict what product each individual will choose among future choices. We modeled this problem as a classification one along the lines of (Evgeniou, Boussios, and Zacharia, 2002). Therefore, the goal is to estimate a classification function for each individual.

We have data from 200 individuals, and for each individual we have 120 data points. The data are three dimensional (the products were described using three attributes, such as color, price, size, etc.) each feature taking only discrete values (for example, the color can be only blue, or black, or red, etc.). To handle the discrete valued attributes, we transformed them into binary ones, having eventually 20-dimensional binary data. We consider each individual as a different "task". Therefore we have 200 classification tasks and 120 20-dimensional data points for each task – for a total of 24000 data points.

We consider a linear SVM classification for each task – trials with non-linear (polynomial of degree 2 and 3) SVM did not improve performance for this data set. To test how multi-task compares to single task as the number of data per task and/or the number of tasks changes, we ran experiments with varying numbers of data per task and number of tasks. In particular, we considered 50, 100, and 200 tasks, splitting the 200 tasks into 4 groups of 50 or 2 groups of 100 (or one group of 200), and then taking the average performance among the 4 groups, the 2 groups (and the 1 group). For each task we split the 120 points into 20, 30, 60, 90 training points, and 100, 90, 60, 30 test points respectively.

Given the limited number of data per task, we chose the regularization parameter γ for the single-task SVM among only a few values (0.1, 1, 10) using the actual test error.[2] On the other hand, the multi-task learning regularization parameter γ and parameter λ in (22) were chosen using a validation set consisting of one (training) data point per task which we then included back to the training data for the final training after the parameter selection. The parameters λ and γ used when we estimated matrix Q through PCA were the same as when we used the identity matrix as Q. We note that one of the advantages of multi-task learning is that, since the data are typically from many tasks, parameters such as regularization parameter γ can be practically chosen using only a few, proportionally to all the data available, validation data without practically "losing" many data for parameter selection – which may be a further important practical reason for multi-task learning. Parameter λ was chosen among values (0, 0.2, 0.4, 0.6, 0.8) – value 1 corresponding to training one SVM per task. Below we also record the results indicating how the test performance is affected by parameter λ.

We display all the results in Table 4.1. Notice that the performance of the single-task SVM does not change as the number of tasks increases – as expected. We also note that when we use one SVM for all the tasks—treating the data as if they come from the same task—we get a very poor performance: between 38 and 42 percent test error for the (data $\times$ tasks) cases considered.

From these results we draw the following conclusions:

1. The data are proprietary were provided to the authors by Research International Inc. and are available upon request.
2. This lead to some overfitting of the single task SVM, however it only gave our competitor an advantage over our approach.

Tasks	Data	One SVM	Indiv SVM	Identity	PCA
50	20	41.97	29.86	**28.72**	**29.16**
100	20	41.41	29.86	**28.30**	29.26
200	20	40.08	29.86	**27.79**	28.53
50	30	40.73	26.84	**25.53**	**25.65**
100	30	40.66	26.84	**25.25**	**24.79**
200	30	39.43	26.84	25.16	**24.13**
50	60	40.33	22.84	22.06	**21.08**
100	60	40.02	22.84	22.27	**20.79**
200	60	39.74	22.84	21.86	**20.00**
50	90	38.51	19.84	19.68	**18.45**
100	90	38.97	19.84	19.34	**18.08**
200	90	38.77	19.84	19.27	**17.53**

Table 1: Comparison of Methods as the number of data per task and the number of tasks changes. "One SVM" stands for training one SVM with all the data from all the task, "Indiv SVM" stands for training for each task independently, "Identity" stands for the multi-task SVM with the identity matrix, and "PCA" is the multi-task SVM using the PCA approach. Misclassification errors are reported. Best performance(s) at the 5% significance level is in bold.

- When there are few data per task (20, 30, or 60), both multi-task SVMs significantly outperform the single-task SVM.
- As the number of tasks increases the advantage of multi-task learning increases – for example for 20 data per task, the improvement in performance relative to single-task SVM is 1.14, 1.56, and 2.07 percent for the 50, 100, and 200 tasks respectively.
- When we have many data per task (90), the simple multi-task SVM does not provide any advantage relative to the single-task SVM. However, the PCA based multi-task SVM significantly outperforms the other two methods.
- When there are few data per task, the simple multi-task SVM performs better than the PCA multi-task SVM. It may be that in this case the PCA multi-task SVM overfits the data.

The last two observations indicate that it is important to have a good estimate of matrix Q in (25) for the multi-task learning method that uses matrix Q. Achieving this is currently an open questions that can be approached, for example, using convex optimization techniques, see, for example, (Lanckriet et al., 2004; Micchelli and Pontil, 2005b)

To explore the second point further, we show in Figure 1 the change in performance for the identity matrix based multi-task SVM relative to the single-task SVM in the case of 20 data per task. We use $\lambda = 0.6$ as before. We notice the following:

- When there are only a few tasks (for example, less than 20 in this case), multi-task can hurt the performance relative to single-task. Notice that this depends on the parameter λ used.

For example, setting λ close to 1 leads to using a single-task SVM. Hence our experimental findings indicate that *for few tasks one should use either a single-task SVM or a multi-task one with parameter λ selected near 1*.

- As the number of tasks increases, performance improves – surpassing the performance of the single-task SVM after 20 tasks in this case.

As discussed in (Baxter, 1997, 2000; Ben-David, Gehrke, and Schuller, 2002; Ben-David and Schuller, 2003), an important theoretical question is to study the effects of adding additional tasks on the generalization performance (Ben-David, Gehrke, and Schuller, 2002; Ben-David and Schuller, 2003). What our experiments show is that, for few tasks it may be inappropriate to follow a multi-task approach if a small λ is used, but as the number of tasks increases performance relative to single-task learning improves. Therefore one should choose parameter λ depending on the number of tasks, much like one should choose regularization parameter γ depending on the number of data.

We tested the effects of parameter λ in Equation (22) on the performance of the proposed approach. In Figure 2 we plot the test error for the simple multi-task learning method using the identity matrix (kernel (22)) for the case of 20 data per task when there are 200 tasks (third row in Table 4.1), or 10 tasks (for which single-task SVM outperforms multi-task SVM for $\lambda = 0.6$ as shown in Figure 1). Parameter λ varies from 0 (one SVM for all tasks) to 1 (one SVM per task). Notice that for the 200 tasks the error drops and then increases, having a flat minimum between $\lambda = 0.4$ and 0.6. Moreover, for any λ between 0.2 and 1 we get a better performance than the single-task SVM. The same behavior holds for the 10 tasks, except that now the space of λ's for which the multi-task approach outperforms the single-task one is smaller – only for λ between 0.7 and 1. Hence, *for a few tasks multi-task learning can still help if a large enough λ is used*. However, as we noted above, it is an open question as to how to choose parameter λ in practice – other than using a validation set.

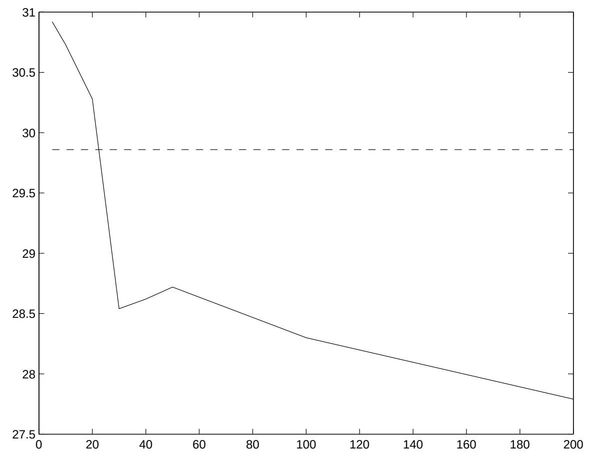


Figure 1: The horizontal axis is the number of tasks used. The vertical axis is the total test misclassification error among the tasks. There are 20 training points per task. We also show the performance of a single-task SVM (dashed line) which, of course, is not changing as the number of tasks increases.

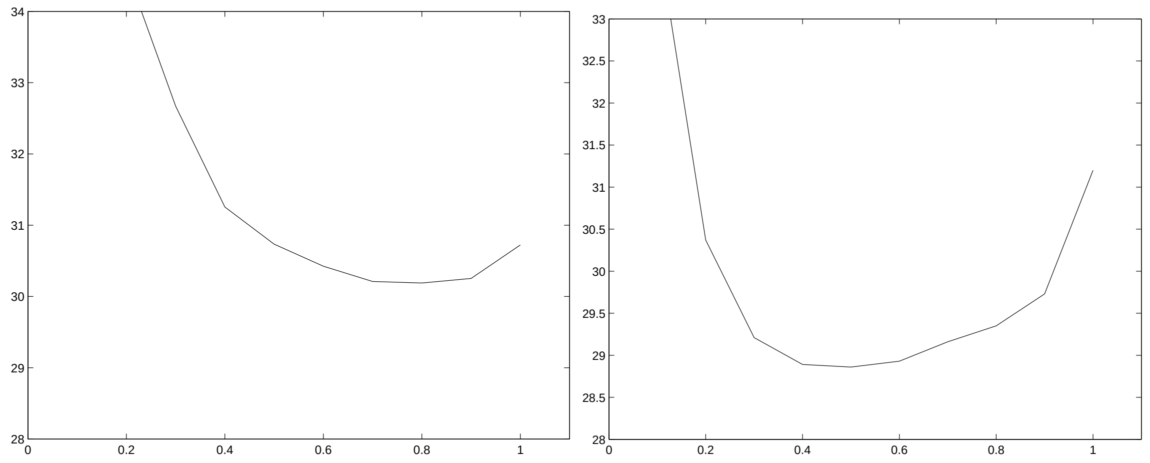


Figure 2: The horizontal axis is the parameter λ for the simple multi-task method with the identity matrix kernel (22). The vertical axis is the total test misclassification error among the tasks. There are 200 tasks with 20 training points and 100 test points per task. Left is for 10 tasks, and right is for 200 tasks.

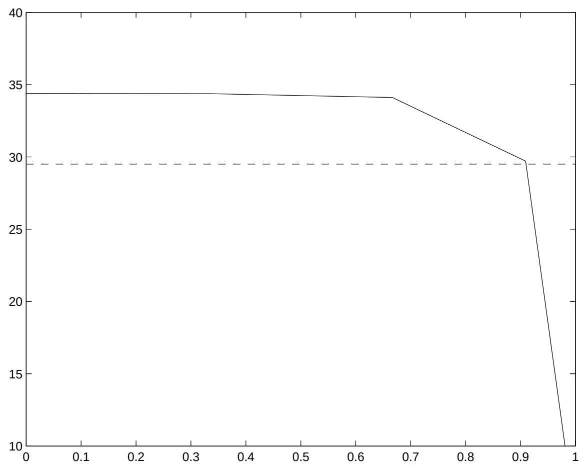


Figure 3: Performance on the school data. The horizontal axis is the parameter λ for the simple multi-task method with the identity matrix while the vertical is the explained variance (percentage) on the test data. The solid line is the performance of the proposed approach while the dashed line is the best performance reported in (Bakker and Heskes, 2003).

4.2 School Data Experiment

We also tested the proposed approach using the “school data” from the Inner London Education Authority available at *multilevel.ioe.ac.uk/intro/datasets.html*. This experiment is also discussed in (Evgeniou and Pontil, 2004) where some of the ideas of this paper were first presented. We selected this data set so that we can also compare our method directly with the work of Bakker and Heskes (2003) where a number of multi-task learning methods are applied to this data set. This data consists of examination records of 15362 students from 139 secondary schools. The goal is to predict the exam scores of the students based on the following inputs: year of the exam, gender, VR band, ethnic group, percentage of students eligible for free school meals in the school, percentage of students in VR band one in the school, gender of the school (i.e. male, female, mixed), and school denomination. We represented the categorical variables using binary (dummy) variables, so the total number of inputs for each student in each of the schools was 27. Since the goal is to predict the exam scores of the students we ran regression using the SVM ε–loss function (Vapnik, 1998) for the multi–task learning method proposed. We considered each school to be "one task". Therefore, we had 139 tasks in total. We made 10 random splits of the data into training (75% of the data, hence around 70 students per school on average) and test (the remaining 25% of the data, hence around 40 students per school on average) data and we measured the generalization performance using the explained variance of the test data as a measure in order to have a direct comparison with (Bakker and Heskes, 2003) where this error measure is used. The explained variance is defined in (Bakker and Heskes, 2003) to be the total variance of the data minus the sum–squared error on the test set as a percentage of the total data variance, which is a percentage version of the standard R^2 error measure for regression for the test data. Finally, we used a simple linear kernel for each of the tasks.

The results for this experiment are shown in Figure 3. We set regularization parameter γ to be 1 and used a linear kernel for simplicity. We used the simple multi-task learning method proposed with the identity matrix. We let the parameter λ vary to see the effects. For comparison we also report on the performance of the task clustering method described in (Bakker and Heskes, 2003) – the dashed line in the figure.

The results show again the advantage of learning all tasks (for all schools) simultaneously instead of learning them one by one. Indeed, learning each task separately in this case hurts performance a lot. Moreover, even the simple identity matrix based approach significantly outperforms the Bayesian method of (Bakker and Heskes, 2003), which in turn in better than other methods as compared in (Bakker and Heskes, 2003). Note, however, that for this data set one SVM for all tasks performs the best, which is also similar to using a small enough λ (any λ between 0 and 0.7 in this case). Hence, it appears that the particular data set may come from a single task (despite this observation, we use this data set for direct comparison with (Bakker and Heskes, 2003)). This result also indicates that when the tasks are the same task, using the proposed multi-task learning method does not hurt as long as a small enough λ is used. Notice that for this data set the performance does not change significantly for λ between 0 and 0.7, which shows that, as for the customer data above, the proposed method is not very sensitive to λ. A theoretical study of the sensitivity of our approach to the choice of the parameter λ is an open research direction which may also lead to a better understanding of the effects of increasing the number of tasks on the generalization performance as discussed in (Baxter, 1997, 2000; Ben-David and Schuller, 2003).

5. Discussion and Conclusions

In this final section we outline the extensions of the ideas presented above to non-linear functions, discuss some open problems on multi-tasks learning and draw our conclusions.

5.1 Nonlinear Multi-Task Kernels

We discuss a non-linear extension of the multi-task learning methods presented above. This gives us an opportunity to provide a wide variety of multi-task kernels which may be useful for applications. Our presentation builds upon earlier work on learning vector–valued functions (Micchelli and Pontil, 2005) which developed the theory of RKHS of functions whose range is a Hilbert space.

As in the linear case we view the vector-valued function $f = (f_\ell : \ell \in \mathbb{N}_n)$ as a real-valued function on the input space $\mathcal{X} \times \mathbb{N}_n$. We express f in terms of the feature maps $\Phi_\ell : \mathcal{X} \to \mathcal{W}$, $\ell \in \mathbb{N}_n$ where $\mathcal{W}$ is a Hilbert space with inner product $\langle \cdot, \cdot \rangle$. That is, we have that

$$f_\ell(x) = \langle w, \Phi_\ell(x) \rangle, \ \ x \in \mathcal{X}, \ \ \ell \in \mathbb{N}_n.$$

The vector w is computed by minimizing the single-task functional

$$S(w) := \frac{1}{nm} \sum_{\ell \in N_n} \sum_{j \in N_m} L(y_{j\ell}, \langle w, \Phi_\ell(x_{j\ell}) \rangle) + \gamma \langle w, w \rangle, \ \ w \in \mathcal{W}. \tag{31}$$

By the representer theorem, the minimizer of functional S has the form in Equation (14) where the multi-task kernel is given by the formula

$$K((x,\ell),(t,q)) = \langle \Phi_\ell(x), \Phi_q(t) \rangle \ \ x,t \in \mathcal{X}, \ \ell,q \in \mathbb{N}_n. \tag{32}$$

In Section 3 we have discussed this approach in the case that $\mathcal{W}$ is a finite dimensional Euclidean space and Φ_ℓ the linear map $\Phi_\ell(x) = B_\ell x$, thereby obtaining the linear multi-task kernel (10). In order to generalize this case it is useful to recall a result of Schur which states that the elementwise product of two positive semidefinite matrices is also positive semidefinite, (Aronszajn, 1950, p. 358). This implies that the elementwise product of two kernels is a kernel. Consequently, we conclude that, for any $r \in \mathbb{N}$,

$$K((x,\ell),(t,q)) = (x' B'_\ell B_q t)^r \tag{33}$$

is a polynomial multi-task kernel.

More generally we have the following lemma.

Lemma 2 *If G is a kernel on $\mathcal{T} \times \mathcal{T}$ and, for every $\ell \in \mathbb{N}_n$, there are prescribed mappings $z_\ell : \mathcal{X} \to \mathcal{T}$ such that*

$$K((x,\ell),(t,q)) = G(z_\ell(x), z_q(t)), \ \ \ x,t \in \mathcal{X}, \ \ell,q \in \mathbb{N}_n \tag{34}$$

then K is a multi-task kernel.

PROOF. We note that for every $\{c_{i\ell} : i \in \mathbb{N}_m, \ell \in \mathbb{N}_n\} \subset \mathbb{R}$ and $\{x_{i\ell} : i \in \mathbb{N}_m, \ell \in \mathbb{N}_n\} \subset \mathcal{X}$ we have

$$\sum_{i,j \in \mathbb{N}_m} \sum_{\ell,q \in \mathbb{N}_n} c_{i\ell} c_{jq} G(z_\ell(x_{i\ell}), z_q(x_{jq})) = \sum_{i,\ell} \sum_{jq} c_{i\ell} c_{jq} G(\tilde{z}_{i\ell}, \tilde{z}_{jq}) \geq 0$$

where we have defined $\tilde{z}_{i\ell} = z_\ell(x_{i\ell})$ and the last inequality follows by the hypothesis that G is a kernel. □

For the special case that $\mathcal{T} = \mathbb{R}^p$, $z_\ell(x) = B_\ell x$ with B_ℓ a $p \times d$ matrix, $\ell \in \mathbb{N}_n$, and $G : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ is the homogeneous polynomial kernel, $G(t,s) = (t's)^r$, the lemma confirms that the function (33) is a multi-task kernel. Similarly, when G is chosen to be a Gaussian kernel, we conclude that

$$K((x,\ell),(t,q)) = \exp(-\beta \|B_\ell x - B_q t\|^2)$$

is a multi-task kernel for every $\beta > 0$.

Lemma 2 also allows us to generalize multi-task learning to the case that each task function f_ℓ has a *different* input domain $\mathcal{X}_\ell$, a situation which is important in applications, see, for example, (Ben-David, Gehrke, and Schuller, 2002) for a discussion. To this end, we specify sets $\mathcal{X}_\ell$, $\ell \in \mathbb{N}_n$, functions $g_\ell : \mathcal{X}_\ell \to \mathbb{R}$, and note that multi–task learning can be placed in the above framework by defining the input space

$$\mathcal{X} := \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_n.$$

We are interested in the functions $f_\ell(x) = g_\ell(P_\ell x)$, where $x = (x_1, \ldots, x_n)$ and $P_\ell : \mathcal{X} \to \mathcal{X}_\ell$ is defined, for every $x \in \mathcal{X}$ by $P_\ell(x) = x_\ell$. Let G be a kernel on $\mathcal{T} \times \mathcal{T}$ and choose $z_\ell(\cdot) = \phi_\ell(P_\ell(\cdot))$ where $\phi_\ell : \mathcal{X}_\ell \to \mathcal{T}$ are some prescribed functions. Then by lemma 2 the kernel defined by Equation (34) can be used to represent the functions g_ℓ. In particular, in the case of linear functions, we choose $\mathcal{X}_\ell = \mathbb{R}^{d_\ell}$, where $d_\ell \in \mathbb{N}$, $\mathcal{T} = \mathbb{R}^p$, $p \in \mathbb{N}$, $G(s,t) = s't$ and $z_\ell = D_\ell P_\ell$ where D_ℓ is a $p \times d_\ell$ matrix. In this case, the multi-task kernel is given by

$$K((x,\ell),(t,q)) = x' P_\ell' D_\ell' D_q P_q t$$

which is of the form in Equation (10) for $B_\ell = D_\ell P_\ell$, $\ell \in \mathbb{N}_n$.

We note that ideas related to those presented in this section appear in (Girosi, 2003).

5.2 Conclusion and Future Work

We developed a framework for multi-task learning in the context of regularization in reproducing kernel Hilbert spaces. This naturally extends standard single-task kernel learning methods, such as SVM and RN. The framework allows to model relationships between the tasks and to learn the task parameters simultaneously. For this purpose, we showed that multi-task learning can be seen as single-task learning if a particular family of kernels, that we called multi-task kernels, is used. We also characterized the non-linear multi-task kernels.

Within the proposed framework, we defined particular linear multi-task kernels that correspond to particular choices of regularizers which model relationships between the function parameters. For example, in the case of SVM, appropriate choices of this kernel/regularizer implemented a trade–off between large margin of each per–task individual SVM and closeness of each SVM to linear combinations of the individual SVMs such as their average.

We tested some of the proposed methods using real data. The experimental results show that the proposed multi-task learning methods can lead to significant performance improvements relative to the single-task learning methods, especially when many tasks with few data each are learned.

A number of research questions can be studied starting from the framework and methods we developed here. We close with commenting on some issues which stem out of the main theme of this paper.

- *Learning a multi-task kernel.* The kernel in Equation (22) is perhaps the simplest nontrivial example of a multi-task kernel. This kernel is a convex combination of two kernels, the first of which corresponds to learning independent tasks and the second one is a rank one kernel which corresponds to learning all tasks as the same task. Thus this kernel linearly combines two opposite models to form a more flexible one. Our experimental results above indicate the value of this approach provided the parameter λ is chosen for the application at hand. Recent work by Micchelli and Pontil (2004) shows that, under rather general conditions, the optimal convex combination of kernels can be learned by minimizing the functional in Equation (1) with respect to K and $f \in \mathcal{H}_K$, where K is a kernel in the convex set of kernels, see also (Lanckriet et al., 2004). Indeed, in our specific case we can show—along the lines in (Micchelli and Pontil, 2004)—that the regularizer (24) is convex in λ and u. This approach is rather general and can be adapted also for learning the matrix Q in the kernel in Equation (25) which in our experiment we estimated by our "ad hoc" PCA approach.

- *Bounds on the generalization error.* Yet another important question is how to bound the generalization error for multi-task learning. Recently developed bounds relying on the notion of algorithmic stability or Rademacher complexity should be easily applicable to our context. This should highlight the role played by the matrices B_ℓ in Equation (10). Intuitively, if $B_\ell = B_0$ we should have a simple (low-complexity) model whereas if the B_ℓ are orthogonal a more complex model. More specifically, this analysis should say how the generalization error, when using the kernel (22), depends on λ.

- *Computational considerations.* A drawback of our proposed multi-task kernel method is that its computational complexity time is $O(p(mn))$ which is worst than the complexity of solving n independent kernel methods, this being $nO(p(m))$. The function p depends on the loss function used and, typically, $p(m) = m^a$ with a a positive constant. For example for the square loss $a = 3$. Future work will focus on the study of efficient decomposition methods for solving the multi-task SVM or RN. This decomposition should exploit the structure provided by the matrices B_ℓ in the kernel (10). For example, if we use the kernel (22) and the tasks share the same input examples it is possible to show that the linear system of mn Equations (15) can be reduced to solving $n+1$ systems of m equations, which is essentially the same as solving n independent ridge regression problems.

- *Multi-task feature selection.* Continuing on the discussion above, we observe that if we restrict the matrix Q to be diagonal then learning Q corresponds to a form of feature selection across tasks. Other feature selection formulations where the tasks may share only some of their features should also be possible. See also the recent work by Jebara (2004) for related work on this direction.

- *Online multi-task learning.* An interesting problem deserving of investigation is the question of how to learn a set of tasks online where at each instance of time a set of examples for a *new task* is sampled. This problem is valuable in applications where an environment is explored and new data/tasks are provided during this exploration. For example, the environment could be a market of customers in our application above, or a set of scenes in computer vision which contains different objects we want to recognize.

- *Multi-task learning extensions.* Finally it would be interesting to extent the framework presented here to other learning problems beyond classification and regression. Two example which come to mind are kernel density estimation, see, for example, (Vapnik, 1998), or one-class SVM (Tax and Duin, 1999).

Acknowledgments

The authors would like to thank Phillip Cartwright and Simon Trusler from Research International for their help with this data set.

References

G. M. Allenby and P. E. Rossi. Marketing models of consumer heterogeneity. *Journal of Econometrics, 89, p. 57–78*, 1999.

R. K. Ando and T. Zhang. A Framework for Learning Predictive Structures from Multiple Tasks and Unlabeled Data. Technical Report RC23462, IBM T.J. Watson Research Center, 2004.

N. Arora G.M Allenby, and J. Ginter. A hierarchical Bayes model of primary and secondary demand. *Marketing Science*, 17,1, p. 29–44, 1998.

N. Aronszajn. Theory of reproducing kernels. *Trans. Amer. Math. Soc.*, 686, pp. 337–404, 1950.

B. Bakker and T. Heskes. Task clustering and gating for Bayesian multi–task learning. *Journal of Machine Learning Research*, 4: 83–99, 2003.

J. Baxter. A Bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine Learning, 28, pp. 7–39*, 1997.

J. Baxter. A model for inductive bias learning. *Journal of Artificial Intelligence Research, 12, p. 149–198*, 2000.

S. Ben-David, J. Gehrke, and R. Schuller. A theoretical framework for learning from a pool of disparate data sources. Proceedings of Knowledge Discovery and Datamining (KDD), 2002.

S. Ben-David and R. Schuller. Exploiting task relatedness for multiple task learning. Proceedings of Computational Learning Theory (COLT), 2003.

L. Breiman and J.H Friedman. Predicting multivariate responses in multiple linear regression. *Royal Statistical Society Series B*, 1998.

P. J. Brown and J. V. Zidek. Adaptive multivariate ridge regression. *The Annals of Statistics, Vol. 8, No. 1, p. 64–74*, 1980.

R. Caruana. Multi–task learning. *Machine Learning, 28, p. 41–75*, 1997.

F. R. K. Chung. *Spectral Graph Theory* CBMS Series, AMS, Providence, 1997.

T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13:1–50, 2000.

T. Evgeniou, C. Boussios, and G. Zacharia. Generalized robust conjoint estimation. *Marketing Science*, 2005 (forthcoming).

T. Evgeniou and M. Pontil. Regularized multi-task learning. Proceedings of the 10th Conference on '*Knowledge Discovery and Data Mining*, Seattle, WA, August 2004.

F. Girosi. *Demographic Forecasting*. PhD Thesis, Harvard University, 2003.

W. Greene. *Econometric Analysis*. Prentice Hall, fifth edition, 2002.

B. Heisele, T. Serre, M. Pontil, T. Vetter, and T. Poggio. Categorization by learning and combining object parts. In *Advances in Neural Information Processing Systems 14*, Vancouver, Canada, Vol. 2, 1239–1245, 2002.

T. Heskes. Empirical Bayes for learning to learn. Proceedings of ICML–2000, ed. Langley, P., pp. 367–374, 2000.

T. Jebara. Multi-Task Feature and Kernel Selection for SVMs. International Conference on Machine Learning, ICML, July 2004.

M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 1993.

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, pp. 27–72, 2004.

G. R. G. Lanckriet, T. De Bie, N. Cristianini, M. I. Jordan, and W. S. Noble. A framework for genomic data fusion and its application to membrane protein prediction. Technical Report CSD–03–1273, Division of Computer Science, University of California, Berkeley, 2003.

O. L. Mangasarian. *Nonlinear Programming*. Classics in Applied Mathematics. SIAM, 1994.

C. A. Micchelli and M. Pontil. Learning the kernel via regularization. Research Note RN/04/11, Dept of Computer Science, UCL, September, 2004.

C. A. Micchelli and M. Pontil. On learning vector–valued functions. *Neural Computation*, 17, pp. 177–204, 2005.

C. A. Micchelli and M. Pontil. Kernels for multi-task learning. Proc. of the 18–th Conf. on Neural Information Processing Systems, 2005.

R. Rifkin, S. Mukherjee, P. Tamayo, S. Ramaswamy, C. Yeang, M. Angelo, M. Reich, T. Poggio, T. Poggio, E. Lander, T. Golub, and J. Mesirov. An analytical method for multi-class molecular cancer classification *SIAM Review*, Vol. 45, No. 4, p. 706-723, 2003.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis.* Cambridge University Press, 2004.

B. Schölkopf and A. J. Smola. *Learning with Kernels*. The MIT Press, Cambridge, MA, USA, 2002.

D. L. Silver and R.E Mercer. The parallel transfer of task knowledge using dynamic learning rates based on a measure of relatedness. *Connection Science, 8, p. 277–294*, 1996.

V. Srivastava and T. Dwivedi. Estimation of seemingly unrelated regression equations: A brief survey *Journal of Econometrics*, 10, p. 15–32, 1971.

D. M. J. Tax and R. P. W. Duin. Support vector domain description. *Pattern Recognition Letters*, 20 (11-13), pp. 1191–1199, 1999.

S. Thrun and L. Pratt. *Learning to Learn*. Kluwer Academic Publishers, November 1997.

S. Thrun and J. O'Sullivan. Clustering learning tasks and the selective cross–task transfer of knowledge. *In Learning to Learn*, S. Thrun and L. Y. Pratt Eds., Kluwer Academic Publishers, 1998.

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.

A. Zellner. An efficient method for estimating seemingly unrelated regression equations and tests for aggregation bias. *Journal of the American Statistical Association*, 57, p. 348–368, 1962.

Journal of Machine Learning Research 6 (2005) 639–660 Submitted 10/04; Revised 3/05; Published 4/05

Adaptive Online Prediction by Following the Perturbed Leader

Marcus Hutter MARCHUS@IDSIA.CH
Jan Poland JAN@IDSIA.CH
IDSIA, Galleria 2
6928 Manno-Lugano, Switzerland

Editor: Manfred Warmuth

Abstract

When applying aggregating strategies to Prediction with Expert Advice (PEA), the learning rate must be adaptively tuned. The natural choice of $\sqrt{\text{complexity/current loss}}$ renders the analysis of Weighted Majority (WM) derivatives quite complicated. In particular, for arbitrary weights there have been no results proven so far. The analysis of the alternative Follow the Perturbed Leader (FPL) algorithm from Kalai and Vempala (2003) based on Hannan's algorithm is easier. We derive loss bounds for adaptive learning rate and both finite expert classes with uniform weights and countable expert classes with arbitrary weights. For the former setup, our loss bounds match the best known results so far, while for the latter our results are new.

Keywords: prediction with expert advice, follow the perturbed leader, general weights, adaptive learning rate, adaptive adversary, hierarchy of experts, expected and high probability bounds, general alphabet and loss, online sequential prediction

1. Introduction

In Prediction with Expert Advice (PEA) one considers an ensemble of sequential predictors (experts). A master algorithm is constructed based on the historical performance of the predictors. The goal of the master algorithm is to perform nearly as well as the best expert in the class, on any sequence of outcomes. This is achieved by making (randomized) predictions close to the better experts.

PEA theory has rapidly developed in the recent past. Starting with the Weighted Majority (WM) algorithm of Littlestone and Warmuth (1989, 1994) and the aggregating strategy of Vovk (1990), a vast variety of different algorithms and variants have been published. A key parameter in all these algorithms is the *learning rate*. While this parameter had to be fixed in the early algorithms such as WM, Cesa-Bianchi et al. (1997) established the so-called doubling trick to make the learning rate coarsely adaptive. A little later, incrementally adaptive algorithms were developed by Auer and Gentile (2000); Auer et al. (2002); Yaroshinsky et al. (2004); Gentile (2003), and others. In Section 10, we will compare our results with these works more in detail. Unfortunately, the loss bound proofs for the incrementally adaptive WM variants are quite complex and technical, despite the typically simple and elegant proofs for a static learning rate.

The complex growing proof techniques also had another consequence. While for the original WM algorithm, assertions are proven for countable classes of experts with arbitrary weights, the modern variants usually restrict to finite classes with uniform weights (an exception being Gentile

(2003); see the discussion section therein.) This might be sufficient for many practical purposes but it prevents the application to more general classes of predictors. Examples are extrapolating (=predicting) data points with the help of a polynomial (=expert) of degree $d = 1, 2, 3, \ldots$ –or– the (from a computational point of view largest) class of all computable predictors. Furthermore, most authors have concentrated on predicting *binary* sequences, often with the 0/1 loss for $\{0, 1\}$-valued and the absolute loss for $[0, 1]$-valued predictions. Arbitrary losses are less common. Nevertheless, it is easy to abstract completely from the predictions and consider the resulting losses only. Instead of predicting according to a "weighted majority" in each time step, one chooses one *single* expert with a probability depending on his past cumulated loss. This is done e.g. by Freund and Schapire (1997), where an elegant WM variant, the Hedge algorithm, is analyzed.

A different, general approach to achieve similar results is Follow the Perturbed Leader (FPL). The principle dates back to as early as 1957, now called Hannan's algorithm (Hannan, 1957). In 2003, Kalai and Vempala published a simpler proof of the main result of Hannan and also succeeded to improve the bound by modifying the distribution of the perturbation. The resulting algorithm (which they call FPL*) has the same performance guarantees as the WM-type algorithms for fixed learning rate, save for a factor of $\sqrt{2}$. A major advantage we will discover in this work is that its analysis remains easy for an adaptive learning rate, in contrast to the WM derivatives. Moreover, it generalizes to online decision problems other than PEA.

In this work,[1] we study the FPL algorithm for PEA. The problems of WM algorithms mentioned above are addressed. Bounds on the cumulative regret of the standard form $\sqrt{kL}$ (where k is the complexity and L is the cumulative loss of the best expert in hindsight) are shown for countable expert classes with arbitrary weights, adaptive learning rate, and arbitrary losses. Regarding the adaptive learning rate, we obtain proofs that are simpler and more elegant than for the corresponding WM algorithms. (In particular, the proof for a self-confident choice of the learning rate, Theorem 7, is less than half a page.) Further, we prove the first loss bounds for *arbitrary weights* and adaptive learning rate. In order to obtain the optimal $\sqrt{kL}$ bound in this case, we will need to introduce a hierarchical version of FPL, while without hierarchy we show a worse bound $k\sqrt{L}$. (For self-confident learning rate together with uniform weights and arbitrary losses, one can prove corresponding results for a variant of WM by adapting an argument by Auer et al. 2002.)

PEA usually refers to an *online worst case* setting: n experts that deliver sequential predictions over a time range $t = 1, \ldots, T$ are given. At each time t, we know the actual predictions and the *past* losses. The goal is to give a prediction such that the overall loss after T steps is "not much worse" than the best expert's loss *on any sequence of outcomes*. If the prediction is deterministic, then an adversary could choose a sequence which provokes maximal loss. So we have to *randomize* our predictions. Consequently, we ask for a prediction strategy such that the *expected* loss on any sequence is small.

This paper is structured as follows. In Section 2 we give the basic definitions. While Kalai and Vempala consider general online decision problems in finite-dimensional spaces, we focus on online prediction tasks based on a countable number of experts. Like Kalai and Vempala (2003) we exploit the infeasible FPL predictor (IFPL) in our analysis. Sections 3 and 4 derive the main analysis tools. In Section 3 we generalize (and marginally improve) the upper bound (Kalai and Vempala, 2003, Lem.3) on IFPL to arbitrary weights. The main difficulty we faced was to appropriately distribute the weights to the various terms. For the corresponding lower bound (Section 7) this

1. A shorter version appeared in the proceedings of the ALT 2004 conference (Hutter and Poland, 2004).

is an open problem. In Section 4 we exploit our restricted setup to significantly improve (Kalai and Vempala, 2003, Eq.(3)) allowing for bounds logarithmic rather than linear in the number of experts. The upper and lower bounds on IFPL are combined to derive various regret bounds on FPL in Section 5. Bounds for static and dynamic learning rate in terms of the sequence length follow straight-forwardly. The proof of our main bound in terms of the loss is much more elegant than the analysis of previous comparable results. Section 6 proposes a novel hierarchical procedure to improve the bounds for non-uniform weights. In Section 7, a lower bound is established. In Section 8, we consider the case of independent randomization more seriously. In particular, we show that the derived bounds also hold for an adaptive adversary. Section 9 treats some additional issues, including bounds with high probability, computational aspects, deterministic predictors, and the absolute loss. Finally, in Section 10 we discuss our results, compare them to references, and state some open problems.

2. Setup and Notation

Setup. Prediction with expert advice proceeds as follows. We are asked to perform sequential predictions $y_t \in \mathcal{Y}$ at times $t = 1,2,\ldots$. At each time step t, we have access to the predictions $(y_t^i)_{1\le i\le n}$ of n experts $\{e_1,...,e_n\}$, where the size of the expert pool is $n \in \mathbb{N}\cup\{\infty\}$. It is convenient to use the same notation for finite ($n \in \mathbb{N}$) and countably infinite ($n = \infty$) expert pool. After having made a prediction, we make some observation $x_t \in \mathcal{X}$, and a Loss is revealed for our and each expert's prediction. (E.g. the loss might be 1 if the expert made an erroneous prediction and 0 otherwise. This is the 0/1 loss.) Our goal is to achieve a total loss "not much worse" than the best expert, after t time steps.

We admit $n \in \mathbb{N}\cup\{\infty\}$ experts, each of which is assigned a known complexity $k^i \ge 0$. Usually we require $\sum_i \mathrm{e}^{-k^i} \le 1$, which implies that the k^i are valid lengths of prefix code words, for instance $k^i = \ln n$ if $n < \infty$ or $k^i = \frac{1}{2} + 2\ln i$ if $n = \infty$. Each complexity defines a weight by means of e^{-k^i} and vice versa. In the following we will talk of complexities rather than of weights. If n is finite, then usually one sets $k^i = \ln n$ for all i; this is the case of *uniform complexities/weights*. If the set of experts is countably infinite ($n = \infty$), uniform complexities are not possible. The vector of all complexities is denoted by $k = (k^i)_{1\le i\le n}$. At each time t, each expert i suffers a loss[2] $s_t^i = \text{Loss}(x_t, y_t^i) \in [0,1]$, and $s_t = (s_t^i)_{1\le i\le n}$ is the vector of all losses at time t. Let $s_{<t} = s_1 + \ldots + s_{t-1}$ (respectively $s_{1:t} = s_1 + \ldots + s_t$) be the total past loss vector (including current loss s_t) and $s_{1:t}^{min} = \min_i\{s_{1:t}^i\}$ be the loss of the *best expert in hindsight (BEH)*. Usually we do not know in advance the time $t \ge 0$ at which the performance of our predictions are evaluated.

General decision spaces. The setup can be generalized as follows. Let $\mathcal{S} \subset \mathbb{R}^n$ be the *state space* and $\mathcal{D} \subset \mathbb{R}^n$ the *decision space*. At time t the state is $s_t \in \mathcal{S}$, and a decision $d_t \in \mathcal{D}$ (which is made before the state is revealed) incurs a loss $d_t \circ s_t$, where "$\circ$" denotes the inner product. This implies that the loss function is *linear* in the states. Conversely, each linear loss function can be represented in this way. The decision which minimizes the loss in state $s \in \mathcal{S}$ is

$$M(s) := \arg\min_{d\in\mathcal{D}}\{d \circ s\} \tag{1}$$

if the minimum exists. The application of this general framework to PEA is straightforward: $\mathcal{D}$ is identified with the space of all unit vectors $\mathcal{E} = \{e_i : 1 \le i \le n\}$, since a decision consists of selecting

2. The setup, analysis and results easily scale to $s_t^i \in [0,S]$ for $S > 0$ other than 1.

a single expert, and $s_t \in [0,1]^n$, so states are identified with losses. Only Theorems 2 and 10 will be stated in terms of general decision space. Our main focus is $\mathcal{D} = \mathcal{E}$. (Even for this special case, the scalar product notation is not too heavy, but will turn out to be convenient.) All our results generalize to the simplex $\mathcal{D} = \Delta = \{v \in [0,1]^n : \sum_i v^i = 1\}$, since the minimum of a linear function on Δ is always attained on $\mathcal{E}$.

Follow the Perturbed Leader. Given $s_{<t}$ at time t, an immediate idea to solve the expert problem is to "Follow the Leader" (FL), i.e. selecting the expert e_i which performed best in the past (minimizes $s^i_{<t}$), that is predict according to expert $M(s_{<t})$. This approach fails for two reasons. First, for $n = \infty$ the minimum in (1) may not exist. Second, for $n = 2$ and $s = \binom{0\,1\,0\,1\,0\,1\ldots}{\frac{1}{2}\,0\,1\,0\,1\,0\ldots}$, FL always chooses the wrong prediction (Kalai and Vempala, 2003). We solve the first problem by penalizing each expert by its complexity, i.e. predicting according to expert $M(s_{<t} + k)$. The *FPL (Follow the Perturbed Leader)* approach solves the second problem by adding to each expert's loss $s^i_{<t}$ a random perturbation. We choose this perturbation to be negative *exponentially distributed*, either independent in each time step or once and for all at the very beginning at time $t = 0$. The former choice is preferable in order to protect against an adaptive adversary who generates the s_t, and in order to get bounds with high probability (Section 9). For the main analysis however, the latter choice is more convenient. Due to linearity of expectations, these two possibilities are equivalent when dealing with *expected losses* (this is straightforward for oblivious adversary, for adaptive adversary see Section 8), so we can henceforth assume without loss of generality one initial perturbation q.

The FPL algorithm is defined as follows:

```
Choose random vector $q \overset{d.}{\sim} \exp$, i.e. $P[q^1 \ldots q^n] = e^{-q^1} \cdot \ldots \cdot e^{-q^n}$ for $q \geq 0$.
For $t = 1, \ldots, T$
- Choose learning rate $\eta_t$.
- Output prediction of expert $i$ which minimizes $s^i_{<t} + (k^i - q^i)/\eta_t$.
- Receive loss $s^i_t$ for all experts $i$.
```

Other than $s_{<t}$, k and q, FPL depends on the *learning rate* η_t. We will give choices for η_t in Section 5, after having established the main tools for the analysis. The expected loss at time t of FPL is $\ell_t := E[M(s_{<t} + \frac{k-q}{\eta_t}) \circ s_t]$. The key idea in the FPL analysis is the use of an intermediate predictor *IFPL* (for *Implicit or Infeasible FPL*). IFPL predicts according to $M(s_{1:t} + \frac{k-q}{\eta_t})$, thus under the knowledge of s_t (which is of course not available in reality). By $r_t := E[M(s_{1:t} + \frac{k-q}{\eta_t}) \circ s_t]$ we denote the expected loss of IFPL at time t. The losses of IFPL will be upper-bounded by BEH in Section 3 and lower-bounded by FPL in Section 4. Note that our definition of the FPL algorithm deviates from that of Kalai and Vempala. It uses an exponentially distributed perturbation similar to their FPL* but one-sided and a non-stationary learning rate like Hannan's algorithm.

Notes. Observe that we have stated the FPL algorithm regardless of the actual *predictions* of the experts and possible *observations*, only the *losses* are relevant. Note also that an expert can implement a highly complicated strategy depending on past outcomes, despite its trivializing identification with a constant unit vector. The complex expert's (and environment's) behavior is summarized and hidden in the state vector $s_t = \text{Loss}(x_t, y^i_t)_{1 \leq i \leq n}$. Our results therefore apply to *arbitrary prediction and observation spaces $\mathcal{Y}$ and $\mathcal{X}$ and arbitrary bounded loss functions*. This is in contrast to the major part of PEA work developed for binary alphabet and 0/1 or absolute loss only. Finally note that the setup allows for losses generated by an adversary who tries to maximize the regret of FPL and knows the FPL algorithm and all experts' past predictions/losses. If the adversary also has access

Symbol	Definition / Explanation
n	$\in \mathbf{N}\cup\{\infty\}$ ($n=\infty$ means countably infinite $\mathcal{E}$). Number of experts.
x^i	= ith component of vector $x\in\mathbf{R}^n$.
$\mathcal{E}$	$:=\{e_i:1\leq i\leq n\}=$ set of unit vectors ($e_i^j=\delta_{ij}$).
Δ	$:=\{v\in[0,1]^n:\sum_i v^i=1\}$= simplex.
$s_t\in[0,1]^n$	= environmental state/loss vector at time t.
$s_{1:t}$	$:=s_1+...+s_t$= state/loss (similar for ℓ_t and r_t).
$s_{1:T}^{min}$	$=\min_i\{s_{1:T}^i\}$= loss of Best Expert in Hindsight (BEH).
$s_{<t}$	$:=s_1+...+s_{t-1}$= state/loss summary ($s_{<0}=0$).
$M(s)$	$:=\text{argmin}_{d\in\mathcal{D}}\{d\circ s\}$= best decision on s.
$T\in\mathbf{N}_0$	= total time=step, $t\in\mathbf{N}$= current time=step.
$k^i\geq 0$	= penalization = complexity of expert i.
$q\in\mathbf{R}^n$	= random vector with independent exponentially distributed components.
I_t	$:=\text{argmin}_{i\in\mathcal{E}}\{s_{<t}^i+\frac{k^i-q^i}{\eta_t}\}$= randomized prediction of FPL.
ℓ_t	$:=E[M(s_{<t}+\frac{k-q}{\eta_t})\circ s_t]$= expected loss at time t of FPL (=$E[s_t^{I_t}]$ for $\mathcal{D}=\mathcal{E}$).
r_t	$:=E[M(s_{1:t}+\frac{k-q}{\eta_t})\circ s_t]$= expected loss at time t of IFPL.
u_t	$:=M(s_{<t}+\frac{k-q}{\eta_t})\circ s_t$= actual loss at time t of FPL (=$s_t^{I_t}$ for $\mathcal{D}=\mathcal{E}$).

Table 1: List of notation.

to FPL's past decisions, then FPL must use independent randomization at each time step in order to achieve good regret bounds. Table 1 summarizes notation.

Motivation of FPL. Let $d(s_{<t})$ be any predictor with decision based on $s_{<t}$. The following identity is easy to show:

$$\underbrace{\sum_{t=1}^T d(s_{<t})\circ s_t}_{\text{“FPL”}} \;\equiv\; \underbrace{d(s_{1:T})\circ s_{1:T}}_{\text{“BEH”}} + \overbrace{\underbrace{\sum_{t=1}^T [d(s_{<t})-d(s_{1:t})]\circ s_{<t}}_{\text{“IFPL}-\text{BEH”}}}^{\leq 0 \text{ if } d\approx M} + \overbrace{\underbrace{\sum_{t=1}^T [d(s_{<t})-d(s_{1:t})]\circ s_t}_{\text{“FPL}-\text{IFPL”}}}^{\text{small if } d(\cdot) \text{ is continuous}}\,. \tag{2}$$

For a good bound of FPL in terms of BEH we need the first term on the r.h.s. to be close to BEH and the last two terms to be small. The first term is close to BEH if $d\approx M$. The second to last term is even negative if $d=M$, hence small if $d\approx M$. The last term is small if $d(s_{<t})\approx d(s_{1:t})$, which is the case if $d(\cdot)$ is a sufficiently smooth function. Randomization smoothes the discontinuous function M: The function $d(s):=E[M(s-q)]$, where $q\in\mathbf{R}^n$ is some random perturbation, is a continuous function in s. If the mean and variance of q are small, then $d\approx M$, if the variance of q is large, then $d(s_{<t})\approx d(s_{1:t})$. An intermediate variance makes the last two terms of (2) simultaneously small enough, leading to excellent bounds for FPL.

3. IFPL bounded by Best Expert in Hindsight

In this section we provide tools for comparing the loss of IFPL to the loss of the best expert in hindsight. The first result bounds the expected error induced by the exponentially distributed perturbation.

Lemma 1 (Maximum of Shifted Exponential Distributions) *Let $q^1,...,q^n$ be (not necessarily independent) exponentially distributed random variables, i.e. $P[q^i]=\mathrm{e}^{-q^i}$ for $q^i\geq 0$ and $1\leq i\leq n\leq\infty$, and $k^i\in\boldsymbol{R}$ be real numbers with $u:=\sum_{i=1}^n\mathrm{e}^{-k^i}$. Then*

$$\begin{aligned}
P[\max_i\{q^i-k^i\}\geq a] &= 1-\prod_{i=1}^n\max\{0,1-\mathrm{e}^{-a-k^i}\} \quad \textit{if} \quad q^1,...,q^n \textit{ are independent},\\
P[\max_i\{q^i-k^i\}\geq a] &\leq \min\{1,u\,\mathrm{e}^{-a}\},\\
E[\max_i\{q^i-k^i\}] &\leq 1+\ln u.
\end{aligned}$$

Proof. Using

$$P[q^i<a]=\max\{0,1-\mathrm{e}^{-a}\}\geq 1-\mathrm{e}^{-a} \quad \text{and} \quad P[q^i\geq a]=\min\{1,\mathrm{e}^{-a}\}\leq\mathrm{e}^{-a},$$

valid for any $a\in\boldsymbol{R}$, the exact expression for $P[\max]$ in Lemma 1 follows from

$$P[\max_i\{q^i-k^i\}<a]=P[q^i-k^i<a\ \forall i]=\prod_{i=1}^n P[q^i<a+k^i]=\prod_{i=1}^n\max\{0,\mathrm{e}^{-a-k^i}\},$$

where the second equality follows from the independence of the q^i. The bound on $P[\max]$ for any $a\in\boldsymbol{R}$ (including negative a) follows from

$$P[\max_i\{q^i-k^i\}\geq a]=P[\exists i:q^i-k^i\geq a]\leq\sum_{i=1}^n P[q^i-k^i\geq a]\leq\sum_{i=1}^n\mathrm{e}^{-a-k^i}=u\cdot\mathrm{e}^{-a}$$

where the first inequality is the union bound. Using $E[z]\leq E[\max\{0,z\}]=\int_0^\infty P[\max\{0,z\}\geq y]dy=\int_0^\infty P[z\geq y]dy$ (valid for any real-valued random variable z) for $z=\max_i\{q^i-k^i\}-\ln u$, this implies

$$E[\max_i\{q^i-k^i\}-\ln u]\leq\int_0^\infty P[\max_i\{q^i-k^i\}\geq y+\ln u]dy\leq\int_0^\infty\mathrm{e}^{-y}dy\ =\ 1,$$

which proves the bound on $E[\max]$. □

If n is finite, a lower bound $E[\max_i q^i]\geq 0.57721+\ln n$ can be derived, showing that the upper bound on $E[\max]$ is quite tight (at least) for $k^i=0\ \forall i$. The following bound generalizes (Kalai and Vempala, 2003, Lem.3) to arbitrary weights, establishing a relation between IFPL and the best expert in hindsight.

Theorem 2 (IFPL bounded by BEH) *Let $\mathcal{D}\subseteq\boldsymbol{R}^n$, $s_t\in\boldsymbol{R}^n$ for $1\leq t\leq T$ (both $\mathcal{D}$ and s may even have negative components, but we assume that all required extrema are attained), and $q,k\in\boldsymbol{R}^n$. If $\eta_t>0$ is decreasing in t, then the loss of the infeasible FPL knowing s_t at time t in advance (l.h.s.) can be bounded in terms of the best predictor in hindsight (first term on r.h.s.) plus additive corrections:*

$$\sum_{t=1}^T M(s_{1:t}+\frac{k-q}{\eta_t})\circ s_t\leq\min_{d\in\mathcal{D}}\{d\circ(s_{1:T}+\frac{k}{\eta_T})\}+\frac{1}{\eta_T}\max_{d\in\mathcal{D}}\{d\circ(q-k)\}-\frac{1}{\eta_T}M(s_{1:T}+\frac{k}{\eta_T})\circ q.$$

Note that if $\mathcal{D}=\mathcal{E}$ (or $\mathcal{D}=\Delta$) and $s_t\geq 0$, then all extrema in the theorem are attained almost surely. The same holds for all subsequent extrema in the proof and throughout the paper.

Proof. For notational convenience, let $\eta_0=\infty$ and $\tilde{s}_{1:t}=s_{1:t}+\frac{k-q}{\eta_t}$. Consider the losses $\tilde{s}_t=s_t+(k-q)(\frac{1}{\eta_t}-\frac{1}{\eta_{t-1}})$ for the moment. We first show by induction on T that the infeasible predictor $M(\tilde{s}_{1:t})$ has zero regret for any loss $\tilde{s}$, i.e.

$$\sum_{t=1}^{T} M(\tilde{s}_{1:t})\circ\tilde{s}_t \leq M(\tilde{s}_{1:T})\circ\tilde{s}_{1:T}. \tag{3}$$

For $T=1$ this is obvious. For the induction step from $T-1$ to T we need to show

$$M(\tilde{s}_{1:T})\circ\tilde{s}_T \leq M(\tilde{s}_{1:T})\circ\tilde{s}_{1:T} - M(\tilde{s}_{<T})\circ\tilde{s}_{<T}. \tag{4}$$

This follows from $\tilde{s}_{1:T}=\tilde{s}_{<T}+\tilde{s}_T$ and $M(\tilde{s}_{1:T})\circ\tilde{s}_{<T}\geq M(\tilde{s}_{<T})\circ\tilde{s}_{<T}$ by minimality of M. Rearranging terms in (3), we obtain

$$\sum_{t=1}^{T} M(\tilde{s}_{1:t})\circ s_t \;\leq\; M(\tilde{s}_{1:T})\circ\tilde{s}_{1:T} - \sum_{t=1}^{T} M(\tilde{s}_{1:t})\circ(k-q)\Big(\frac{1}{\eta_t}-\frac{1}{\eta_{t-1}}\Big) \tag{5}$$

Moreover, by minimality of M,

$$\begin{aligned} M(\tilde{s}_{1:T})\circ\tilde{s}_{1:T} &\leq M\Big(s_{1:T}+\frac{k}{\eta_T}\Big)\circ\Big(s_{1:T}+\frac{k-q}{\eta_T}\Big) \\ &= \min_{d\in\mathcal{D}}\Big\{d\circ(s_{1:T}+\frac{k}{\eta_T})\Big\} - M\Big(s_{1:T}+\frac{k}{\eta_T}\Big)\circ\frac{q}{\eta_T} \end{aligned} \tag{6}$$

holds. Using $\frac{1}{\eta_t}-\frac{1}{\eta_{t-1}}\geq 0$ and again minimality of M, we have

$$\begin{aligned} \sum_{t=1}^{T}(\frac{1}{\eta_t}-\frac{1}{\eta_{t-1}})M(\tilde{s}_{1:t})\circ(q-k) &\leq \sum_{t=1}^{T}(\frac{1}{\eta_t}-\frac{1}{\eta_{t-1}})M(k-q)\circ(q-k) \\ &= \frac{1}{\eta_T}M(k-q)\circ(q-k) = \frac{1}{\eta_T}\max_{d\in\mathcal{D}}\{d\circ(q-k)\}. \end{aligned} \tag{7}$$

Inserting (6) and (7) back into (5) we obtain the assertion. □

Assuming q random with $E[q^i]=1$ and taking the expectation in Theorem 2, the last term reduces to $-\frac{1}{\eta_T}\sum_{i=1}^{n}M(s_{1:T}+\frac{k}{\eta_T})^i$. If $\mathcal{D}\geq 0$, the term is negative and may be dropped. In case of $\mathcal{D}=\mathcal{E}$ or Δ, the last term is identical to $-\frac{1}{\eta_T}$ (since $\sum_i d^i=1$) and keeping it improves the bound. Furthermore, we need to evaluate the expectation of the second to last term in Theorem 2, namely $E[\max_{d\in\mathcal{D}}\{d\circ(q-k)\}]$. For $\mathcal{D}=\mathcal{E}$ and q being exponentially distributed, using Lemma 1, the expectation is bounded by $1+\ln u$. We hence get the following bound:

Corollary 3 (IFPL bounded by BEH) *For $\mathcal{D}=\mathcal{E}$ and $\sum_i e^{-k^i}\leq 1$ and $P[q^i]=e^{-q^i}$ for $q\geq 0$ and decreasing $\eta_t>0$, the expected loss of the infeasible FPL exceeds the loss of expert i by at most k^i/η_T:*

$$r_{1:T} \;\leq\; s^i_{1:T}+\frac{1}{\eta_T}k^i \quad \forall i.$$

Theorem 2 can be generalized to expert dependent factorizable $\eta_t \rightsquigarrow \eta_t^i = \eta_t \cdot \eta^i$ by scaling $k^i \rightsquigarrow k^i/\eta^i$ and $q^i \rightsquigarrow q^i/\eta^i$. Using $E[\max_i\{\frac{q^i-k^i}{\eta^i}\}] \leq E[\max_i\{q^i-k^i\}]/\min_i\{\eta^i\}$, Corollary 3, generalizes to

$$E[\sum_{t=1}^{T} M(s_{1:t} + \frac{k-q}{\eta_t^i})\circ s_t] \;\leq\; s_{1:T}^i + \frac{1}{\eta_T^i}k^i + \frac{1}{\eta_T^{min}} \quad \forall i,$$

where $\eta_T^{min} := \min_i\{\eta_T^i\}$. For example, for $\eta_t^i = \sqrt{k^i/t}$ we get the desired bound $s_{1:T}^i + \sqrt{T\cdot(k^i+4)}$. Unfortunately we were not able to generalize Theorem 4 to expert-dependent η, necessary for the final bound on FPL. In Section 6 we solve this problem by a hierarchy of experts.

4. Feasible FPL bounded by Infeasible FPL

This section establishes the relation between the FPL and IFPL losses. Recall that $\ell_t = E[M(s_{<t} + \frac{k-q}{\eta_t})\circ s_t]$ is the expected loss of FPL at time t and $r_t = E[M(s_{1:t} + \frac{k-q}{\eta_t})\circ s_t]$ is the expected loss of IFPL at time t.

Theorem 4 (FPL bounded by IFPL) *For $\mathcal{D} = \mathcal{E}$ and $0 \leq s_t^i \leq 1\ \forall i$ and arbitrary $s_{<t}$ and $P[q] = \mathrm{e}^{-\sum_i q^i}$ for $q \geq 0$, the expected loss of the feasible FPL is at most a factor $\mathrm{e}^{\eta_t} > 1$ larger than for the infeasible FPL:*

$$\ell_t \leq \mathrm{e}^{\eta_t} r_t, \quad \textit{which implies} \quad \ell_{1:T} - r_{1:T} \leq \sum_{t=1}^{T} \eta_t \ell_t.$$

Furthermore, if $\eta_t \leq 1$, then also $\ell_t \leq (1+\eta_t+\eta_t^2)r_t \leq (1+2\eta_t)r_t$.

Proof. Let $s = s_{<t} + \frac{1}{\eta}k$ be the past cumulative penalized state vector, q be a vector of independent exponential distributions, i.e. $P[q^i] = \mathrm{e}^{-q^i}$, and $\eta = \eta_t$. Then

$$\frac{P[q^j \geq \eta(s^j - m + 1)]}{P[q^j \geq \eta(s^j - m)]} = \left\{ \begin{array}{ccc} \mathrm{e}^{-\eta} & \text{if} & s^j \geq m \\ \mathrm{e}^{-\eta(s^j-m+1)} & \text{if} & m-1 \leq s^j \leq m \\ 1 & \text{if} & s^j \leq m-1 \end{array} \right\} \geq \mathrm{e}^{-\eta}$$

We now define the random variables $I := \mathrm{argmin}_i\{s^i - \frac{1}{\eta}q^i\}$ and $J := \mathrm{argmin}_i\{s^i + s_t^i - \frac{1}{\eta}q^i\}$, where $0 \leq s_t^i \leq 1\ \forall i$. Furthermore, for fixed vector $x \in \boldsymbol{R}^n$ and fixed j we define $m := \min_{i\neq j}\{s^i - \frac{1}{\eta}x^i\} \leq \min_{i\neq j}\{s^i + s_t^i - \frac{1}{\eta}x^i\} =: m'$. With this notation and using the independence of q^j from q^i for all $i \neq j$, we get

$$P[I = j|q^i = x^i\,\forall i \neq j] \;=\; P[s^j - \tfrac{1}{\eta}q^j \leq m|q^i = x^i\,\forall i \neq j] \;=\; P[q^j \geq \eta(s^j - m)]$$

$$\leq\; \mathrm{e}^{\eta}P[q^j \geq \eta(s^j - m + 1)] \;\leq\; \mathrm{e}^{\eta}P[q^j \geq \eta(s^j + s_t^j - m')]$$

$$=\; \mathrm{e}^{\eta}P[s^j + s_t^j - \tfrac{1}{\eta}q^j \leq m'|q^i = x^i\,\forall i \neq j] \;=\; \mathrm{e}^{\eta}P[J = j|q^i = x^i\,\forall i \neq j].$$

Since this bound holds under any condition x, it also holds unconditionally, i.e. $P[I = j] \leq \mathrm{e}^{\eta}P[J = j]$. For $\mathcal{D} = \mathcal{E}$ we have $s_t^I = M(s_{<t} + \frac{k-q}{\eta})\circ s_t$ and $s_t^J = M(s_{1:t} + \frac{k-q}{\eta})\circ s_t$, which implies

$$\ell_t \;=\; E[s_t^I] \;=\; \sum_{j=1}^{n} s_t^j \cdot P[I = j] \;\leq\; \mathrm{e}^{\eta}\sum_{j=1}^{n} s_t^j \cdot P[J = j] \;=\; \mathrm{e}^{\eta}E[s_t^J] \;=\; \mathrm{e}^{\eta}r_t.$$

Finally, $\ell_t - r_t \leq \eta_t \ell_t$ follows from $r_t \geq \mathrm{e}^{-\eta_t}\ell_t \geq (1-\eta_t)\ell_t$, and $\ell_t \leq \mathrm{e}^{\eta_t} r_t \leq (1+\eta_t+\eta_t^2) r_t \leq (1+2\eta_t) r_t$ for $\eta_t \leq 1$ is elementary. $\square$

Remark. As done by Kalai and Vempala (2003), one can prove a similar statement for general decision space $\mathcal{D}$ as long as $\sum_i |s_t^i| \leq A$ is guaranteed for some $A > 0$: In this case, we have $\ell_t \leq \mathrm{e}^{\eta_t A} r_t$. If n is finite, then the bound holds for $A = n$. For $n = \infty$, the assertion holds under the somewhat unnatural assumption that $\mathcal{S}$ is l^1-bounded.

5. Combination of Bounds and Choices for η_t

Throughout this section, we assume

$$\mathcal{D} = \mathcal{E}, \quad s_t \in [0,1]^n \ \forall t, \quad P[q] = \mathrm{e}^{-\sum_i q^i} \text{ for } q \geq 0, \quad \text{and} \quad \sum_i \mathrm{e}^{-k^i} \leq 1. \tag{8}$$

We distinguish *static* and *dynamic* bounds. Static bounds refer to a constant $\eta_t \equiv \eta$. Since this value has to be chosen in advance, a static choice of η_t requires certain prior information and therefore is not practical in many cases. However, the static bounds are very easy to derive, and they provide a good means to compare different PEA algorithms. If on the other hand the algorithm shall be applied without appropriate prior knowledge, a dynamic choice of η_t depending only on t and/or past observations, is necessary.

Theorem 5 (FPL bound for static $\eta_t = \eta \propto 1/\sqrt{L}$) *Assume (8) holds, then the expected loss ℓ_t of feasible FPL, which employs the prediction of the expert i minimizing $s^i_{<t} + \frac{k^i - q^i}{\eta_t}$, is bounded by the loss of the best expert in hindsight in the following way:*

$$\begin{array}{rl} i) & \textit{For} \quad \eta_t = \eta = 1/\sqrt{L} \quad \textit{with} \quad L \geq \ell_{1:T} \quad \textit{we have} \\ & \ell_{1:T} \ \leq \ s^i_{1:T} + \sqrt{L}(k^i+1) \quad \forall i. \\ ii) & \textit{For} \quad \eta_t = \sqrt{K/L} \quad \textit{with} \quad L \geq \ell_{1:T} \quad \textit{and} \quad k^i \leq K\ \forall i \quad \textit{we have} \\ & \ell_{1:T} \ \leq \ s^i_{1:T} + 2\sqrt{LK} \quad \forall i. \\ iii) & \textit{For} \quad \eta_t = \sqrt{k^i/L} \quad \textit{with} \quad L \geq \max\{s^i_{1:T}, k^i\} \quad \textit{we have} \\ & \ell_{1:T} \ \leq \ s^i_{1:T} + 2\sqrt{Lk^i} + 3k^i. \end{array}$$

Note that according to assertion (iii), knowledge of only the *ratio* of the complexity and the loss of the best expert is sufficient in order to obtain good static bounds, even for non-uniform complexities.

Proof. (i,ii) For $\eta_t = \sqrt{K/L}$ and $L \geq \ell_{1:T}$, from Theorem 4 and Corollary 3, we get

$$\ell_{1:T} - r_{1:T} \leq \sum_{t=1}^{T} \eta_t \ell_t = \ell_{1:T}\sqrt{K/L} \leq \sqrt{LK} \quad \text{and} \quad r_{1:T} - s^i_{1:T} \leq k^i/\eta_T = k^i\sqrt{L/K}.$$

Combining both, we get $\ell_{1:T} - s^i_{1:T} \leq \sqrt{L}(\sqrt{K} + k^i/\sqrt{K})$. (i) follows from $K=1$ and (ii) from $k^i \leq K$. (iii) For $\eta = \sqrt{k^i/L} \leq 1$ we get

$$\begin{aligned} \ell_{1:T} &\leq \mathrm{e}^{\eta} r_{1:T} \leq (1+\eta+\eta^2) r_{1:T} \leq \Big(1 + \sqrt{\frac{k^i}{L}} + \frac{k^i}{L}\Big)\Big(s^i_{1:T} + \sqrt{\frac{L}{k^i}}k^i\Big) \\ &\leq s^i_{1:T} + \sqrt{Lk^i} + \Big(\sqrt{\frac{k^i}{L}} + \frac{k^i}{L}\Big)(L + \sqrt{Lk^i}) = s^i_{1:T} + 2\sqrt{Lk^i} + \Big(2 + \sqrt{\frac{k^i}{L}}\Big)k^i. \end{aligned}$$

□

The static bounds require knowledge of an upper bound L on the loss (or the ratio of the complexity of the best expert and its loss). Since the instantaneous loss is bounded by 1, one may set $L=T$ if T is known in advance. For finite n and $k^i=K=\ln n$, bound (ii) gives the classic regret $\propto\sqrt{T\ln n}$. If neither T nor L is known, a dynamic choice of η_t is necessary. We first present bounds with regret $\propto\sqrt{T}$, thereafter with regret $\propto\sqrt{s^i_{1:T}}$.

Theorem 6 (FPL bound for dynamic $\eta_t\propto 1/\sqrt{t}$) *Assume (8) holds.*

$$
\begin{array}{lll}
i) & \textit{For}\quad \eta_t=1/\sqrt{t}\quad \textit{we have}\quad \ell_{1:T}\ \le\ s^i_{1:T}+\sqrt{T}(k^i+2)\quad \forall i. \\
ii) & \textit{For}\quad \eta_t=\sqrt{K/2t}\ \ \textit{and}\ \ k^i\le K\ \forall i\ \ \textit{we have}\ \ \ell_{1:T}\ \le\ s^i_{1:T}+2\sqrt{2TK}\quad \forall i.
\end{array}
$$

Proof. For $\eta_t=\sqrt{K/2t}$, using $\sum_{t=1}^T\frac{1}{\sqrt{t}}\le\int_0^T\frac{dt}{\sqrt{t}}=2\sqrt{T}$ and $\ell_t\le 1$ we get

$$\ell_{1:T}-r_{1:T}\le\sum_{t=1}^T\eta_t\le\sqrt{2TK}\quad\text{and}\quad r_{1:T}-s^i_{1:T}\le k^i/\eta_T=k^i\sqrt{\frac{2T}{K}}.$$

Combining both, we get $\ell_{1:T}-s^i_{1:T}\le\sqrt{2T}(\sqrt{K}+k^i/\sqrt{K})$. (i) follows from $K=2$ and (ii) from $k^i\le K$. □

In Theorem 5 we assumed knowledge of an upper bound L on $\ell_{1:T}$. In an adaptive form, $L_t:=\ell_{<t}+1$, known at the beginning of time t, could be used as an upper bound on $\ell_{1:t}$ with corresponding adaptive $\eta_t\propto 1/\sqrt{L_t}$. Such choice of η_t is also called *self-confident* (Auer et al., 2002).

Theorem 7 (FPL bound for self-confident $\eta_t\propto 1/\sqrt{\ell_{<t}}$) *Assume (8) holds.*

$$
\begin{array}{ll}
i) & \textit{For}\quad \eta_t=1/\sqrt{2(\ell_{<t}+1)}\quad \textit{we have} \\
 & \ell_{1:T}\ \le\ s^i_{1:T}+(k^i{+}1)\sqrt{2(s^i_{1:T}{+}1)}+2(k^i{+}1)^2\quad \forall i. \\
ii) & \textit{For}\quad \eta_t=\sqrt{K/2(\ell_{<t}+1)}\quad \textit{and}\quad k^i\le K\ \forall i\quad \textit{we have} \\
 & \ell_{1:T}\ \le\ s^i_{1:T}+2\sqrt{2(s^i_{1:T}{+}1)K}+8K\quad \forall i.
\end{array}
$$

Proof. Using $\eta_t=\sqrt{K/2(\ell_{<t}{+}1)}\le\sqrt{K/2\ell_{1:t}}$ and $\frac{b-a}{\sqrt{b}}=(\sqrt{b}-\sqrt{a})(\sqrt{b}+\sqrt{a})\frac{1}{\sqrt{b}}\le 2(\sqrt{b}-\sqrt{a})$ for $a\le b$ and $t_0:=\min\{t:\ell_{1:t}>0\}$ we get

$$\ell_{1:T}-r_{1:T}\le\sum_{t=t_0}^T\eta_t\ell_t\le\sqrt{\frac{K}{2}}\sum_{t=t_0}^T\frac{\ell_{1:t}-\ell_{<t}}{\sqrt{\ell_{1:t}}}\le\sqrt{2K}\sum_{t=t_0}^T[\sqrt{\ell_{1:t}}-\sqrt{\ell_{<t}}]=\sqrt{2K}\sqrt{\ell_{1:T}}.$$

Adding $r_{1:T}-s^i_{1:T}\le\frac{k^i}{\eta_T}\le k^i\sqrt{2(\ell_{1:T}{+}1)/K}$ we get

$$\ell_{1:T}-s^i_{1:T}\le\sqrt{2\bar\kappa^i(\ell_{1:T}{+}1)},\quad\text{where}\quad\sqrt{\bar\kappa^i}:=\sqrt{K}+k^i/\sqrt{K}.$$

Taking the square and solving the resulting quadratic inequality w.r.t. $\ell_{1:T}$ we get

$$\ell_{1:T}\le s^i_{1:T}+\bar\kappa^i+\sqrt{2(s^i_{1:T}{+}1)\bar\kappa^i+(\bar\kappa^i)^2}\le s^i_{1:T}+\sqrt{2(s^i_{1:T}{+}1)\bar\kappa^i}+2\bar\kappa^i.$$

For $K=1$ we get $\sqrt{\overline{\kappa}^i}=k^i+1$ which yields (i). For $k^i\leq K$ we get $\overline{\kappa}^i\leq 4K$ which yields (ii). □

The proofs of results similar to (ii) for WM for 0/1 loss all fill several pages (Auer et al., 2002; Yaroshinsky et al., 2004). The next result establishes a similar bound, but instead of using the *expected* value $\ell_{<t}$, the *best loss so far* $s^{min}_{<t}$ is used. This may have computational advantages, since $s^{min}_{<t}$ is immediately available, while $\ell_{<t}$ needs to be evaluated (see discussion in Section 9).

Theorem 8 (FPL bound for adaptive $\eta_t\propto 1/\sqrt{s^{min}_{<t}}$) *Assume (8) holds.*

$$\begin{aligned}
i)\quad & \textit{For}\quad \eta_t = 1/\min_i\{k^i+\sqrt{(k^i)^2+2s^i_{<t}+2}\} \quad \textit{we have} \\
& \ell_{1:T} \;\leq\; s^i_{1:T}+(k^i+2)\sqrt{2s^i_{1:T}}+2(k^i+2)^2 \quad \forall i. \\
ii)\quad & \textit{For}\quad \eta_t = \sqrt{\tfrac{1}{2}}\cdot\min\{1,\sqrt{K/s^{min}_{<t}}\} \quad \textit{and}\quad k^i\leq K\ \forall i \quad \textit{we have} \\
& \ell_{1:T} \;\leq\; s^i_{1:T}+2\sqrt{2Ks^i_{1:T}}+5K\ln(s^i_{1:T})+3K+6 \quad \forall i.
\end{aligned}$$

We briefly motivate the strange looking choice for η_t in (i). The first naive candidate, $\eta_t\propto 1/\sqrt{s^{min}_{<t}}$, turns out too large. The next natural trial is requesting $\eta_t=1/\sqrt{2\min\{s^i_{<t}+\frac{k^i}{\eta_t}\}}$. Solving this equation results in $\eta_t=1/(k^i+\sqrt{(k^i)^2+2s^i_{<t}})$, where i be the index for which $s^i_{<t}+\frac{k^i}{\eta_t}$ is minimal.

Proof. Define the minimum of a vector as its minimum component, e.g. $\min(k)=k^{min}$. For notational convenience, let $\eta_0=\infty$ and $\tilde{s}_{1:t}=s_{1:t}+\frac{k-q}{\eta_t}$. Like in the proof of Theorem 2, we consider one exponentially distributed perturbation q. Since $M(\tilde{s}_{1:t})\circ\tilde{s}_t\leq M(\tilde{s}_{1:t})\circ\tilde{s}_{1:t}-M(\tilde{s}_{<t})\circ\tilde{s}_{<t}$ by (4), we have

$$M(\tilde{s}_{1:t})\circ s_t \leq M(\tilde{s}_{1:t})\circ\tilde{s}_{1:t}-M(\tilde{s}_{<t})\circ\tilde{s}_{<t}-M(\tilde{s}_{1:t})\circ\left(\frac{k-q}{\eta_t}-\frac{k-q}{\eta_{t-1}}\right)$$

Since $\eta_t\leq\sqrt{1/2}$, Theorem 4 asserts $\ell_t\leq E[(1+\eta_t+\eta_t^2)M(\tilde{s}_{1:t})\circ s_t]$, thus $\ell_{1:T}\leq A+B$, where

$$\begin{aligned}
A &= \sum_{t=1}^{T}E\left[(1+\eta_t+\eta_t^2)(M(\tilde{s}_{1:t})\circ\tilde{s}_{1:t}-M(\tilde{s}_{<t})\circ\tilde{s}_{<t})\right] \\
&= E[(1+\eta_T+\eta_T^2)M(\tilde{s}_{1:T})\circ\tilde{s}_{1:T}]-E[(1+\eta_1+\eta_1^2)\min(\frac{k-q}{\eta_1})] \\
&\quad+\sum_{t=1}^{T-1}E\left[(\eta_t-\eta_{t+1}+\eta_t^2-\eta_{t+1}^2)M(\tilde{s}_{1:t})\circ\tilde{s}_{1:t}\right] \quad \text{and} \\
B &= \sum_{t=1}^{T}E\left[(1+\eta_t+\eta_t^2)M(\tilde{s}_{1:t})\circ\left(\frac{q-k}{\eta_t}-\frac{q-k}{\eta_{t-1}}\right)\right] \\
&\leq \sum_{t=1}^{T}(1+\eta_t+\eta_t^2)\left(\frac{1}{\eta_t}-\frac{1}{\eta_{t-1}}\right)=\frac{1+\eta_T+\eta_T^2}{\eta_T}+\sum_{t=1}^{T-1}\frac{\eta_t-\eta_{t+1}+\eta_t^2-\eta_{t+1}^2}{\eta_t}.
\end{aligned}$$

Here, the estimate for B follows from $\frac{1}{\eta_t}-\frac{1}{\eta_{t-1}}\geq 0$ and $E[M(\eta_t s_{1:t}+k-q)\circ(q-k)]\leq E[\max_i\{q^i-k^i\}]\leq 1$, which in turn holds by minimality of M, $\sum_i e^{-k^i}\leq 1$ and Lemma 1. In order to estimate A, we set $\overline{s}_{1:t}=s_{1:t}+\frac{k}{\eta_t}$ and observe $M(\tilde{s}_{1:t})\circ\tilde{s}_{1:t}\leq M(\overline{s}_{1:t})\circ(\overline{s}_{1:t}-\frac{q}{\eta_t})$ by minimality of M. The expectations of q can then be evaluated to $E[M(\bar{s}_{1:t})\circ q]=1$, and as before we have $E[-\min(k-q)]\leq 1$. Hence

$$\begin{aligned}
\ell_{1:T} &\leq A+B \leq (1+\eta_T+\eta_T^2)\left(M(\bar{s}_{1:T})\circ\bar{s}_{1:T}-\frac{1}{\eta_T}\right)+\frac{1+\eta_1+\eta_1^2}{\eta_1}\\
&\quad+\sum_{t=1}^{T-1}(\eta_t-\eta_{t+1}+\eta_t^2-\eta_{t+1}^2)\left(M(\bar{s}_{1:t})\circ\bar{s}_{1:t}-\frac{1}{\eta_t}\right)+B \qquad (9)\\
&\leq (1+\eta_T+\eta_T^2)\min(\bar{s}_{1:T})+\sum_{t=1}^{T-1}(\eta_t-\eta_{t+1}+\eta_t^2-\eta_{t+1}^2)\min(\bar{s}_{1:t})+\frac{1}{\eta_1}+2.
\end{aligned}$$

We now proceed by considering the two parts of the theorem separately.

(i) Here, $\eta_t=1/\min(k+\sqrt{k^2+2s_{<t}+2})$. Fix $t\leq T$ and choose m such that $k^m+\sqrt{(k^m)^2+2s^m_{<t}+2}$ is minimal. Then

$$\min(s_{1:t}+\frac{k}{\eta_t})\leq s^m_{<t}+1+\frac{k^m}{\eta_t}=\tfrac{1}{2}(k^m+\sqrt{(k^m)^2+2s^m_{<t}+2})^2=\frac{1}{2\eta_t^2}\leq\frac{1}{2\eta_t\eta_{t+1}}.$$

We may overestimate the quadratic terms η_t^2 in (9) by η_t – the easiest justification is that we could have started with the cruder estimate $\ell_t\leq(1+2\eta_t)r_t$ from Theorem 4. Then

$$\begin{aligned}
\ell_{1:T} &\leq (1+2\eta_T)\min(s_{1:T}+\frac{k}{\eta_T})+2\sum_{t=1}^{T-1}(\eta_t-\eta_{t+1})\min(s_{1:t}+\frac{k}{\eta_t})+\frac{1}{\eta_1}+2\\
&\leq (1+2\eta_T)\frac{1}{2\eta_T^2}+2\sum_{t=1}^{T-1}(\eta_t-\eta_{t+1})\frac{1}{2\eta_t^2}+\frac{1}{\eta_1}+2\\
&\leq \frac{1}{2\eta_T^2}+\frac{1}{\eta_T}+\sum_{t=1}^{T-1}\left(\frac{1}{\eta_{t+1}}-\frac{1}{\eta_t}\right)+\frac{1}{\eta_1}+2\\
&\leq \tfrac{1}{2}\min(k+\sqrt{k^2+2s_{<T}+2})^2+2\min(k+\sqrt{k^2+2s_{<T}+2})+2\\
&\leq s^i_{1:T}+(k^i+2)\sqrt{2s^i_{1:T}}+2(k^i)^2+6k^i+6 \quad \text{for all } i.
\end{aligned}$$

This proves the first part of the theorem.

(ii) Here we have $K\geq k^i$ for all i. Abbreviate $a_t=\max\{K,s^{min}_{1:t}\}$ for $1\leq t\leq T$, then $\eta_t=\sqrt{\frac{K}{2a_{t-1}}}$, $a_t\geq K$, and $a_t-a_{t-1}\leq 1$ for all t. Observe $M(\bar{s}_{1:t})=M(s_{1:t})$, $\eta_t-\eta_{t+1}=\frac{\sqrt{K}(a_t-a_{t-1})}{\sqrt{2}\sqrt{a_t}\sqrt{a_{t-1}}(\sqrt{a_t}+\sqrt{a_{t-1}})}$, $\eta_t^2-\eta_{t+1}^2=\frac{K(a_t-a_{t-1})}{2a_ta_{t-1}}$, and $\frac{a_t-a_{t-1}}{2a_{t-1}}\leq\ln(1+\frac{a_t-a_{t-1}}{a_{t-1}})=\ln(a_t)-\ln(a_{t-1})$ which is true for $\frac{a_t-a_{t-1}}{a_{t-1}}\leq\frac{1}{K}\leq\frac{1}{\ln 2}$. This implies

$$\begin{aligned}
\frac{(\eta_t-\eta_{t+1})K}{\eta_t} &\leq \frac{K(a_t-a_{t-1})}{2a_{t-1}}\leq K\ln\left(1+\frac{a_t-a_{t-1}}{a_{t-1}}\right)=K(\ln(a_t)-\ln(a_{t-1})),\\
(\eta_t-\eta_{t+1})s^{min}_{1:t} &\leq \frac{\sqrt{K}(a_t-a_{t-1})(\sqrt{a_{t-1}}+\sqrt{a_t}-\sqrt{a_{t-1}})}{\sqrt{2}\sqrt{a_{t-1}}(\sqrt{a_t}+\sqrt{a_{t-1}})}\\
&= \sqrt{\frac{K}{2}}(\sqrt{a_t}-\sqrt{a_{t-1}})+\frac{\sqrt{K}(a_t-a_{t-1})^2}{\sqrt{2a_{t-1}}(\sqrt{a_t}+\sqrt{a_{t-1}})^2}\\
\boxed{\substack{\text{use } a_t-a_{t-1}\leq 1\\ \text{and } a_{t-1}\geq K}}\quad &\leq \sqrt{\frac{K}{2}}(\sqrt{a_t}-\sqrt{a_{t-1}})+\frac{1}{2\sqrt{2}}(\ln(a_t)-\ln(a_{t-1})),
\end{aligned}$$

$$\begin{aligned}\frac{(\eta_t^2-\eta_{t+1}^2)K}{\eta_t} &= \frac{K\sqrt{K}(a_t-a_{t-1})}{\sqrt{2}a_t\sqrt{a_{t-1}}} \overset{\boxed{a_{t-1}\geq K}}{\leq} \sqrt{2}K(\ln(a_t)-\ln(a_{t-1})), \text{ and} \\ (\eta_t^2-\eta_{t+1}^2)s_{1:t}^{min} &\leq \frac{K(a_t-a_{t-1})}{2a_{t-1}} \leq K(\ln(a_t)-\ln(a_{t-1})).\end{aligned}$$

The logarithmic estimate in the second and third bound is unnecessarily rough and for convenience only. Therefore, the coefficient of the log-term in the final bound of the theorem can be reduced to $2K$ without much effort. Plugging the above estimates back into (9) yields

$$\begin{aligned}\ell_{1:T} \leq\ & s_{1:T}^{min}+\sqrt{\frac{K}{2}s_{1:T}^{min}}+\sqrt{2Ks_{1:T}^{min}}+3K+2+\sqrt{\frac{K}{2}s_{1:T}^{min}}+(\tfrac{7}{2}K+\tfrac{1}{2\sqrt{2}})\ln(s_{1:T}^{min}) \\ & +\frac{1}{\eta_1}+2 \leq s_{1:T}^{min}+2\sqrt{2Ks_{1:T}^{min}}+5K\ln(s_{1:T}^{min})+3K+6.\end{aligned}$$

This completes the proof. □

Theorem 7 and Theorem 8 (i) immediately imply the following bounds on the $\sqrt{\text{Loss}}$-regrets: $\sqrt{\ell_{1:T}} \leq \sqrt{s_{1:T}^i+1}+\sqrt{8K}$, $\sqrt{\ell_{1:T}} \leq \sqrt{s_{1:T}^i+1}+\sqrt{2}(k^i+1)$, and $\sqrt{\ell_{1:T}} \leq \sqrt{s_{1:T}^i}+\sqrt{2}(k^i+2)$, respectively.

Remark. The same analysis as for Theorems [5–8](ii) applies to general $\mathcal{D}$, using $\ell_t \leq e^{\eta_t n} r_t$ instead of $\ell_t \leq e^{\eta_t} r_t$, and leading to an additional factor $\sqrt{n}$ in the regret. Compare the remark at the end of Section 4.

6. Hierarchy of Experts

We derived bounds which do not need prior knowledge of L with regret $\propto \sqrt{TK}$ and $\propto \sqrt{s_{1:T}^i K}$ for a finite number of experts with equal penalty $K = k^i = \ln n$. For an infinite number of experts, unbounded expert-dependent complexity penalties k^i are necessary (due to constraint $\sum_i e^{-k^i} \leq 1$). Bounds for this case (without prior knowledge of T) with regret $\propto k^i\sqrt{T}$ and $\propto k^i\sqrt{s_{1:T}^i}$ have been derived. In this case, the complexity k^i is no longer under the square root. Although this already implies Hannan consistency, i.e. the average per round regret tends to zero as $t\to\infty$, improved regret bounds $\propto \sqrt{Tk^i}$ and $\propto \sqrt{s_{1:T}^i k^i}$ are desirable and likely to hold. We were not able to derive such improved bounds for FPL, but for a (slight) modification. We consider a two-level hierarchy of experts. First consider an FPL for the subclass of experts of complexity K, for each $K \in \mathbb{N}$. Regard these FPL^K as (meta) experts and use them to form a (meta) FPL. The class of meta experts now contains for each complexity only one (meta) expert, which allows us to derive good bounds. In the following, quantities referring to complexity class K are superscripted by K, and meta quantities are superscripted by $\tilde{}$.

Consider the class of experts $\mathcal{E}^K := \{i : K-1 < k^i \leq K\}$ of complexity K, for each $K \in \mathbb{N}$. FPL^K makes randomized prediction $I_t^K := \text{argmin}_{i\in\mathcal{E}^K}\{s_{<t}^i + \frac{k^i-q^i}{\eta_t^K}\}$ with $\eta_t^K := \sqrt{K/2t}$ and suffers loss $u_t^K := s_t^{I_t^K}$ at time t. Since $k^i \leq K\ \forall i \in \mathcal{E}^k$ we can apply Theorem 6(ii) to FPL^K:

$$E[u_{1:T}^K] \;=\; \ell_{1:T}^K \;\leq\; s_{1:T}^i + 2\sqrt{2TK} \quad \forall i \in \mathcal{E}^K \quad \forall K \in \mathbb{N}. \tag{10}$$

We now define a meta state $\tilde{s}_t^K = u_t^K$ and regard FPL^K for $K \in \mathbb{N}$ as meta experts, so meta expert K suffers loss $\tilde{s}_t^K$. (Assigning expected loss $\tilde{s}_t^K = E[u_t^K] = \ell_t^K$ to FPL^K would also work.) Hence the setting is again an expert setting and we define the meta $\widetilde{\text{FPL}}$ to predict $\tilde{I}_t := \text{argmin}_{K\in N}\{\tilde{s}^K_{<t} + \frac{\tilde{k}^K - \tilde{q}^K}{\tilde{\eta}_t}\}$ with $\tilde{\eta}_t = 1/\sqrt{t}$ and $\tilde{k}^K = \frac{1}{2} + 2\ln K$ (implying $\sum_{K=1}^\infty e^{-\tilde{k}^K} \leq 1$). Note that $\tilde{s}^K_{1:t} = \tilde{s}^K_1 + ... + \tilde{s}^K_t = s_1^{I_1^K} + ... + s_t^{I_t^K}$ sums over the same meta state components K, but over different components I_t^K in normal state representation.

By Theorem 6(i) the $\tilde{q}$-expected loss of $\widetilde{\text{FPL}}$ is bounded by $\tilde{s}^K_{1:T} + \sqrt{T}(\tilde{k}^K + 2)$. As this bound holds for all q it also holds in q-expectation. So if we define $\tilde{\ell}_{1:T}$ to be the q *and* $\tilde{q}$ expected loss of $\widetilde{\text{FPL}}$, and chain this bound with (10) for $i \in \mathcal{E}^K$ we get:

$$\begin{aligned} \tilde{\ell}_{1:T} &\leq E[\tilde{s}^K_{1:T} + \sqrt{T}(\tilde{k}^K + 2)] \;=\; \ell^K_{1:T} + \sqrt{T}(\tilde{k}^K + 2) \\ &\leq s^i_{1:T} + \sqrt{T}[2\sqrt{2(k^i + 1)} + \tfrac{1}{2} + 2\ln(k^i + 1) + 2], \end{aligned}$$

where we have used $K \leq k^i + 1$. This bound is valid for all i and has the desired regret $\propto \sqrt{Tk^i}$. Similarly we can derive regret bounds $\propto \sqrt{s^i_{1:T}k^i}$ by exploiting that the bounds in Theorems 7 and 8 are concave in $s^i_{1:T}$ and using Jensen's inequality.

Theorem 9 (Hierarchical FPL bound for dynamic η_t) *The hierarchical $\widetilde{FPL}$ employs at time t the prediction of expert $i_t := I_t^{\tilde{I}_t}$, where*

$$I_t^K := \mathop{\arg\min}_{i:\lceil k^i\rceil = K}\left\{s^i_{<t} + \frac{k^i - q^i}{\eta^K_t}\right\} \quad \textit{and} \quad \tilde{I}_t := \mathop{\arg\min}_{K\in N}\left\{s_1^{I_1^K} + ... + s_{t-1}^{I_{t-1}^K} + \frac{\frac{1}{2} + 2\ln K - \tilde{q}^K}{\tilde{\eta}_t}\right\}$$

Under assumptions (8) and independent $P[\tilde{q}^K] = e^{-\tilde{q}^K}$ $\forall K \in N$, the expected loss $\tilde{\ell}_{1:T} = E[s_1^{i_1} + ... + s_T^{i_T}]$ of $\widetilde{FPL}$ is bounded as follows:

$$\begin{aligned} &a) \quad \textit{For} \quad \eta^K_t = \sqrt{K/2t} \quad \textit{and} \quad \tilde{\eta}_t = 1/\sqrt{t} \quad \textit{we have} \\ &\qquad \tilde{\ell}_{1:T} \;\leq\; s^i_{1:T} + 2\sqrt{2Tk^i}\cdot(1 + O(\tfrac{\ln k^i}{\sqrt{k^i}})) \quad \forall i. \\ &b) \quad \textit{For } \tilde{\eta}_t \textit{ as in } (i) \textit{ and } \eta^K_t \textit{ as in } (ii) \textit{ of Theorem } \{{}^7_8\} \textit{ we have} \\ &\qquad \tilde{\ell}_{1:T} \;\leq\; s^i_{1:T} + 2\sqrt{2s^i_{1:T}k^i}\cdot(1 + O(\tfrac{\ln k^i}{\sqrt{k^i}})) + \left\{ {O(k^i) \atop O(k^i \ln s^i_{1:T})} \right\} \quad \forall i. \end{aligned}$$

The hierarchical $\widetilde{\text{FPL}}$ differs from a direct FPL over all experts $\mathcal{E}$. One potential way to prove a bound on direct FPL may be to show (if it holds) that FPL performs better than $\widetilde{\text{FPL}}$, i.e. $\ell_{1:T} \leq \tilde{\ell}_{1:T}$. Another way may be to suitably generalize Theorem 4 to expert dependent η.

7. Lower Bound on FPL

A lower bound on FPL similar to the upper bound in Theorem 2 can also be proven.

Theorem 10 (FPL lower-bounded by BEH) *Let n be finite. Assume $\mathcal{D} \subseteq \mathbf{R}^n$ and $s_t \in \mathbf{R}^n$ are chosen such that the required extrema exist (possibly negative), $q \in \mathbf{R}^n$, and $\eta_t > 0$ is a decreasing sequence. Then the loss of FPL for uniform complexities (l.h.s.) can be lower-bounded in terms of the best predictor in hindsight (first term on r.h.s.) plus/minus additive corrections:*

$$\sum_{t=1}^T M(s_{<t} - \frac{q}{\eta_t})\circ s_t \geq \min_{d\in\mathcal{D}}\{d\circ s_{1:T}\} - \frac{1}{\eta_T}\max_{d\in\mathcal{D}}\{d\circ q\} + \sum_{t=1}^T(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}})M(s_{<t})\circ q$$

Proof. For notational convenience, let $\eta_0=\infty$ and $\tilde{s}_{1:t}=s_{1:t}-\frac{q}{\eta_t}$. Consider the losses $\tilde{s}_t=s_t-q(\frac{1}{\eta_t}-\frac{1}{\eta_{t-1}})$ for the moment. We first show by induction on T that the predictor $M(\tilde{s}_{<t})$ has nonnegative regret, i.e.

$$\sum_{t=1}^{T} M(\tilde{s}_{<t})\circ\tilde{s}_t \geq M(\tilde{s}_{1:T})\circ\tilde{s}_{1:T}. \tag{11}$$

For $T=1$ this follows immediately from minimality of M ($\tilde{s}_{<1}:=0$). For the induction step from $T-1$ to T we need to show

$$M(\tilde{s}_{<T})\circ\tilde{s}_T \geq M(\tilde{s}_{1:T})\circ\tilde{s}_{1:T} - M(\tilde{s}_{<T})\circ\tilde{s}_{<T}.$$

Due to $\tilde{s}_{1:T}=\tilde{s}_{<T}+\tilde{s}_T$, this is equivalent to $M(\tilde{s}_{<T})\circ\tilde{s}_{1:T}\geq M(\tilde{s}_{1:T})\circ\tilde{s}_{1:T}$, which holds by minimality of M. Rearranging terms in (11) we obtain

$$\sum_{t=1}^{T} M(\tilde{s}_{<t})\circ s_t \geq M(\tilde{s}_{1:T})\circ\tilde{s}_{1:T} + \sum_{t=1}^{T} M(\tilde{s}_{<t})\circ q\Big(\frac{1}{\eta_t}-\frac{1}{\eta_{t-1}}\Big), \quad \text{with} \tag{12}$$

$$M(\tilde{s}_{1:T})\circ\tilde{s}_{1:T} = M(s_{1:T}-\frac{q}{\eta_T})\circ s_{1:T} - M(s_{1:T}-\frac{q}{\eta_T})\circ\frac{q}{\eta_T} \geq \min_{d\in\mathcal{D}}\{d\circ s_{1:T}\} - \frac{1}{\eta_T}\max_{d\in\mathcal{D}}\{d\circ q\}$$

$$\text{and} \quad \sum_{t=1}^{T} M(\tilde{s}_{<t})\circ q\Big(\frac{1}{\eta_t}-\frac{1}{\eta_{t-1}}\Big) \;\geq\; \sum_{t=1}^{T}\Big(\frac{1}{\eta_t}-\frac{1}{\eta_{t-1}}\Big)M(s_{<t})\circ q.$$

Again, the last bound follows from the minimality of M, which asserts that $[M(s-q)-M(s)]\circ s\geq 0\geq[M(s-q)-M(s)]\circ(s-q)$ and thus implies that $M(s-q)\circ q\geq M(s)\circ q$. So Theorem 10 follows from (12). $\square$

Assuming q random with $E[q^i]=1$ and taking the expectation in Theorem 10, the last term reduces to $\sum_t(\frac{1}{\eta_t}-\frac{1}{\eta_{t-1}})\sum_i M(s_{<t})^i$. If $\mathcal{D}\geq 0$, the term is positive and may be dropped. In case of $\mathcal{D}=\mathcal{E}$ or Δ, the last term is identical to $\frac{1}{\eta_T}$ (since $\sum_i d^i=1$) and keeping it improves the bound. Furthermore, we need to evaluate the expectation of the second to last term in Theorem 10, namely $E[\max_{d\in\mathcal{D}}\{d\circ q\}]$. For $\mathcal{D}=\mathcal{E}$ and q being exponentially distributed, using Lemma 1 with $k^i=0\ \forall i$, the expectation is bounded by $1+\ln n$. We hence get the following lower bound:

Corollary 11 (FPL lower-bounded by BEH) *For $\mathcal{D}=\mathcal{E}$ and any $\mathcal{S}$ and all k^i equal and $P[q^i]=e^{-q^i}$ for $q\geq 0$ and decreasing $\eta_t>0$, the expected loss of FPL is at most $\ln n/\eta_T$ lower than the loss of the best expert in hindsight:*

$$\ell_{1:T} \;\geq\; s^{min}_{1:T} - \frac{\ln n}{\eta_T}$$

The upper and lower bounds on $\ell_{1:T}$ (Theorem 4 and Corollaries 3 and 11) together show that

$$\frac{\ell_{1:t}}{s^{min}_{1:t}} \to 1 \qquad \text{if} \qquad \eta_t\to 0 \quad \text{and} \quad \eta_t\cdot s^{min}_{1:t}\to\infty \quad \text{and} \quad k^i=K\ \forall i. \tag{13}$$

For instance, $\eta_t=\sqrt{K/2s^{min}_{<t}}$. For $\eta_t=\sqrt{K/2(\ell_{<t}+1)}$ we proved the bound in Theorem 7(ii). Knowing that $\sqrt{K/2(\ell_{<t}+1)}$ converges to $\sqrt{K/2s^{min}_{<t}}$ due to (13), we can derive a bound similar to Theorem 7(ii) for $\eta_t=\sqrt{K/2s^{min}_{<t}}$. This choice for η_t has the advantage that we do not have to compute $\ell_{<t}$ (cf. Section 9), as also achieved by Theorem 8(ii).

We do not know whether Theorem 10 can be generalized to expert dependent complexities k^i.

8. Adaptive Adversary

In this section we show that bounds that hold against an oblivious adversary automatically also hold against an adaptive one.

Initial versus independent randomization. So far we assumed that the perturbations q are sampled only once at time $t=0$. As already indicated, under the expectation this is equivalent to generating a new perturbation q_t at each time step t, i.e. Theorems 4–9 remain valid for this case. While the former choice was favorable for the analysis, the latter has two advantages. First, repeated sampling of the perturbations guarantees better bounds with high probability (see next section). Second, if the losses are generated by an adaptive adversary (not to be confused with an adaptive learning rate) which has access to FPL's past decisions, then he may after some time figure out the initial random perturbation and use it to force FPL to have a large loss. We now show that the bounds for FPL remain valid, even in case of an adaptive adversary, if independent randomization $q \rightsquigarrow q_t$ is used.

Oblivious versus adaptive adversary. Recall the protocol for FPL: After each expert i made its prediction y_t^i, and FPL combined them to form its own prediction y_t^{FPL}, we observe x_t, and Loss(x_t,$y_t^{...}$) is revealed for FPL's and each expert's prediction. For independent randomization, we have $y_t^{\text{FPL}} = y_t^{\text{FPL}}(x_{<t},y_{1:t},q_t)$. For an oblivious (non-adaptive) adversary, $x_t = x_t(x_{<t},y_{<t})$. Recursively inserting and eliminating the experts $y_t^i = y_t^i(x_{<t},y_{<t})$ and y_t^{FPL}, we get the dependencies

$$u_t := \text{Loss}(x_t, y_t^{\text{FPL}}) = u_t(x_{1:t}, q_t) \quad \text{and} \quad s_t^i := \text{Loss}(x_t, y_t^i) = s_t^i(x_{1:t}), \tag{14}$$

where $x_{1:t}$ is a "fixed" sequence. With this notation, Theorems 5–8 read $\ell_{1:T} \equiv E[\sum_{t=1}^T u_t(x_{1:t},q_t)] \leq f(x_{1:T})$ for all $x_{1:T} \in \mathcal{X}^T$, where $f(x_{1:T})$ is one of the r.h.s. in Theorems 5–8. Noting that f is independent of $q_{1:T}$, we can write this as

$$A_1 \leq 0, \quad \text{where} \quad A_t(x_{<t}, q_{<t}) := \max_{x_{t:T}} E_{q_{t:T}} \Big[\sum_{\tau=1}^{T} u_\tau(x_{1:\tau}, q_\tau) - f(x_{1:T}) \Big], \tag{15}$$

where $E_{q_{t:T}}$ is the expectation w.r.t. $q_t...q_T$ (keeping $q_{<t}$ fixed).

For an adaptive adversary, $x_t = x_t(x_{<t},y_{<t},y_{<t}^{\text{FPL}})$ can additionally depend on $y_{<t}^{\text{FPL}}$. Eliminating y_t^i and y_t^{FPL} we get, again, (14), but $x_t = x_t(x_{<t},q_{<t})$ is no longer fixed, but an (arbitrary) random function. So we have to replace x_t by $x_t(x_{<t},q_{<t})$ in (15) for $t = 1..T$. The maximization is now a functional maximization over all functions $x_t(\cdot,\cdot)...x_T(\cdot,\cdot)$. Using "$\max_{x(\cdot)} E_q[g(x(q),q)] = E_q \max_x [g(x,q)]$," we can write this as

$$B_1 \overset{?}{\leq} 0, \quad \text{where} \quad B_t(x_{<t}, q_{<t}) := \max_{x_t} E_{q_t} ... \max_{x_T} E_{q_T} \Big[\sum_{\tau=1}^{T} u_\tau(x_{1:\tau}, q_\tau) - f(x_{1:T}) \Big]. \tag{16}$$

So, establishing $B_1 \leq 0$ would show that all bounds also hold in the adaptive case.

Lemma 12 (Adaptive=Oblivious) *Let $q_1...q_T \in \mathbb{R}^T$ be independent random variables, E_{q_t} be the expectation w.r.t. q_t, f any function of $x_{1:T} \in \mathcal{X}^T$, and u_t arbitrary functions of $x_{1:t}$ and q_t. Then, $A_t(x_{<t},q_{<t}) = B_t(x_{<t},q_{<t})$ for all $1 \leq t \leq T$, where A_t and B_t are defined in (15) and (16). In particular, $A_1 \leq 0$ implies $B_1 \leq 0$.*

Proof. We prove $B_t = A_t$ by induction on t, which establishes the theorem. $B_T = A_T$ is obvious. Assume $B_t = A_t$. Then

$$\begin{aligned}
B_{t-1} &= \max_{x_{t-1}} E_{q_{t-1}} B_t \;=\; \max_{x_{t-1}} E_{q_{t-1}} A_t \\
&= \max_{x_{t-1}} E_{q_{t-1}} \left[\max_{x_{t:T}} E_{q_{t:T}} \left[\sum_{\tau=1}^{T} u_\tau(x_{1:\tau}, q_\tau) - f(x_{1:T}) \right] \right] \\
&= \max_{x_{t-1}} E_{q_{t-1}} \left[\underbrace{\sum_{\tau=1}^{t-1} u_\tau(x_{1:\tau}, q_\tau)}_{\text{independent } x_{t:T} \text{ and } q_{t:T}} + \underbrace{\max_{x_{t:T}} E_{q_{t:T}} \left[\sum_{\tau=t}^{T} u_\tau(x_{1:\tau}, q_\tau) - f(x_{1:T}) \right]}_{\text{independent } q_{t-1}\text{, since the } q_t \text{ are i.d.}} \right] \\
&= \max_{x_{t-1}} \left[\overbrace{E_{q_{t-1}} \left[\sum_{\tau=1}^{t-1} u_\tau(x_{1:\tau}, q_\tau) \right]} + \overbrace{\max_{x_{t:T}} E_{q_{t:T}} \left[\sum_{\tau=t}^{T} u_\tau(x_{1:\tau}, q_\tau) - f(x_{1:T}) \right]} \right] \\
&= \max_{x_{t-1}} \max_{x_{t:T}} E_{q_{t:T}} \left[E_{q_{t-1}} \sum_{\tau=1}^{t-1} u_\tau(x_{1:\tau}, q_\tau) + \sum_{\tau=t}^{T} u_\tau(x_{1:\tau}, q_\tau) - f(x_{1:T}) \right] \;=\; A_{t-1}.
\end{aligned}$$

□

Corollary 13 (FPL Bounds for adaptive adversary) *Theorems 5–8 also hold for an adaptive adversary in case of independent randomization $q \rightsquigarrow q_t$.*

Lemma 12 shows that every bound of the form $A_1 \le 0$ proven for an oblivious adversary, implies an analogous bound $B_1 \le 0$ for an adaptive adversary. Note that this strong statement holds only for the *full observation game*, i.e. if after each time step we learn all losses. In partial observation games such as the Bandit case (Auer et al., 1995), our actual action may depend on our past action by means of our past observation, and the assertion no longer holds. In this case, FPL with an adaptive adversary can be analyzed as shown by McMahan and Blum (2004); Poland and Hutter (2005). Finally, y_t^{IFPL} can additionally depend on x_t, but the “reduced” dependencies (14) are the same as for FPL, hence, IFPL bounds also hold for adaptive adversary.

9. Miscellaneous

Bounds with high probability. We have derived several bounds for the expected loss $\ell_{1:T}$ of FPL. The *actual* loss at time t is $u_t = M(s_{<t} + \frac{k-q}{\eta_t}) \circ s_t$. A simple Markov inequality shows that the total actual loss $u_{1:T}$ exceeds the total expected loss $\ell_{1:T} = E[u_{1:T}]$ by a factor of $c > 1$ with probability at most $1/c$:

$$P[u_{1:T} \ge c \cdot \ell_{1:T}] \;\le\; 1/c.$$

Randomizing independently for each t as described in the previous Section, the actual loss is $u_t = M(s_{<t} + \frac{k-q_t}{\eta_t}) \circ s_t$ with the same expected loss $\ell_{1:T} = E[u_{1:T}]$ as before. The advantage of independent randomization is that we can get a much better high-probability bound. We can exploit a Chernoff-Hoeffding bound (McDiarmid, 1989, Cor.5.2b), valid for arbitrary independent random variables $0 \le u_t \le 1$ for $t = 1,...,T$:

$$P\Big[|u_{1:T} - E[u_{1:T}]| \ge \delta E[u_{1:T}] \Big] \;\le\; 2\exp(-\tfrac{1}{3}\delta^2 E[u_{1:T}]), \qquad 0 \le \delta \le 1.$$

For $\delta = \sqrt{3c/\ell_{1:T}}$ we get

$$P[|u_{1:T} - \ell_{1:T}| \geq \sqrt{3c\ell_{1:T}}] \leq 2\mathrm{e}^{-c} \quad \text{as soon as} \quad \ell_{1:T} \geq 3c. \tag{17}$$

Using (17), the bounds for $\ell_{1:T}$ of Theorems 5–8 can be rewritten to yield similar bounds with high probability $(1-2\mathrm{e}^{-c})$ for $u_{1:T}$ with small extra regret $\propto \sqrt{c\cdot L}$ or $\propto \sqrt{c\cdot s^i_{1:T}}$. Furthermore, (17) shows that with probability 1, $u_{1:T}/\ell_{1:T}$ converges rapidly to 1 for $\ell_{1:T} \to \infty$. Hence we may use the easier to compute $\eta_t = \sqrt{K/2u_{<t}}$ instead of $\eta_t = \sqrt{K/2(\ell_{<t}+1)}$, likely with similar bounds on the regret.

Computational Aspects. It is easy to generate the randomized decision of FPL. Indeed, only a single initial exponentially distributed vector $q \in \mathbb{R}^n$ is needed. Only for self-confident $\eta_t \propto 1/\sqrt{\ell_{<t}}$ (see Theorem 7) we need to compute expectations explicitly. Given η_t, from $t \rightsquigarrow t+1$ we need to compute ℓ_t in order to update η_t. Note that $\ell_t = w_t \circ s_t$, where $w^i_t = P[I_t = i]$ and $I_t := \mathrm{argmin}_{i\in\mathcal{E}}\{s^i_{<t} + \frac{k^i - q^i}{\eta_t}\}$ is the actual (randomized) prediction of FPL. With $s := s_{<t} + k/\eta_t$, $P[I_t = i]$ has the following representation:

$$\begin{aligned}
P[I_t = i] &= P[s - \frac{q^i}{\eta_t} \leq s - \frac{q^j}{\eta_t}\ \forall j \neq i] \\
&= \int P[s - \frac{q^i}{\eta_t} = m \ \wedge\ s - \frac{q^j}{\eta_t} \geq m\ \forall j \neq i] dm \\
&= \int P[q^i = \eta_t(s^i - m)] \cdot \prod_{j\neq i} P[q^j \leq \eta_t(s^j - m)] dm \\
&= \int_{-\infty}^{s^{min}} \eta_t \mathrm{e}^{-\eta_t(s^i - m)} \prod_{j\neq i}(1 - \mathrm{e}^{-\eta_t(s^j - m)}) dm \\
&= \sum_{\mathcal{M}:\{i\}\subseteq\mathcal{M}\subseteq\mathcal{N}} \frac{(-)^{|\mathcal{M}|-1}}{|\mathcal{M}|} \mathrm{e}^{-\eta_t \sum_{j\in\mathcal{M}}(s^j - s^{min})}.
\end{aligned}$$

In the last equality we expanded the product and performed the resulting exponential integrals. For finite n, the second to last one-dimensional integral should be numerically feasible. Once the product $\prod_{j=1}^n (1 - \mathrm{e}^{-\eta_t(s^j - m)})$ has been computed in time $O(n)$, the argument of the integral can be computed for each i in time $O(1)$, hence the overall time to compute ℓ_t is $O(c\cdot n)$, where c is the time to numerically compute one integral. For infinite n, the last sum may be approximated by the dominant contributions. Alternatively, one can modify the algorithm by considering only a finite pool of experts in each time step; see next paragraph. The expectation may also be approximated by (Monte Carlo) sampling I_t several times.

Recall that approximating $\ell_{<t}$ can be avoided by using $s^{min}_{<t}$ (Theorem 8) or $u_{<t}$ (bounds with high probability) instead.

Finitized expert pool. In the case of an infinite expert class, FPL has to compute a minimum over an infinite set in each time step, which is not directly feasible. One possibility to address this is to choose the experts from a *finite pool* in each time step. This is the case in the algorithm of Gentile (2003), and also discussed by Littlestone and Warmuth (1994). For FPL, we can obtain this behavior by introducing an *entering time* $\tau^i \geq 1$ for each expert. Then expert i is not considered for $i < \tau^i$. In the bounds, this leads to an additional $\frac{1}{\eta_T}$ in Theorem 2 and Corollary 3 and a further additional τ^i in the final bounds (Theorems 5–8), since we must add the regret of the best expert in hindsight which has already entered the game and the best expert in hindsight at all. Selecting $\tau^i = k^i$ implies bounds for FPL with entering times similar to the ones we derived here. The details and proofs for this construction can be found in (Poland and Hutter, 2005).

Deterministic prediction and absolute loss. Another use of w_t from the second last paragraph is the following: If the decision space is $\mathcal{D}=\Delta$, then FPL may make a deterministic decision $d=w_t\in\Delta$ at time t with bounds now holding for sure, instead of selecting e_i with probability w_t^i. For example for the absolute loss $s_t^i = |x_t - y_t^i|$ with observation $x_t \in [0,1]$ and predictions $y_t^i \in [0,1]$, a master algorithm predicting deterministically $w_t{\circ}y_t\in[0,1]$ suffers absolute loss $|x_t-w_t{\circ}y_t|\leq\sum_i w_t^i|x_t-y_t^i|=\ell_t$, and hence has the same (or better) performance guarantees as FPL. In general, masters can be chosen deterministic if prediction space $\mathcal{Y}$ and loss-function $\text{Loss}(x,y)$ are convex. For $x_t,y_t^i \in \{0,1\}$, the absolute loss $|x_t - p_t|$ of a master deterministically predicting $p_t \in [0,1]$ actually coincides with the p_t-expected 0/1 loss of a master predicting 1 with probability p_t. Hence a regret bound for the absolute loss also implies the same regret for the 0/1 loss.

10. Discussion and Open Problems

How does FPL compare with other expert advice algorithms? We briefly discuss four issues, summarized in Table 2.

Static bounds. Here the coefficient of the regret term $\sqrt{KL}$, referred to as the *leading constant* in the sequel, is 2 for FPL (Theorem 5). It is thus a factor of $\sqrt{2}$ worse than the Hedge bound for arbitrary loss by Freund and Schapire (1997), which is sharp in some sense (Vovk, 1995). This is the price one pays for the elegance of FPL. There is evidence that this (worst-case) difference really exists and is not only a proof artifact. For special loss functions, the bounds can sometimes be improved, e.g. to a leading constant of 1 in the static (randomized) WM case with 0/1 loss (Cesa-Bianchi et al., 1997)[3]. Because of the structure of the FPL algorithm however, it is questionable if corresponding bounds hold there.

Dynamic bounds. Not knowing the right learning rate in advance usually costs a factor of $\sqrt{2}$. This is true for Hannan's algorithm (Kalai and Vempala, 2003) as well as in all our cases. Also for binary prediction with uniform complexities and 0/1 loss, this result has been established recently – Yaroshinsky et al. (2004) show a dynamic regret bound with leading constant $\sqrt{2}(1+\varepsilon)$. Remarkably, the best dynamic bound for a WM variant proven by Auer et al. (2002) has a leading constant $2\sqrt{2}$, which matches ours. Considering the difference in the static case, we therefore conjecture that a bound with leading constant of 2 holds for a dynamic Hedge algorithm.

General weights. While there are several dynamic bounds for uniform weights, the only previous result for non-uniform weights we know of is (Gentile, 2003, Cor.16), which gives the dynamic bound $\ell_{1:T}^{\text{Gentile}} \leq s_{1:T}^i + i + O\left[\sqrt{(s_{1:T}^i+i)\ln(s_{1:T}^i+i)}\right]$ for a p-norm algorithm for the absolute loss. This is comparable to our bound for rapidly decaying weights $w^i = \exp(-i)$, i.e. $k^i = i$. Our hierarchical FPL bound in Theorem 9 (b) generalizes this to arbitrary weights and losses and strengthens it, since both, asymptotic order and leading constant, are smaller.

It seems that the analysis of all experts algorithms, including Weighted Majority variants and FPL, gets more complicated for general weights together with adaptive learning rate, because the

3. While FPL and Hedge and WMR (Littlestone and Warmuth, 1994) can sample an expert without knowing its prediction, Cesa-Bianchi et al. (1997) need to know the experts' predictions. Note also that for many (smooth) loss-functions like the quadratic loss, finite regret can be achieved (Vovk, 1990).

η	Loss	conjecture	Lower Bound	Upper Bound
static	0/1	1	1?	1 (Cesa-Bianchi et al., 1997)
static	any	$\sqrt{2}$!	$\sqrt{2}$ (Vovk, 1995)	$\sqrt{2}$ (Hedge), 2 (FPL)
dynamic	0/1	$\sqrt{2}$	1? (Hutter, 2003b)	$\sqrt{2}$ (Yaroshinsky) , $2\sqrt{2}$ (Auer 2002)
dynamic	any	2	$\sqrt{2}$ (Vovk, 1995)	$2\sqrt{2}$ (FPL), 2 (Hutter, 2003b, Bayes)

Table 2: Comparison of the constants c in regrets $c\sqrt{\text{Loss}\times\ln n}$ for various settings and algorithms.

choice of the learning rate must account for both the weight of the best expert (in hindsight) and its loss. Both quantities are not known in advance, but may have a different impact on the learning rate: While increasing the current loss estimate always decreases η_t, the optimal learning rate for an expert with higher complexity would be larger. On the other hand, all analyses known so far require decreasing η_t. Nevertheless we conjecture that the bounds $\propto\sqrt{Tk^i}$ and $\propto\sqrt{s^i_{1:T}k^i}$ also hold without the hierarchy trick, probably by using expert dependent learning rate η^i_t.

Comparison to Bayesian sequence prediction. We can also compare the *worst-case* bounds for FPL obtained in this work to similar bounds for *Bayesian sequence prediction*. Let $\{\nu_i\}$ be a class of probability distributions over sequences and assume that the true sequence is sampled from $\mu\in\{\nu_i\}$ with complexity k^μ ($\sum_i e^{-k^{\nu_i}}\leq 1$). Then it is known that the Bayes optimal predictor based on the $e^{-k^{\nu_i}}$-weighted mixture of ν_i's has an expected total loss of at most $L^\mu+2\sqrt{L^\mu k^\mu}+2k^\mu$, where L^μ is the expected total loss of the Bayes optimal predictor based on μ (Hutter, 2003a, Thm.2), (Hutter, 2004b, Thm.3.48). Using FPL, we obtained the same bound except for the leading order constant, but for any sequence independently of the assumption that it is generated by μ. This is another indication that a PEA bound with leading constant 2 could hold. See Hutter (2004a), Hutter (2003b, Sec.6.3) and Hutter (2004b, Sec.3.7.4) for a more detailed comparison of Bayes bounds with PEA bounds.

Acknowledgments

This work was supported by SNF grant 2100-67712.02.

References

P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Proc. 36th Annual Symposium on Foundations of Computer Science (FOCS 1995)*, pages 322–331, Los Alamitos, CA, 1995. IEEE Computer Society Press.

P. Auer, N. Cesa-Bianchi, and C. Gentile. Adaptive and self-confident on-line learning algorithms. *Journal of Computer and System Sciences*, 64:48–75, 2002.

P. Auer and C. Gentile. Adaptive and self-confident on-line learning algorithms. In *Proc. 13th Conference on Computational Learning Theory*, pages 107–117. Morgan Kaufmann, San Francisco, 2000.

N. Cesa-Bianchi, Y. Freund, D. Haussler, D. Helmbold, R. Schapire, and M. K. Warmuth. How to use expert advice. *Journal of the ACM*, 44(3):427–485, 1997.

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

C. Gentile. The robustness of the p-norm algorithm. *Machine Learning*, 53(3):265–299, 2003.

J. Hannan. Approximation to Bayes risk in repeated plays. In M. Dresher, A. W. Tucker, and P. Wolfe, editors, *Contributions to the Theory of Games 3*, pages 97–139. Princeton University Press, 1957.

M. Hutter. Convergence and loss bounds for Bayesian sequence prediction. *IEEE Trans. on Information Theory*, 49(8):2061–2067, 2003a. URL `http://arxiv.org/abs/cs.LG/0301014`.

M. Hutter. Optimality of universal Bayesian prediction for general loss and alphabet. *Journal of Machine Learning Research*, 4:971–1000, 2003b. URL `http://arxiv.org/abs/cs.LG/0311014`.

M. Hutter. Online prediction – Bayes versus experts. Technical report, July 2004a. URL `http://www.idsia.ch/~marcus/ai/bayespea.htm`. Presented at the EU PASCAL Workshop on Learning Theoretic and Bayesian Inductive Principles (LTBIP-2004).

M. Hutter. *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability*. Springer, Berlin. 300 pages, 2004b. URL `http://www.idsia.ch/~marcus/ai/uaibook.htm`.

M. Hutter and J. Poland. Prediction with expert advice by following the perturbed leader for general weights. In *Proc. 15th International Conf. on Algorithmic Learning Theory (ALT-2004)*, volume 3244 of *LNAI*, pages 279–293, Padova, 2004. Springer, Berlin. URL `http://arxiv.org/abs/cs.LG/0405043`.

A. Kalai and S. Vempala. Efficient algorithms for online decision. In *Proc. 16th Annual Conference on Learning Theory (COLT-2003)*, Lecture Notes in Artificial Intelligence, pages 506–521, Berlin, 2003. Springer.

N. Littlestone and M. K. Warmuth. The weighted majority algorithm. In *30th Annual Symposium on Foundations of Computer Science*, pages 256–261, Research Triangle Park, North Carolina, 1989. IEEE.

N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.

C. McDiarmid. On the method of bounded differences. *Surveys in Combinatorics*, 141, London Mathematical Society Lecture Notes Series:148–188, 1989.

H. B. McMahan and A. Blum. Online geometric optimization in the bandit setting against an adaptive adversary. In *17th Annual Conference on Learning Theory (COLT)*, volume 3120 of *LNCS*, pages 109–123. Springer, 2004.

J. Poland and M. Hutter. Master algorithms for active experts problems based on increasing loss values. In *Annual Machine Learning Conference of Belgium and the Netherlands (Benelearn-2005)*, Enschede, 2005. URL `http://arxiv.org/abs/cs.LG/0502067`.

V. G. Vovk. Aggregating strategies. In *Proc. Third Annual Workshop on Computational Learning Theory*, pages 371–383, Rochester, New York, 1990. ACM Press.

V. G. Vovk. A game of prediction with expert advice. In *Proc. 8th Annual Conference on Computational Learning Theory*, pages 51–60. ACM Press, New York, NY, 1995.

R. Yaroshinsky, R. El-Yaniv, and S. Seiden. How to better use expert advice. *Machine Learning*, 55 (3):271–309, 2004.

Journal of Machine Learning Research 6 (2005) 661–694 Submitted 2/04; Revised 11/04; Published 4/05

Variational Message Passing

John Winn JWINN@MICROSOFT.COM
Christopher M. Bishop CMBISHOP@MICROSOFT.COM
Microsoft Research Cambridge
Roger Needham Building
7 J. J. Thomson Avenue
Cambridge CB3 0FB, U.K.

Editor: Tommi Jaakkola

Abstract

Bayesian inference is now widely established as one of the principal foundations for machine learning. In practice, exact inference is rarely possible, and so a variety of approximation techniques have been developed, one of the most widely used being a deterministic framework called variational inference. In this paper we introduce Variational Message Passing (VMP), a general purpose algorithm for applying variational inference to Bayesian Networks. Like belief propagation, VMP proceeds by sending messages between nodes in the network and updating posterior beliefs using local operations at each node. Each such update increases a lower bound on the log evidence (unless already at a local maximum). In contrast to belief propagation, VMP can be applied to a very general class of conjugate-exponential models because it uses a factorised variational approximation. Furthermore, by introducing additional variational parameters, VMP can be applied to models containing non-conjugate distributions. The VMP framework also allows the lower bound to be evaluated, and this can be used both for model comparison and for detection of convergence. Variational message passing has been implemented in the form of a general purpose inference engine called VIBES (‘Variational Inference for BayEsian networkS’) which allows models to be specified graphically and then solved variationally without recourse to coding.

Keywords: Bayesian networks, variational inference, message passing

1. Introduction

Variational inference methods (Neal and Hinton, 1998; Jordan et al., 1998) have been used successfully for a wide range of models, and new applications are constantly being explored. In each previous application, the equations for optimising the variational approximation have been worked out by hand, a process which is both time consuming and error prone. For several other inference methods, general purpose algorithms have been developed which can be applied to large classes of probabilistic models. For example, *belief propagation* can be applied to any acyclic discrete network (Pearl, 1986) or mixed-Gaussian network (Lauritzen, 1992), and the Monte Carlo algorithm described in Thomas et al. (1992) can perform Gibbs sampling in almost any Bayesian network. Similarly, *expectation propagation* (Minka, 2001) has been successfully applied to a wide range of models. Each of these algorithms relies on being able to decompose the required computation into calculations that are local to each node in the graph and which require only messages passed along the edges connected to that node.

However, Monte Carlo methods are computationally very intensive, and also suffer from difficulties in diagnosing convergence, while belief propagation is only guaranteed to converge for tree-structured graphs. Expectation propagation is limited to certain classes of model for which the required expectations can be evaluated, is also not guaranteed to converge in general, and is prone to finding poor solutions in the case of multi-modal distributions. For these reasons the framework of variational inference has received much attention.

In this paper, the variational message passing algorithm is developed, which optimises a variational bound using a set of local computations for each node, together with a mechanism for passing messages between the nodes. VMP allows variational inference to be applied automatically to a large class of Bayesian networks, without the need to derive application-specific update equations. In VMP, the messages are exponential family distributions, summarised either by their natural parameter vector (for child-to-parent messages) or by a vector of moments (for parent-to-child messages). These messages are defined so that the optimal variational distribution for a node can be found by summing the messages from its children together with a function of the messages from its parents, where this function depends on the conditional distribution for the node.

The VMP framework applies to models described by directed acyclic graphs in which the conditional distributions at each node are members of the exponential family, which therefore includes discrete, Gaussian, Poisson, gamma, and many other common distributions as special cases. For example, VMP can handle a general DAG of discrete nodes, or of linear-Gaussian nodes, or an arbitrary combination of the two provided there are no links going from continuous to discrete nodes. This last restriction can be lifted by introducing further variational bounds, as we shall discuss. Furthermore, the marginal distribution of observed data represented by the graph is not restricted to be from the exponential family, but can come from a very broad class of distributions built up from exponential family building blocks. The framework therefore includes many well known machine learning algorithms such as hidden Markov models, probabilistic PCA, factor analysis and Kalman filters, as well as mixtures and hierarchical mixtures of these.

Note that, since we work in a fully Bayesian framework, latent variables and parameters appear on an equal footing (they are all unobserved stochastic variables which are marginalised out to make predictions). If desired, however, point estimates for parameters can be made simply by maximising the same bound as used for variational inference.

As an illustration of the power of the message passing viewpoint, we use VMP within a software tool called VIBES (Variational Inference in BayEsian networkS) which allows a model to be specified by drawing its graph using a graphical interface, and which then performs variational inference automatically on this graph.

The paper is organised as follows. Section 2 gives a brief review of variational inference methods. Section 3 contains the derivation of the variational message passing algorithm, along with an example of its use. In Section 4 the class of models which can be handled by the algorithm is defined, while Section 5 describes the VIBES software. Some extensions to the algorithm are given in Section 6, and Section 7 concludes with an overall discussion and suggestions for future research directions.

2. Variational Inference

In this section, variational inference will be reviewed briefly with particular focus on the case where the variational distribution has a factorised form. The random variables in the model will be denoted by $\mathbf{X} = (\mathbf{V}, \mathbf{H})$ where $\mathbf{V}$ are the visible (observed) variables and $\mathbf{H}$ are the hidden (latent) variables. We assume that the model has the form of a Bayesian network and so the joint distribution $P(\mathbf{X})$ can be expressed in terms of the conditional distributions at each node i,

$$P(\mathbf{X}) = \prod_i P(X_i \,|\, \mathrm{pa}_i) \tag{1}$$

where pa_i denotes the set of variables corresponding to the parents of node i and X_i denotes the variable or group of variables associated with node i.

Ideally, we would like to perform exact inference within this model to find posterior marginal distributions over individual latent variables. Unfortunately, exact inference algorithms, such as the junction tree algorithm (Cowell et al., 1999), are typically only applied to discrete or linear-Gaussian models and are computationally intractable for all but the simplest models. Instead, we must turn to approximate inference methods. Here, we consider the deterministic approximation method of variational inference.

The goal in variational inference is to find a tractable variational distribution $Q(\mathbf{H})$ that closely approximates the true posterior distribution $P(\mathbf{H} \,|\, \mathbf{V})$. To do this we note the following decomposition of the log marginal probability of the observed data, which holds for any choice of distribution: $Q(\mathbf{H})$

$$\ln P(\mathbf{V}) = \mathcal{L}(Q) + \mathrm{KL}(Q \,||\, P). \tag{2}$$

Here

$$\begin{aligned} \mathcal{L}(Q) &= \sum_{\mathbf{H}} Q(\mathbf{H}) \ln \frac{P(\mathbf{H}, \mathbf{V})}{Q(\mathbf{H})} \\ \mathrm{KL}(Q \,||\, P) &= -\sum_{\mathbf{H}} Q(\mathbf{H}) \ln \frac{P(\mathbf{H} \,|\, \mathbf{V})}{Q(\mathbf{H})} \end{aligned} \tag{3}$$

and the sums are replaced by integrals in the case of continuous variables. Here $\mathrm{KL}(Q \,||\, P)$ is the Kullback-Leibler divergence between the true posterior $P(\mathbf{H} \,|\, \mathbf{V})$ and the variational approximation $Q(\mathbf{H})$. Since this satisfies $\mathrm{KL}(Q \,||\, P) \geqslant 0$ it follows from (2) that the quantity $\mathcal{L}(Q)$ forms a lower bound on $\ln P(\mathbf{V})$.

We now choose some family of distributions to represent $Q(\mathbf{H})$ and then seek a member of that family that maximises the lower bound $\mathcal{L}(Q)$ and hence minimises the Kullback-Leibler divergence $\mathrm{KL}(Q \,||\, P)$. If we allow $Q(\mathbf{H})$ to have complete flexibility then we see that the maximum of the lower bound occurs when the Kullback-Leibler divergence is zero. In this case, the variational posterior distribution equals the true posterior and $\mathcal{L}(Q) = \ln P(\mathbf{V})$. However, working with the true posterior distribution is computationally intractable (otherwise we wouldn't be resorting to variational methods). We must therefore consider a more restricted family of Q distributions which has the property that the lower bound (3) can be evaluated and optimised efficiently and yet which is still sufficiently flexible as to give a good approximation to the true posterior distribution.

This method leads to minimisation of 'exclusive' divergence $\mathrm{KL}(Q \,||\, P)$ rather than the 'inclusive' divergence $\mathrm{KL}(P \,||\, Q)$. Minimising the exclusive divergence can lead to a Q which ignores modes of P. However, minimising the inclusive divergence can lead to Q assigning posterior mass to regions where P has vanishing density. If the latter behaviour is preferred, then there are alternative approximation techniques for minimising the inclusive divergence, including expectation propagation (Minka, 2001).

2.1 Factorised Variational Distributions

We wish to choose a variational distribution $Q(\mathbf{H})$ with a simpler structure than that of the model, so as to make calculation of the lower bound $\mathcal{L}(Q)$ tractable. One way to simplify the dependency structure is by choosing a variational distribution where disjoint groups of variables are independent. This is equivalent to choosing Q to have a factorised form

$$Q(\mathbf{H}) = \prod_i Q_i(\mathbf{H}_i), \tag{4}$$

where $\{\mathbf{H}_i\}$ are the disjoint groups of variables. This approximation has been successfully used in many applications of variational methods (Attias, 2000; Ghahramani and Beal, 2001; Bishop, 1999). Substituting (4) into (3) gives

$$\mathcal{L}(Q) = \sum_{\mathbf{H}} \prod_i Q_i(\mathbf{H}_i) \ln P(\mathbf{H},\mathbf{V}) - \sum_i \sum_{\mathbf{H}_i} Q_i(\mathbf{H}_i) \ln Q_i(\mathbf{H}_i).$$

We now separate out all terms in one factor Q_j,

$$\begin{aligned}\mathcal{L}(Q) &= \sum_{\mathbf{H}_j} Q_j(\mathbf{H}_j) \langle \ln P(\mathbf{H},\mathbf{V})\rangle_{\sim Q_j(\mathbf{H}_j)} + \mathbb{H}(Q_j) + \sum_{i \neq j} \mathbb{H}(Q_i) \\ &= -\mathrm{KL}(Q_j \,||\, Q_j^\star) + \text{terms not in } Q_j \end{aligned} \tag{5}$$

where $\mathbb{H}$ denotes entropy and we have introduced a new distribution $Q_j^\star$, defined by

$$\ln Q_j^\star(\mathbf{H}_j) = \langle \ln P(\mathbf{H},\mathbf{V})\rangle_{\sim Q(\mathbf{H}_j)} + \text{const.} \tag{6}$$

and $\langle \cdot \rangle_{\sim Q(\mathbf{H}_j)}$ denotes an expectation with respect to all factors except $Q_j(\mathbf{H}_j)$. The bound is maximised with respect to Q_j when the KL divergence in (5) is zero, which occurs when $Q_j = Q_j^\star$. Therefore, we can maximise the bound by setting Q_j equal to $Q_j^\star$. Taking exponentials of both sides we obtain

$$Q_j^\star(\mathbf{H}_j) = \frac{1}{Z} \exp\langle \ln P(\mathbf{H},\mathbf{V})\rangle_{\sim Q(\mathbf{H}_j)}, \tag{7}$$

where Z is the normalisation factor needed to make $Q_j^\star$ a valid probability distribution. Note that the equations for all of the factors are coupled since the solution for each $Q_j(\mathbf{H}_j)$ depends on expectations with respect to the other factors $Q_{i \neq j}$. The variational optimisation proceeds by initialising each of the $Q_j(\mathbf{H}_j)$ and then cycling through each factor in turn replacing the current distribution with a revised estimate given by (7).

3. Variational Message Passing

In this section, the variational message passing algorithm will be derived and shown to optimise a factorised variational distribution using a message passing procedure on a graphical model. For the initial derivation, it will be assumed that the variational distribution is factorised with respect to each hidden variable H_i and so can be written

$$Q(\mathbf{H}) = \prod_i Q_i(H_i).$$

From (6), the optimised form of the jth factor is given by

$$\ln Q_j^\star(H_j) = \langle \ln P(\mathbf{H},\mathbf{V})\rangle_{\sim Q(H_j)} + \text{const.}$$

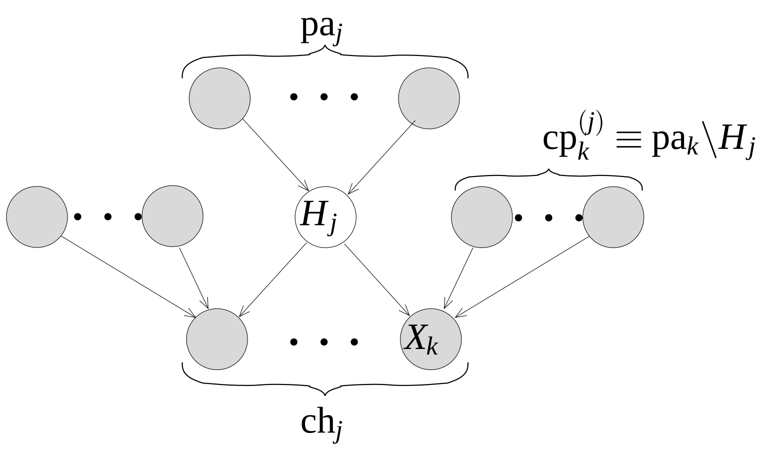


Figure 1: A key observation is that the variational update equation for a node H_j depends only on expectations over variables in the Markov blanket of that node (shown shaded), defined as the set of parents, children and co-parents of that node.

We now substitute in the form of the joint probability distribution of a Bayesian network, as given in (1),

$$\ln Q_j^\star(H_j) = \left\langle \sum_i \ln P(X_i \,|\, \mathrm{pa}_i) \right\rangle_{\sim Q(H_j)} + \text{const.}$$

Any terms in the sum over i that do not depend on H_j will be constant under the expectation and can be subsumed into the constant term. This leaves only the conditional $P(H_j \,|\, \mathrm{pa}_j)$ together with the conditionals for all the children of H_j, as these have H_j in their parent set,

$$\ln Q_j^\star(H_j) = \langle \ln P(H_j \,|\, \mathrm{pa}_j) \rangle_{\sim Q(H_j)} + \sum_{k \in \mathrm{ch}_j} \langle \ln P(X_k \,|\, \mathrm{pa}_k) \rangle_{\sim Q(H_j)} + \text{const.} \tag{8}$$

where ch_j are the children of node j in the graph. Thus, the expectations required to evaluate $Q_j^\star$ involve only those variables lying in the Markov blanket of H_j, consisting of its parents, children and co-parents[1] $\mathrm{cp}_k^{(j)}$. This is illustrated in the form of a directed graphical model in Figure 1. Note that we use the notation X_k to denote both a random variable and the corresponding node in the graph. The optimisation of Q_j can therefore be expressed as a local computation at the node H_j. This computation involves the sum of a term involving the parent nodes, along with one term from each of the child nodes. These terms can be thought of as 'messages' from the corresponding nodes. Hence, we can decompose the overall optimisation into a set of local computations that depend only on messages from neighbouring (i.e. parent and child) nodes in the graph.

3.1 Conjugate-Exponential Models

The exact form of the messages in (8) will depend on the functional form of the conditional distributions in the model. It has been noted (Attias, 2000; Ghahramani and Beal, 2001) that important simplifications to the variational update equations occur when the distributions of variables, condi-

1. The co-parents of a node X are all the nodes with at least one child which is also a child of X (excluding X itself).

tioned on their parents, are drawn from the exponential family and are conjugate[2] with respect to the distributions over these parent variables. A model where both of these constraints hold is known as a *conjugate-exponential* model.

A conditional distribution is in the exponential family if it can be written in the form

$$P(\mathbf{X} \mid \mathbf{Y}) = \exp[\phi(\mathbf{Y})^{\mathrm{T}}\mathbf{u}(\mathbf{X}) + f(\mathbf{X}) + g(\mathbf{Y})] \tag{9}$$

where $\phi(\mathbf{Y})$ is called the *natural parameter* vector and $\mathbf{u}(\mathbf{X})$ is called the *natural statistic* vector. The term $g(\mathbf{Y})$ acts as a normalisation function that ensures the distribution integrates to unity for any given setting of the parameters $\mathbf{Y}$. The exponential family contains many common distributions, including the Gaussian, gamma, Dirichlet, Poisson and discrete distributions. The advantages of exponential family distributions are that expectations of their logarithms are tractable to compute and their state can be summarised completely by the natural parameter vector. The use of conjugate distributions means that the posterior for each factor has the same form as the prior and so learning changes only the values of the parameters, rather than the functional form of the distribution.

If we know the natural parameter vector $\phi(\mathbf{Y})$ for an exponential family distribution, then we can find the expectation of the natural statistic vector with respect to the distribution. Rewriting (9) and defining $\widetilde{g}$ as a reparameterisation of g in terms of ϕ gives,

$$P(\mathbf{X} \mid \phi) = \exp[\phi^{\mathrm{T}}\mathbf{u}(\mathbf{X}) + f(\mathbf{X}) + \widetilde{g}(\phi)].$$

We integrate with respect to $\mathbf{X}$,

$$\int_{\mathbf{X}} \exp[\phi^{\mathrm{T}}\mathbf{u}(\mathbf{X}) + f(\mathbf{X}) + \widetilde{g}(\phi)]\, d\mathbf{X} = \int_{\mathbf{X}} P(\mathbf{X} \mid \phi)\, d\mathbf{X} \; = \; 1$$

and then differentiate with respect to ϕ

$$\begin{aligned}
\int_{\mathbf{X}} \frac{d}{d\phi} \exp[\phi^{\mathrm{T}}\mathbf{u}(\mathbf{X}) + f(\mathbf{X}) + \widetilde{g}(\phi)]\, d\mathbf{X} &= \frac{d}{d\phi}(1) \; = \; 0 \\
\int_{\mathbf{X}} P(\mathbf{X} \mid \phi) \left[\mathbf{u}(\mathbf{X}) + \frac{d\widetilde{g}(\phi)}{d\phi}\right] d\mathbf{X} &= 0.
\end{aligned}$$

And so the expectation of the natural statistic vector is given by

$$\langle \mathbf{u}(\mathbf{X}) \rangle_{P(\mathbf{X} \mid \phi)} \;=\; -\frac{d\widetilde{g}(\phi)}{d\phi}. \tag{10}$$

We will see later that the factors of our Q distribution will also be in the exponential family and will have the same natural statistic vector as the corresponding factor of P. Hence, the expectation of $\mathbf{u}$ under the Q distribution can also be found using (10).

3.2 Optimisation of Q in Conjugate-Exponential Models

We will now demonstrate how the optimisation of the variational distribution can be carried out, given that the model is conjugate-exponential. We consider the general case of optimising a factor

2. A parent distribution $P(X \mid Y)$ is said to be *conjugate* to a child distribution $P(W \mid X)$ if $P(X \mid Y)$ has the same functional form, with respect to X, as $P(W \mid X)$.

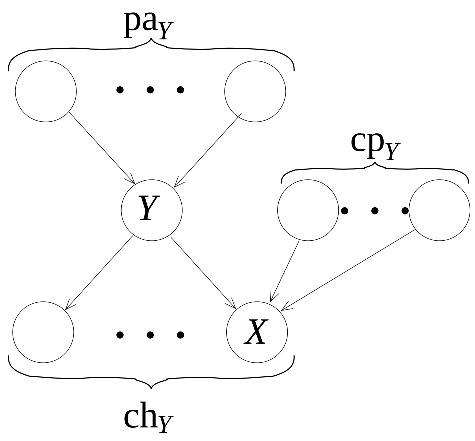


Figure 2: Part of a graphical model showing a node Y, the parents and children of Y, and the co-parents of Y with respect to a child node X.

$Q(Y)$ corresponding to a node Y, whose children include X, as illustrated in Figure 2. From (9), the log conditional probability of the variable Y given its parents can be written

$$\ln P(Y \mid \mathrm{pa}_Y) = \phi_Y(\mathrm{pa}_Y)^{\mathrm{T}} \mathbf{u}_Y(Y) + f_Y(Y) + g_Y(\mathrm{pa}_Y). \tag{11}$$

The subscript Y on each of the functions $\phi_Y, \mathbf{u}_Y, f_Y, g_Y$ is required as these functions differ for different members of the exponential family and so need to be defined separately for each node.

Consider a node $X \in \mathrm{ch}_Y$ which is a child of Y. The conditional probability of X given its parents will also be in the exponential family and so can be written in the form

$$\ln P(X \mid Y, \mathrm{cp}_Y) = \phi_X(Y, \mathrm{cp}_Y)^{\mathrm{T}} \mathbf{u}_X(X) + f_X(X) + g_X(Y, \mathrm{cp}_Y) \tag{12}$$

where cp_Y are the co-parents of Y with respect to X, in other words, the set of parents of X excluding Y itself. The quantity $P(Y \mid \mathrm{pa}_Y)$ in(11) can be thought of as a prior over Y, and $P(X \mid Y, \mathrm{cp}_Y)$ as a (contribution to) the likelihood of Y.

Example

If X is Gaussian distributed with mean Y and precision β, it follows that the co-parent set cp_Y contains only β, and the log conditional for X is

$$\ln P(X \mid Y, \beta) = \begin{bmatrix} \beta Y \\ -\beta/2 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} X \\ X^2 \end{bmatrix} + \tfrac{1}{2}(\ln \beta - \beta Y^2 - \ln 2\pi). \tag{13}$$

Conjugacy requires that the conditionals of (11) and (12) have the same functional form with respect to Y, and so the latter can be rewritten in terms of $\mathbf{u}_Y(Y)$ by defining functions ϕ_{XY} and λ as follows

$$\ln P(X \mid Y, \mathrm{cp}_Y) = \phi_{XY}(X, \mathrm{cp}_Y)^{\mathrm{T}} \mathbf{u}_Y(Y) + \lambda(X, \mathrm{cp}_Y). \tag{14}$$

It may appear from this expression that the function ϕ_{XY} depends on the form of the parent conditional $P(Y \mid \mathrm{pa}_Y)$ and so cannot be determined locally at X. This is not the case, because the conjugacy constraint dictates $\mathbf{u}_Y(Y)$ for any parent Y of X, implying that ϕ_{XY} can be found directly from the form of the conditional $P(X \mid \mathrm{pa}_X)$.

Example

Continuing the above example, we can find ϕ_{XY} by rewriting the log conditional in terms of Y to give

$$\ln P(X \mid Y, \beta) = \begin{bmatrix} \beta X \\ -\beta/2 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} Y \\ Y^2 \end{bmatrix} + \tfrac{1}{2}(\ln\beta - \beta X^2 - \ln 2\pi),$$

which lets us define ϕ_{XY} and dictate what $\mathbf{u}_Y(Y)$ must be to enforce conjugacy,

$$\phi_{XY}(X,\beta) \stackrel{\text{def}}{=} \begin{bmatrix} \beta X \\ -\beta/2 \end{bmatrix}, \qquad \mathbf{u}_Y(Y) = \begin{bmatrix} Y \\ Y^2 \end{bmatrix}. \tag{15}$$

From (12) and (14), it can be seen that $\ln P(X \mid Y, \mathrm{cp}_Y)$ is linear in $\mathbf{u}_X(X)$ and $\mathbf{u}_Y(Y)$ respectively. Conjugacy also dictates that this log conditional will be linear in $\mathbf{u}_Z(Z)$ for each co-parent $Z \in \mathrm{cp}_Y$. Hence, $\ln P(X \mid Y, \mathrm{cp}_Y)$ must be a multi-linear[3] function of the natural statistic functions $\mathbf{u}$ of X and its parents. This result is general, for any variable A in a conjugate-exponential model, the log conditional $\ln P(A \mid \mathrm{pa}_A)$ must be a multi-linear function of the natural statistic functions of A and its parents.

Example

The log conditional $\ln P(X \mid Y, \beta)$ in (13) is multi-linear in each of the vectors,

$$\mathbf{u}_X(X) = \begin{bmatrix} X \\ X^2 \end{bmatrix}, \quad \mathbf{u}_Y(Y) = \begin{bmatrix} Y \\ Y^2 \end{bmatrix}, \quad \mathbf{u}_\beta(\beta) = \begin{bmatrix} \beta \\ \ln\beta \end{bmatrix}.$$

Returning to the variational update equation (8) for a node Y, it follows that all the expectations on the right hand side can be calculated in terms of the $\langle \mathbf{u} \rangle$ for each node in the Markov blanket of Y. Substituting for these expectations, we get

$$\begin{aligned} \ln Q_Y^*(Y) &= \left\langle \phi_Y(\mathrm{pa}_Y)^{\mathrm{T}} \mathbf{u}_Y(Y) + f_Y(Y) + g_Y(\mathrm{pa}_Y) \right\rangle_{\sim Q(Y)} \\ &\quad + \sum_{k \in \mathrm{ch}_Y} \left\langle \phi_{XY}(X_k, \mathrm{cp}_k)^{\mathrm{T}} \mathbf{u}_Y(Y) + \lambda(X_k, \mathrm{cp}_k) \right\rangle_{\sim Q(Y)} + \text{const.} \end{aligned}$$

which can be rearranged to give

$$\begin{aligned} \ln Q_Y^*(Y) &= \left[\langle \phi_Y(\mathrm{pa}_Y) \rangle_{\sim Q(Y)} + \sum_{k \in \mathrm{ch}_Y} \langle \phi_{XY}(X_k, \mathrm{cp}_k) \rangle_{\sim Q(Y)} \right]^{\mathrm{T}} \mathbf{u}_Y(Y) \\ &\quad + f_Y(Y) + \text{const.} \end{aligned} \tag{16}$$

It follows that Q_Y^* is an exponential family distribution of the same form as $P(Y \mid \mathrm{pa}_Y)$ but with a natural parameter vector ϕ_Y^* such that

$$\phi_Y^* = \langle \phi_Y(\mathrm{pa}_Y) \rangle + \sum_{k \in \mathrm{ch}_Y} \langle \phi_{XY}(X_k, \mathrm{cp}_k) \rangle \tag{17}$$

where all expectations are with respect to Q. As explained above, the expectations of ϕ_Y and each ϕ_{XY} are multi-linear functions of the expectations of the natural statistic vectors corresponding to their dependent variables. It is therefore possible to reparameterise these functions in terms of these

3. A function f is a multi-linear function of variables $a, b \ldots$ if it varies linearly with respect to each variable, for example, $f(a,b) = ab + 3b$ is multi-linear in a and b. Although, strictly, this function is *affine* in a because of the constant term, we follow common usage and refer to it as linear.

expectations

$$\begin{aligned}\widetilde{\phi}_Y\left(\{\langle \mathbf{u}_i\rangle\}_{i\in \mathrm{pa}_Y}\right) &= \langle \phi_Y(\mathrm{pa}_Y)\rangle \\ \widetilde{\phi}_{XY}\left(\langle \mathbf{u}_k\rangle, \{\langle \mathbf{u}_j\rangle\}_{j\in \mathrm{cp}_k}\right) &= \langle \phi_{XY}(X_k, \mathrm{cp}_k)\rangle\,.\end{aligned}$$

The final step is to show that we can compute the expectations of the natural statistic vectors $\mathbf{u}$ under Q. From (16) any variable A has a factor Q_A with the same exponential family form as $P(A\,|\,\mathrm{pa}_A)$. Hence, the expectations of $\mathbf{u}_A$ can be found from the natural parameter vector of that distribution using (10). In the case where A is observed, the expectation is irrelevant and we can simply calculate $\mathbf{u}_A(A)$ directly.

Example

In (15), we defined $\phi_{XY}(X,\beta) = \begin{bmatrix} \beta X \\ -\beta/2 \end{bmatrix}$. We now reparameterise it as

$$\widetilde{\phi}_{XY}\left(\langle \mathbf{u}_X\rangle, \langle \mathbf{u}_\beta\rangle\right) \overset{\text{def}}{=} \begin{bmatrix} \langle \mathbf{u}_\beta\rangle_0 \langle \mathbf{u}_X\rangle_0 \\ -\frac{1}{2}\langle \mathbf{u}_\beta\rangle_0 \end{bmatrix}$$

where $\langle \mathbf{u}_X\rangle_0$ and $\langle \mathbf{u}_\beta\rangle_0$ are the first elements of the vectors $\langle \mathbf{u}_X\rangle$ and $\langle \mathbf{u}_\beta\rangle$ respectively (and so are equal to $\langle X\rangle$ and $\langle \beta\rangle$). As required, we have reparameterised ϕ_{XY} into a function $\widetilde{\phi}_{XY}$ which is a multi-linear function of natural statistic vectors.

3.3 Definition of the Variational Message Passing Algorithm

We have now reached the point where we can specify exactly what form the messages between nodes must take and so define the variational message passing algorithm. The message from a parent node Y to a child node X is just the expectation under Q of the natural statistic vector

$$\mathbf{m}_{Y\to X} = \langle \mathbf{u}_Y\rangle. \tag{18}$$

The message from a child node X to a parent node Y is

$$\mathbf{m}_{X\to Y} = \widetilde{\phi}_{XY}\left(\langle \mathbf{u}_X\rangle, \{\mathbf{m}_{i\to X}\}_{i\in \mathrm{cp}_Y}\right) \tag{19}$$

which relies on X having received messages previously from all the co-parents. If any node A is observed then the messages are as defined above but with $\langle \mathbf{u}_A\rangle$ replaced by $\mathbf{u}_A$.

Example

If X is Gaussian distributed with conditional $P(X\,|\,Y,\beta)$, the messages to its parents Y and β are

$$\mathbf{m}_{X\to Y} = \begin{bmatrix} \langle \beta\rangle \langle X\rangle \\ -\langle \beta\rangle/2 \end{bmatrix}, \qquad \mathbf{m}_{X\to \beta} = \begin{bmatrix} -\frac{1}{2}\left(\langle X^2\rangle - 2\langle X\rangle\langle Y\rangle + \langle Y^2\rangle\right) \\ \frac{1}{2} \end{bmatrix}$$

and the message from X to any child node is $\begin{bmatrix} \langle X\rangle \\ \langle X^2\rangle \end{bmatrix}$.

When a node Y has received messages from all parents and children, we can finds its updated posterior distribution Q_Y^* by finding its updated natural parameter vector ϕ_Y^*. This vector ϕ_Y^* is computed from all the messages received at a node using

$$\phi_Y^* = \widetilde{\phi}_Y\left(\{\mathbf{m}_{i\to Y}\}_{i\in \mathrm{pa}_Y}\right) + \sum_{j\in \mathrm{ch}_Y} \mathbf{m}_{j\to Y}, \tag{20}$$

which follows from (17). The new expectation of the natural statistic vector $\langle \mathbf{u}_Y \rangle_{Q_Y^*}$ can then be found, as it is a deterministic function of ϕ_Y^*.

The variational message passing algorithm uses these messages to optimise the variational distribution iteratively, as described in Algorithm 1 below. This algorithm requires that the lower bound $\mathcal{L}(Q)$ be evaluated, which will be discussed in Section 3.6.

Algorithm 1 The variational message passing algorithm

```
1. Initialise each factor distribution $Q_j$ by initialising the corresponding moment vector $\langle \mathbf{u}_j(X_j) \rangle$.
2. For each node $X_j$ in turn,
    • Retrieve messages from all parent and child nodes, as defined in (18) and (19). This will require child nodes to retrieve messages from the co-parents of $X_j$.
    • Compute updated natural parameter vector $\phi_j^*$ using (20).
    • Compute updated moment vector $\langle \mathbf{u}_j(X_j) \rangle$ given the new setting of the parameter vector.
3. Calculate the new value of the lower bound $\mathcal{L}(Q)$ (if required).
4. If the increase in the bound is negligible or a specified number of iterations has been reached, stop. Otherwise repeat from step 2.
```

3.4 Example: the Univariate Gaussian Model

To illustrate how variational message passing works, let us apply it to a model which represents a set of observed one-dimensional data $\{x_n\}_{n=1}^N$ with a univariate Gaussian distribution of mean μ and precision γ,

$$P(\mathbf{x} \mid \mathcal{H}) = \prod_{n=1}^{N} \mathcal{N}(x_n \mid \mu, \gamma^{-1}).$$

We wish to infer the posterior distribution over the parameters μ and γ. In this simple model the exact solution is tractable, which will allow us to compare the approximate posterior with the true posterior. Of course, for any practical application of VMP, the exact posterior would not be tractable otherwise we would not be using approximate inference methods.

In this model, the conditional distribution of each data point x_n is a univariate Gaussian, which is in the exponential family and so its logarithm can be expressed in standard form as

$$\ln P(x_n \mid \mu, \gamma^{-1}) = \begin{bmatrix} \gamma\mu \\ -\gamma/2 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} x_n \\ x_n^2 \end{bmatrix} + \frac{1}{2}(\ln\gamma - \gamma\mu^2 - \ln 2\pi)$$

and so $\mathbf{u}_x(x_n) = [x_n, x_n^2]^{\mathrm{T}}$. This conditional can also be written so as to separate out the dependencies on μ and γ

$$\ln P(x_n \mid \mu, \gamma^{-1}) = \begin{bmatrix} \gamma x_n \\ -\gamma/2 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \mu \\ \mu^2 \end{bmatrix} + \frac{1}{2}(\ln\gamma - \gamma x_n^2 - \ln 2\pi) \qquad (21)$$

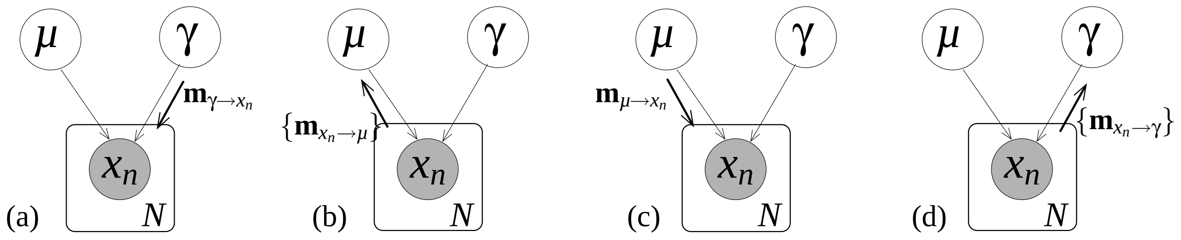

μ
γ
$\mathbf{m}_{\gamma\to x_n}$
$\{\mathbf{m}_{x_n\to\mu}\}$
$\mathbf{m}_{\mu\to x_n}$
$\{\mathbf{m}_{x_n\to\gamma}\}$
x_n
N
(a)
(b)
(c)
(d)

Figure 3: (a)-(d) Message passing procedure for variational inference in a univariate Gaussian model. The box around the x_i node denotes a *plate*, which indicates that the contained node and its connected edges are duplicated N times. The braces around the messages leaving the plate indicate that a set of N distinct messages are being sent.

$$= \begin{bmatrix} -\frac{1}{2}(x_n-\mu)^2 \\ \frac{1}{2} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \gamma \\ \ln\gamma \end{bmatrix} - \ln 2\pi \tag{22}$$

which shows that, for conjugacy, $\mathbf{u}_\mu(\mu)$ must be $[\mu,\mu^2]^{\mathrm{T}}$ and $\mathbf{u}_\gamma(\gamma)$ must be $[\gamma,\ln\gamma]^{\mathrm{T}}$ or linear transforms of these.[4] If we use a separate conjugate prior for each parameter then μ must have a Gaussian prior and γ a gamma prior since these are the exponential family distributions with these natural statistic vectors. Alternatively, we could have chosen a normal-gamma prior over both parameters which leads to a slightly more complicated message passing procedure. We define the parameter priors to have hyper-parameters m, β, a and b, so that

$$\begin{aligned} \ln P(\mu\,|\,m,\beta) &= \begin{bmatrix} \beta m \\ -\beta/2 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \mu \\ \mu^2 \end{bmatrix} + \frac{1}{2}(\ln\beta - \beta m^2 - \ln 2\pi) \\ \ln P(\gamma\,|\,a,b) &= \begin{bmatrix} -b \\ a-1 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \gamma \\ \ln\gamma \end{bmatrix} + a\ln b - \ln\Gamma(a). \end{aligned}$$

3.4.1 VARIATIONAL MESSAGE PASSING IN THE UNIVARIATE GAUSSIAN MODEL

We can now apply variational message passing to infer the distributions over μ and γ variationally. The variational distribution is fully factorised and takes the form

$$Q(\mu,\gamma) = Q_\mu(\mu)Q_\gamma(\gamma).$$

We start by initialising $Q_\mu(\mu)$ and $Q_\gamma(\gamma)$ and find initial values of $\langle\mathbf{u}_\mu(\mu)\rangle$ and $\langle\mathbf{u}_\gamma(\gamma)\rangle$. Let us choose to update $Q_\mu(\mu)$ first, in which case variational message passing will proceed as follows (illustrated in Figure 3a-d).

(a) As we wish to update $Q_\mu(\mu)$, we must first ensure that messages have been sent to the children of μ by any co-parents. Thus, messages $\mathbf{m}_{\gamma\to x_n}$ are sent from γ to each of the observed nodes x_n. These messages are the same, and are just equal to $\langle\mathbf{u}_\gamma(\gamma)\rangle = [\langle\gamma\rangle,\langle\ln\gamma\rangle]^{\mathrm{T}}$, where the expectation are with respect to the initial setting of Q_γ.

4. To prevent the need for linear transformation of messages, a normalised form of natural statistic vectors will always be used, for example $[\mu,\mu^2]^{\mathrm{T}}$ or $[\gamma,\ln\gamma]^{\mathrm{T}}$.

(b) Each x_n node has now received messages from all co-parents of μ and so can send a message to μ which is the expectation of the natural parameter vector in (21),

$$\mathbf{m}_{x_n \to \mu} = \begin{bmatrix} \langle \gamma \rangle x_n \\ -\langle \gamma \rangle / 2 \end{bmatrix}.$$

(c) Node μ has now received its full complement of incoming messages and can update its natural parameter vector,

$$\phi_\mu^* = \begin{bmatrix} \beta m \\ -\beta/2 \end{bmatrix} + \sum_{n=1}^{N} \mathbf{m}_{x_n \to \mu}.$$

The new expectation $\langle \mathbf{u}_\mu(\mu) \rangle$ can then be computed under the updated distribution Q_μ^* and sent to each x_n as the message $\mathbf{m}_{\mu \to x_n} = [\langle \mu \rangle, \langle \mu^2 \rangle]^{\mathrm{T}}$.

(d) Finally, each x_n node sends a message back to γ which is

$$\mathbf{m}_{x_n \to \gamma} = \begin{bmatrix} -\frac{1}{2}(x_n^2 - 2x_n \langle \mu \rangle + \langle \mu^2 \rangle) \\ \frac{1}{2} \end{bmatrix}$$

and γ can update its variational posterior

$$\phi_\gamma^* = \begin{bmatrix} -b \\ a-1 \end{bmatrix} + \sum_{n=1}^{N} \mathbf{m}_{x_n \to \gamma}.$$

As the expectation of $\mathbf{u}_\gamma(\gamma)$ has changed, we can now go back to step (a) and send an updated message to each x_n node and so on. Hence, in variational message passing, the message passing procedure is repeated again and again until convergence (unlike in belief propagation on a junction tree where the exact posterior is available after a message passing is performed once). Each round of message passing is equivalent to one iteration of the update equations in standard variational inference.

Figure 4 gives an indication of the accuracy of the variational approximation in this model, showing plots of both the true and variational posterior distributions for a toy example. The difference in shape between the two distributions is due to the requirement that Q be factorised. Because $\mathrm{KL}(Q\,||\,P)$ has been minimised, the optimal Q is the factorised distribution which lies slightly *inside* P.

3.5 Initialisation and Message Passing Schedule

The variational message passing algorithm is guaranteed to converge to a local minimum of the KL divergence. As with many approximate inference algorithms, including Expectation-Maximisation and Expectation Propagation, it is important to have a good initialisation to ensure that the local minimum that is found is sufficiently close to the global minimum. What makes a good initialisation will depend on the model. In some cases, initialising each factor to a broad distribution will suffice, whilst in others it may be necessary to use a heuristic, such as using K-means to initialise a mixture model.

The variational distribution in the example of Section 3.4 contained only two factors and so messages were passed back-and-forth so as to update these alternately. In fact, unlike belief propagation,

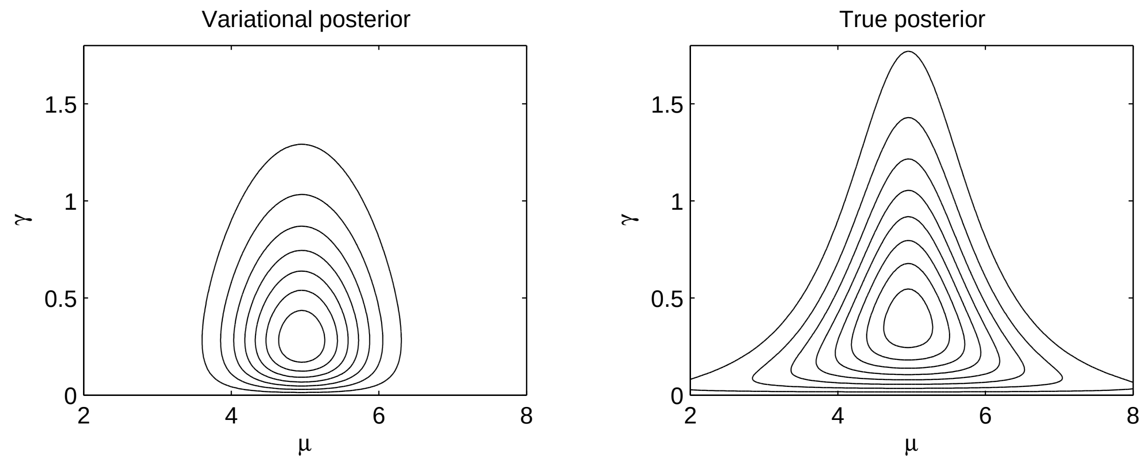


Figure 4: Contour plots of the variational and true posterior over the mean μ and precision γ of a Gaussian distribution, given four samples from $\mathcal{N}(x \mid 5, 1)$. The parameter priors are $P(\mu) = \mathcal{N}(0, 1000)$ and $P(\gamma) = \text{Gamma}(0.001, 0.001)$.

messages in VMP can be passed according to a very flexible schedule. At any point, any factor can be selected and it can be updated locally using only messages from its neighbours and co-parents. There is no requirement that factors be updated in any particular order. However, changing the update order can change which stationary point the algorithm converges to, even if the initialisation is unchanged.

Another constraint on belief propagation is that it is only exact for graphs which are trees and suffers from double-counting if loops are included. VMP does not have this restriction and can be applied to graphs of general form.

3.6 Calculation of the Lower Bound $\mathcal{L}(Q)$

The variational message passing algorithm makes use of the lower bound $\mathcal{L}(Q)$ as a diagnostic of convergence. Evaluating the lower bound is also useful for performing model selection, or model averaging, because it provides an estimate of the log evidence for the model.

The lower bound can also play a useful role in helping to check the correctness both of the analytical derivation of the update equations and of their software implementation, simply by evaluating the bound after updating each factor in the variational posterior distribution and checking that the value of the bound does not decrease. This can be taken a stage further (Bishop and Svensén, 2003) by using numerical differentiation applied to the lower bound. After each update, the gradient of the bound is evaluated in the subspace corresponding to the parameters of the updated factor, to check that it is zero (within numerical tolerances). This requires that the differentiation take account of any constraints on the parameters (for instance that they be positive or that they sum to one). These checks, of course, provide necessary but not sufficient conditions for correctness. Also, they add computational cost so would typically only be employed whilst debugging the implementation.

In previous applications of variational inference, however, the evaluation of the lower bound has typically been done using separate code from that used to implement the update equations.

Although the correctness tests discussed above also provide a check on the mutual consistency of the two bodies of code, it would clearly be more elegant if their evaluation could be unified.

This is achieved naturally in the variational message passing framework by providing a way to calculate the bound automatically, as will now be described. To recap, the lower bound on the log evidence is defined to be

$$\mathcal{L}(Q) = \langle \ln P(\mathbf{H}, \mathbf{V}) \rangle - \langle \ln Q(\mathbf{H}) \rangle ,$$

where the expectations are with respect to Q. In a Bayesian network, with a factorised Q distribution, the bound becomes

$$\begin{aligned} \mathcal{L}(Q) &= \sum_i \langle \ln P(X_i \mid \mathrm{pa}_i) \rangle - \sum_{i \in \mathbf{H}} \langle \ln Q_i(H_i) \rangle \\ &\overset{\text{def}}{=} \sum_i \mathcal{L}_i \end{aligned}$$

where it has been decomposed into contributions from the individual nodes $\{\mathcal{L}_i\}$. For a particular latent variable node H_j, the contribution is

$$\mathcal{L}_j = \big\langle \ln P(H_j \mid \mathrm{pa}_j) \big\rangle - \big\langle \ln Q_j(H_j) \big\rangle .$$

Given that the model is conjugate-exponential, we can substitute in the standard form for the exponential family

$$\begin{aligned} \mathcal{L}_j &= \langle \phi_j(\mathrm{pa}_j)^{\mathrm{T}} \rangle \langle \mathbf{u}_j(H_j) \rangle + \langle f_j(H_j) \rangle + \langle g_j(\mathrm{pa}_j) \rangle \\ &\quad - \Big[\phi_j^{*\mathrm{T}} \langle \mathbf{u}_j(H_j) \rangle + \langle f_j(H_j) \rangle + \tilde{g}_j(\phi_j^*) \Big] , \end{aligned}$$

where the function $\tilde{g}_j$ is a reparameterisation of g_j so as to make it a function of the natural parameter vector rather than the parent variables. This expression simplifies to

$$\mathcal{L}_j = (\langle \phi_j(\mathrm{pa}_j) \rangle - \phi_j^*)^{\mathrm{T}} \langle \mathbf{u}_j(H_j) \rangle + \langle g_j(\mathrm{pa}_j) \rangle - \tilde{g}_j(\phi_j^*). \tag{23}$$

Three of these terms are already calculated during the variational message passing algorithm: $\langle \phi_j(\mathrm{pa}_j) \rangle$ and ϕ_j^* when finding the posterior distribution over H_j in (20), and $\langle \mathbf{u}_j(H_j) \rangle$ when calculating outgoing messages from H_j. Thus, considerable saving in computation are made compared to when the bound is calculated separately.

Each observed variable V_k also makes a contribution to the bound

$$\begin{aligned} \mathcal{L}_k &= \langle \ln P(V_k \mid \mathrm{pa}_k) \rangle \\ &= \langle \phi_k(\mathrm{pa}_k) \rangle^{\mathrm{T}} \mathbf{u}_k(V_k) + f_k(V_k) + \tilde{g}_k(\langle \phi_k(\mathrm{pa}_k) \rangle) . \end{aligned}$$

Again, computation can be saved by computing $\mathbf{u}_k(V_k)$ during the initialisation of the message passing algorithm.

Example 1 ***Calculation of the Bound for the Univariate Gaussian Model***
In the univariate Gaussian model, the bound contribution from each observed node x_n is

$$\mathcal{L}_{x_n} = \begin{bmatrix} \langle \gamma \rangle \langle \mu \rangle \\ -\langle \gamma \rangle / 2 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} x_n \\ x_n^2 \end{bmatrix} + \frac{1}{2} \left(\langle \ln \gamma \rangle - \langle \gamma \rangle \langle \mu^2 \rangle - \ln 2\pi \right)$$

and the contributions from the parameter nodes μ and γ are

$$\begin{aligned}
\mathcal{L}_\mu &= \begin{bmatrix} \beta m - \beta' m' \\ -\beta/2 + \beta'/2 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \langle \mu \rangle \\ \langle \mu^2 \rangle \end{bmatrix} + \frac{1}{2}\left(\ln\beta - \beta m^2 - \ln\beta' + \beta' m'^2\right) \\
\mathcal{L}_\gamma &= \begin{bmatrix} -b + b' \\ a - a' \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \langle \gamma \rangle \\ \langle \ln\gamma \rangle \end{bmatrix} + a\ln b - \ln\Gamma(a) - a'\ln b' + \ln\Gamma(a').
\end{aligned}$$

The bound for this univariate Gaussian model is given by the sum of the contributions from the μ and γ nodes and all x_n nodes.

4. Allowable Models

The variational message passing algorithm can be applied to a wide class of models, which will be characterised in this section.

4.1 Conjugacy Constraints

The main constraint on the model is that each parent–child edge must satisfy the constraint of conjugacy. Conjugacy allows a Gaussian variable to have a Gaussian parent for its mean and we can extend this hierarchy to any number of levels. Each Gaussian node has a gamma parent as the distribution over its precision. Furthermore, each gamma distributed variable can have a gamma distributed scale parameter b, and again this hierarchy can be extended to multiple levels.

A discrete variable can have multiple discrete parents with a Dirichlet prior over the entries in the conditional probability table. This allows for an arbitrary graph of discrete variables. A variable with an Exponential or Poisson distribution can have a gamma prior over its scale or mean respectively, although, as these distributions do not lead to hierarchies, they may be of limited interest.

These constraints are listed in Table 1. This table can be encoded in implementations of the variational message passing algorithm and used during initialisation to check the conjugacy of the supplied model.

4.1.1 TRUNCATED DISTRIBUTIONS

The conjugacy constraint does not put any restrictions on the $f_X(X)$ term in the exponential family distribution. If we choose f_X to be a step function

$$f_X(X) = \begin{cases} 0 & : \quad X \geq 0 \\ -\infty & : \quad X < 0 \end{cases}$$

then we end up with a rectified distribution, so that $P(X \mid \theta) = 0$ for $X < 0$. The choice of such a truncated distribution will change the form of messages to parent nodes (as the g_X normalisation function will also be different) but will not change the form of messages that are passed to child nodes. However, truncation will affect how the moments of the distribution are calculated from the updated parameters, which will lead to different values of child messages. For example, the moments of a rectified Gaussian distribution are expressed in terms of the standard 'erf' function. Similarly, we can consider doubly truncated distributions which are non-zero only over some finite interval, as long as the calculation of the moments and parent messages remains tractable. One

Distribution	**1st parent**	**Conjugate dist.**	**2nd parent**	**Conjugate dist.**
Gaussian	mean μ	Gaussian	precision γ	gamma
gamma	shape a	None	scale b	gamma
discrete	probabilities $\mathbf{p}$	Dirichlet	parents $\{x_i\}$	discrete
Dirichlet	pseudo-counts $\mathbf{a}$	None		
Exponential	scale a	gamma		
Poisson	mean λ	gamma		

Table 1: Distributions for each parameter of a number of exponential family distributions if the model is to satisfy conjugacy constraints. Conjugacy also holds if the distributions are replaced by their multivariate counterparts e.g. the distribution conjugate to the precision matrix of a multivariate Gaussian is a Wishart distribution. Where “None” is specified, no standard distribution satisfies conjugacy.

potential problem with the use of a truncated distribution is that no standard distributions may exist which are conjugate for each distribution parameter.

4.2 Deterministic Functions

We can considerably enlarge the class of tractable models if variables are allowed to be defined as deterministic functions of the states of their parent variables. This is achieved by adding deterministic nodes into the graph, as have been used to similar effect in the BUGS software (see Section 5).

Consider a deterministic node X which has stochastic parents $\mathbf{Y} = \{Y_1, \ldots, Y_M\}$ and which has a stochastic child node Z. The state of X is given by a deterministic function f of the state of its parents, so that $X = f(\mathbf{Y})$. If X were stochastic, the conjugacy constraint with Z would require that $P(X \mid \mathbf{Y})$ must have the same functional form, with respect to X, as $P(Z \mid X)$. This in turn would dictate the form of the natural statistic vector $\mathbf{u}_X$ of X, whose expectation $\langle \mathbf{u}_X(X) \rangle_Q$ would be the message from X to Z.

Returning to the case where X is deterministic, it is still necessary to provide a message to Z of the form $\langle \mathbf{u}_X(X) \rangle_Q$ where the function $\mathbf{u}_X$ is dictated by the conjugacy constraint. This message can be evaluated only if it can be expressed as a function of the messages from the parent variables, which are the expectations of their natural statistics functions $\{\langle \mathbf{u}_{Y_i}(Y_i) \rangle_Q\}$. In other words, there must exist a vector function ψ_X such that

$$\langle \mathbf{u}_X(f(\mathbf{Y})) \rangle_Q = \psi_X(\langle \mathbf{u}_{Y_1}(Y_1) \rangle_Q, \ldots, \langle \mathbf{u}_{Y_M}(Y_M) \rangle_Q).$$

As was discussed in Section 3.2, this constrains $\mathbf{u}_X(f(\mathbf{Y}))$ to be a multi-linear function of the set of functions $\{\mathbf{u}_{Y_i}(Y_i)\}$.

A deterministic node can be viewed as a having a conditional distribution which is a delta function, so that $P(X \mid \mathbf{Y}) = \delta(X - f(\mathbf{Y}))$. If X is discrete, this is the distribution that assigns probability one to the state $X = f(\mathbf{Y})$ and zero to all other states. If X is continuous, this is the distribution with the property that $\int g(X)\, \delta(X - f(\mathbf{Y}))\, dX = g(f(\mathbf{Y}))$. The contribution to the lower bound from a deterministic node is zero.

Example 2 ***Using a Deterministic Function as the Mean of a Gaussian***
Consider a model where a deterministic node X is to be used as the mean of a child Gaussian distribution $\mathcal{N}(Z\,|\,X,\beta^{-1})$ and where X equals a function f of Gaussian-distributed variables $Y_1,\ldots,Y_M$. The natural statistic vectors of X (as dictated by conjugacy with Z) and those of $Y_1,\ldots,Y_M$ are

$$\mathbf{u}_X(X) = \begin{bmatrix} X \\ X^2 \end{bmatrix}, \quad \mathbf{u}_{Y_i}(Y_i) = \begin{bmatrix} Y_i \\ Y_i^2 \end{bmatrix} \text{ for } i = 1\ldots M$$

The constraint on f is that $\mathbf{u}_X(f)$ must be multi-linear in $\{\mathbf{u}_{Y_i}(Y_i)\}$ and so both f and f^2 must be multi-linear in $\{Y_i\}$ and $\{Y_i^2\}$. Hence, f can be any multi-linear function of $Y_1,\ldots,Y_M$. In other words, the mean of a Gaussian can be the sum of products of other Gaussian-distributed variables.

Example 3 ***Using a Deterministic Function as the Precision of a Gaussian***
As another example, consider a model where X is to be used as the precision of a child Gaussian distribution $\mathcal{N}(Z\,|\,\mu,X^{-1})$ and where X is a function f of gamma-distributed variables $Y_1,\ldots,Y_M$. The natural statistic vectors of X and $Y_1,\ldots,Y_M$ are

$$\mathbf{u}_X(X) = \begin{bmatrix} X \\ \ln X \end{bmatrix}, \quad \mathbf{u}_{Y_i}(Y_i) = \begin{bmatrix} Y_i \\ \ln Y_i \end{bmatrix} \text{ for } i = 1\ldots M.$$

and so both f and $\ln f$ must be multi-linear in $\{Y_i\}$ and $\{\ln Y_i\}$. This restricts f to be proportional to a product of the variables $Y_1,\ldots,Y_M$ as the logarithm of a product can be found in terms of the logarithms of terms in that product. Hence $f = c\prod_i Y_i$ where c is a constant. A function containing a summation, such as $f = \sum_i Y_i$, would not be valid as the logarithm of the sum cannot be expressed as a multi-linear function of Y_i and $\ln Y_i$.

4.2.1 VALIDATING CHAINS OF DETERMINISTIC FUNCTIONS

The validity of a deterministic function for a node X is dependent on the form of the stochastic nodes it is connected to, as these dictate the functions $\mathbf{u}_X$ and $\{\mathbf{u}_{Y_i}(Y_i)\}$. For example, if the function was a summation $f = \sum_i Y_i$, it would be valid for the first of the above examples but not for the second. In addition, it is possible for deterministic functions to be chained together to form more complicated expressions. For example, the expression $X = Y_1 + Y_2Y_3$ can be achieved by having a deterministic product node A with parents Y_2 and Y_3 and a deterministic sum node X with parents Y_1 and A. In this case, the form of the function $\mathbf{u}_A$ is not determined directly by its immediate neighbours, but instead is constrained by the requirement of consistency for the connected deterministic subgraph.

In a software implementation of variational message passing, the validity of a particular deterministic structure can most easily be checked by requiring that the function $\mathbf{u}_{X_i}$ be specified explicitly for each deterministic node X_i, thereby allowing the existing mechanism for checking conjugacy to be applied uniformly across both stochastic and deterministic nodes.

4.2.2 DETERMINISTIC NODE MESSAGES

To examine message passing for deterministic nodes, we must consider the general case where the deterministic node X has multiple children $\{Z_j\}$. The message from the node X to any child Z_j is simply

$$\begin{aligned}\mathbf{m}_{X\to Z_j} &= \langle \mathbf{u}_X(f(\mathbf{Y}))\rangle_Q \\ &= \psi_X(\mathbf{m}_{Y_1\to X},\ldots,\mathbf{m}_{Y_M\to X}).\end{aligned}$$

For a particular parent Y_k, the function $\mathbf{u}_X(f(\mathbf{Y}))$ is linear with respect to $\mathbf{u}_{Y_k}(Y_k)$ and so it can be written as

$$\mathbf{u}_X(f(\mathbf{Y})) = \Psi_{X,Y_k}(\{\mathbf{u}_{Y_i}(Y_i)\}_{i\neq k}) \,.\, \mathbf{u}_{Y_k}(Y_k) + \lambda(\{\mathbf{u}_{Y_i}(Y_i)\}_{i\neq k})$$

where Ψ_{X,Y_k} is a matrix function of the natural statistics vectors of the co-parents of Y_k. The message from a deterministic node to a parent Y_k is then

$$\mathbf{m}_{X\to Y_k} = \left[\sum_j \mathbf{m}_{Z_j\to X}\right] \Psi_{X,Y_k}(\{\mathbf{m}_{Y_i\to X}\}_{i\neq k})$$

which relies on having received messages from all the child nodes and from all the co-parents. The sum of child messages can be computed and stored locally at the node and used to evaluate all child-to-parent messages. In this sense, it can be viewed as the natural parameter vector of a distribution which acts as a kind of pseudo-posterior over the value of X.

4.3 Mixture Distributions

So far, only distributions from the exponential family have been considered. Often it is desirable to use richer distributions that better capture the structure of the system that generated the data. Mixture distributions, such as mixtures of Gaussians, provide one common way of creating richer probability densities. A mixture distribution over a variable X is a weighted sum of a number of component distributions

$$P(X \mid \{\pi_k\}, \{\theta_k\}) = \sum_{k=1}^{K} \pi_k P_k(X \mid \theta_k)$$

where each P_k is a component distribution with parameters θ_k and a corresponding mixing coefficient π_k indicating the weight of the distribution in the weighted sum. The K mixing coefficients must be non-negative and sum to one.

A mixture distribution is not in the exponential family and therefore cannot be used directly as a conditional distribution within a conjugate-exponential model. Instead, we can introduce an additional discrete latent variable λ which indicates from which component distribution each data point was drawn, and write the distribution as

$$P(X \mid \lambda, \{\theta_k\}) = \prod_{k=1}^{K} P_k(X \mid \theta_k)^{\delta_{\lambda k}}.$$

Conditioned on this new variable, the distribution is now in the exponential family provided that all of the component distributions are also in the exponential family. In this case, the log conditional probability of X given all the parents (including λ) can be written as

$$\ln P(X \mid \lambda, \{\theta_k\}) = \sum_k \delta(\lambda, k) \left[\phi_k(\theta_k)^{\mathrm{T}} \mathbf{u}_k(X) + f_k(X) + g_k(\theta_k)\right].$$

If X has a child Z, then conjugacy will require that all the component distributions have the same natural statistic vector, which we can then call $\mathbf{u}_X$ so: $\mathbf{u}_1(X) = \mathbf{u}_2(X) = \ldots = \mathbf{u}_K(X) \overset{\text{def}}{=} \mathbf{u}_X(X)$. In addition, we may choose to specify, as part of the model, that all these distributions have exactly the same form (that is, $f_1 = f_2 = \ldots = f_K \overset{\text{def}}{=} f_X$), although this is not required by conjugacy. In this case, where all the distributions are the same, the log conditional becomes

$$\begin{aligned}
\ln P(X \mid \lambda, \{\theta_k\}) &= \left[\sum_k \delta(\lambda,k)\phi_k(\theta_k)\right]^{\mathrm{T}} \mathbf{u}_X(X) + f_X(X) \\
&\quad + \sum_k \delta(\lambda,k) g_k(\theta_k) \\
&= \phi_X(\lambda,\{\theta_k\})^T \mathbf{u}_X(X) + f_X(X) + \widetilde{g}_X(\phi_X(\lambda,\{\theta_k\}))
\end{aligned}$$

where we have defined $\phi_X = \sum_k \delta(\lambda,k)\phi_k(\theta_k)$ to be the natural parameter vector of this mixture distribution and the function $\widetilde{g}_X$ is a reparameterisation of g_X to make it a function of ϕ_X (as in Section 3.6). The conditional is therefore in the same exponential family form as each of the components.

We can now apply variational message passing. The message from the node X to any child is $\langle \mathbf{u}_X(X) \rangle$ as calculated from the mixture parameter vector $\phi_X(\lambda,\{\theta_k\})$. Similarly, the message from X to a parent θ_k is the message that would be sent by the corresponding component if it were not in a mixture, scaled by the variational posterior over the indicator variable $Q(\lambda = k)$. Finally, the message from X to λ is the vector of size K whose kth element is $\langle \ln P_k(X \mid \theta_k) \rangle$.

4.4 Multivariate Distributions

Until now, only scalar variables have been considered. It is also possible to handle vector variables in this framework (or to handle scalar variables which have been grouped into a vector to capture posterior dependencies between the variables). In each case, a multivariate conditional distribution is defined in the overall joint distribution P and the corresponding factor in the variational posterior Q will also be multivariate, rather than factorised with respect to the elements of the vector. To understand how multivariate distributions are handled, consider the d-dimensional Gaussian distribution with mean μ and precision matrix[5] Λ:

$$P(\mathbf{x} \mid \mu, \Lambda^{-1}) = \sqrt{\frac{|\Lambda|}{(2\pi)^d}} \exp\left(-\tfrac{1}{2}(\mathbf{x}-\mu)^{\mathrm{T}} \Lambda\, (\mathbf{x}-\mu)\right).$$

This distribution can be written in exponential family form

$$\ln \mathcal{N}(\mathbf{x} \mid \mu, \Lambda^{-1}) = \begin{bmatrix} \Lambda\mu \\ -\frac{1}{2}\mathrm{vec}(\Lambda) \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \mathbf{x} \\ \mathrm{vec}(\mathbf{x}\mathbf{x}^{\mathrm{T}}) \end{bmatrix} + \tfrac{1}{2}(\ln|\Lambda| - \mu^{\mathrm{T}}\Lambda\mu - d\ln 2\pi)$$

where $\mathrm{vec}(\cdot)$ is a function that re-arranges the elements of a matrix into a column vector in some consistent fashion, such as by concatenating the columns of the matrix. The natural statistic function for a multivariate distribution therefore depends on both the type of the distribution and its dimensionality d. As a result, the conjugacy constraint between a parent node and a child node will also constrain the dimensionality of the corresponding vector-valued variables to be the same. Multivariate conditional distributions can therefore be handled by VMP like any other exponential family distribution, which extends the class of allowed distributions to include multivariate Gaussian and Wishart distributions.

5. The precision matrix of a multivariate Gaussian is the inverse of its covariance matrix.

A group of scalar variables can act as a single parent of a vector-valued node. This is achieved using a deterministic *concatenation* function which simply concatenates a number of scalar values into a vector. In order for this to be a valid function, the scalar distributions must still be conjugate to the multivariate distribution. For example, a set of d univariate Gaussian distributed variables can be concatenated to act as the mean of a d-dimensional multivariate Gaussian distribution.

4.4.1 NORMAL-GAMMA DISTRIBUTION

The mean μ and precision γ parameters of a Gaussian distribution can be grouped together into a single bivariate variable $\mathbf{c} = \{\mu, \gamma\}$. The conjugate distribution for this variable is the normal-gamma distribution, which is written

$$\ln P(\mathbf{c} \mid m, \lambda, a, b) = \begin{bmatrix} m\lambda \\ -\frac{1}{2}\lambda \\ -b - \frac{1}{2}\lambda m^2 \\ a - \frac{1}{2} \end{bmatrix}^{T} \begin{bmatrix} \mu\gamma \\ \mu^2\gamma \\ \gamma \\ \ln\gamma \end{bmatrix} + \tfrac{1}{2}(\ln\lambda - \ln 2\pi) + a\ln b - \ln\Gamma(a).$$

This distribution therefore lies in the exponential family and can be used within VMP instead of separate Gaussian and gamma distributions. In general, grouping these variables together will improve the approximation and so increase the lower bound. The multivariate form of this distribution, the normal-Wishart distribution, is handled as described above.

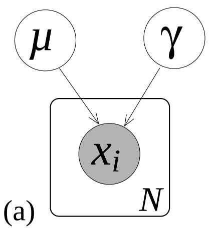
4.5 Summary of Allowable Models

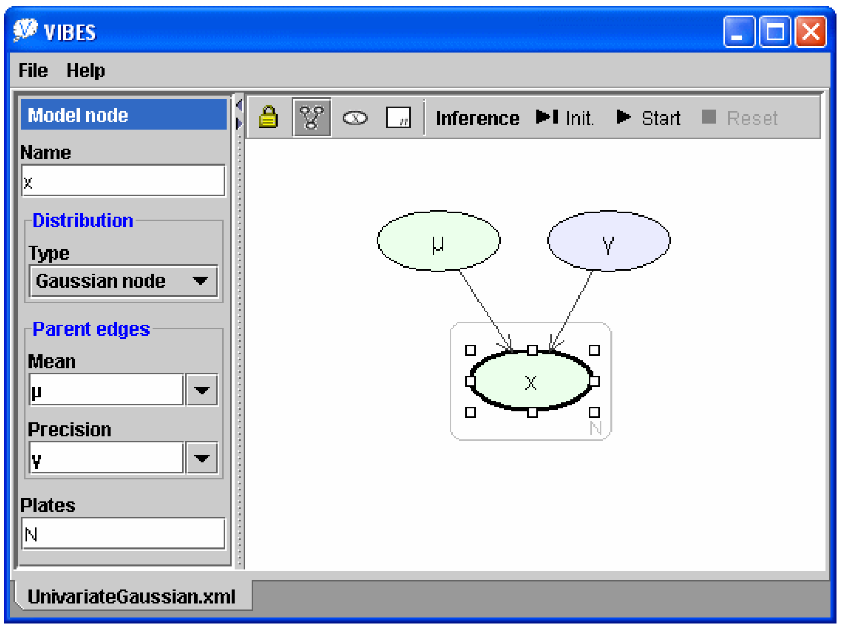
In summary, the variational message passing algorithm can handle probabilistic models with the following very general architecture: arbitrary directed acyclic subgraphs of multinomial discrete variables (each having Dirichlet priors) together with arbitrary subgraphs of univariate and multivariate linear Gaussian nodes (having gamma and Wishart priors), with arbitrary mixture nodes providing connections from the discrete to the continuous subgraphs. In addition, deterministic nodes can be included to allow parameters of child distributions to be deterministic functions of parent variables. Finally, any of the continuous distributions can be singly or doubly truncated to restrict the range of allowable values, provided that the appropriate moments under the truncated distribution can be calculated along with any necessary parent messages.

This architecture includes as special cases models such as hidden Markov models, Kalman filters, factor analysers, principal component analysers and independent component analysers, as well as mixtures and hierarchical mixtures of these.

5. VIBES: An Implementation of Variational Message Passing

The variational message passing algorithm has been implemented in a software package called VIBES (Variational Inference in BayEsian networkS), first described by Bishop et al. (2002). Inspired by WinBUGS (a graphical user interface for BUGS by Lunn et al., 2000), VIBES allows for models to be specified graphically, simply by constructing the Bayesian network for the model. This involves drawing the graph for the network (using operations similar to those in a drawing package) and then assigning properties to each node such as its name, the functional form of the conditional distribution, its dimensionality and its parents. As an example, Figure 5 shows the Bayesian network for the univariate Gaussian model along with a screenshot of the same model in

(a)


(b)


Model node

Name
x

Distribution
Type
Gaussian node

Parent edges
Mean
μ
Precision
γ

Plates
N

Figure 5: (a) Bayesian network for the univariate Gaussian model. (b) Screenshot of VIBES showing how the same model appears as it is being edited. The node x is selected and the panel to the left shows that it has a Gaussian conditional distribution with mean μ and precision γ. The plate surrounding x shows that it is duplicated N times and the heavy border indicates that it is observed (according to the currently attached data file).

VIBES. Models can also be specified in a text file, which contains XML according to a pre-defined model definition schema. VIBES is written in Java and so can be used on Windows, Linux or any operating system with a Java 1.3 virtual machine.

As in WinBUGS, the convention of making deterministic nodes explicit in the graphical representation has been adopted, as this greatly simplifies the specification and interpretation of the model. VIBES also uses the plate notation of a box surrounding one or more nodes to denote that those nodes are replicated some number of times, specified by the parameter in the bottom right hand corner of the box.

Once the model is specified, data can be attached from a separate data file which contains observed values for some of the nodes, along with sizes for some or all of the plates. The model can then be *initialised* which involves: (i) checking that the model is valid by ensuring that conjugacy and dimensionality constraints are satisfied and that all parameters are specified; (ii) checking that the observed data is of the correct dimensionality; (iii) allocating memory for all moments and messages; (iv) initialisation of the individual distributions Q_i.

Following a successful initialisation, inference can begin immediately. As inference proceeds, the current state of the distribution Q_i for any node can be inspected using a range of diagnostics including tables of values and Hinton diagrams. If desired, the lower bound $\mathcal{L}(Q)$ can be monitored (at the expense of slightly increased computation), in which case the optimisation can be set to terminate automatically when the change in the bound during one iteration drops below a small value. Alternatively, the optimisation can be stopped after a fixed number of iterations.

The VIBES software can be downloaded from `http://vibes.sourceforge.net`. This software was written by one of the authors (John Winn) whilst a Ph.D. student at the University of Cambridge and is free and open source. Appendix A contains a tutorial for applying VIBES to an example problem involving a Gaussian Mixture model. The VIBES web site also contains an online version of this tutorial.

6. Extensions to Variational Message Passing

In this section, three extensions to the variational message passing algorithm will be described. These extensions are intended to illustrate how the algorithm can be modified to perform alternative inference calculations and to show how the conjugate-exponential constraint can be overcome in certain circumstances.

6.1 Further Variational Approximations: The Logistic Sigmoid Function

As it stands, the VMP algorithm requires that the model be conjugate-exponential. However, it is possible to sidestep the conjugacy requirement by introducing additional variational parameters and approximating non-conjugate conditional distributions by valid conjugate ones. We will now illustrate how this can be achieved using the example of a conditional distribution over a binary variable $x \in 0,1$ of the form

$$\begin{aligned} P(x \mid a) &= \sigma(a)^x [1 - \sigma(a)]^{1-x} \\ &= e^{ax} \sigma(-a) \end{aligned}$$

where

$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$

is the logistic sigmoid function.

We take the approach of Jaakkola and Jordan (1996) and use a variational bound for the logistic sigmoid function defined as

$$\sigma(a) \geqslant F(a,\xi) \overset{\text{def}}{=} \sigma(\xi) \exp[(a-\xi)/2 + \lambda(\xi)(a^2 - \xi^2)]$$

where $\lambda(\xi) = [1/2 - g(\xi)]/2\xi$ and ξ is a variational parameter. For any given value of a we can make this bound exact by setting $\xi^2 = a^2$. The bound is illustrated in Figure 6 in which the solid curve shows the logistic sigmoid function $\sigma(a)$ and the dashed curve shows the lower bound $F(a,\xi)$ for $\xi = 2$.

We use this result to define a new lower bound $\widetilde{\mathcal{L}} \leqslant \mathcal{L}$ by replacing each expectation of the form $\langle \ln[e^{ax}\sigma(-a)] \rangle$ with its lower bound $\langle \ln[e^{ax}F(-a,\xi)] \rangle$. The effect of this transformation is to replace the logistic sigmoid function with an exponential, therefore restoring conjugacy to the model. Optimisation of each ξ parameter is achieved by maximising this new bound $\widetilde{\mathcal{L}}$, leading to the re-estimation equation

$$\xi^2 = \langle a^2 \rangle_Q .$$

It is important to note that, as the quantity $\widetilde{\mathcal{L}}$ involves expectations of $\ln F(-a,\xi)$, it is no longer guaranteed to be exact for any value of ξ.

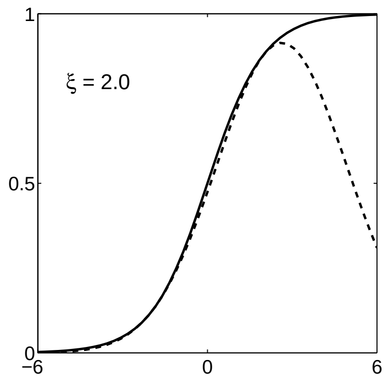


Figure 6: The logistic sigmoid function $\sigma(a)$ and variational bound $F(a,\xi)$.

It follows from (8) that the factor in Q corresponding to $P(x\,|\,a)$ is updated using

$$\begin{aligned}\ln Q_x^\star(x) &= \langle \ln(e^{ax}F(-a,\xi))\rangle_{\sim Q_x(x)} + \sum_{k\in ch_x}\langle \ln P(X_k|\mathrm{pa}_k)\rangle_{\sim Q_x(x)} + \text{const.}\\ &= \langle ax\rangle_{\sim Q_x(x)} + \sum_{k\in ch_x}\langle b_k x\rangle_{\sim Q_x(x)} + \text{const.}\\ &= a^\star x + \text{const.}\end{aligned}$$

where $a^\star = \langle a\rangle + \sum_k \langle b_k\rangle$ and the $\{b_k\}$ arise from the child terms which must be in the form $(b_k x +$ const.$)$ due to conjugacy. Therefore, the variational posterior $Q_x(x)$ takes the form

$$Q_x(x) = \sigma(a^\star)^x[1-\sigma(a^\star)]^{1-x}.$$

6.1.1 USING THE LOGISTIC APPROXIMATION WITHIN VMP

We will now explain how this additional variational approximation can be used within the VMP framework. The lower bound $\widetilde{\mathcal{L}}$ contains terms like $\langle \ln(e^{ax}F(-a,\xi))\rangle$ which need to be evaluated and so we must be able to evaluate $[\langle a\rangle \ \langle a^2\rangle]^{\mathrm{T}}$. The conjugacy constraint on a is therefore that its distribution must have a natural statistic vector $\mathbf{u}_a(a) = [a\ a^2]$. Hence it could, for example, be Gaussian.

For consistency with general discrete distributions, we write the bound on the log conditional $\ln P(x\,|\,a)$ as

$$\begin{aligned}\ln P(x\,|\,a) &\geqslant \begin{bmatrix}0\\ a\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}\delta(x-0)\\ \delta(x-1)\end{bmatrix} + (-a-\xi)/2 + \lambda(\xi)(a^2-\xi^2) + \ln\sigma(\xi)\\ &= \begin{bmatrix}\delta(x-1)-\frac{1}{2}\\ \lambda(\xi)\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}a\\ a^2\end{bmatrix} - \xi/2 - \lambda(\xi)\xi^2 + \ln\sigma(\xi).\end{aligned}$$

The message from node x to node a is therefore

$$\mathbf{m}_{x\to a} = \begin{bmatrix}\langle\delta(x-1)\rangle - \frac{1}{2}\\ \lambda(\xi)\end{bmatrix}$$

and all other messages are as in standard VMP. The update of variational factors can then be carried out as normal except that each ξ parameter must also be re-estimated during optimisation. This can be carried out, for example, just before sending a message from x to a. The only remaining modification is to the calculation of the lower bound in (23), where the term $\langle g_j(\mathrm{pa}_j)\rangle$ is replaced by the expectation of its bound,

$$\langle g_j(\mathrm{pa}_j)\rangle \geqslant (-\langle a\rangle - \xi)/2 + \lambda(\xi)(\langle a^2\rangle - \xi^2) + \ln\sigma(\xi).$$

This extension to VMP enables discrete nodes to have continuous parents, further enlarging the class of allowable models. In general, the introduction of additional variational parameters enormously extends the class of models to which VMP can be applied, as the constraint that the model distributions must be conjugate no longer applies.

6.2 Finding a Maximum A Posteriori Solution

The advantage of using a variational distribution is that it provides a posterior distribution over latent variables. It is, however, also possible to use VMP to find a Maximum A Posteriori (MAP) solution, in which values of each latent variable are found that maximise the posterior probability. Consider choosing a variational distribution which is a delta function

$$Q^{\mathrm{MAP}}(\mathbf{H}) = \delta(\mathbf{H} - \mathbf{H}^\star)$$

where $H^\star$ is the MAP solution. From (3), the lower bound is

$$\begin{aligned}\mathcal{L}(Q) &= \langle \ln P(\mathbf{H},\mathbf{V})\rangle - \langle \ln Q(\mathbf{H})\rangle \\ &= \ln P(\mathbf{H}^\star,\mathbf{V}) + h_\delta\end{aligned}$$

where h_δ is the differential entropy of the delta function. By considering the differential entropy of a Gaussian in the limit as the variance goes to 0, we can see that $h_\delta = \log a, a \to 0$. Thus h_δ does not depend on $\mathbf{H}^\star$ and so maximising $\mathcal{L}(Q)$ is equivalent to finding the MAP solution. However, since the entropy h_δ tends to $-\infty$, so does $\mathcal{L}(Q)$ and so, whilst it is still trivially a lower bound on the log evidence, it is not an informative one. In other words, knowing the probability density of the posterior at a point is uninformative about the posterior mass.

The variational distribution can be written in factorised form as

$$Q^{\mathrm{MAP}}(\mathbf{H}) = \prod_j Q_j(H_j).$$

with $Q_j(H_j) = \delta(H_j - H_j^\star)$. The KL divergence between the approximating distribution and the true posterior is minimised if $\mathrm{KL}(Q_j \,||\, Q_j^\star)$ is minimised, where $Q_j^\star$ is the standard variational solution given by (6). Normally, Q_j is unconstrained so we can simply set it to $Q_j^\star$. However, in this case, Q_j is a delta function and so we have to find the value of $H_j^\star$ that minimises $\mathrm{KL}(\delta(H_j - H_j^\star)\,||\,Q_j^\star)$. Unsurprisingly, this is simply the value of $H_j^\star$ that maximises $Q_j^\star(H_j^\star)$.

In the message passing framework, a MAP solution can be obtained for a particular latent variable H_j directly from the updated natural statistic vector $\phi_j^\star$ using

$$(\phi_j^\star)^{\mathrm{T}} \frac{d\mathbf{u}_j(H_j)}{dH_j} = 0.$$

For example, if $Q_j^\star$ is Gaussian with mean μ then $H_j^\star = \mu$ or if $Q_j^\star$ is gamma with parameters a, b, then $H_j^\star = (a-1)/b$.

Given that the variational posterior is now a delta function, the expectation of any function $\langle f(H_j)\rangle$ under the variational posterior is just $f(H_j^\star)$. Therefore, in any outgoing messages, $\langle \mathbf{u}_j(H_j)\rangle$ is replaced by $\mathbf{u}_j(H_j^\star)$. Since all surrounding nodes can process these messages as normal, a MAP solution may be obtained for any chosen subset of variables (such as particular hyper-parameters), whilst a full posterior distribution is retained for all other variables.

6.3 Learning Non-conjugate Priors by Sampling

For some exponential family distribution parameters, there is no standard probability distribution which can act as a conjugate prior. For example, there is no standard distribution which can act as a conjugate prior for the shape parameter a of the gamma distribution. This implies that we cannot learn a posterior distribution over a gamma shape parameter within the basic VMP framework. As discussed above, we can sometimes introduce conjugate approximations by adding variational parameters, but this may not always be possible.

The purpose of the conjugacy constraint is two-fold. First, it means that the posterior distribution of each variable, conditioned on its neighbours, has the same form as the prior distribution. Hence, the updated variational distribution factor for that variable has the same form and inference involves just updating the parameters of that distribution. Second, conjugacy results in variational distributions being in standard exponential family form allowing their moments to be calculated analytically.

If we ignore the conjugacy constraint, we get non-standard posterior distributions and we must resort to using sampling or other methods to determine the moments of these distributions. The disadvantages of using sampling include computational expense, inability to calculate an analytical lower bound and the fact that inference is no longer deterministic for a given initialisation and ordering. The use of sampling methods will now be illustrated by an example showing how to sample from the posterior over the shape parameter of a gamma distribution.

Example 4 ***Learning a Gamma Shape Parameter***
Let us assume that there is a latent variable a which is to be used as the shape parameter of K gamma distributed variables $\{x_1 \dots x_K\}$. We choose a to have a non-conjugate *prior of an inverse-gamma distribution:*

$$P(a \mid \alpha, \beta) \propto a^{-\alpha-1} \exp\left(\frac{-\beta}{a}\right).$$

The form of the gamma distribution means that messages sent to the node a are with respect to a natural statistic vector

$$\mathbf{u}_a = \begin{bmatrix} a \\ \ln\Gamma(a) \end{bmatrix}$$

which means that the updated factor distribution $Q_a^\star$ has the form

$$\ln Q_a^\star(a) = \left[\sum_{i=1}^{K} \mathbf{m}_{x_i \to a}\right]^{\mathrm{T}} \begin{bmatrix} a \\ \ln\Gamma(a) \end{bmatrix} + (-\alpha - 1)\ln a - \frac{\beta}{a} + \text{const.}$$

This density is not of standard form, but it can be shown that $Q^\star(\ln a)$ is log-concave, so we can generate independent samples from the distribution for $\ln a$ using Adaptive Rejection Sampling from Gilks and Wild (1992). These samples are then transformed to get samples of a from $Q_a^\star(a)$, which *is used to estimate the expectation $\langle \mathbf{u}_a(a) \rangle$. This expectation is then sent as the outgoing message to each of the child nodes.*

Each factor distribution is normally updated during every iteration and so, in this case, a number of independent samples from $Q_a^\star$ would have to be drawn during every iteration. If this proved too computationally expensive, then the distribution need only be updated intermittently.

It is worth noting that, as in this example, BUGS also uses Adaptive Rejection Sampling for sampling when the posterior distribution is log-concave but non-conjugate, whilst also providing techniques for sampling when the posterior is not log-concave. This suggests that non-conjugate parts of a general graphical model could be handled within a BUGS-style framework whilst variational message passing is used for the rest of the model. The resulting hybrid variational/sampling framework would, to a certain extent, capture the advantages of both techniques.

7. Discussion

The variational message passing algorithm allows approximate inference using a factorised variational distribution in any conjugate-exponential model, and in a range of non-conjugate models. As a demonstration of its utility, this algorithm has already been used to solve problems in the domain of machine vision and bioinformatics (see Winn, 2003; Bishop and Winn, 2000). In general, variational message passing dramatically simplifies the construction and testing of new variational models and readily allows a range of alternative models to be tested on a given problem.

The general form of VMP also allows the inclusion of arbitrary nodes in the graphical model provided that each node is able to receive and generate appropriate messages in the required form, whether or not the model remains conjugate-exponential. The extensions to VMP concerning the logistic function and sampling illustrate this flexibility.

One limitation of the current algorithm is that it uses a variational distribution which is factorised across nodes, giving an approximate posterior which is separable with respect to individual (scalar or vector) variables. In general, an improved approximation will be achieved if a posterior distribution is used which retains some dependency structure. Whilst Wiegerinck (2000) has presented a general framework for such structured variational inference, he does not provide a general-purpose algorithm for applying this framework. Winn (2003) and Bishop and Winn (2003) have therefore proposed an extended version of variational message passing which allows for structured variational distributions. VIBES has been extended to implement a limited version of this algorithm that can only be applied to a constrained set of models. However, a complete implementation and evaluation of this extended algorithm has yet to be undertaken.

The VIBES software is free and open source and can be downloaded from the VIBES web site at `http://vibes.sourceforge.net`. The web site also contains a tutorial that provides an introduction to using VIBES.

Acknowledgments

The authors would like to thank David Spiegelhalter for his help with the VIBES project. We would also like to thank Zoubin Ghahramani, David MacKay, Matthew Beal and Michael Jordan for many helpful discussions about variational inference.

This work was carried out whilst John Winn was a Ph.D. student at the University of Cambridge, funded by a Microsoft Research studentship.

Appendix A. VIBES Tutorial

In this appendix, we demonstrate the application of VIBES to an example problem involving a Gaussian Mixture model. We then demonstrate the flexibility of VIBES by changing the model to fit the data better, using the lower bound as an estimate of the log evidence for each model. An online version of this tutorial is available at `http://vibes.sourceforge.net/tutorial`.

The data used in this tutorial is two-dimensional and consists of nine clusters in a three-by-three grid, as illustrated in Figure 7.

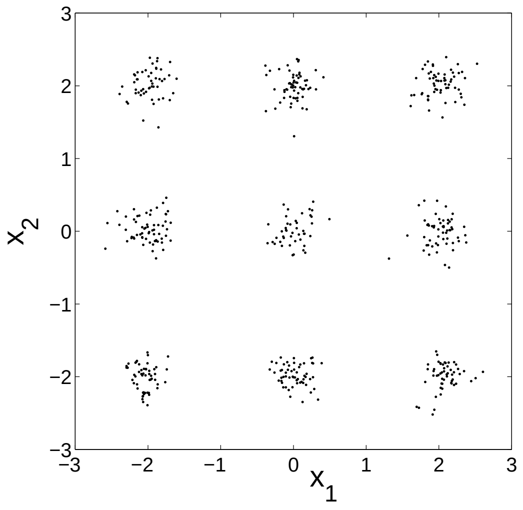


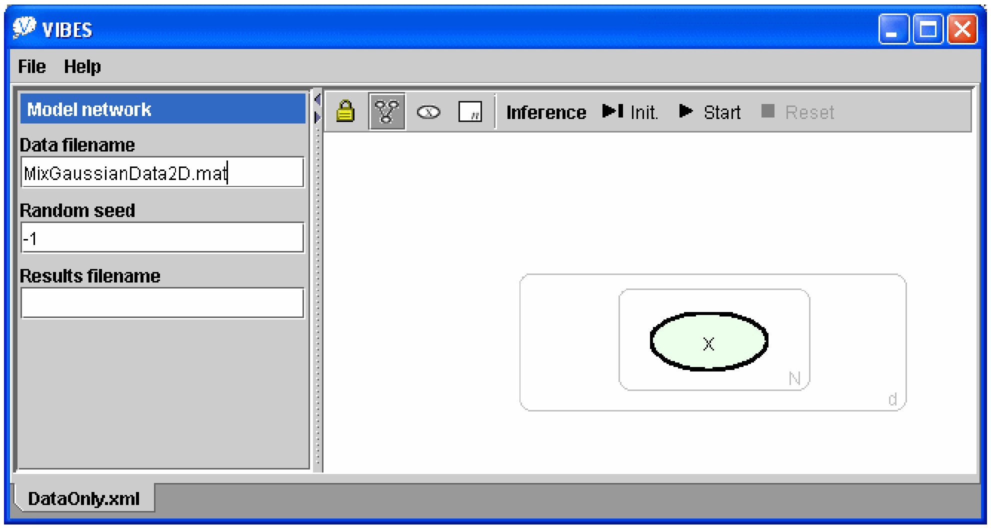
Figure 7: The two-dimensional data set used in the tutorial, which consists of nine clusters in a three-by-three grid.

A.1 Loading Matlab Data into VIBES

The first step is to load the data set into VIBES. This is achieved by creating a node with the name *x* which corresponds to a matrix *x* in a Matlab `.mat` file. As the data matrix is two dimensional, the node is placed inside two plates *N* and *d* and the data filename (in this case `MixGaussianData2D.mat`) is entered. Selecting `File`→`Load data` loads the data into the node and also sets the size of the *N* and *d* plates to 500 and 2 respectively. The node is marked as observed (shown with a bold edge) and the observed data can be inspected by double-clicking the node with the mouse. At this point, the display is as shown in Figure 8.

A.2 Creating and Learning a Gaussian Model

The node *x* has been marked as Gaussian by default and so the model is invalid as neither the mean nor the precision of the Gaussian have been set (attempting to initialise the model by pressing the `Init.` button will give an error message to this effect). We can specify latent variables for these



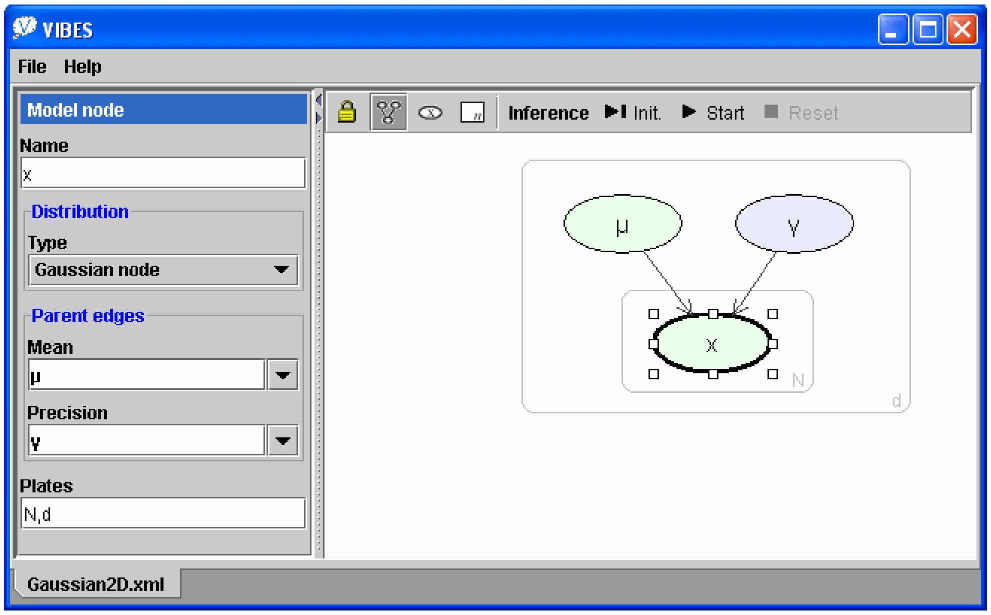
Data filename
MixGaussianData2D.mat

Random seed
-1

Results filename

Figure 8: A VIBES model with a single observed node x which has attached data.

parameters by creating a node μ for the mean parameter and a node γ for the precision parameter. These nodes are created within the d plate to give a model which is separable over each data dimension. These are then set as the `Mean` and `Precision` properties of x, as shown in Figure 9.



Name
x

Distribution
Type
Gaussian node

Parent edges
Mean
μ
Precision
γ

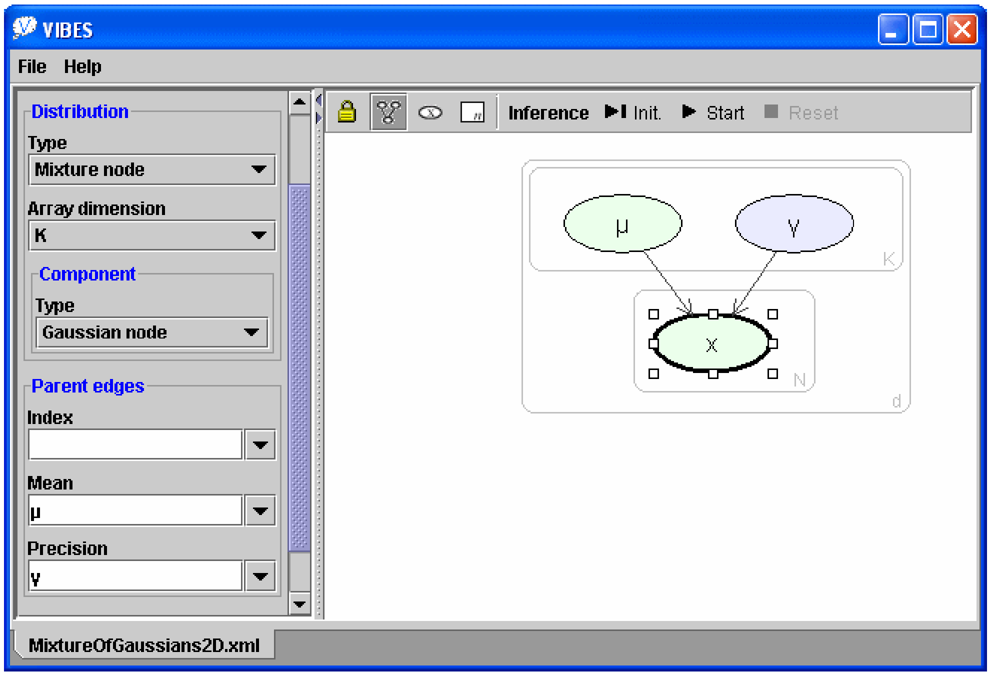
Plates
N,d

Figure 9: A two-dimensional Gaussian model, showing that the variables μ and γ are being used as the mean and precision parameters of the conditional distribution over x.

The model is still invalid as the parameters of μ and γ are unspecified. In this case, rather than create further latent variables, these parameters will be set to fixed values to give appropriate priors (for example setting μ to have mean $= 0$ and precision $= 0.3$ and γ to have $a = 10$ and $b = 1$). The network now corresponds to a two-dimensional Gaussian model and variational inference can be performed automatically by pressing the `Start` button (which also performs initialisation). For this data set, inference converges after four iterations and gives a bound of -1984 nats. At this point, the expected values of each latent variable under the fully-factorised Q distribution can be displayed or graphed by double-clicking on the corresponding node.

A.3 Extending the Gaussian model to a Gaussian Mixture Model

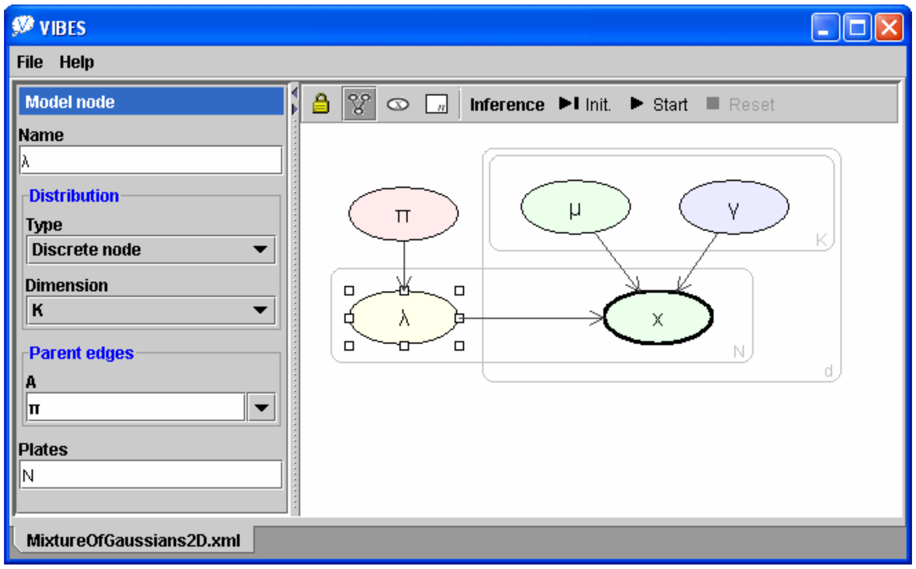
Our aim is to create a Gaussian mixture model and so we must extend our simple Gaussian model to be a mixture with K Gaussian components. As there will now be K sets of the latent variables μ and γ, these are placed in a new plate, called K, whose size is set to 20. We modify the conditional distribution for the x node to be a mixture of dimension K, with each component being Gaussian. The display is then as shown in Figure 10.



Distribution
Type: Mixture node
Array dimension: K
Component
Type: Gaussian node
Parent edges
Index:
Mean: μ
Precision: γ

Figure 10: An incomplete model which shows that x is now a mixture of K Gaussians. There are now K sets of parameters and so μ and γ have been placed in a plate K. The model is incomplete as the `Index` parent of x has not been specified.

The model is currently incomplete as making x a mixture requires a new discrete `Index` parent to indicate which component distribution each data point was drawn from. We must therefore create a new node λ, sitting in the N plate, to represent this new discrete latent variable. We also create a node π with a Dirichlet distribution which provides a prior over λ. The completed mixture model is shown in Figure 11.


VIBES

File Help

Model node

Name
λ

Distribution
Type
Discrete node
Dimension
K

Parent edges
A
π

Plates
N

Inference ▶I Init. ▶ Start ■ Reset

MixtureOfGaussians2D.xml

Figure 11: The completed Gaussian mixture model showing the discrete indicator node λ.

A.4 Inference Using the Gaussian Mixture Model

With the model complete, inference can once again proceed automatically by pressing the `Start` button. A Hinton diagram of the expected value of π can be displayed by double-clicking on the π node, giving the result shown in Figure 12. As can be seen, nine of the twenty components have been retained.

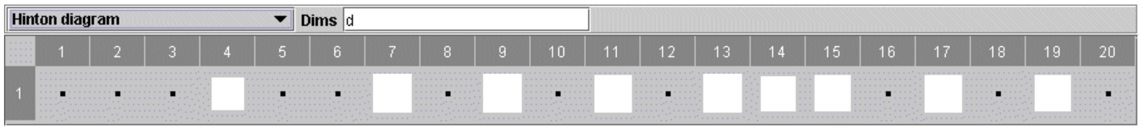

Hinton diagram Dims d

Figure 12: A Hinton diagram showing the expected value of π for each mixture component. The learned mixture consists of only nine components.

The means of the retained components can be inspected by double-clicking on the μ node, giving the Hinton diagram of Figure 13. These learned means correspond to the centres of each of the data clusters.

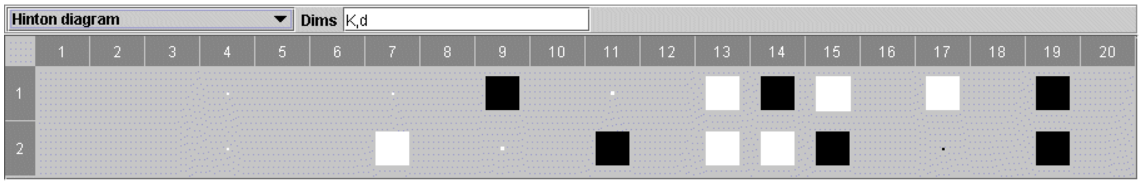

Hinton diagram Dims K,d

Figure 13: A Hinton diagram whose columns give the expected two-dimensional value of the mean μ for each mixture component. The mean of each of the eleven unused components is just the expected value under the prior which is $(0,0)$. Column 4 corresponds to a retained component whose mean is roughly $(0,0)$.

A graph of the evolution of the bound can be displayed by clicking on the bound value and is shown in Figure 14. The converged lower bound of this new model is -1019 nats, which is significantly higher than that of the single Gaussian model, showing that there is much greater evidence for this model. This is unsurprising since a mixture of 20 Gaussians has significantly more parameters than a single Gaussian and hence can give a much closer fit to the data. Note, however, that the model automatically chooses only to exploit 9 of these components, with the remainder being suppressed (by virtue of their mixing coefficients going to zero). This provides an elegant example of automatic model complexity selection within a Bayesian setting.

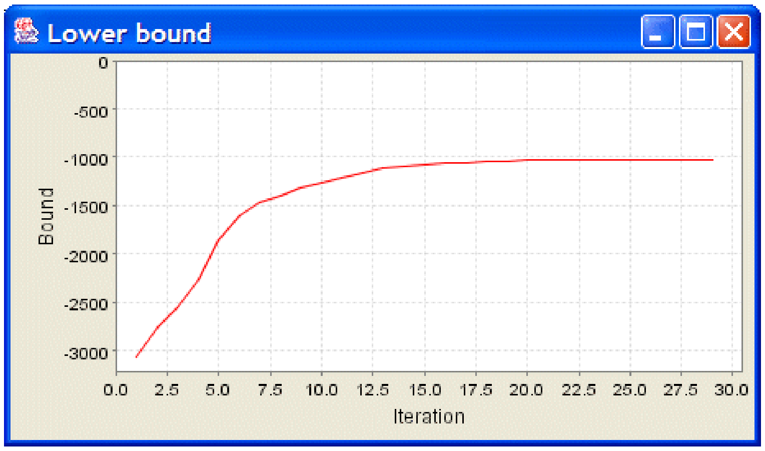


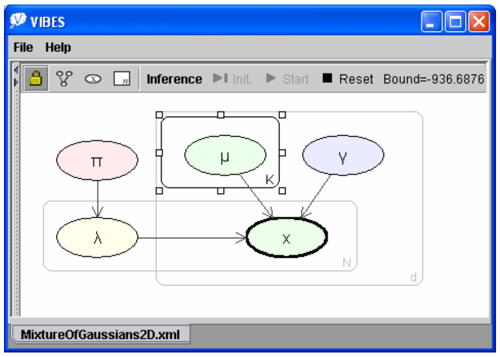
Figure 14: A graph of the evolution of the lower bound during inference.

A.5 Modifying the Mixture Model

The rapidity with which models can be constructed using VIBES allows new models to be quickly developed and compared. For example, we can take our existing mixture of Gaussians model and modify it to try and find a more probable model.

First, we may hypothesise that each of the clusters has similar size and so they may be modelled by a mixture of Gaussian components having a common variance in each dimension. Graphically, this corresponds to shrinking the K plate so that it no longer contains the γ node, as shown in Figure 15a. The converged lower bound for this new model is -937 nats showing that this modified model is better at explaining this data set than the standard mixture of Gaussians model. Note that the increase in model probability does not arise from an improved fit to the data, since this model and the previous one both contain 20 Gaussian components and in both cases 9 of these components contribute to the data fit. Rather, the constrained model having a single variance parameter can achieve almost as good a data fit as the unconstrained model yet with far fewer parameters. Since a Bayesian approach automatically penalises complexity, the simpler (constrained) model has the higher probability as indicated by the higher value for the variational lower bound.

We may further hypothesise that the data set is separable with respect to its two dimensions (i.e. the two dimensions are independent). Graphically this consists of moving all nodes inside the d plate (so we effectively have two copies of a one-dimensional mixture of Gaussians model with common variance). A VIBES screenshot of this further modification is shown in Figure 15b.

(a) 

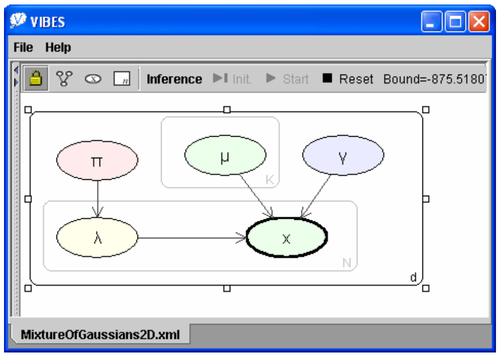
(b) 

Figure 15: (a) Mixture of Gaussians model with shared precision parameter γ (the γ node is no longer inside the K plate). (b) Model with independent data dimensions, each a univariate Gaussian mixture with common variance.

Performing variational inference on this separable model leads to each one-dimensional mixture having three retained mixture components and gives an improved bound of -876 nats.

We will consider one final model. In this model both the π and the γ nodes are common to both data dimensions, as shown in Figure 16. This change corresponds to the assumption that the mixture coefficients are the same for each of the two mixtures and that the component variances are the same for all components in both mixtures. Inference leads to a final improved bound of -856 nats. Whilst this tutorial has been on a toy data set, the principles of model construction, modification and comparison can be applied just as readily to real data sets.

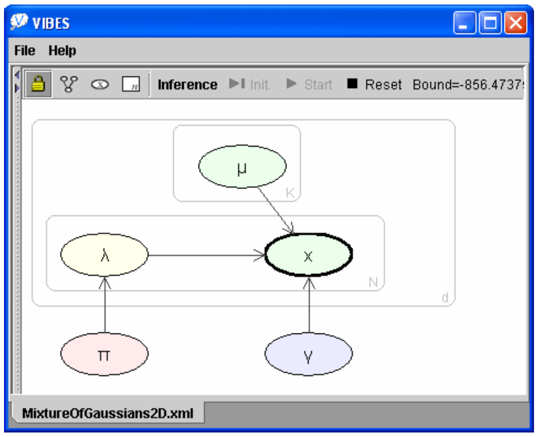


Figure 16: Further modified mixture model where the π and γ nodes are now common to all data dimensions.

References

H. Attias. A variational Bayesian framework for graphical models. In S. Solla, T. K. Leen, and K-L Muller, editors, *Advances in Neural Information Processing Systems*, volume 12, pages 209–215, Cambridge MA, 2000. MIT Press.

C. M. Bishop. Variational principal components. In *Proceedings Ninth International Conference on Artificial Neural Networks, ICANN'99*, volume 1, pages 509–514. IEE, 1999.

C. M. Bishop and M. Svensén. Bayesian Hierarchical Mixtures of Experts. In U. Kjaerulff and C. Meek, editors, *Proceedings Nineteenth Conference on Uncertainty in Artificial Intelligence*, pages 57–64. Morgan Kaufmann, 2003.

C. M. Bishop and J. M. Winn. Non-linear Bayesian image modelling. In *Proceedings Sixth European Conference on Computer Vision*, volume 1, pages 3–17. Springer-Verlag, 2000.

C. M. Bishop and J. M. Winn. Structured variational distributions in VIBES. In *Proceedings Artificial Intelligence and Statistics*, Key West, Florida, 2003. Society for Artificial Intelligence and Statistics.

C. M. Bishop, J. M. Winn, and D. Spiegelhalter. VIBES: A variational inference engine for Bayesian networks. In *Advances in Neural Information Processing Systems*, volume 15, 2002.

R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Statistics for Engineering and Information Science. Springer-Verlag, 1999.

Z. Ghahramani and M. J. Beal. Propagation algorithms for variational Bayesian learning. In T. K. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13, Cambridge MA, 2001. MIT Press.

W. R. Gilks and P. Wild. Adaptive rejection sampling for Gibbs sampling. *Applied Statistics*, 41(2): 337–348, 1992.

T. Jaakkola and M. Jordan. A variational approach to Bayesian logistic regression problems and their extensions. In *In Proceedings of the 6th international workshop on artificial intelligence and statistics.*, 1996.

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 105–162. Kluwer, 1998.

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.

D. J. Lunn, A. Thomas, N. G. Best, and D. J. Spiegelhalter. WinBUGS – a Bayesian modelling framework: concepts, structure and extensibility. *Statistics and Computing*, 10:321–333, 2000. http://www.mrc-bsu.cam.ac.uk/bugs/.

T. P. Minka. Expectation propagation for approximate Bayesian inference. In *Proceedings of the 17th Annual Conference on Uncertainty in Artificial Intelligence*, pages 362–369. Morgan Kauffmann, 2001.

R. M. Neal and G. E. Hinton. A new view of the EM algorithm that justifies incremental and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. Kluwer, 1998.

J. Pearl. Fusion, propagation and structuring in belief networks. *Artificial Intelligence*, 29:241–288, 1986.

A. Thomas, D. J. Spiegelhalter, and W. R. Gilks. BUGS: A program to perform Bayesian inference using Gibbs sampling. In J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, editors, *Bayesian Statistics*, Oxford: Clarendon Press, 1992.

W. Wiegerinck. Variational approximations between mean field theory and the junction tree algorithm. In *Uncertainty in Artificial Intelligence*. Morgan Kauffmann, 2000.

J. M. Winn. *Variational Message Passing and its Applications*. PhD thesis, University of Cambridge, October 2003.

Journal of Machine Learning Research 6 (2005) 695–709 Submitted 11/04; Revised 3/05; Published 4/05

Estimation of Non-Normalized Statistical Models by Score Matching

Aapo Hyvärinen AAPO.HYVARINEN@HELSINKI.FI
Helsinki Institute for Information Technology (BRU)
Department of Computer Science
FIN-00014 University of Helsinki, Finland

Editor: Peter Dayan

Abstract

One often wants to estimate statistical models where the probability density function is known only up to a multiplicative normalization constant. Typically, one then has to resort to Markov Chain Monte Carlo methods, or approximations of the normalization constant. Here, we propose that such models can be estimated by minimizing the expected squared distance between the gradient of the log-density given by the model and the gradient of the log-density of the observed data. While the estimation of the gradient of log-density function is, in principle, a very difficult non-parametric problem, we prove a surprising result that gives a simple formula for this objective function. The density function of the observed data does not appear in this formula, which simplifies to a sample average of a sum of some derivatives of the log-density given by the model. The validity of the method is demonstrated on multivariate Gaussian and independent component analysis models, and by estimating an overcomplete filter set for natural image data.

Keywords: statistical estimation, non-normalized densities, pseudo-likelihood, Markov chain Monte Carlo, contrastive divergence

1. Introduction

In many cases, probabilistic models in machine learning, statistics, or signal processing are given in the form of non-normalized probability densities. That is, the model contains an unknown normalization constant whose computation is too difficult for practical purposes.

Assume we observe a random vector $\mathbf{x} \in \mathbb{R}^n$ which has a probability density function (pdf) denoted by $p_{\mathbf{x}}(.)$. We have a parametrized density model $p(.;\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is an m-dimensional vector of parameters. We want to estimate the parameter $\boldsymbol{\theta}$ from $\mathbf{x}$, i.e. we want to approximate $p_{\mathbf{x}}(.)$ by $p(.;\hat{\boldsymbol{\theta}})$ for the estimated parameter value $\hat{\boldsymbol{\theta}}$. (We shall here consider the case of continuous-valued variables only.)

The problem we consider here is that we only are able to compute the pdf given by the model up to a multiplicative constant $Z(\boldsymbol{\theta})$:

$$p(\boldsymbol{\xi};\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} q(\boldsymbol{\xi};\boldsymbol{\theta}).$$

That is, we do know the functional form of q as an analytical expression (or any form that can be easily computed), but we do *not* know how to easily compute Z which is given by an integral that is often analytically intractable:

$$Z(\boldsymbol{\theta}) = \int_{\boldsymbol{\xi} \in \mathbb{R}^n} q(\boldsymbol{\xi}; \boldsymbol{\theta}) \, d\boldsymbol{\xi}.$$

In higher dimensions (in fact, for almost any $n > 2$), the numerical computation of this integral is practically impossible as well.

Usually, estimation of non-normalized models is approached by Markov Chain Monte Carlo (MCMC) methods, which are very slow, or by making some approximations, which may be quite poor (Mackay, 2003).

Non-normalized models are often encountered in continous-valued Markov random fields, which are widely used in image modelling, see e.g. (Bouman and Sauer, 1993; Li, 2001). In general, undirected graphical models cannot be normalized except in the Gaussian case. Other recent work in image modelling also includes non-normalized models (Hyvärinen and Hoyer, 2001; Teh et al., 2003). Presumably, the number of useful applications for non-normalized models is much larger than the present literature suggests. Non-normalized models have been avoided because their estimation has been considered too difficult; the advent of efficient estimation methods may significantly increase their utility.

In this paper, we propose a simple method for estimating such non-normalized models. This is based on minimizing the expected squared distance of the score function of $\mathbf{x}$ and the score function given by the model. (By score function, we mean here the gradient of log-density.) We show that this distance can be estimated by a very simple formula involving only sample averages of some derivatives of the logarithm of the pdf given by the model. Thus, the computations involved are essentially not more complicated than in the case where we know an analytical expression for the normalization constant. The proposed formula is exact and does not involve any approximations, which is why we are able to prove the local consistency of the resulting method. Minimization of the proposed objective function thus provides an estimation method that is computationally simple yet statistically locally consistent.

2. Estimation by Score Matching

In the following, we use extensively the gradient of the log-density with respect to the data vector. For simplicity, we call this the score function, although according the conventional definition, it is actually the score function with respect to a hypothetical location parameter (Schervish, 1995). For the model density, we denote the score function by $\boldsymbol{\psi}(\boldsymbol{\xi}; \boldsymbol{\theta})$:

$$\boldsymbol{\psi}(\boldsymbol{\xi}; \boldsymbol{\theta}) = \begin{pmatrix} \frac{\partial \log p(\boldsymbol{\xi}; \boldsymbol{\theta})}{\partial \xi_1} \\ \vdots \\ \frac{\partial \log p(\boldsymbol{\xi}; \boldsymbol{\theta})}{\partial \xi_n} \end{pmatrix} = \begin{pmatrix} \psi_1(\boldsymbol{\xi}; \boldsymbol{\theta}) \\ \vdots \\ \psi_n(\boldsymbol{\xi}; \boldsymbol{\theta}) \end{pmatrix} = \nabla_{\boldsymbol{\xi}} \log p(\boldsymbol{\xi}; \boldsymbol{\theta}).$$

The point in using the score function is that it does not depend on $Z(\boldsymbol{\theta})$. In fact we obviously have

$$\boldsymbol{\psi}(\boldsymbol{\xi}; \boldsymbol{\theta}) = \nabla_{\boldsymbol{\xi}} \log q(\boldsymbol{\xi}; \boldsymbol{\theta}). \tag{1}$$

Likewise, we denote by $\boldsymbol{\psi}_{\mathbf{x}}(.) = \nabla_{\boldsymbol{\xi}} \log p_{\mathbf{x}}(.)$ the score function of the distribution of observed data $\mathbf{x}$. This could in principle be estimated by computing the gradient of the logarithm of a non-parametric estimate of the pdf—but we will see below that no such computation is necessary. Note that score functions are mappings from $\mathbb{R}^n$ to $\mathbb{R}^n$.

We now propose that the model is estimated by minimizing the expected squared distance between the model score function $\boldsymbol{\psi}(.;\boldsymbol{\theta})$ and the data score function $\boldsymbol{\psi}_{\mathbf{x}}(.)$. We define this squared distance as

$$J(\boldsymbol{\theta}) = \frac{1}{2}\int_{\boldsymbol{\xi}\in\mathbb{R}^n} p_{\mathbf{x}}(\boldsymbol{\xi})\|\boldsymbol{\psi}(\boldsymbol{\xi};\boldsymbol{\theta}) - \boldsymbol{\psi}_{\mathbf{x}}(\boldsymbol{\xi})\|^2 d\boldsymbol{\xi}. \tag{2}$$

Thus, our *score matching* estimator of $\boldsymbol{\theta}$ is given by

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}).$$

The motivation for this estimator is that the score function can be directly computed from q as in (1), and we do not need to compute Z. However, this may still seem to be a very difficult way of estimating $\boldsymbol{\theta}$, since we might have to compute an estimator of the data score function $\boldsymbol{\psi}_{\mathbf{x}}$ from the observed sample, which is basically a non-parametric estimation problem. However, no such non-parametric estimation is needed. This is because we can use a simple trick of partial integration to compute the objective function very easily, as shown by the following theorem:

Theorem 1 *Assume that the model score function $\boldsymbol{\psi}(\boldsymbol{\xi};\boldsymbol{\theta})$ is differentiable, as well as some weak regularity conditions.*[1]

Then, the objective function J in (2) can be expressed as

$$J(\boldsymbol{\theta}) = \int_{\boldsymbol{\xi}\in\mathbb{R}^n} p_{\mathbf{x}}(\boldsymbol{\xi})\sum_{i=1}^{n}\left[\partial_i\psi_i(\boldsymbol{\xi};\boldsymbol{\theta}) + \frac{1}{2}\psi_i(\boldsymbol{\xi};\boldsymbol{\theta})^2\right] d\boldsymbol{\xi} + \mathit{const.} \tag{3}$$

where the constant does not depend on $\boldsymbol{\theta}$,

$$\psi_i(\boldsymbol{\xi};\boldsymbol{\theta}) = \frac{\partial \log q(\boldsymbol{\xi};\boldsymbol{\theta})}{\partial \xi_i}$$

is the i-th element of the model score function, and

$$\partial_i\psi_i(\boldsymbol{\xi};\boldsymbol{\theta}) = \frac{\partial\psi_i(\boldsymbol{\xi};\boldsymbol{\theta})}{\partial\xi_i} = \frac{\partial^2 \log q(\boldsymbol{\xi};\boldsymbol{\theta})}{\partial\xi_i^2}$$

is the partial derivative of the i-th element of the model score function with respect to the i-th variable.

The proof, given in the Appendix, is based a simple trick of partial integration that has previously been used in the theory of independent component analysis for modelling the densities of the independent components (Pham and Garrat, 1997).

We have thus proven the remarkable fact that the squared distance of the model score function from the data score function can be computed as a simple expectation of certain

1. Namely: the data pdf $p_{\mathbf{x}}(\boldsymbol{\xi})$ is differentiable, the expectations $E_{\mathbf{x}}\{\|\boldsymbol{\psi}(\mathbf{x};\boldsymbol{\theta})\|^2\}$ and $E_{\mathbf{x}}\{\|\boldsymbol{\psi}_{\mathbf{x}}(\mathbf{x})\|^2\}$ are finite for any $\boldsymbol{\theta}$, and $p_{\mathbf{x}}(\boldsymbol{\xi})\boldsymbol{\psi}(\boldsymbol{\xi};\boldsymbol{\theta})$ goes to zero for any $\boldsymbol{\theta}$ when $\|\boldsymbol{\xi}\|\to\infty$.

functions of the non-normalized model pdf. If we have an analytical expression for the non-normalized density function q, these functions are readily obtained by derivation using (1) and taking further derivatives.

In practice, we have T observations of the random vector $\mathbf{x}$, denoted by $\mathbf{x}(1), \ldots, \mathbf{x}(T)$. The sample version of J is obviously obtained from (3) as

$$\tilde{J}(\boldsymbol{\theta}) = \frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{n} \left[\partial_i \psi_i(\mathbf{x}(t); \boldsymbol{\theta}) + \frac{1}{2} \psi_i(\mathbf{x}(t); \boldsymbol{\theta})^2 \right] + \text{const.} \tag{4}$$

which is asymptotically equivalent to J due to the law of large numbers. We propose to estimate the model by minimization of $\tilde{J}$ in the case of a real, finite sample.

One may wonder whether it is enough to minimize J to estimate the model, or whether the distance of the score functions can be zero for different parameter values. Obviously, if the model is degenerate in the sense that two different values of $\boldsymbol{\theta}$ give the same pdf, we cannot estimate $\boldsymbol{\theta}$. If we assume that the model is not degenerate, and that $q > 0$ always, we have local consistency as shown by the following theorem and the corollary:

Theorem 2 *Assume the pdf of $\mathbf{x}$ follows the model: $p_{\mathbf{x}}(.) = p(.; \boldsymbol{\theta}^*)$ for some $\boldsymbol{\theta}^*$. Assume further that no other parameter value gives a pdf that is equal*[2] *to $p(.; \boldsymbol{\theta}^*)$, and that $q(\boldsymbol{\xi}; \boldsymbol{\theta}) > 0$ for all $\boldsymbol{\xi}, \boldsymbol{\theta}$. Then*

$$J(\boldsymbol{\theta}) = 0 \Leftrightarrow \boldsymbol{\theta} = \boldsymbol{\theta}^*.$$

For a proof, see the Appendix.

Corollary 3 *Under the assumptions of the preceding Theorems, the score matching estimator obtained by minimization of $\tilde{J}$ is consistent, i.e. it converges in probability towards the true value of $\boldsymbol{\theta}$ when sample size approaches infinity, assuming that the optimization algorithm is able to find the global minimum.*

The corollary is proven by applying the law of large numbers. As sample size approaches infinity, $\tilde{J}$ converges to J (in probability). Thus, the estimator converges to a point where J is globally minimized. By Theorem 2, the global minimum is unique and found at the true parameter value (obviously, J cannot be negative).

This result of consistency assumes that the global minimum of $\tilde{J}$ is found by the optimization algorithm used in the estimation. In practice, this may not be true, in particular because there may be several local minima. Then, the consistency is of local nature, i.e., the estimator is consistent if the optimization iteration is started sufficiently close to the true value. Note that consistency implies asymptotic unbiasedness.

3. Examples

Here, we provide three simulations to illustrate how score matching works, as well as to confirm its consistency and applicability to real data.

2. In this theorem and its proof, equalities of pdf's are to be taken in the sense of equal almost everywhere with respect to the Lebesgue measure.

3.1 Multivariate Gaussian Density

As a very simple illustrative example, we consider estimation of the parameters of the multivariate Gaussian density.

3.1.1 Estimation

The probability density function is given by

$$p(\mathbf{x};\mathbf{M},\boldsymbol{\mu}) = \frac{1}{Z(\mathbf{M},\boldsymbol{\mu})}\exp(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\mathbf{M}(\mathbf{x}-\boldsymbol{\mu})),$$

where $\mathbf{M}$ is a symmetric positive-definite matrix (the inverse of the covariance matrix). Of course, the expression for Z is well-known in this case, but this serves as an illustration of the method. As long as there is no chance of confusion, we use $\mathbf{x}$ here as the general n-dimensional vector. Thus, here we have

$$q(\mathbf{x}) = \exp(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\mathbf{M}(\mathbf{x}-\boldsymbol{\mu})), \tag{5}$$

and we obtain

$$\boldsymbol{\psi}(\mathbf{x};\mathbf{M},\boldsymbol{\mu}) = -\mathbf{M}(\mathbf{x}-\boldsymbol{\mu}),$$

and

$$\partial_i\boldsymbol{\psi}(\mathbf{x};\mathbf{M},\boldsymbol{\mu}) = -m_{ii}.$$

Thus, we obtain

$$\tilde{J}(\mathbf{M},\boldsymbol{\mu}) = \frac{1}{T}\sum_{t=1}^{T}[\sum_i -m_{ii} + \frac{1}{2}(\mathbf{x}(t)-\boldsymbol{\mu})^T\mathbf{M}\mathbf{M}(\mathbf{x}(t)-\boldsymbol{\mu})]. \tag{6}$$

To minimize this with respect to $\boldsymbol{\mu}$, it is enough to compute the gradient

$$\nabla_{\boldsymbol{\mu}}\tilde{J} = \mathbf{M}\mathbf{M}\boldsymbol{\mu} - \mathbf{M}\mathbf{M}\frac{1}{T}\sum_{t=1}^{T}\mathbf{x}(t),$$

which is obviously zero if and only if $\boldsymbol{\mu}$ is the sample average $\frac{1}{T}\sum_{t=1}^{T}\mathbf{x}(t)$. This is truly a minimum because the matrix $\mathbf{MM}$ that defines the quadratic form is positive-definite.

Next, we compute the gradient with respect to $\mathbf{M}$, which gives

$$\nabla_{\mathbf{M}}\tilde{J} = -\mathbf{I} + \mathbf{M}\frac{1}{2T}\sum_{t=1}^{T}(\mathbf{x}(t)-\boldsymbol{\mu})(\mathbf{x}(t)-\boldsymbol{\mu})^T + \frac{1}{2T}[\sum_{t=1}^{T}(\mathbf{x}(t)-\boldsymbol{\mu})(\mathbf{x}(t)-\boldsymbol{\mu})^T]\mathbf{M},$$

which is zero if and only if $\mathbf{M}$ is the inverse of the sample covariance matrix $\frac{1}{T}\sum_{t=1}^{T}(\mathbf{x}(t)-\boldsymbol{\mu})(\mathbf{x}(t)-\boldsymbol{\mu})^T$, which thus gives the score matching estimate.

Interestingly, we see that score matching gives exactly the same estimator as maximum likelihood estimation. In fact, the estimators are identical for any sample (and not just asymptotically). The maximum likelihood estimator is known to be consistent, so the score matching estimator is consistent as well.

3.1.2 Intuitive Interpretation

This example also gives some intuitive insight into the principle of score matching. Let us consider what happened if we just maximized the non-normalized log-likelihood, i.e., log of q in (5). It is maximized when the scale parameters in $\mathbf{M}$ are zero, i.e., the model variances are infinite and the pdf is completely flat. This is because then the model assigns the same probability to all possible values of $\mathbf{x}(t)$, which is equal to 1. In fact, the same applies to the second term in (6), which thus seems to be closely connected to maximization of the non-normalized log-likelihood.

Therefore, the first term in (3) and (6), involving second derivatives of the logarithm of q, seems to act as a kind of a normalization term. Here it is equal to $-\sum_i m_{ii}$. To minimize this, the m_{ii} should be made as large (and positive) as possible. Thus, this term has the opposite effect to the second term. Since the first term is linear and the second term polynomial in $\mathbf{M}$, the minimum of the sum is different from zero.

A similar interpretation applies to the general non-Gaussian case. The second term in (3), expectation of the norm of score function, is closely related to maximization of non-normalized likelihood: if the norm of this gradient is zero, then in fact the data point is in a local extremum of the non-normalized log-likelihood. The first term then measures what kind of an extremum this is. If it is a minimum, the first term is positive and the value of J is increased. To minimize J, the first term should be negative, in which case the extremum is a maximum. In fact, the extremum should be as steep a maximum (as opposed to a flat maximum) as possible to minimize J. This counteracts, again, the tendency to assign the same probability to all data points that is often inherent in the maximization of the non-normalized likelihood.

3.2 Estimation of Basic Independent Component Analysis Model

Next, we show the validity of score matching in estimating the following model

$$\log p(\mathbf{x}) = \sum_{k=1}^{n} G(\mathbf{w}_k^T \mathbf{x}) + Z(\mathbf{w}_1, \dots, \mathbf{w}_n), \tag{7}$$

which is the basic form of the independent component analysis (ICA) model. Again, the normalization constant is well-known and equal to $-\log|\det \mathbf{W}|$ where the matrix $\mathbf{W}$ has the vectors $\mathbf{w}_i$ as rows, but this serves as an illustration of our method.

The nice thing about this model is that we can easily generate data that follows this model. In fact, if latent variables $s_i, i = 1 \dots, n$ are independently distributed and have the pdf given by $\exp(G(s_i))$, the linear transformation

$$\mathbf{x} = \mathbf{A}\mathbf{s} \tag{8}$$

with $\mathbf{A} = \mathbf{W}^{-1}$ follows the pdf's given in (7), see e.g. (Hyvärinen et al., 2001). Thus, we will be estimating the generative model in (8) using the non-normalized likelihood in (7).

Here, we choose the distribution of the components s_i to be so-called logistic with

$$G(s) = -2 \log\cosh(\frac{\pi}{2\sqrt{3}} s) - \log 4.$$

This distribution is normalized to unit variance as typical in the theory of ICA. The score function of the model in (7 is given by

$$\boldsymbol{\psi}(\mathbf{x};\mathbf{W}) = \sum_{k=1}^{n} \mathbf{w}_k g(\mathbf{w}_k^T \mathbf{x}), \tag{9}$$

where the scalar nonlinear function g is given by

$$g(s) = -\frac{\pi}{3}\tanh(\frac{\pi}{2\sqrt{3}}s).$$

The relevant derivatives of the score function are given by:

$$\partial_i \psi_i(x) = \sum_{k=1}^{n} w_{ki}^2 g'(\mathbf{w}_k^T \mathbf{x}),$$

and the sample version of the objective function $\tilde{J}$ is given by

$$\begin{aligned}\tilde{J} &= \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{n}\left[\sum_{k=1}^{n} w_{ki}^2 g'(\mathbf{w}_k^T\mathbf{x}(t)) + \frac{1}{2}\sum_{j=1}^{n} w_{ji} g(\mathbf{w}_j^T\mathbf{x}(t))\sum_{k=1}^{n} w_{ki} g(\mathbf{w}_k^T\mathbf{x}(t))\right]\\ &= \sum_{k=1}^{n}\|\mathbf{w}_k\|^2\frac{1}{T}\sum_{t=1}^{T} g'(\mathbf{w}_k^T\mathbf{x}(t)) + \frac{1}{2}\sum_{j,k=1}^{n}\mathbf{w}_j^T\mathbf{w}_k\frac{1}{T}\sum_{t=1}^{T} g(\mathbf{w}_k^T\mathbf{x}(t))g(\mathbf{w}_j^T\mathbf{x}(t)).\end{aligned} \tag{10}$$

We performed simulations to validate the consistency of score matching estimation, and to compare its efficiency with respect to maximum likelihood estimation. We generated data following the model as described above, where the dimension was chosen to be $n = 4$. Score matching estimation consisted of minimizing $\tilde{J}$ in (10) by a simple gradient descent; likelihood was maximized using a natural gradient method (Amari et al., 1996; Hyvärinen et al., 2001), using the true value of Z. We repeated the estimation for several different sample sizes: 500, 1000, 2000, 4000, 8000, and 16000. For each sample size, the estimation was repeated 11 times using different random initial points in the optimization, and different random data sets. For each estimate, a measure of asymptotic variance was computed as follows. The matrix $\hat{\mathbf{W}}\mathbf{A}$, where $\hat{\mathbf{W}}$ is the estimate was normalized row-by-row so that the largest value on each row had an absolute value of 1. Then, the sum of squares of all the elements was computed, and 4 (i.e. the sum of the squares of the four elements equal to one) was subtracted. This gives a measure of the squared error of the estimate (we cannot simply compare $\hat{\mathbf{W}}\mathbf{A}$ with identity because the order of the components is not well-defined). For each sample size and estimator type (score matching vs. maximum likelihood) we then computed the median error.

Figure 1 shows the results. The error of score matching seems to go to zero, which validates the theoretical consistency result of Theorem 2. Score matching gives slightly larger errors than maximum likelihood, which is to be expected because of the efficiency results of maximum likelihood estimation (Pham and Garrat, 1997).

In the preceding simulation, we knew exactly the proper function g to be used in the score function. To investigate the robustness of the method to misspecification of the score

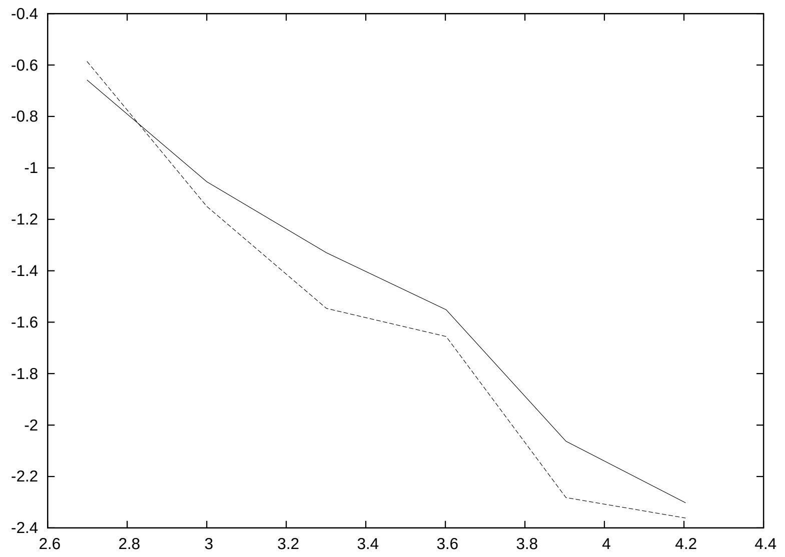


Figure 1: The estimation errors of score matching (solid line) compared with errors of maximum likelihood estimation (dashed line) for the basic ICA model. Horizontal axis: $\log_{10}$ of sample size. Vertical axis: $\log_{10}$ of estimation error.

function (a well-known problem in ICA estimation), we ran the same estimation methods, score matching and maximum likelihood, for data that was generated by a slightly different distribution. Specifically, we generated the data so that the independent components s_i had Laplacian distributions of unit variance (Hyvärinen et al., 2001). We then estimated the model using exactly the same g as before, which was not theoretically correct. The estimation errors are shown in Figure 2. We see that score matching still seems consistent. Interestingly, it now performs slightly better than maximum likelihood estimation (which would more properly be called quasi-maximum likelihood estimation due to the misspecification (Pham and Garrat, 1997)).

3.3 Estimation of an Overcomplete Model for Image Data

Finally, we show image analysis results using an overcomplete version of the ICA model. The likelihood is defined almost as in (7), but the number of components m is larger than the dimension of the data n, see e.g. (Teh et al., 2003), and we introduce some extra parameters. The likelihood is given by

$$\log p(\mathbf{x}) = \sum_{k=1}^{m} \alpha_k G(\mathbf{w}_k^T \mathbf{x}) + Z(\mathbf{w}_1, \ldots, \mathbf{w}_n, \alpha_1, \ldots, \alpha_n), \tag{11}$$

where the vectors $\mathbf{w}_k = (w_{k1}, \ldots, w_{kn})$ are constrained to unit norm (unlike in the preceding example), and the α_k are scaling parameters. We introduce here the extra parameters α_k to account for different distributions for different projections. Constraining $\alpha_k = 1$ and $m = n$

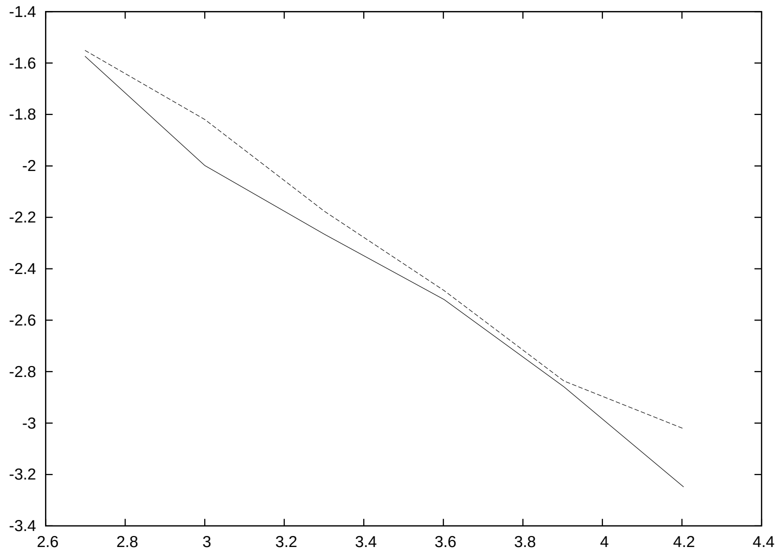


Figure 2: The estimation errors of score matching compared with errors of maximum likelihood estimation for the basic ICA model. This time, the pdf of the independent components was slightly misspecified. Legend as in Fig. 1.

and allowing the $\mathbf{w}_k$ to have any norm, this becomes the basic ICA model of the preceding subsection.

The model is related to ICA with overcomplete bases (Hyvärinen et al., 2001; Hyvärinen and Inki, 2002; Olshausen and Field, 1997), i.e. the case where there are more independent components and basis vectors than observed variables. In contrast to most ICA models, the overcompleteness is expressed as overcompleteness of *filters* $\mathbf{w}_k$ which seems to make the problem a bit simpler because no latent variables need to be inferred. However, the normalization constant Z is not known when G is non-quadratic, i.e. when the model is non-Gaussian, which is why previous research had to resort to MCMC methods (Teh et al., 2003) or some approximations (Hyvärinen and Inki, 2002).

We have the score function

$$\boldsymbol{\psi}(\mathbf{x}; \mathbf{W}, \alpha_1, \dots, \alpha_m) = \sum_{k=1}^{m} \alpha_k w_k g(\mathbf{w}_k^T \mathbf{x}),$$

where g is the first derivative of G. Going through similar developments as in the case of the basic ICA model, the sample version of the objective function $\tilde{J}$ can be shown to equal

$$\tilde{J} = \sum_{k=1}^{m} \alpha_k \frac{1}{T} \sum_{t=1}^{T} g'(\mathbf{w}_k^T \mathbf{x}(t)) + \frac{1}{2} \sum_{j,k=1}^{m} \alpha_j \alpha_k \mathbf{w}_j^T \mathbf{w}_k \frac{1}{T} \sum_{t=1}^{T} g(\mathbf{w}_k^T \mathbf{x}(t)) g(\mathbf{w}_j^T \mathbf{x}(t)). \tag{12}$$

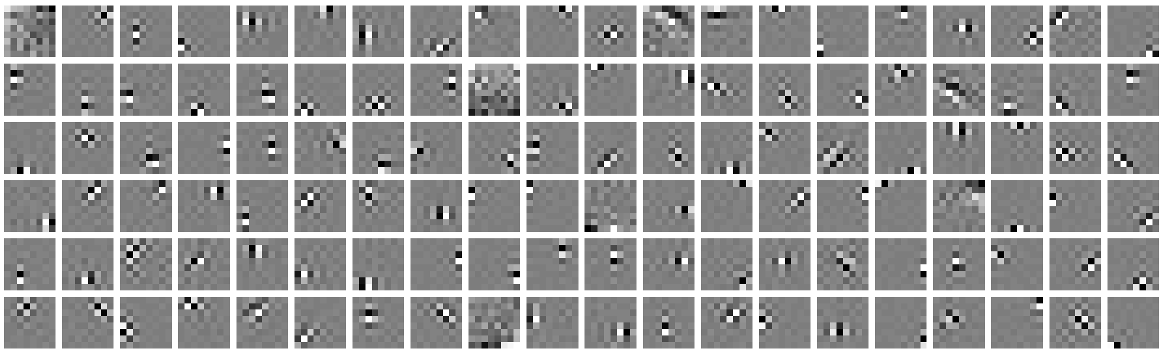


Figure 3: The overcomplete set of filters $\mathbf{w}_i$ estimated from natural image data. Note that no dimension reduction was performed, and we show filters instead of basis vectors, which is why the results are much less smooth and "beautiful" than some published ICA results (Hyvärinen et al., 2001).

We estimated the model for image patches of 8×8 pixels taken from natural images, see P.O. Hoyer's *imageica* package.[3] As preprocessing, the DC component (i.e. the mean gray-scale value) was removed from each image patch, reducing the effective dimensionality of the data to $n = 63$. The data was also whitened, i.e. the model was used in a linearly transformed space (the exact method of whitening has no significance). We set $m = 200$. We also took the tanh function as g, which corresponds to $G(u) = \log\cosh(u)$ (we did not bother to find the right scaling as in the basic ICA case). The objective function $\tilde{J}$ in (12) was optimized by gradient descent. The $\mathbf{w}_i$ were set to random initial values, and the α_i were all set to the initial value 1.5 that was found to be close to the optimal value in pilot experiments.

The obtained vectors $\mathbf{w}_i$ are shown in Figure 3. For the purposes of visualization, the vectors were converted back to the original space from the whitened space. The optimal α_i were in the range $0.5 \ldots 2$.

To show that the method correctly found different vectors and not duplicates of a smaller set of vectors, we computed the dot-products between the vectors, and for each $\mathbf{w}_i$, we selected the largest absolute value of dot-product $|\mathbf{w}_i^T \mathbf{w}_j|, j \neq i$. The dot-products were computed in the whitened space. The histogram of these maximal dot-products is shown in Figure 4. They are all much smaller than 1 (in absolute value), in fact all are smaller than 0.5. Since the vectors $\mathbf{w}_i$ were normalized to unit norm, this shows that no two $\mathbf{w}_i$ were close to equal, and we did find m different vectors.

4. Discussion

Here we discuss the connections of our method to two well-known methods before concluding the paper.

3. The package can be downloaded at http://www.cs.helsinki.fi/patrik.hoyer/.

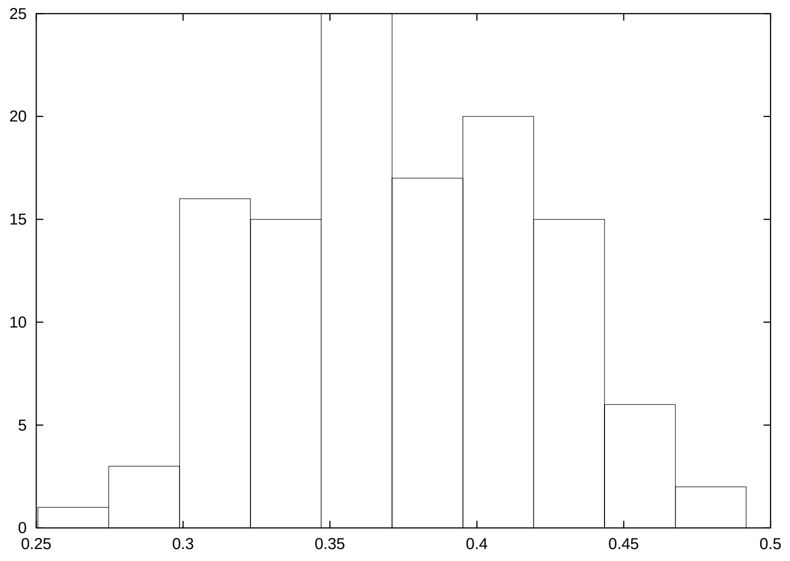


Figure 4: The distribution of maximal dot-products of a filter $\mathbf{w}_i$ with all other filters, computed in the whitened space.

4.1 Comparison with Pseudo-Likelihood Estimation

A related method for estimating non-normalized models is maximization of pseudo-likelihood (Besag, 1974). The idea is to maximize the product of marginal conditional likelihoods. The pdf is approximated by

$$\log p_{pseudo}(\mathbf{x}) = \sum_{i=1}^{n} p(x_i | x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n), \tag{13}$$

and the likelihood is computed using this approximation. The idea was originally developed in connection with Markov random fields, in which context it is quite natural because the conditional probabilities are often given as part of the model specification. The idea can still be used in the general case considered in this article. However, the conditional probabilities in (13) are not necessarily readily available and need to be computed. In particular, these conditional densities need to be normalized. The computational burden needed in the normalization is reduced from the original problem since we only need to numerically compute n one-dimensional integrals which is far more feasible than a single n-dimensional integral. However, compared to score matching, this is a computationally expensive method since score matching avoids the need for numerical integration altogether.

The question of consistency of pseudo-likelihood estimation seems to be unclear. Some consistency proofs were provided by Besag (1974, 1977), but these only apply to special cases such as Gaussian or binary random fields. Sufficiently general consistency results on pseudo-likelihood estimation seem to be lacking. This is another disadvantage with respect to score matching, which was shown above to be (locally) consistent.

4.2 Comparison with Contrastive Divergence

An interesting approximative MCMC method called contrastive divergence was recently proposed by Hinton (2002). The basic principle is to use an MCMC method for computing the derivative of the logarithm of the normalization factor Z, but the MCMC is allowed to run for only a single iteration (or a few iterations) before doing the gradient step.

The method is generally biased, even asymptotically (Carreira-Perpiñán and Hinton, 2005b), except in some special cases such as the multivariate Gaussian distribution (Carreira-Perpiñán and Hinton, 2005a). Score matching is thus preferable if a consistent estimator is wanted.

The computational efficiency of contrastive divergence is difficult to evaluate since it is not really a single method but a family of methods, depending on the MCMC method used. For the case of continuous-valued variables that we consider here, a Metropolis-type algorithm would probably be the method of choice, but there is a large number of different variants whose performances are likely to be quite different.

Nevertheless, contrastive divergence is a much more general method than score matching since it is applicable to intractable latent variable models. It can also handle binary/discrete variables—in fact, it is probably much easier to implement, using Gibbs sampling, for binary variables than for continous-valued variables. Extension of score matching to these two cases is an important problem for future research.

4.3 Conclusion

We have proposed a new method, score matching, to estimate statistical models in the case where the normalization constant is unknown. Although the estimation of the score function is computationally difficult, we showed that the distance of data and model score functions is very easy to compute. The main assumptions in the method are: 1) all the variables are continuous-valued and defined over $\mathbb{R}^n$, 2) the model pdf is smooth enough. Score matching provides a computationally simple yet locally consistent alternative to existing methods, such as MCMC and various approximative methods.

Acknowledgments

I am grateful to Patrik Hoyer, Jarmo Hurri, and Shohei Shimizu for comments on the manuscript, to Sam Roweis for interesting discussions, and to Miguel Carreira-Perpiñán and Geoffrey Hinton for providing access to unpublished results. The work was supported by the Academy of Finland, Academy Research Fellow position and project #48593.

Appendix A. Proof of Theorem 1

Definition (2) gives

$$J(\boldsymbol{\theta}) = \int p_{\mathbf{x}}(\boldsymbol{\xi}) \left[\frac{1}{2} \|\boldsymbol{\psi}_{\mathbf{x}}(\boldsymbol{\xi})\|^2 + \frac{1}{2} \|\boldsymbol{\psi}(\boldsymbol{\xi}; \boldsymbol{\theta})\|^2 - \boldsymbol{\psi}_{\mathbf{x}}(\boldsymbol{\xi})^T \boldsymbol{\psi}(\boldsymbol{\xi}; \boldsymbol{\theta}) \right] d\boldsymbol{\xi}. \tag{14}$$

(For simplicity, we omit the integration domain in here.) The first term in brackets does not depend on $\boldsymbol{\theta}$, and can be ignored. The integral of the second term is simply the integral of the sum of the second terms in brackets in (3). Thus, the difficult thing to prove is that integral of the third term in brackets in (14) equals the integral of the sum of the first terms in brackets in (3). This term equals

$$-\sum_i \int p_{\mathbf{x}}(\boldsymbol{\xi})\psi_{\mathbf{x},i}(\boldsymbol{\xi})\psi_i(\boldsymbol{\xi};\theta)d\boldsymbol{\xi},$$

where $\psi_{\mathbf{x},i}(\boldsymbol{\xi})$ denotes the i-th element of the vector $\boldsymbol{\psi}_{\mathbf{x}}(\boldsymbol{\xi})$. We can consider the integral for a single i separately, which equals

$$-\int p_{\mathbf{x}}(\boldsymbol{\xi})\frac{\partial \log p_{\mathbf{x}}(\boldsymbol{\xi})}{\partial \xi_i}\psi_i(\boldsymbol{\xi};\boldsymbol{\theta})d\boldsymbol{\xi} = -\int \frac{p_{\mathbf{x}}(\boldsymbol{\xi})}{p_{\mathbf{x}}(\boldsymbol{\xi})}\frac{\partial p_{\mathbf{x}}(\boldsymbol{\xi})}{\partial \xi_i}\psi_i(\boldsymbol{\xi};\boldsymbol{\theta})d\boldsymbol{\xi} = -\int \frac{\partial p_{\mathbf{x}}(\boldsymbol{\xi})}{\partial \xi_i}\psi_i(\boldsymbol{\xi};\boldsymbol{\theta})d\boldsymbol{\xi}.$$

The basic trick of partial integration needed the proof is simple: for any one-dimensional pdf p and any function f, we have

$$\int p(x)(\log p)'(x)f(x)dx = \int p(x)\frac{p'(x)}{p(x)}f(x)dx = \int p'(x)f(x)dx = -\int p(x)f'(x)dx$$

under some regularity assumptions that will be dealt with below.

To proceed with the proof, we need to use a multivariate version of such partial integration:

Lemma 4

$$\lim_{a\to\infty,b\to-\infty} f(a,\xi_2,\dots,\xi_n)g(a,\xi_2,\dots,\xi_n) - f(b,\xi_2,\dots,\xi_n)g(b,\xi_2,\dots,\xi_n)$$
$$= \int_{-\infty}^{\infty} f(\boldsymbol{\xi})\frac{\partial g(\boldsymbol{\xi})}{\partial \xi_1}d\xi_1 + \int_{-\infty}^{\infty} g(\boldsymbol{\xi})\frac{\partial f(\boldsymbol{\xi})}{\partial \xi_1}d\xi_1,$$

assuming that f and g are differentiable. The same applies for all indices of ξ_i, but for notational simplicity we only write the case $i = 1$ here.

Proof of lemma:

$$\frac{\partial f(\boldsymbol{\xi})g(\boldsymbol{\xi})}{\partial \xi_1} = f(\boldsymbol{\xi})\frac{\partial g(\boldsymbol{\xi})}{\partial \xi_1} + g(\boldsymbol{\xi})\frac{\partial f(\boldsymbol{\xi})}{\partial \xi_1}.$$

We can now consider this as a function of ξ_1 alone, all other variables being fixed. Then, integrating over $\xi_1 \in \mathbb{R}$, we have proven the lemma.

Now, we can apply this lemma on $p_{\mathbf{x}}$ and $\psi_1(\boldsymbol{\xi};\boldsymbol{\theta})$ which were both assumed to be differentiable in the theorem, and we obtain:

$$-\int \frac{\partial p_{\mathbf{x}}(\boldsymbol{\xi})}{\partial \xi_1}\psi_1(\boldsymbol{\xi};\boldsymbol{\theta})d\boldsymbol{\xi} = -\int \left[\int \frac{\partial p_{\mathbf{x}}(\boldsymbol{\xi})}{\partial \xi_1}\psi_1(\boldsymbol{\xi};\boldsymbol{\theta})d\xi_1\right] d(\xi_2,\dots,\xi_n)$$
$$= -\int \Bigg[\lim_{a\to\infty,b\to-\infty}[p_{\mathbf{x}}(a,\xi_2,\dots,\xi_n)\psi_1(a,\xi_2,\dots,\xi_n;\boldsymbol{\theta})$$
$$- p_{\mathbf{x}}(b,\xi_2,\dots,\xi_n)\psi_1(b,\xi_2,\dots,\xi_n;\boldsymbol{\theta})]$$
$$-\int \frac{\partial \psi_1(\boldsymbol{\xi};\boldsymbol{\theta})}{\partial \xi_1}p_{\mathbf{x}}(\boldsymbol{\xi})d\xi_1\Bigg] d(\xi_2,\dots,\xi_n).$$

For notational simplicity, we consider the case of $i = 1$ only, but this is true for any i.

The limit in the above expression is zero for any $\xi_2, \ldots, \xi_n, \boldsymbol{\theta}$ because we assumed that $p_{\mathbf{x}}(\boldsymbol{\xi})\boldsymbol{\psi}(\boldsymbol{\xi};\boldsymbol{\theta})$ goes to zero at infinity. Thus, we have proven that

$$-\int \frac{\partial p_{\mathbf{x}}(\boldsymbol{\xi})}{\partial \xi_i}\psi_i(\boldsymbol{\xi};\boldsymbol{\theta})d\boldsymbol{\xi} = \int \frac{\partial \psi_i(\boldsymbol{\xi};\boldsymbol{\theta})}{\partial \xi_i}p_{\mathbf{x}}(\boldsymbol{\xi})d\boldsymbol{\xi},$$

that is, integral of the the third term in brackets in (14) equals the integral of the sum of the first terms in brackets in (3), and the proof of the theorem is complete.

Appendix B. Proof of Theorem 2

Assume $J(\boldsymbol{\theta}) = 0$. Then, the assumption $q > 0$ implies $p_{\mathbf{x}}(\boldsymbol{\xi}) > 0$ for all $\boldsymbol{\xi}$, which implies that $\boldsymbol{\psi}_{\mathbf{x}}(.)$ and $\boldsymbol{\psi}(.;\boldsymbol{\theta})$ are equal. This implies $\log p_{\mathbf{x}}(.) = \log p(.;\boldsymbol{\theta}) + c$ for some constant c. But c is necessarily 0 because both $p_{\mathbf{x}}$ and $p(.;\boldsymbol{\theta})$ are pdf's. Thus, $p_{\mathbf{x}} = p(.;\boldsymbol{\theta})$. By assumption, only $\boldsymbol{\theta} = \boldsymbol{\theta}^*$ fulfills this equality, so necessarily $\boldsymbol{\theta} = \boldsymbol{\theta}^*$, and we have proven the implication from left to right. The converse is trivial.

References

S.-I. Amari, A. Cichocki, and H. H. Yang. A new learning algorithm for blind source separation. In *Advances in Neural Information Processing Systems 8*, pages 757–763. MIT Press, 1996.

J. Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society, Series B*, 36(2):192–236, 1974.

J. Besag. Efficiency of pseudolikelihood estimation for simple gaussian fields. *Biometrika*, 64(3):616–618, 1977.

C. Bouman and K. Sauer. A generalized gaussian image model for edge-preserving MAP estimation. *IEEE Transactions on Image Processing*, 2(3):296–310, 1993.

M. Á. Carreira-Perpiñán and G. E. Hinton. On contrastive divergence (CD) learning. Technical report, Dept of Computer Science, University of Toronto, 2005a. In preparation.

M. Á. Carreira-Perpiñán and G. E. Hinton. On contrastive divergence learning. In *Proceedings of the Workshop on Artificial Intelligence and Statistics (AISTATS2005)*, Barbados, 2005b.

G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

A. Hyvärinen and P. O. Hoyer. A two-layer sparse coding model learns simple and complex cell receptive fields and topography from natural images. *Vision Research*, 41(18):2413–2423, 2001.

A. Hyvärinen and M. Inki. Estimating overcomplete independent component bases from image windows. *Journal of Mathematical Imaging and Vision*, 17:139–152, 2002.

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

S. Z. Li. *Markov Random Field Modeling in Image Analysis.* Springer, 2nd edition, 2001.

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

B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37:3311–3325, 1997.

D.-T. Pham and P. Garrat. Blind separation of mixture of independent sources through a quasi-maximum likelihood approach. *IEEE Transactions on Signal Processing*, 45(7): 1712–1725, 1997.

M. Schervish. *Theory of Statistics.* Springer, 1995.

Y. W. Teh, M. Welling, S. Osindero, and G. E. Hinton. Energy-based models for sparse overcomplete representations. *Journal of Machine Learning Research*, 4:1235–1260, 2003.

Journal of Machine Learning Research 6 (2005) 711–741 Submitted 1/04; Revised 2/05; Published 5/05

Smooth ε-Insensitive Regression by Loss Symmetrization

Ofer Dekel OFERD@CS.HUJI.AC.IL
Shai Shalev-Shwartz SHAIS@CS.HUJI.AC.IL
Yoram Singer SINGER@CS.HUJI.AC.IL
School of Computer Science and Engineering
The Hebrew University
Jerusalem, 91904, Israel

Editors: Kristin P. Bennett and Nicolò Cesa-Bianchi

Abstract

We describe new loss functions for regression problems along with an accompanying algorithmic framework which utilizes these functions. These loss functions are derived by symmetrization of margin-based losses commonly used in boosting algorithms, namely, the logistic loss and the exponential loss. The resulting symmetric logistic loss can be viewed as a smooth approximation to the ε-insensitive hinge loss used in support vector regression. We describe and analyze two parametric families of batch learning algorithms for minimizing these symmetric losses. The first family employs an iterative *log-additive* update which can be viewed as a regression counterpart to recent boosting algorithms. The second family utilizes an iterative *additive* update step. We also describe and analyze online gradient descent (GD) and exponentiated gradient (EG) algorithms for the symmetric logistic loss. A byproduct of our work is a new simple form of regularization for boosting-based classification and regression algorithms. Our regression framework also has implications on classification algorithms, namely, a new additive update boosting algorithm for classification. We demonstrate the merits of our algorithms in a series of experiments.

1. Introduction

The focus of this paper is supervised learning of real-valued functions. We observe a sequence $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ of instance-target pairs, where the instances are vectors in $\mathbb{R}^n$ and the targets are real-valued scalars, $y_i \in \mathbb{R}$. Our goal is to learn a function $f : \mathbb{R}^n \to \mathbb{R}$ which provides a good approximation of the target values from their corresponding instance vectors. Such a function is often referred to as a regression function or a regressor for short. Regression problems have long been the focus of research papers in statistics and learning theory (see for instance the book by Hastie, Tibshirani, and Friedman (2001) and the references therein). In this paper we discuss learning of linear regressors, that is, f is of the form $f(\mathbf{x}) = \boldsymbol{\lambda} \cdot \mathbf{x}$. This setting is also suitable for learning a linear combination of base regressors of the form $f(\mathbf{x}) = \sum_{j=1}^{l} \lambda_j h_j(\mathbf{x})$ where each base regressor h_j is a mapping from an instance domain X into $\mathbb{R}$. The latter form enables us to employ kernels by setting $h_j(\mathbf{x}) = K(\mathbf{x}_j, \mathbf{x})$.

The class of linear regressors is rather restricted. Furthermore, in real applications both the instances and the target values are often corrupted by noise and a perfect mapping such that for all $(\mathbf{x}_i, y_i) \in S$, $f(\mathbf{x}_i) = y_i$ is usually unobtainable. Hence, we employ a loss

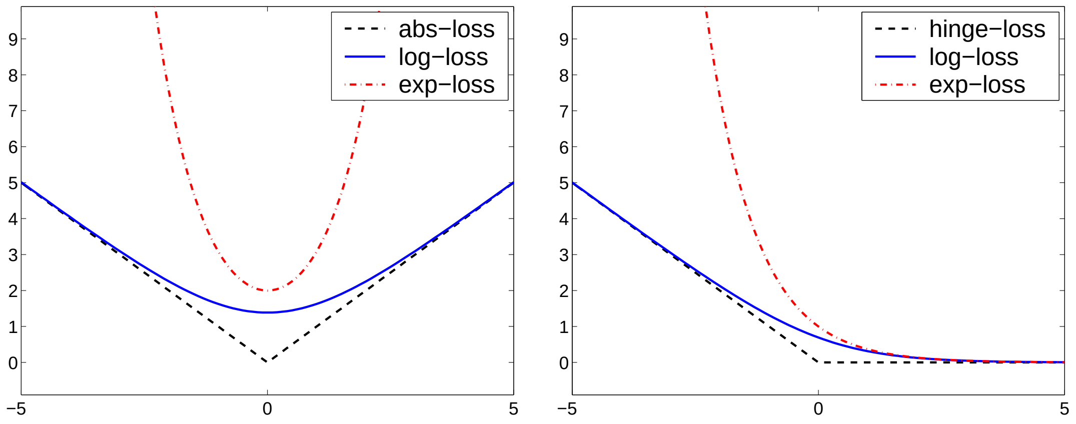


Figure 1: Constructing regression losses (left) by symmetrization of margin losses (right).

function $L : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$ which determines the penalty for a discrepancy between the *predicted* target, $f(\mathbf{x})$, and the *true* (observed) target y. As we discuss shortly, the loss functions we consider in this paper depend only on the discrepancy between the predicted target and the true target $\delta = f(\mathbf{x}) - y$, hence L can be viewed as a function from $\mathbb{R}$ into $\mathbb{R}_+$. We therefore allow ourselves to overload our notation and denote $L(\delta) = L(f(\mathbf{x}), y)$.

Given a loss function L, the goal of a regression algorithm is to find a regressor f which attains a small total loss on the training set S,

$$\text{Loss}(\boldsymbol{\lambda}, S) \;=\; \sum_{i=1}^{m} L(f(\mathbf{x}_i) - y_i) \;=\; \sum_{i=1}^{m} L(\boldsymbol{\lambda} \cdot \mathbf{x}_i - y_i).$$

Denoting the discrepancy $\boldsymbol{\lambda} \cdot \mathbf{x}_i - y_i$ by δ_i, we note that two common approaches to solving regression problems are to minimize either the sum of the absolute discrepancies over the sample ($\sum_i |\delta_i|$) or the sum of squared discrepancies ($\sum_i \delta_i^2$). It has been argued that the squared loss is sensitive to outliers, hence robust regression algorithms often employ the absolute loss (Huber, 1981). Furthermore, it is often the case that the *exact* discrepancy between $\boldsymbol{\lambda} \cdot \mathbf{x}$ and y is unimportant so long as it falls below an insensitivity parameter ε. Formally, the ε-insensitive hinge loss, denoted $|\delta|_\varepsilon$, is zero if $|\delta| \le \varepsilon$ and is $|\delta| - \varepsilon$ for $|\delta| > \varepsilon$ (see also the left hand side of Figure 2). The ε-insensitive hinge loss is not smooth as its derivative is discontinuous at $\delta = \pm\varepsilon$. Several batch learning algorithms have been proposed for minimizing the ε-insensitive hinge loss (see for example Vapnik, 1998; Smola and Schölkopf, 1998). However, these algorithms are based on rather complex constrained optimization techniques since the ε-insensitive hinge loss is a non-smooth function.

The first loss function presented in this paper is a smooth approximation to the ε-insensitive hinge loss. Define the *symmetric ε-insensitive logistic loss*, or *log-loss* for short, to be,

$$L_{\log}(\delta; \varepsilon) = \log\left(1 + e^{\delta - \varepsilon}\right) + \log\left(1 + e^{-\delta - \varepsilon}\right) - \kappa. \tag{1}$$

Whenever it is clear from context we omit the insensitivity parameter ε and denote this loss by $L_{\log}(\delta)$. The constant κ in Eq. (1) equals $2\log(1 + e^{-\varepsilon})$ and is set such that $L_{\log}(0) = 0$. Since additive constants do not affect the value of the minimizer of $L_{\log}(\delta)$, we omit κ

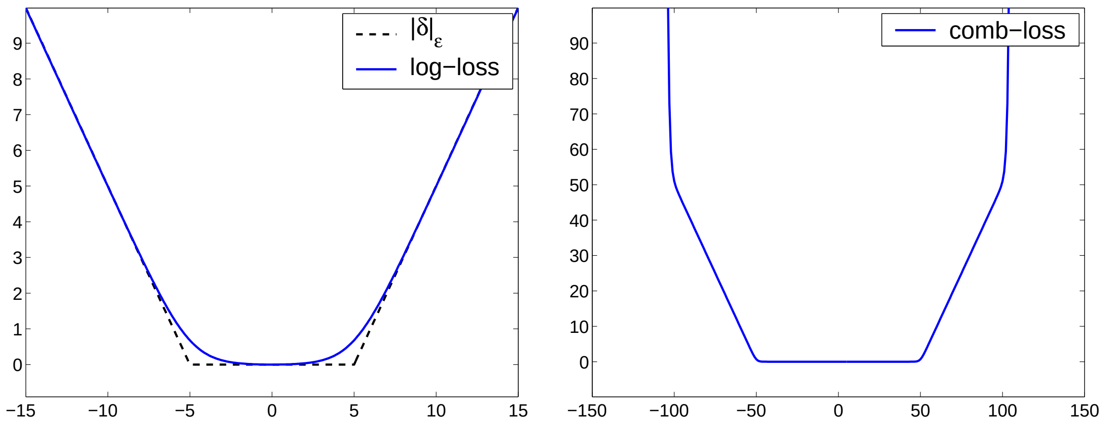


Figure 2: The smooth ε-insensitive log-loss (left) and the comb-loss (right).

henceforth. In Figure 2 we depict the ε-insensitive log-loss along with the ε-insensitive hinge loss for $\varepsilon = 5$. Note that the ε-insensitive log-loss provides a smooth upper bound on the ε-insensitive hinge loss. Moreover, note that for this particular choice of ε and for $|\delta| < 2$ and $|\delta| > 8$ the log-loss and hinge-loss are graphically indistinguishable.

To motivate our construction, let us take a short detour and discuss a recent view of margin-based classification algorithms. In the binary classification setting discussed in Friedman et al. (2000), Collins et al. (2002) and Lebanon and Lafferty (2001), we are provided with *instance-label* pairs, $(\mathbf{x}, y)$, where, in contrast to regression, each label takes one of two values, namely $y \in \{-1, +1\}$. A real-valued classifier is a function f into the reals such that $\operatorname{sign}(f(\mathbf{x}))$ is the predicted label and $|f(\mathbf{x})|$ is the confidence of f in its prediction. The product $yf(\mathbf{x})$ is called the (signed) margin of the instance-label pair $(\mathbf{x}, y)$. The goal of a margin-based classifier is to attain large margin values on as many instances as possible. Learning algorithms for margin-based classifiers typically employ a margin-based loss function $L_{\mathrm{c}}(yf(\mathbf{x}))$ and attempt to minimize the total loss over all instances in a given sample. One of the margin losses discussed is the logistic loss, which takes the form

$$L_{\mathrm{c}}(yf(\mathbf{x})) = \log\left(1 + e^{-yf(\mathbf{x})}\right) . \tag{2}$$

We discuss a general technique for reducing a regression problem to a margin-based classification problem called *loss symmetrization*. The symmetric log-loss given in Eq. (1) is obtained by applying this technique to the classification logistic loss in Eq. (2). The technique of loss symmetrization was previously discussed in Bi and Bennett (2003) in the context of support vector regression.

Formally, let $[\mathbf{u}\,;\,v]$ denote the concatenation of an additional element v to the end of a vector $\mathbf{u}$. We replace every instance-*target* pair $(\mathbf{x}, y)$ from the regression problem with *two* classification instance-*label* pairs,

$$(\mathbf{x}, y) \;\mapsto\; \left\{ \begin{array}{l} ([\mathbf{x}\,;\,-y+\varepsilon]\,,\,+1) \\ ([\mathbf{x}\,;\,-y-\varepsilon]\,,\,-1) \end{array} \right. .$$

In words, we duplicate each regression instance and create two instances of a classification problem. We then increase the dimension of the instance vectors by one and concatenate

$-y + \varepsilon$ to the first newly created instance and set its label to $+1$. Symmetrically, we concatenate $-y - \varepsilon$ to the second copy of the instance and set its label to -1. We define the linear *classifier* to be the vector $[\boldsymbol{\lambda}\,;\,1] \in \mathbb{R}^{n+1}$. It is simple to verify that,

$$L_{\log}(\boldsymbol{\lambda} \cdot \mathbf{x} - y\,;\,\varepsilon) = L_{\mathrm{C}}([\boldsymbol{\lambda}\,;\,1] \cdot [\mathbf{x}\,;\,-y+\varepsilon]) + L_{\mathrm{C}}(-[\boldsymbol{\lambda}\,;\,1] \cdot [\mathbf{x}\,;\,-y-\varepsilon]).$$

In Figure 1 we give an illustration of the above construction. We have thus reduced a regression problem of m instances in $\mathbb{R}^n$ with targets in $\mathbb{R}$ to a classification problem with $2m$ instances in $\mathbb{R}^{n+1}$ and binary labels in $\{\pm 1\}$.

The work in Collins et al. (2002) gave a unified view of two margin losses: the logistic loss defined by Eq. (2) and an exponential loss. An immediate benefit of our construction is a similar unified account of the two respective regression losses. Formally, we define the *symmetric exponential loss*, or *exp-loss* for short, as

$$L_{\exp}(\delta) \;=\; e^{\delta} + e^{-\delta}. \tag{3}$$

The exp-loss was first presented and analyzed by Duffy and Helmbold (2000) in their pioneering work on leveraging regressors. However, their view is somewhat different than ours as it builds upon the notion of weak-learnability, yielding a different (sequential) algorithm for regression. The exp-loss is by far less forgiving than the log-loss, i.e. small discrepancies are amplified exponentially. While this property might be undesirable in regression problems with numerous outliers, it can also serve as a barrier that prevents the existence of any large discrepancy in the training set. To see this, note that the minimizer of $\sum_i L_{\exp}(\delta_i)$ is also the minimizer of $\log(\sum_i L_{\exp}(\delta_i))$ which is a smooth approximation to $\max_i |\delta_i|$.

We can also combine the log-loss and the exp-loss with two different insensitivity parameters and benefit both from a discrepancy insensitivity region and from enforcing a smooth barrier on the maximal discrepancy. Formally, let $\varepsilon_1 > 0$ and $\varepsilon_2 > \varepsilon_1$ be two insensitivity parameters. We define the *combined loss*, abbreviated as *comb-loss*, by

$$L_{\mathrm{comb}}(\delta; \varepsilon_1, \varepsilon_2) \;=\; L_{\log}(\delta; \varepsilon_1) + L_{\exp}(\delta; \varepsilon_2),$$

where $L_{\exp}(\delta; \varepsilon_2) = e^{-\varepsilon_2} L_{\exp}(\delta)$. An illustration of the combined loss with $\varepsilon_1 = 50$ and $\varepsilon_2 = 100$ is given on the right hand side of Figure 2.

The paper is organized as follows. In Section 2 we describe and analyze a family of *log-additive update* algorithms for batch learning settings. The algorithms in this family are in essence boosting algorithms for regression problems. The symmetrization technique outlined above is used to derive these algorithms and to adapt proof techniques from Collins et al. (2002). In Section 3 we describe another family of *additive update* regression algorithms based on modified gradient descent. For both the log-additive and the additive updates, we provide a boosting-style analysis of the decrease in loss. Then, in Section 4, we describe a simple use of the symmetric losses defined above as a means of regularizing our batch learning algorithms and other boosting algorithms as well. In Section 5 we discuss the convergence properties of both log-additive and additive update algorithms, when applied with regularization. We then show the implications of our work on classification problems in Section 6. Specifically, we show how both the additive update algorithm of Section 3 and the regularization scheme of Section 4 extend to the setting of classification. In Section 7 we shift our attention to *online* learning algorithms for the ε-insensitive log-loss. In Section 8 we complement our formal discussion with a set of experimental results obtained on real and synthetic data sets. Specifically, we demonstrate the different properties of the log-loss and exp-loss functions, we compare the different algorithms presented in this paper under different settings and discuss the effect of regularization on the generalization abilities of our algorithms. In this section, we also present a detailed example of boosting a weak-learning regression algorithm using our techniques. We conclude the paper in Section 9.

2. Log-additive Update for Batch Regression

In the previous section we discussed a general reduction from regression problems to margin-based classification problems. As a first application of this reduction, we devise a family of batch regression learning algorithms based on boosting techniques. We term these algorithms *log-additive update* algorithms as they iteratively update $\boldsymbol{\lambda}$ by a logarithmic function of the gradient of the loss.

Our implicit goal is to obtain the (global) minimizer of the empirical loss function $\sum_{i=1}^{m} L(\boldsymbol{\lambda} \cdot \mathbf{x}_i - y_i)$ where L is either the log-loss, the exp-loss or the comb-loss. We first prove that progress is made on every iteration of the learning algorithm. For the sake of clarity, the main theorem of this section is stated and proven only for the log-loss. We then complete our presentation with a brief discussion on how the theorem is easily adapted to the exp-loss and comb-loss cases. In Section 5 we show how progress on every iteration leads to convergence to the global minimum of the respective loss function.

Following the general paradigm of boosting, we make the assumption that we have access to a set of predefined base regressors. These base regressors are analogous to the weak hypotheses commonly discussed in boosting. The goal of the learning algorithm is to select a subset of base regressors and combine them linearly to obtain a highly accurate strong regressor. We assume that the set of base regressors is of finite cardinality though our algorithms can be generalized to deal with a countably infinite number of base regressors. In the finite case we can simply map each input instance to the vector of images generated by each of the base-regressors, $\mathbf{x} \mapsto (h_1(\mathbf{x}), \ldots, h_n(\mathbf{x}))$, where n is the number of base-regressors. Using this transformation, each input instance is a vector $\mathbf{x}_i \in \mathbb{R}^n$ and the strong regressor's prediction is $\boldsymbol{\lambda} \cdot \mathbf{x}$. The j'th element of $\boldsymbol{\lambda}$, namely λ_j, should be regarded as the weight associated with the base regressor h_j.

Boosting was initially described and analyzed as a *sequential* algorithm that iteratively selects a single base-hypothesis or feature h_j and updates its weight λ_j. All of the elements of $\boldsymbol{\lambda}$ are initialized to zero, so after performing T sequential update iterations, at most T elements of $\boldsymbol{\lambda}$ are non-zero. Thus, this form of sequential update can be used for feature selection as well as loss minimization. An alternative approach is to simultaneously update all of the elements of $\boldsymbol{\lambda}$ on every iteration. This approach is the more common among regression algorithms. Collins et al. (2002) described a unified framework of boosting algorithms for *classification*. In that framework, the sequential and parallel update schemes become two extremes of a general approach for applying iterative updates to $\boldsymbol{\lambda}$. Following Collins et al. we describe and analyze an algorithm that employs *update templates* to determine specifically which subsets of the coordinates of $\boldsymbol{\lambda}$ may be updated in parallel. This algorithm includes both sequential update and parallel update paradigms as special cases

```
Input: Training set $S = \{(\mathbf{x}_i, y_i) \,|\, \mathbf{x}_i \in \mathbb{R}^n,\ y_i \in \mathbb{R}\}_{i=1}^m$ ; Insensitivity $\varepsilon \in \mathbb{R}_+$
       Update templates $\mathcal{A} \subseteq \mathbb{R}_+^n$ s.t. $\forall \mathbf{a} \in \mathcal{A}\ \ \max_i \left(\sum_{j=1}^n a_j |x_{i,j}|\right) \le 1$
Initialize: $\boldsymbol{\lambda}_1 = (0, 0, \ldots, 0)$
Iterate: For $t = 1, 2, \ldots$
    $\delta_{t,i} = \boldsymbol{\lambda}_t \cdot \mathbf{x}_i - y_i$
    [ if log-loss ]  $q^-_{t,i} = \frac{e^{\delta_{t,i} - \varepsilon}}{1 + e^{\delta_{t,i} - \varepsilon}}$   $q^+_{t,i} = \frac{e^{-\delta_{t,i} - \varepsilon}}{1 + e^{-\delta_{t,i} - \varepsilon}}$   $(1 \le i \le m)$
    [ if exp-loss ]  $q^-_{t,i} = e^{\delta_{t,i}}$   $q^+_{t,i} = e^{-\delta_{t,i}}$   $(1 \le i \le m)$
    $W^-_{t,j} = \sum_{i: x_{i,j} \ge 0} q^-_{t,i}\, x_{i,j} - \sum_{i: x_{i,j} < 0} q^+_{t,i}\, x_{i,j}$   $(1 \le j \le n)$
    $W^+_{t,j} = \sum_{i: x_{i,j} \ge 0} q^+_{t,i}\, x_{i,j} - \sum_{i: x_{i,j} < 0} q^-_{t,i}\, x_{i,j}$   $(1 \le j \le n)$
    $\mathbf{a}_t = \operatorname{argmax}_{\mathbf{a} \in \mathcal{A}} \sum_{j=1}^n a_j \left(\sqrt{W^-_{t,j}} - \sqrt{W^+_{t,j}}\right)^2$
    $\Lambda_{t,j} = \frac{a_{t,j}}{2} \log\left(\frac{W^+_{t,j}}{W^-_{t,j}}\right)$   $(1 \le j \le n)$
    $\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t + \boldsymbol{\Lambda}_t$
```

Figure 3: A log-additive update algorithm for minimizing either the log-loss or the exp-loss.

by setting the templates accordingly, and allows us to discuss and prove the correctness of both paradigms in a unified manner.

In this unified approach, we are required to pre-specify to the algorithm which subsets of the coordinates of $\boldsymbol{\lambda}$ may be updated simultaneously. Formally, the algorithm is given a set of update templates $\mathcal{A}$, where every template $\mathbf{a} \in \mathcal{A}$ is a vector in $\mathbb{R}^n_+$. On every iteration, the algorithm selects a template $\mathbf{a} \in \mathcal{A}$ and updates only those elements λ_j for which a_j is non-zero. We require that every $\mathbf{a} \in \mathcal{A}$ conform with the constraint $\sum_j a_j |x_{i,j}| \le 1$ for all of the instances $\mathbf{x}_i$ in the training set. The purpose of this requirement will become apparent in the proof of Theorem 1. The parallel update is obtained by setting $\mathcal{A}$ to contain the single vector $(\rho, \ldots, \rho)$ where $\rho = (\max_i \|\mathbf{x}_i\|_1)^{-1}$. The sequential update is obtained by setting $\mathcal{A}$ to be the set of vectors $\mathbf{a}_1, \ldots, \mathbf{a}_n$ defined by

$$a_{k,j} = \begin{cases} (\max_i |x_{i,j}|)^{-1} & \text{if } j = k \\ 0 & \text{if } j \ne k \ . \end{cases}$$

The algorithm that we discuss is outlined in Figure 3 and operates as follows: during the process of building $\boldsymbol{\lambda}$, we may encounter two different types of discrepancies: underestimation and overestimation. If the predicted target $\boldsymbol{\lambda} \cdot \mathbf{x}_i$ is less than the correct target y_i, we say that $\boldsymbol{\lambda}$ underestimates y_i and if it is greater we say that $\boldsymbol{\lambda}$ overestimates y_i. For every instance-target pair in the training set, we use a pair of weights $q^-_{t,i}$ and $q^+_{t,i}$ to represent its discrepancies: $q^-_{t,i}$ represents the degree to which y_i is overestimated by $\boldsymbol{\lambda}_t$ and analogously $q^+_{t,i}$ represents the degree to which y_i is underestimated by $\boldsymbol{\lambda}_t$. We then proceed to calculate two weighted sums over each coordinate of the instances: $W^-_{t,j}$ can be thought of as the degree to which $\boldsymbol{\lambda}_{t,j}$ should be decreased in order to compensate for overestimation discrepancies. Symmetrically, $W^+_{t,j}$ represents the degree to which $\boldsymbol{\lambda}_{t,j}$ should be increased. At this point, the algorithm selects the update template $\mathbf{a}_t \in \mathcal{A}$ with respect to which it will apply the update to $\boldsymbol{\lambda}$. $\mathbf{a}_t$ is selected so as to maximize the decrease in loss, according to a criterion that follows directly from Theorem 1 below. In the sequential version of the algorithm, selecting an update template is equivalent to selecting a single base regressor and updating its weight. In this case, the template selection criterion should be viewed as the *weak learning criterion* of the boosting procedure. The algorithm's iteration concludes with an update of $\boldsymbol{\lambda}$. Each element λ_j is updated by half the log ratio between the respective elements of W^+_t and W^-_t, times the scaling factor $a_{t,j}$.

The following theorem states a non-negative lower bound on the decrease in loss on every iteration of the algorithm for the case of the log-loss.

Theorem 1 *Let $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$ be a training set of instance-target pairs where for all i in $1, \ldots, m$, $\mathbf{x}_i \in \mathbb{R}^n$ and $y_i \in \mathbb{R}$. Then using the notation defined in the algorithm outlined in Figure 3, on every iteration t the decrease in the log-loss satisfies,*

$$\mathrm{Loss}(\boldsymbol{\lambda}_t, S) \;-\; \mathrm{Loss}(\boldsymbol{\lambda}_{t+1}, S) \;\geq\; \sum_{j=1}^{n} a_{t,j} \left(\sqrt{W^-_{t,j}} - \sqrt{W^+_{t,j}}\right)^2 \; .$$

Proof Define $\Delta_t(i)$ to be the difference between the loss attained by $\boldsymbol{\lambda}_t$ and that attained by $\boldsymbol{\lambda}_{t+1}$ on an instance-target pair $(\mathbf{x}_i, y_i)$ in the training set, namely

$$\Delta_t(i) \;=\; L_{\log}(\delta_{t,i}) - L_{\log}(\delta_{t+1,i}). \tag{4}$$

Since $\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t + \boldsymbol{\Lambda}_t$ then $\delta_{t+1,i} = \delta_{t,i} + \boldsymbol{\Lambda}_t \cdot \mathbf{x}_i$. Using this equality, and the identity $1/(1+e^{\alpha}) = 1 - 1/(1+e^{-\alpha})$, $\Delta_t(i)$ can be rewritten as

$$\begin{aligned}
\Delta_t(i) \;&=\; -\log\left(\frac{1+e^{\delta_{t+1,i}-\varepsilon}}{1+e^{\delta_{t,i}-\varepsilon}}\right) - \log\left(\frac{1+e^{-\delta_{t+1,i}-\varepsilon}}{1+e^{-\delta_{t,i}-\varepsilon}}\right) \\
&=\; -\log\left(1 - \frac{1}{1+e^{-(\delta_{t,i}-\varepsilon)}} + \frac{e^{\boldsymbol{\Lambda}_t \cdot \mathbf{x}_i}}{1+e^{-(\delta_{t,i}-\varepsilon)}}\right) \\
&\quad\; -\log\left(1 - \frac{1}{1+e^{-(-\delta_{t,i}-\varepsilon)}} + \frac{e^{-\boldsymbol{\Lambda}_t \cdot \mathbf{x}_i}}{1+e^{-(-\delta_{t,i}-\varepsilon)}}\right) .
\end{aligned}$$

We can now plug the definitions of $q^+_{t,i}$ and $q^-_{t,i}$ into this expression to get

$$\Delta_t(i) \;=\; -\log\left(1 - q^-_{t,i}\left(1 - e^{\boldsymbol{\Lambda}_t \cdot \mathbf{x}_i}\right)\right) - \log\left(1 - q^+_{t,i}\left(1 - e^{-\boldsymbol{\Lambda}_t \cdot \mathbf{x}_i}\right)\right) .$$

Next we apply the inequality $-\log(1-\alpha) \geq \alpha$ (which holds wherever $\log(1-\alpha)$ is defined):

$$\Delta_t(i) \;\geq\; q^-_{t,i}\left(1 - e^{\boldsymbol{\Lambda}_t\cdot\mathbf{x}_i}\right) \;+\; q^+_{t,i}\left(1 - e^{-\boldsymbol{\Lambda}_t\cdot\mathbf{x}_i}\right). \tag{5}$$

We rewrite the scalar product $\boldsymbol{\Lambda}_t\cdot\mathbf{x}_i$ in a more convenient form,

$$\begin{aligned}\boldsymbol{\Lambda}_t\cdot\mathbf{x}_i &= \sum_{j=1}^{n}\frac{a_{t,j}}{2}\log\left(W^+_{t,j}/W^-_{t,j}\right)x_{i,j}\\ &= \sum_{j=1}^{n}(a_{t,j}|x_{i,j}|)\ \operatorname{sign}(x_{i,j})\ \log\left(\sqrt{W^+_{t,j}/W^-_{t,j}}\right).\end{aligned} \tag{6}$$

Recall the assumptions made on the vectors in $\mathcal{A}$, namely that $\mathbf{a}_t$ and $\mathbf{x}_i$ comply with $\sum_{j=1}^{n} a_{t,j}|x_{i,j}| \leq 1$ and that $a_{t,j}|x_{i,j}|$ is non-negative. This assumption is used in conjunction with the fact that $(1-e^{\alpha})$ is a concave function and is equal to zero at $\alpha = 0$. We can replace $\boldsymbol{\Lambda}_t\cdot\mathbf{x}_i$ in Eq. (5) with the form given in Eq. (6) and use Jensen's inequality to get,

$$\begin{aligned}\Delta_t(i) &\geq q^-_{t,i}\left(1-e^{\boldsymbol{\Lambda}_t\cdot\mathbf{x}_i}\right) + q^+_{t,i}\left(1-e^{-\boldsymbol{\Lambda}_t\cdot\mathbf{x}_i}\right)\\ &\geq \sum_{j=1}^{n} a_{t,j}q^-_{t,i}|x_{i,j}|\left(1-e^{\operatorname{sign}(x_{i,j})\log\left(\sqrt{W^+_{t,j}/W^-_{t,j}}\right)}\right)\\ &+ \sum_{j=1}^{n} a_{t,j}q^+_{t,i}|x_{i,j}|\left(1-e^{-\operatorname{sign}(x_{i,j})\log\left(\sqrt{W^+_{t,j}/W^-_{t,j}}\right)}\right).\end{aligned}$$

We now rewrite,

$$\begin{aligned}\Delta_t(i) &\geq \sum_{j:x_{i,j}>0} a_{t,j}q^-_{t,i}|x_{i,j}|\left(1-\sqrt{\frac{W^+_{t,j}}{W^-_{t,j}}}\right) + \sum_{j:x_{i,j}<0} a_{t,j}q^-_{t,i}|x_{i,j}|\left(1-\sqrt{\frac{W^-_{t,j}}{W^+_{t,j}}}\right)\\ &+ \sum_{j:x_{i,j}>0} a_{t,j}q^+_{t,i}|x_{i,j}|\left(1-\sqrt{\frac{W^-_{t,j}}{W^+_{t,j}}}\right) + \sum_{j:x_{i,j}<0} a_{t,j}q^+_{t,i}|x_{i,j}|\left(1-\sqrt{\frac{W^+_{t,j}}{W^-_{t,j}}}\right).\end{aligned}$$

Summing $\Delta_t(i)$ over i and using the definition of the q's and W's we finally get that,

$$\begin{aligned}\sum_{i=1}^{m}\Delta_t(i) &\geq \sum_{j=1}^{n} a_{t,j}\left(W^-_{t,j}\left(1-\sqrt{W^+_{t,j}/W^-_{t,j}}\right) + W^+_{t,j}\left(1-\sqrt{W^-_{t,j}/W^+_{t,j}}\right)\right)\\ &= \sum_{j=1}^{n} a_{t,j}\left(\sqrt{W^-_{t,j}}-\sqrt{W^+_{t,j}}\right)^2.\end{aligned}$$

This concludes the proof. ■

Theorem 1 focuses on the log-loss function, but is easily adapted to case of the exp-loss. Note that the only difference between the log-loss and exp-loss cases in the algorithm pseudocode (Figure 3) is in the definitions of the overestimation and underestimation weights q^- and q^+. When our goal is to minimize the exp-loss, we define

$$q^-_{t,i} = e^{\delta_{t,i}} \qquad q^+_{t,i} = e^{-\delta_{t,i}}. \tag{7}$$

To show that Theorem 1 still holds for the exp-loss we modify the definition of $\Delta_t(i)$ from Eq. (4) in accordance to the change in the loss. Specifically, let

$$\begin{aligned}\Delta_t(i) &= L_{\exp}(\delta_{t,i}) - L_{\exp}(\delta_{t+1,i}) \\ &= e^{\delta_{t,i}} - e^{\delta_{t+1,i}} + e^{-\delta_{t,i}} - e^{-\delta_{t+1,i}}.\end{aligned}$$

As before, we plug the definitions of $q^+_{t,i}$ and $q^-_{t,i}$ from Eq. (7) into the above and rewrite Δ_t as,

$$\Delta_t(i) \;=\; q^-_{t,i}\left(1 - e^{\mathbf{\Lambda}_t\cdot\mathbf{x}_i}\right) + q^+_{t,i}\left(1 - e^{-\mathbf{\Lambda}_t\cdot\mathbf{x}_i}\right).$$

Eq. (5) in the proof of Theorem 1 now holds with equality and the rest of the proof proceeds as before. Consequently, we get the same lower bound for the exp-loss as was stated in Theorem 1 for the log-loss.

Similarly, we can redefine q^- and q^+ to minimize the comb-loss. Recall that the comb-loss function is defined by a pair of insensitivity parameters, ε_1 and ε_2. To minimize the comb-loss, we define

$$q^-_{t,i} \;=\; \frac{e^{\delta_{t,i}-\varepsilon_1}}{1+e^{\delta_{t,i}-\varepsilon_1}} \;+\; e^{\delta_{t,i}-\varepsilon_2} \qquad\qquad q^+_{t,i} \;=\; \frac{e^{-\delta_{t,i}-\varepsilon_1}}{1+e^{-\delta_{t,i}-\varepsilon_1}} \;+\; e^{-\delta_{t,i}-\varepsilon_2}.$$

Again, the formal discussion given in this section carries over to the comb-loss case with only minor technical adaptations necessary.

To conclude this section, we note that the log-additive algorithm can be used verbatim in the case of a *weighted* loss. In Section 4 we use this extension to devise a simple regularization scheme. Formally, let $\boldsymbol{\nu} \in \mathbb{R}^m_+$ be a vector of non-negative weights such that ν_i is the weight of the i'th example. The weighted loss is defined as,

$$\mathrm{Loss}(\boldsymbol{\lambda}, \boldsymbol{\nu}, S) \;=\; \sum_{i=1}^{m} \nu_i L(\boldsymbol{\lambda}\cdot\mathbf{x}_i - y_i),$$

where $L(\cdot)$ is any of the loss functions discussed above. The sole change to the algorithm resides in the calculation of the weights $q^+_{t,i}$ and $q^-_{t,i}$ which must now be scaled by ν_i, namely, $q^+_{t,i} \leftarrow \nu_i q^+_{t,i}$ and $q^-_{t,i} \leftarrow \nu_i q^-_{t,i}$. It is easy to verify that Theorem 1 still holds for this extended definition of weighted-loss.

3. Additive Update for Batch Regression

In this section we describe a family of additive batch learning algorithms that advance on each iteration in a direction which is a linear transformation of the gradient of the loss. We term these algorithms *additive update* algorithms. These algorithms bear a resemblance to the log-additive algorithms described in the previous section, as do their proofs of progress. As in the previous section, we first restrict the discussion to the log-loss and then outline the adaptation to the exp-loss at the end of the section.

We again devise a template-based family of updates. This family includes a parallel update which modifies all the elements of $\boldsymbol{\lambda}$ simultaneously and a sequential update which updates a single element of $\boldsymbol{\lambda}$ on each iteration. The parallel update amounts to a gradient descent approach to minimizing the loss. The sequential update applied to an element

```
INPUT: Training set $S = \{(\mathbf{x}_i, y_i) \,|\, \mathbf{x}_i \in \mathbb{R}^n,\ y_i \in \mathbb{R}\}_{i=1}^m$ ; Insensitivity $\varepsilon \in \mathbb{R}_+$
       Update templates $\mathcal{A} \subseteq \mathbb{R}_+^n$ s.t. $\forall \mathbf{a} \in \mathcal{A}\ \sum_{i=1}^m \sum_{j=1}^n a_j x_{i,j}^2 \le 2$
INITIALIZE: $\boldsymbol{\lambda}_1 = (0, 0, \ldots, 0)$
ITERATE: For $t = 1, 2, \ldots$
    $\delta_{t,i} = \boldsymbol{\lambda}_t \cdot \mathbf{x}_i - y_i$
    [ IF LOG-LOSS ]  $q_{t,i}^- = \frac{e^{\delta_{t,i}-\varepsilon}}{1+e^{\delta_{t,i}-\varepsilon}}$   $q_{t,i}^+ = \frac{e^{-\delta_{t,i}-\varepsilon}}{1+e^{-\delta_{t,i}-\varepsilon}}$   $(1 \le i \le m)$
    [ IF EXP-LOSS ]  $q_{t,i}^- = \frac{e^{\delta_{t,i}}}{Z_t}$   $q_{t,i}^+ = \frac{e^{-\delta_{t,i}}}{Z_t}$   $(1 \le i \le m)$
                     where $Z_t = \sum_{i=1}^m \left(e^{\delta_{t,i}} + e^{-\delta_{t,i}} + 2\right)$
    $W_{t,j} = \sum_{i=1}^m (q_{t,i}^+ - q_{t,i}^-)\, x_{i,j}$   $(1 \le j \le n)$
    $\mathbf{a}_t = \operatorname{argmax}_{\mathbf{a} \in \mathcal{A}} \sum_{j=1}^n a_j W_{t,j}^2$
    $\Lambda_{t,j} = a_{t,j} W_{t,j}$   $(1 \le j \le n)$
    $\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t + \boldsymbol{\Lambda}_t$
```

Figure 4: An additive update algorithm for minimizing either the log-loss or the exp-loss.

λ_j is an axis-parallel gradient descent step. We denote the set of update templates by $\mathcal{A}$ and assume that every $\mathbf{a} \in \mathcal{A}$ is a vector in $\mathbb{R}_+^n$. For each $\mathbf{a} \in \mathcal{A}$ we require that $\sum_{i=1}^m \sum_{j=1}^n a_j x_{i,j}^2 \le 2$.

The pseudo-code of the additive update algorithm is given in Figure 4. Intuitively, on each iteration t, the algorithm computes the negative of the gradient with respect to $\boldsymbol{\lambda}_t$, denoted $(W_{t,1}, \ldots, W_{t,n})$. It then selects the update template $\mathbf{a}_t \in \mathcal{A}$ which, as we shortly show in Theorem 2, guarantees a maximal drop in the loss. Finally, $\lambda_{t,j}$ is updated by $a_{t,j} W_{t,j}$.

Theorem 2 *Let $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$ be a training set of instance-target pairs where for all i in $1, \ldots, m$, $\mathbf{x}_i \in \mathbb{R}^n$ and $y_i \in \mathbb{R}$. Then using the notation defined in the algorithm outlined in Figure 4, on every iteration t the decrease in the log-loss, denoted Δ_t, satisfies*

$$\Delta_t \;=\; \text{Loss}(\boldsymbol{\lambda}_t, S) \;-\; \text{Loss}(\boldsymbol{\lambda}_{t+1}, S) \;\ge\; \frac{1}{2} \sum_{j=1}^n a_{t,j} W_{t,j}^2 \quad .$$

Proof We begin by defining a quadratic function $Q : \mathbb{R} \to \mathbb{R}$ which is parameterized by two parameters, $\boldsymbol{\lambda}$ and $\boldsymbol{\Lambda}$. $Q_{\boldsymbol{\lambda},\boldsymbol{\Lambda}}$ will be shown to be an upper bound on the log-loss along the direction $\boldsymbol{\Lambda}$ from $\boldsymbol{\lambda}$. Concretely, $Q_{\boldsymbol{\lambda},\boldsymbol{\Lambda}}$ is defined as,

$$Q_{\boldsymbol{\lambda},\boldsymbol{\Lambda}}(\alpha) \;=\; \mathrm{Loss}(\boldsymbol{\lambda}, S) \;+\; (\nabla\mathrm{Loss}(\boldsymbol{\lambda}, S) \cdot \boldsymbol{\Lambda}) \left(\alpha - \alpha^2/2\right).$$

Formally, we show that for all α, $Q_{\boldsymbol{\lambda}_t,\boldsymbol{\Lambda}_t}(\alpha) \geq \mathrm{Loss}(\boldsymbol{\lambda}_t + \alpha\boldsymbol{\Lambda}_t, S)$ where $\boldsymbol{\Lambda}_t$ is defined as in Figure 4. For convenience, we define $\Gamma(\alpha) = Q_{\boldsymbol{\lambda}_t,\boldsymbol{\Lambda}_t}(\alpha) - \mathrm{Loss}(\boldsymbol{\lambda}_t + \alpha\boldsymbol{\Lambda}_t, S)$ and instead prove that Γ is a non-negative function.

By construction, we get that $\Gamma(0) = 0$. Since the derivative of $Q_{\boldsymbol{\lambda}_t,\boldsymbol{\Lambda}_t}$ at zero is equal to $\nabla\mathrm{Loss}(\boldsymbol{\lambda}_t, S) \cdot \boldsymbol{\Lambda}_t$, we get that the derivative of Γ at zero is also zero. To prove that Γ is a non-negative function it remains to show that Γ is convex and thus $\alpha = 0$ attains its global minimum. To prove convexity it is sufficient to show that the second derivative of Γ (denoted Γ'') is non-negative. Routine calculations yield that,

$$\Gamma''(\alpha) \;=\; -\,\boldsymbol{\Lambda} \cdot \nabla\mathrm{Loss}(\boldsymbol{\lambda}, S) - \boldsymbol{\Lambda}^{\mathrm{T}} H \boldsymbol{\Lambda}, \tag{8}$$

where $H = \sum_{i=1}^{m} L''_{\log}(\boldsymbol{\lambda} + \alpha\boldsymbol{\Lambda})\mathbf{x}_i\mathbf{x}_i^{\mathrm{T}}$ and $L''_{\log}$ is the second derivative of the log-loss function. It is simple to show that this derivative is bounded in $[0, 1/2]$. Plugging the value of H into Eq. (8) we get that,

$$\Gamma''(\alpha) \;\geq\; -\,\boldsymbol{\Lambda} \cdot \nabla\mathrm{Loss}(\boldsymbol{\lambda}, S) - \frac{1}{2}\sum_{i=1}^{m}(\boldsymbol{\Lambda} \cdot \mathbf{x}_i)^2. \tag{9}$$

Note that on the t'th iteration, the j'th element of $\boldsymbol{\Lambda}_t$ equals $a_{t,j}W_{t,j}$ where $W_{t,j} = -\nabla_j\mathrm{Loss}(\boldsymbol{\lambda}_t, S)$. Therefore, we rewrite Eq. (9) as,

$$\begin{aligned}
\Gamma''(\alpha) \quad &\geq \quad \sum_{j=1}^{n} a_{t,j}W_{t,j}^2 - \frac{1}{2}\sum_{i=1}^{m}\left(\sum_{j=1}^{n} a_{t,j}W_{t,j}x_{i,j}\right)^2 \\
&= \quad \sum_{j=1}^{n} a_{t,j}W_{t,j}^2 - \frac{1}{2}\sum_{i=1}^{m}\left(\sum_{j=1}^{n} \sqrt{a_{t,j}}\,W_{t,j}\,\sqrt{a_{t,j}}\,x_{i,j}\right)^2 .
\end{aligned} \tag{10}$$

Using the Cauchy-Schwartz inequality ($\mathbf{u} \cdot \mathbf{v} \leq \|\mathbf{u}\|\|\mathbf{v}\|$) we further bound Γ'' by,

$$\begin{aligned}
\Gamma''(\alpha) \quad &\geq \quad \sum_{j=1}^{n} a_{t,j}W_{t,j}^2 - \frac{1}{2}\sum_{i=1}^{m}\left(\sum_{j=1}^{n} a_{t,j}W_{t,j}^2\right)\left(\sum_{k=1}^{n} a_{t,k}x_{i,k}^2\right) \\
&= \quad \sum_{j=1}^{n} a_{t,j}W_{t,j}^2\left(1 - \frac{1}{2}\sum_{i=1}^{m}\sum_{k=1}^{n} a_{t,k}x_{i,k}^2\right).
\end{aligned} \tag{11}$$

Finally, we use the constraint $\sum_i \sum_{k=1}^{n} a_{t,k}x_{i,k}^2 \leq 2$ which immediately implies that $\Gamma''(\alpha) \geq 0$. Summing up, we have shown that $\mathrm{Loss}(\boldsymbol{\lambda}_t + \alpha\boldsymbol{\Lambda}_t, S)$ is upper bounded by $Q_{\boldsymbol{\lambda}_t,\boldsymbol{\Lambda}_t}(\alpha)$. Therefore, $\mathrm{Loss}(\boldsymbol{\lambda}_{t+1}, S) = \mathrm{Loss}(\boldsymbol{\lambda}_t + \boldsymbol{\Lambda}_t, S) \;\leq\; Q_{\boldsymbol{\lambda}_t,\boldsymbol{\Lambda}_t}(1)$, hence,

$$\Delta_t \;\geq\; \mathrm{Loss}(\boldsymbol{\lambda}_t, S) \;-\; Q_{\boldsymbol{\lambda}_t,\boldsymbol{\Lambda}_t}(1) \;=\; \frac{1}{2}\sum_{j=1}^{n} a_{t,j}W_{t,j}^2.$$

This concludes the proof. ■

To conclude this section, we briefly outline the adaptation of the additive update algorithm to the exp-loss. Recall that in the exp-loss setting, our goal is to minimize,

$$\sum_{i=1}^{m}\left(e^{\delta_i}+e^{-\delta_i}\right) \qquad \text{where} \quad \delta_i=\boldsymbol{\lambda}\cdot\mathbf{x}_i-y_i .$$

Since the gradient of the exp-loss is itself exponential, we cannot hope to minimize the exp-loss by straightforward gradient descent. However, instead of minimizing the exp-loss function over the sample, we can minimize the loss,

$$\log\left(\sum_{i=1}^{m}\left(e^{\delta_i}+e^{-\delta_i}+2\right)\right). \tag{12}$$

Clearly, both functions attain the same (global) minimum. We can now repeat verbatim the proof technique of Theorem 2 using q^- and q^+ as defined for the exp-loss case in Figure 4.

The additive update family of algorithms can accommodate a weighted loss just as log-additive update algorithms do. The algorithm is adapted to cope with weights in the same way that the log-additive algorithm was adapted in the end of Section 2, namely by an appropriate rescaling of the weights q^- and q^+.

4. Regularization

Regularization is a means of controlling the complexity of the regressor being learned. In particular for linear regressors, regularization serves as a soft limit on the magnitude of the elements of $\boldsymbol{\lambda}$ (cf. (Poggio and Girosi, 1990)). The loss functions discussed in the previous sections can also be used as a new form of regularization. Using the log-loss, we can apply the following regularization to the j'th coordinate of $\boldsymbol{\lambda}$,

$$\log\left(1+e^{\lambda_j}\right)+\log\left(1+e^{-\lambda_j}\right).$$

The minimum of the above equation is obtained at $\lambda_j=0$. It is straightforward to show that the regularization term above is bounded from below by $|\lambda_j|$ and from above by $|\lambda_j|+2$. Therefore, summing over all possible indices j, the regularization term on $\boldsymbol{\lambda}$ lies between $\|\boldsymbol{\lambda}\|_1$ and $\|\boldsymbol{\lambda}\|_1+2n$. Thus, this form of regularization can be viewed as a smooth approximation to the ℓ_1 norm of $\boldsymbol{\lambda}$. A similar form of regularization can be imposed using the exp-loss, namely,

$$e^{\lambda_j}+e^{-\lambda_j}.$$

For both losses, the j'th regularization term equals $L(\lambda_j\,;0)$. When the set of base hypotheses is finite, an equivalent way to impose this form of regularization is to introduce a set of pseudo examples $S_{\text{reg}}=\{\mathbf{x}_k,0\}_{k=1}^{n}$ where $\mathbf{x}_k=\mathbf{1}_k$ (the vector with 1 in its k'th position and zeros elsewhere). Let $\nu>0$ be a regularization parameter that governs the relative importance of the regularization term with respect to the empirical loss. Slightly overloading our notation, let $\text{Loss}(\boldsymbol{\lambda},\nu,S)$ denote the regularized empirical loss, defined by,

$$\text{Loss}(\boldsymbol{\lambda},S)\ +\ \nu\,\text{Loss}(\boldsymbol{\lambda},S_{\text{reg}}).$$

As noted in Section 2 and Section 3, both the log-additive and additive update batch algorithms easily accommodate a weighted loss. Therefore, by introducing a set of n pseudo-examples, each of which weighted by ν, we can incorporate regularization into our batch algorithms without any modification to the algorithm core. Concretely, we set the weight of each example in S to 1 and of each pseudo-example in S_{reg} to ν. We can now use either the log-additive or the additive algorithm to minimize the weighted loss.

In practice, we do not need to explicitly add pseudo-examples to our sample in order to incorporate a regularization term into the loss function. A more efficient way of achieving the same effect is to modify our algorithms to behave as if such a pseudo sample was presented to them. For instance, for the log-additive log-loss update (Figure 3) the term $\nu/(1+e^{-\lambda_{t,j}})$ should be added to the definition of $W_{t,j}^{-}$ for every coordinate being updated. Analogously, the term $\nu/(1+e^{\lambda_{t,j}})$ should be added to $W_{t,j}^{+}$. Applying this modification is equivalent to adding pseudo-examples which correspond to the coordinates being updated.

Another useful property of this regularized loss is that it is *strictly* convex. To see that $\mathrm{Loss}(\boldsymbol{\lambda}, \nu, S)$ is strictly convex it suffices to show that its Hessian is positive definite. The Hessian of $\mathrm{Loss}(\boldsymbol{\lambda}, \nu, S)$ can be written as a sum of two matrices $H + H_{\mathrm{reg}}$ where the first is the matrix of second order derivatives of $\mathrm{Loss}(\boldsymbol{\lambda}, S)$ and the second contains the second order derivatives of $\nu\,\mathrm{Loss}(\boldsymbol{\lambda}, S_{\mathrm{reg}})$. Since $\mathrm{Loss}(\boldsymbol{\lambda}, S)$ is the sum of convex losses, H is positive semi-definite. It is simple to verify that the matrix H_{reg} is a diagonal matrix with $H_{i,i} = 2\nu/\left(1+e^{-\lambda_i}\right)\left(1+e^{\lambda_i}\right)$ for the log-loss or $H_{i,i} = \nu\left(e^{-\lambda_i}+e^{\lambda_i}\right)$ for the exp-loss. Clearly, the diagonal elements are strictly positive for both losses for any finite $\boldsymbol{\lambda}$. Therefore, H_{reg} is positive definite and thus $H + H_{\mathrm{reg}}$ is positive definite as well. Furthermore, since the regularization term tends to infinity at least as fast as $\|\boldsymbol{\lambda}\|_1$, the regularized loss has an *attainable* global minimum. In other words, this form of regularization enforces uniqueness of the solution in our loss minimization problem. We denote the unique global minimum of $\mathrm{Loss}(\boldsymbol{\lambda}, \nu, S)$ by $\boldsymbol{\lambda}^{\star}$. We use the uniqueness of $\boldsymbol{\lambda}^{\star}$ in the next section where the convergence of our batch algorithms is discussed.

5. Convergence

In the previous section we have argued that the regularized loss attains a unique minimum at the point denoted $\boldsymbol{\lambda}^{\star}$. In this section we show that the batch algorithms described so far converge to this unique minimizer. For simplicity, we assume that the set of templates $\mathcal{A}$ spans $\mathbb{R}^n$. The following theorem can be tediously generalized to the case where the space spanned by $\mathcal{A}$ is any linear-subspace of $\mathbb{R}^n$ in which case convergence is to the optimal value within this subspace.

Theorem 3 *Assume that the vectors in $\mathcal{A}$ span the entire space $\mathbb{R}^n$. Let $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots, \boldsymbol{\lambda}_t, \ldots$ be the sequence of vectors generated by the log-additive (Figure 3) or the additive (Figure 4) updates, using either of the regularized loss functions discussed in this paper. Then this sequence converges to $\boldsymbol{\lambda}^{\star}$, the global minimizer of the regularized loss $L(\boldsymbol{\lambda}, \nu, S)$.*

Proof Due to the introduction of the regularization term, the loss function is strictly convex and attains its unique minimum at the point denoted $\boldsymbol{\lambda}^{\star}$, as argued in the previous section. In addition, the regularization term guarantees that the entire sequence $\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_t, \ldots$ lies within a compact set C. To see this, note that $\boldsymbol{\lambda}$ is initialized to be the zero vector and therefore the initial regularized loss is

$$\mathrm{Loss}(\mathbf{0}, \nu, S) = \mathrm{Loss}(\mathbf{0}, S) \;+\; \nu\, \mathrm{Loss}(\mathbf{0}, S_{\mathrm{reg}}).$$

Denote the initial loss above by $\mathcal{L}_0$. Since the loss attained by the algorithm on every iteration is non-increasing, the contribution of the regularization term to the total loss certainly does not exceed $\mathcal{L}_0/\nu$. Also, the regularization term for both the exp-loss and the log-loss bounds the ℓ_∞ norm of $\boldsymbol{\lambda}_t$ by

$$\|\boldsymbol{\lambda}_t\|_\infty \;\le\; \mathrm{Loss}(\boldsymbol{\lambda}_t, S_{\mathrm{reg}}) \;\le\; \mathrm{Loss}(\boldsymbol{\lambda}_t, \nu, S)/\nu \;\le\; \mathcal{L}_0/\nu.$$

Therefore, we can define $C = \{\boldsymbol{\lambda} : \|\boldsymbol{\lambda}\|_\infty \le \mathcal{L}_0/\nu\}$ and assert that the sequence $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots$ is contained in C. Next, note that the lower bound on the decrease in loss given in Theorem 1 and Theorem 2 can be thought of as a function of the current regressor $\boldsymbol{\lambda}_t$ and the chosen template $\mathbf{a}_t$. If the bound on the decrease equals zero for all possible $\mathbf{a} \in \mathcal{A}$ then $\boldsymbol{\lambda}_t$ must be equal to $\boldsymbol{\lambda}^\star$. Otherwise, there exists $\mathbf{a} \in \mathcal{A}$ for which the decrease bound is strictly positive. To see this, note that if the decrease bound for the log-additive update is 0 then,

$$\forall j : \quad \left(\sqrt{W_{t,j}^-} - \sqrt{W_{t,j}^+}\right)^2 = 0,$$

which implies that $W_{t,j}^- - W_{t,j}^+ = 0$. Note that $W_{t,j}^- - W_{t,j}^+$ is the j'th partial derivative of the loss function being minimized (log-loss, exp-loss, or comb-loss). Since the regularized loss function is strictly convex, a zero gradient vector is attained only at the optimal point $\boldsymbol{\lambda}^\star$. A similar argument holds for the additive update.

Assume now by contradiction that the sequence of regressors $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots$ does *not* converge to $\boldsymbol{\lambda}^\star$. An immediate consequence of this assumption is that there exists $\gamma > 0$ such that an infinite subsequence of regressors $\boldsymbol{\lambda}_{s_1}, \boldsymbol{\lambda}_{s_2}, \ldots$ remains outside of $B(\boldsymbol{\lambda}^\star, \gamma)$, the open ball of radius γ centered at $\boldsymbol{\lambda}^\star$. The set $C \setminus B(\boldsymbol{\lambda}^\star, \gamma)$ is a compact set and therefore the lower bound from Theorem 1 (or equivalently Theorem 2) attains a minimum value over $C \setminus B(\boldsymbol{\lambda}^\star, \gamma)$ at some point $\tilde{\boldsymbol{\lambda}}$. Denote this minimum by μ. Since $\tilde{\boldsymbol{\lambda}} \notin B(\boldsymbol{\lambda}^\star, \gamma)$ it necessarily follows that $\tilde{\boldsymbol{\lambda}} \neq \boldsymbol{\lambda}^\star$ and therefore μ must be strictly positive. Thus, on each of the iterations $s_1, s_2, \ldots$ the decrease in loss is at least $\mu > 0$. The subsequence $s_1, s_2, \ldots$ is infinite and as a consequence the loss must eventually become negative. We get a contradiction since the loss is a non-negative function. We conclude that the sequence $\boldsymbol{\lambda}_t$ must converge to $\boldsymbol{\lambda}^\star$. ■

6. Back to Classification

To conclude the part of the paper which discusses batch algorithms we would like to briefly draw connections to boosting algorithms for classification. The reader mainly interested in regression problems may skip this section.

The algorithms of previous sections can also be used in classification settings. The log-additive updates simply reduce to the algorithms described in (Collins et al., 2002). The additive update results in a new boosting procedure for classification accompanied with a matching criterion for selecting a base hypothesis. Concretely, in the binary classification setting we receive a training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ where each target y_i is either -1 or $+1$. As in the case of regression, $\mathbf{x}$ is the mapping of an instance into its image under the set of base-classifiers, $\mathbf{x} \mapsto (h_1(\mathbf{x}), \ldots, h_n(\mathbf{x}))$ and the goal is to find a function $f(\mathbf{x})$ that attains a small loss. The function $f(\mathbf{x})$ is a weighted combination of base-hypotheses, $f(\mathbf{x}) = \sum_{j=1}^n \lambda_j h_j(\mathbf{x}) = \boldsymbol{\lambda} \cdot \mathbf{x}$. The skeleton of the additive algorithm for classification is almost the same as the one for regression. In the classification case we define a single (unnormalized) distribution over the examples, setting the weight of the i'th example to,

$$q_{t,i} = e^{-\delta_{t,i}}/Z_t \;\;[\text{ EXP-LOSS }] \quad ; \quad q_{t,i} = \frac{1}{1+e^{-\delta_{t,i}}} \;\;[\text{ LOG-LOSS }],$$

where $\delta_i = y_i \boldsymbol{\lambda}_t \cdot \mathbf{x}_i$ and $Z_t = 1 + \sum_{i=1}^m e^{-\delta_{t,i}}$. On round t we set each variable $W_{t,j}$ to be $W_{t,j} = \sum_{i=1}^n q_{t,i}\, x_{i,j}$. The rest of the algorithm, including the constraint $\sum_{i,j} a_j x_{i,j}^2 \leq 2$, is kept intact. The result is a new boosting-type procedure where base-hypotheses are selected so as to maximize $\sum_j a_j W_{i,j}^2$.

The regularization technique discussed in Section 4 can be used in classification tasks as well. It is worth noting that Schapire et al. (2002) suggested a procedure for incorporating prior knowledge into log-loss boosting which can also be used for regularization. We compare the two regularization techniques in the log-loss case. Using the notation of Section 4, the regularization technique of Schapire et al. can also be described via the introduction of a pseudo-sample. Given a training set $S = \{\mathbf{x}_i, y_i\}_{i=1}^m$ with $y_i \in \{-1, +1\}$ define the pseudo-sample $\bar{S} = \{\mathbf{x}_i, -y_i\}$ and use log-loss boosting to train a classifier whose task is to minimize the loss,

$$(1-\nu)\mathrm{Loss}(\boldsymbol{\lambda}, S) + \nu \mathrm{Loss}(\boldsymbol{\lambda}, \bar{S}),$$

where, as before, ν is a regularization parameter. In this case ν is restricted to the interval $[0, 1/2]$. This construction of a regularization sample implies that even when there exists a strong-hypothesis which attains zero classification error on S the extended sample $S \cup \bar{S}$ is inseparable. If the space spanned by the examples is of a full rank, then this regularization scheme guarantees a unique and attainable global minimizer $\boldsymbol{\lambda}^\star$. However, the two optima due to the two different regularization schemes will be achieved at different points. The regularization scheme presented in this paper penalizes large values of $|\lambda_j|$ whereas Schapire et al. penalize overconfident predictions.

7. Online Regression Algorithms

In this section we describe online regression algorithms for the log-loss defined in Eq. (1). In the previous sections we allowed ourselves to ignore the constant κ which appears in the definition of the log-loss since this did not alter the global minimum of our problem. However, in the online learning setting this constant should not be ignored. Inclusion of κ does not affect the online algorithms themselves as they depend only on the gradient of the loss function, but it will play a role in their analysis.

We follow the notation and techniques presented in (Kivinen and Warmuth, 1997; Cesa-Bianchi, 1999). In online learning settings, we observe a sequence of instance-target pairs, in rounds, one by one. On round t we first receive an instance $\mathbf{x}_t$. Based on the current regressor, $\boldsymbol{\lambda}_t$, we extend a prediction $\boldsymbol{\lambda}_t \cdot \mathbf{x}_t$. We then receive the true target y_t and suffer an instantaneous loss equal to $L_{\log}(\boldsymbol{\lambda}_t \cdot \mathbf{x}_t - y_t)$. Our goal is to suffer a small cumulative loss.

```
Input: Insensitivity parameter $\varepsilon$ ; Upper bound $R$
Initialize:
  [ if GD ]   $\boldsymbol{\lambda}_1 = (0, \ldots, 0)$
  [ if EG ]   $\boldsymbol{\lambda}_1 = (1/n, \ldots, 1/n)$
Iterate: For $t = 1, 2, \ldots$
  Receive an example $\mathbf{x}_t$
  Predict $\boldsymbol{\lambda}_t \cdot \mathbf{x}_t$
  Receive true target $y_t$
  Update:
    $\delta_t = \boldsymbol{\lambda}_t \cdot \mathbf{x}_t$
    $L'_{\log}(\delta_t) = \frac{1}{1+e^{-\delta_t+\varepsilon}} - \frac{1}{1+e^{\delta_t+\varepsilon}}$
    $\beta_t = L'_{\log}(\delta_t)/R$
    [ if GD ]   $\lambda_{t+1,j} = \lambda_{t,j} - \beta_t \, x_{t,j}$                  $(1 \leq j \leq n)$
    [ if EG ]   $\lambda_{t+1,j} = \frac{\lambda_{t,j} e^{-\beta_t \, x_{t,j}}}{\sum_{k=1}^n \lambda_{t,k} e^{-\beta_t \, x_{t,k}}}$   $(1 \leq j \leq n)$
```

Figure 5: The GD and EG algorithms for online regression with the log-loss.

The learning algorithm employs an update rule which modifies its current regressor after each round. We describe and analyze two online regression algorithms for the log-loss that differ in the update rules that they employ. The first is additive in the gradient of the loss and is thus called *Gradient Descent* (GD) while the second is exponential in the gradient of the loss and is analogously called *Exponentiated Gradient* (EG).

The GD algorithm: The pseudo-code of the algorithm is given in Figure 5. Note that the GD algorithm updates its current regressor, $\boldsymbol{\lambda}_t$, by subtracting the gradient of the loss function from it. The GD algorithm assumes an upper bound R on twice the squared norm of all the instances, that is, $2\|\mathbf{x}_t\|_2^2 \leq R$. In the following analysis we give a bound on the cumulative loss attained on any number of rounds. However, rather than bounding the loss per se we bound the cumulative loss *relative* to the cumulative loss suffered by a *fixed* regressor $\boldsymbol{\mu}$. The bound holds for any linear regressor $\boldsymbol{\mu}$ and any number of rounds, hence we get that the GD algorithm is competitive with the optimal (fixed) linear regressor for any number of rounds. Formally, the following theorem states that the cumulative loss attained by the GD algorithm is at most twice the cumulative loss of any fixed linear regressor plus an additive constant.

Theorem 4 *Let $S = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_T, y_T)\}$ be a sequence of instance-target pairs such that $\forall t : 2\|x_t\|_2^2 \leq R$ and let $\boldsymbol{\lambda}_1, ..., \boldsymbol{\lambda}_T$ be the regressors generated by the GD online algorithm*

(Figure 5) on the sequence. Then for any fixed linear regressor $\boldsymbol{\mu} \in \mathbb{R}^n$ we have

$$\sum_{t=1}^{T} L_{log}(\boldsymbol{\lambda}_t \cdot \mathbf{x}_t - y_t) \;\leq\; 2\sum_{t=1}^{T} L_{log}(\boldsymbol{\mu} \cdot \mathbf{x}_t - y_t) \;+\; R\,\|\boldsymbol{\mu}\|_2^2 \;. \tag{13}$$

Note that the statement of the theorem changes if the constant κ is excluded from the definition of the log-loss in Eq. (1), resulting in a looser bound. The proof of the theorem is based on the following lemma that underscores an invariant property of the update rule.

Lemma 5 *Consider the setting of Theorem 4, then for each round t we have*

$$L_{log}(\boldsymbol{\lambda}_t \cdot \mathbf{x}_t - y_t) - 2L_{log}(\boldsymbol{\mu} \cdot \mathbf{x}_t - y_t) \;\leq\; R\,\left(\|\boldsymbol{\lambda}_t - \boldsymbol{\mu}\|_2^2 - \|\boldsymbol{\lambda}_{t+1} - \boldsymbol{\mu}\|_2^2\right) \;. \tag{14}$$

The proof of the lemma is given in Appendix A. Intuitively, the lemma states that if the loss attained by $\boldsymbol{\lambda}_t$ on round t is greater than the loss of a fixed regressor $\boldsymbol{\mu}$, then the algorithm will update $\boldsymbol{\lambda}_t$ such that it gets closer to $\boldsymbol{\mu}$. In contrast, if the loss of $\boldsymbol{\mu}$ is greater than the loss of GD, the algorithm may move its regressor away from $\boldsymbol{\mu}$. With Lemma 5 handy, the proof of Theorem 4 is almost immediate.

Proof of Theorem 4: Summing Eq. (14) for $t = 1, ..., T$ we get

$$\begin{aligned}\sum_{t=1}^{T} L_{\log}(\boldsymbol{\lambda}_t \cdot \mathbf{x}_t - y_t) - 2\sum_{t=1}^{T} L_{\log}(\boldsymbol{\mu} \cdot \mathbf{x}_t - y_t) \;&\leq\; R\,\left(\|\boldsymbol{\lambda}_1 - \boldsymbol{\mu}\|_2^2 - \|\boldsymbol{\lambda}_{T+1} - \boldsymbol{\mu}\|_2^2\right)\\ &\leq\; R\,\|\boldsymbol{\lambda}_1 - \boldsymbol{\mu}\|_2^2\\ &=\; R\,\|\boldsymbol{\mu}\|_2^2,\end{aligned}$$

where in the last equality we use the fact that the initial regressor, $\boldsymbol{\lambda}_1$, is the zero vector. ■

The EG algorithm: The algorithm is described in Figure 5 and works under the assumption that the regressor $\boldsymbol{\lambda}$ is contained in the probability simplex, namely $\boldsymbol{\lambda} \in \mathbb{P}^n$ where $\mathbb{P}^n = \{\boldsymbol{\mu} \;:\; \boldsymbol{\mu} \in \mathbb{R}_+^n,\; \sum_{j=1}^{n} \mu_j = 1\}$. We note in passing that following a construction described in (Kivinen and Warmuth, 1997), it is possible to derive a generalized version of EG in which the elements of $\boldsymbol{\lambda}$ can be either negative or positive, so long as the sum of their absolute values is less than 1. The EG algorithm assumes an upper bound on the squared difference between the maximal and minimal coordinates of the instances it receives, $R \geq (\max_j x_{t,j} - \min_j x_{t,j})^2$. Since EG maintains a regressor from the probability simplex, we measure the cumulative loss of the EG algorithm *relative* to the cumulative loss achieved by any fixed regressor from the probability simplex.

Theorem 6 *Let $S = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_T, y_T)\}$ be a sequence of instance-target pairs such that $\forall t : (\max_j x_{t,j} - \min_j x_{t,j})^2 \leq R$ and let $\boldsymbol{\lambda}_1, ..., \boldsymbol{\lambda}_T$ be the regressors generated by the EG online algorithm (Figure 5) on the sequence. Then, for any fixed regressor $\boldsymbol{\mu} \in \mathbb{P}^n$ we have*

$$\sum_{t=1}^{T} L_{log}(\boldsymbol{\lambda}_t \cdot \mathbf{x}_t - y_t) \;\leq\; \frac{4}{3}\sum_{t=1}^{T} L_{log}(\boldsymbol{\mu} \cdot \mathbf{x}_t - y_t) \;+\; \frac{4R}{3}\; D_{RE}(\boldsymbol{\mu}, \boldsymbol{\lambda}_1) \;, \tag{15}$$

where $D_{RE}(p, q) = \sum_j p_j \log(p_j/q_j)$ is the relative entropy function.

The proof of the theorem is analogous to the proof of Theorem 4 and employs the following relative entropy based progress lemma.

Lemma 7 *Consider the setting of Theorem 6, then for each round t we have*

$$L_{log}(\boldsymbol{\lambda}_t \cdot \mathbf{x}_t - y_t) - \frac{4}{3} L_{log}(\boldsymbol{\mu} \cdot \mathbf{x}_t - y_t) \;\leq\; \frac{4R}{3} \left(D_{RE}(\boldsymbol{\mu}, \boldsymbol{\lambda}_t) - D_{RE}(\boldsymbol{\mu}, \boldsymbol{\lambda}_{t+1}) \right) \; . \tag{16}$$

The proof of the lemma is given in Appendix A.

8. Experiments

In this section we present experimental results that demonstrate different aspects of our algorithms in the light of their formal analysis. In Section 8.1, we start with a synthetic example that underscores the different properties of the log-loss and the exp-loss functions. We then turn to a comparison of the different algorithmic approaches to minimizing these losses. In Section 8.2 we compare the log-additive and additive batch updates by examining how certain properties of the training data influence the rates of convergence of the two updates. Next we demonstrate the different benefits of the sequential and parallel update paradigms. In Section 8.3 we use the sequential form of our update as a boosting procedure which uses regression stumps as base regressors. We compare this boosting technique to the LAD algorithm (Friedman, 2001) on a natural data set. In Section 8.4 we turn our attention to the parallel update paradigm. When using the parallel update, some form of regularization is essential to avoid over-fitting. We therefore demonstrate the effectiveness of the regularization scheme presented in Section 4 and show the effect of the regularization parameter on the generalization ability of our algorithms. More specifically, we learn a kernel-based regressor and compare our log-loss regularization to Support Vector Regression with l_1 regularization. The last two experiments illustrate some properties of our online algorithms and are presented in Section 8.5. In these experiments, we compare the cumulative loss of the online GD algorithm with its theoretical bound given in Theorem 4. We also compare the sensitivity of the online GD and EG regression algorithms to the number of relevant coordinates in the data, demonstrating that EG vastly outperforms GD when the number of relevant coordinates is small.

8.1 Comparison of the Exp-Loss and the Log-Loss

In this section we describe an experiment that underscores the different merits of the log-loss and the exp-loss functions. The end result is that the solution obtained by minimizing the log-loss shares the same asymptotic behavior as the ℓ_1 regression loss ($\sum_i |\delta_i|$). On the other hand, the solution found by minimizing the exp-loss approximately minimizes the l_∞ regression loss on the sample. To exemplify the above properties we created two synthetic data sets and for each one we found the two regressors that minimize the log-loss and the exp-loss respectively. The two data sets and the resulting regressors are depicted in Figure 6. Each of the two data sets was generated by sampling points on the curve of a univariate third degree polynomial, resulting in a sample $S = \{(x_i, y_i)\}$ where $x_i, y_i \in \mathbb{R}$. Then, the target y_i of each point $\mathbf{x}_i$ was contaminated with a small additive noise distributed

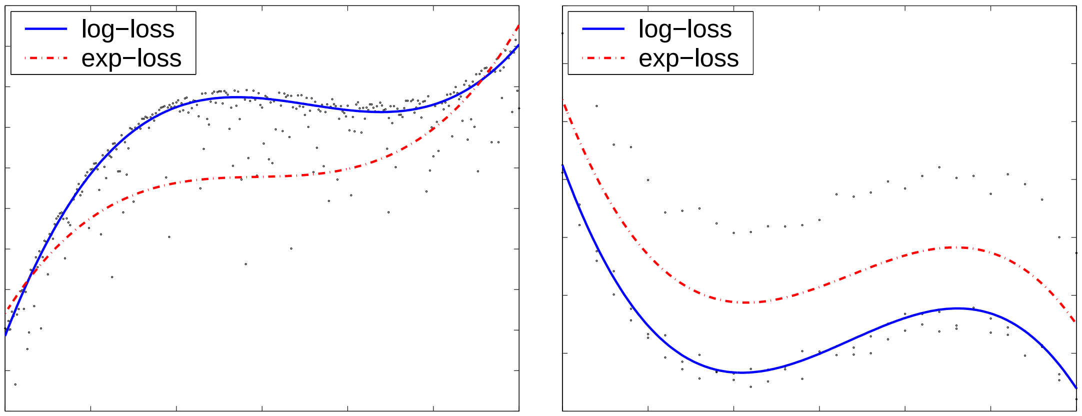


Figure 6: A comparison of log-loss and exp-loss on synthetic data.

normally with a zero mean and a variance of 0.1. In the first experiment, the targets were further contaminated by adding one-sided noise which was generated by subtracting the absolute value of a normal variable with a zero mean and a unit variance (Figure 6, left). Each instance $\mathbf{x}_i$ was expanded by taking powers of x_i, i.e. we performed the mapping $x_i \mapsto (1, x_i, x_i^2, x_i^3)$. This expansion enables us to use our linear algorithms to learn degree three polynomials. It is clear from the figure that the regressor obtained by minimizing the log-loss is very close to the polynomial generating the data, demonstrating the robustness of the log-loss to biased noise. The regressor attained by minimizing the exp-loss, however, approximately minimizes the maximal discrepancy over the entire data set and therefore lies significantly below. The other facet of this behavior is illustrated on the right hand side of Figure 6. In this data set, the additional one sided noise was set to *one* with a probability of 1/3 and otherwise it was set to zero. Thus, about a third of the targets were shifted up by 1. Here, the regressor obtained by minimizing the exp-loss lies between the two groups of points and as such approximately minimizes the ℓ_∞ regression loss on the sample. The regressor found by minimizing the log-loss practically ignores the samples that were shifted by 1 and as such approximately minimizes the ℓ_1 regression loss on the sample.

8.2 A Comparison of the Log-Additive and Additive Updates

In this section we compare the performance of the log-additive update from Figure 3 to that of the additive update from Figure 4. An important difference between the two updates is the type of constraint imposed on the norm of the update templates. In the following, we demonstrate the effect of this difference on the convergence rates of the two update strategies. To make the comparison as simple as possible, we chose the instance space to be $\mathbb{R}^1$. Therefore, the instances are scalars and there is a single update template $a \in \mathbb{R}$. For the log-additive update the constraint on a becomes $a \max_i |x_i| \leq 1$ while for the additive update the constraint is $a \sum_i x_i^2 \leq 2$. To demonstrate the implications of the different norm constraints, we generated two synthetic data sets. The target of each instance was set for both data sets to be equal to the input instance, that is, $y_i = x_i$. Each data set consists of 26 instance-target pairs. For both data sets, we set the value of the first 25 instances to equal 0.01. In the first data set we set the last instance to 0.5 whereas in the second data set

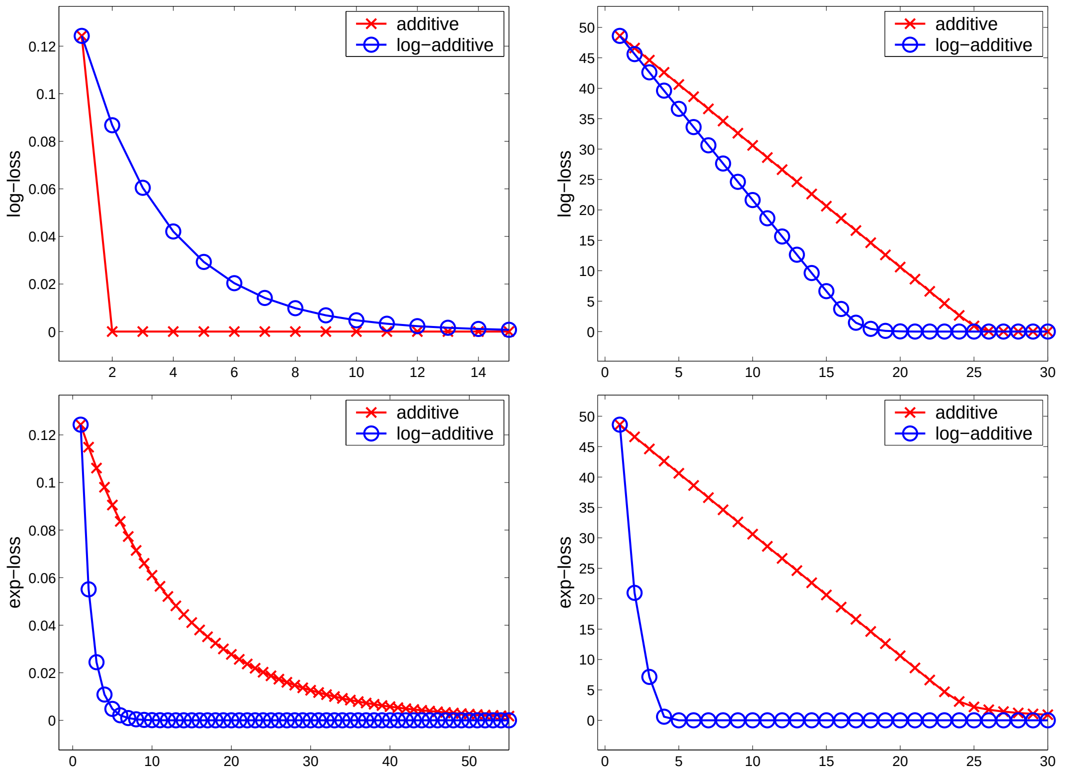


Figure 7: Comparison of the convergence rates of the log-additive and additive updates on two different data sets (see text). The left column corresponds to the first data set and the right column to the second. The top row presents results for the log-loss while the bottom row presents results for the exp-loss.

we set it to 10. Therefore, for the first data set, the constraint on a reduces to $a \leq 2$ when using the log-additive update and to $a \leq 4$ when the additive update is used. In the case of the second data set, the constraint on a becomes $a \leq 0.02$ for the log-additive update, and $a \leq 0.0008$ for the additive one. We would like to note in passing that for the additive update, a typically decreases as the number of examples increases. Hence, the steps that additive update takes are likely to be smaller in large data sets. The end result is slower convergence rates as both the log-additive and the additive updates scale linearly with the value of the template used. Put another way, a small value of a yields an update which changes $\boldsymbol{\lambda}$ rather conservatively. Therefore, in the settings discussed in this section, the additive update should converge faster on the first data set while the log-additive update should converge faster on the second data set. The top row of Figure 7 shows the log-loss obtained on the training set as a function of the number of iterations for the two data sets. It is clear from the graphs that our expectations are met and that the additive update converges faster than the log-additive update on the first data set and slower on the second data set.

Another important difference between the two updates is the construction of q_i^+ and q_i^- when minimizing the exp-loss. Recall that in the case of the log-additive update, the weights of the examples are $q_i^+ = e^{\delta_i}$ and $q_i^- = e^{-\delta_i}$ while for the additive update we further divide these weights by Z. Therefore, when the data set contains examples for which the exp-loss cannot be made small, the value of Z is likely to be rather large. Unlike the log-additive update, the additive form is *sensitive* to scaling of the weights q_i^+ and q_i^-. Thus, whenever Z is large, the resulting normalized weights will be small and therefore the corresponding step sizes taken by the additive update will also be small. The bottom row of Figure 7 reflects this sensitivity of the additive update to scaling. For both data sets described above the log-additive update exhibits much faster convergence than the additive-one.

8.3 Boosting Regression Stumps

The next experiment demonstrates the effectiveness of the log-additive and additive updates in their sequential form, when they are applied as boosting procedures. As in the classic boosting setting, our algorithm has access to an external learning procedure called a base or weak learner. The goal of the boosting algorithm is to construct a highly accurate regressor by combining base regressors obtained from consecutive calls to the base learner. On every boosting iteration the base learner receives the training set along with the weights $q_{t,i}^+$ and $q_{t,i}^-$ generated by the boosting algorithm. The goal of the base learner is to construct a regressor which maximizes the decrease in loss. We denote by $h_t : \mathbb{R}^n \to \mathbb{R}$, the regressor returned by the base learner on round t. We use either the bound in Theorem 1 or the bound in Theorem 2 as the criterion for selecting a base regressor using the log-additive and additive updates respectively. That is, the base learner attempts to maximize the lower bound on the decrease in loss given in Theorem 1 or Theorem 2.

In our experiments, we use *regression stumps* as base regressors. Like decision stumps which are depth-one decision trees, regression stumps are the simplest form of regression trees (cf. Friedman (2001)). Each stump is characterized by two parameters: a feature index parameter, $\ell \in \{1, \ldots, n\}$ and a threshold parameter $\theta \in \mathbb{R}$. The prediction of each stump is either -1 or $+1$ and is defined as $h(\mathbf{x}) = \operatorname{sign}(\theta - x_\ell)$. We now describe the specific base learner we use. Given a training set $S = \{(\mathbf{x}_i, y_i)\}$ of m instance-target pairs, we construct for each feature index $\ell \in \{1, \ldots, n\}$ a set of candidate thresholds. Each set consists of all possible mid-points between two consecutive values of that feature on the training set. Formally, let Θ_ℓ denote the candidate thresholds set for feature ℓ and let $x_{i,\ell}$ denote the ℓth feature of the ith instance in S. Then, the set Θ_ℓ is defined as,

$$\Theta_\ell = \{(x_{i_1,\ell} + x_{i_2,\ell})/2 \,|\, x_{i_1,\ell} < x_{i_2,\ell} \text{ and } \nexists\, r \;\text{ s.t. }\; x_{i_1,\ell} < x_{r,\ell} < x_{i_2,\ell}\} \,. \tag{17}$$

Note that each set Θ_ℓ may contain at most $m-1$ different thresholds and can be *pre-computed* efficiently in time $m\log(m)$ by sorting the training set independently for each feature.

Given the current set of weights, $q_{t,i}^+$ and $q_{t,i}^-$, the base learner constructs a regression stump by choosing a feature index ℓ and a threshold value $\theta \in \Theta_\ell$. This pair is chosen so as to maximize the bound on the decrease in the log-loss as defined in Theorem 1 or in Theorem 2. It is easy to verify that the value of an update template for each base regressor is either $2/m$ in the case of the additive update or 1 in the case of the log-additive update

```
INPUT: Training set $S = \{(\mathbf{x}_i, y_i) \,|\, \mathbf{x}_i \in \mathbb{R}^d,\ y_i \in \mathbb{R}\}_{i=1}^m$ ; Insensitivity $\varepsilon \in \mathbb{R}_+$ ;
   Number of iterations $T$
INITIALIZATION: compute the set of admissible thresholds $\Theta_\ell$ $(\ell = 1, \ldots, d)$
ITERATE:
  For $t = 1, 2, \ldots, T$
      $\delta_{t,i} = \sum_{j=1}^{t-1} \boldsymbol{\lambda}_j h_j(\mathbf{x}_i) - y_i$
      $q^-_{t,i} = \frac{e^{\delta_{t,i}-\varepsilon}}{1+e^{\delta_{t,i}-\varepsilon}}$ , $q^+_{t,i} = \frac{e^{-\delta_{t,i}-\varepsilon}}{1+e^{-\delta_{t,i}-\varepsilon}}$   $(1 \le i \le m)$
      Define:
         [ IF LOG-ADDITIVE ]
         $\Delta(\theta,\ell) = \left(\sqrt{\sum_{i:x_{i,l}>\theta} q^-_{t,i} + \sum_{i:x_{i,l}<\theta} q^+_{t,i}} - \sqrt{\sum_{i:x_{i,l}>\theta} q^+_{t,i} + \sum_{i:x_{i,l}<\theta} q^-_{t,i}}\right)^2$
         [ IF ADDITIVE ]
         $\Delta(\theta,\ell) = \frac{1}{m}\left(\sum_{i:x_{i,l}>\theta} (q^+_{t,i} - q^-_{t,i}) + \sum_{i:x_{i,l}<\theta} (q^-_{t,i} - q^+_{t,i})\right)^2$
      Set $(\theta^\star, \ell^\star) = \operatorname{argmax}_{(\theta,\ell)} \Delta(\theta,\ell)$
      Set $h_t(\mathbf{x}) = \operatorname{sign}(\theta^\star - x_{\ell^\star})$
      Update:
          [ IF LOG-ADDITIVE ] $\lambda_t = \frac{1}{2}\log\left(\frac{\sum_{i:h_t(\mathbf{x}_i)\ge 0} q^+_{t,i} + \sum_{i:h_t(\mathbf{x}_i)<0} q^-_{t,i}}{\sum_{i:h_t(\mathbf{x}_i)\ge 0} q^-_{t,i} + \sum_{i:h_t(\mathbf{x}_i)<0} q^+_{t,i}}\right)$
          [ IF ADDITIVE ]     $\lambda_t = \frac{2}{m}\sum_{i=1}^m (q^+_{t,i} - q^-_{t,i}) h_t(\mathbf{x}_i)$
OUTPUT: $f(\mathbf{x}) = \sum_{t=1}^T \lambda_t h_t(\mathbf{x})$
```

Figure 8: The stumps-based regression algorithm.

since the output of the base regressors is either $+1$ or -1. Hence, given a candidate feature index ℓ and a threshold $\theta \in \Theta_\ell$ the bound on the decrease in loss for the additive update is,

$$\frac{1}{m}\left(\sum_{i\,:\,x_{i,l}>\theta}(q^+_{t,i}-q^-_{t,i})+\sum_{i\,:\,x_{i,l}<\theta}(q^-_{t,i}-q^+_{t,i})\right)^2 , \tag{18}$$

and for the log-additive update the bound is,

$$\left(\sqrt{\sum_{i\,:\,x_{i,l}>\theta}q^-_{t,i}+\sum_{i\,:\,x_{i,l}<\theta}q^+_{t,i}}-\sqrt{\sum_{i\,:\,x_{i,l}>\theta}q^+_{t,i}+\sum_{i\,:\,x_{i,l}<\theta}q^-_{t,i}}\right)^2 . \tag{19}$$

As mentioned above, the base learner evaluates one of the above terms (depending on the update) for each possible ℓ and $\theta \in \Theta_\ell$. It then chooses the pair which maximizes either Eq. (18) or Eq. (19). The pseudocode of the regression learning algorithm using stumps for both the additive and the log-additive updates is given in Figure 8.

We compared the regression algorithm with stumps to an algorithm named Least Absolute Deviation (LAD) due to Friedman (2001). LAD is a boosting-style algorithm which attempts to minimize the hinge-loss by fitting a base hypothesis to the residual error, the approximation error left after applying the combination of base hypotheses found so far. It is not obvious how to conduct a fair comparison between our approach and LAD since the direct objective of our regression learning algorithm is to minimize the log-loss while the goal of LAD is to minimize the hinge-loss. To remove any doubt on the validity of the results, we evaluate both algorithms using the hinge-loss, thus giving a slight advantage to LAD in our empirical evaluation. In addition, we compared the mean squared errors (MSE) of the algorithms.

We ran experiments on two standard data sets for regression: the *Boston housing* data set from the UCI repository and the *body fat* data set (Penrose et al., 1985). To evaluate our results, we used a 10-fold cross validation technique. The plots on the top row of Figure 9 depict the average hinge-loss on the two data sets as a function of the number of sequential iterations while the plots on the bottom row correspond to the mean squared error obtained by the algorithms on the same data sets. The plots underscore a few interesting phenomena. The LAD algorithm appears to be able to decrease the hinge loss and the MSE much faster than our algorithm on both the training data (not shown) and the test data. In no more than 3 iterations, LDA is able to achieve rather low loss. It takes about an order of magnitude more iterations for our algorithm to obtain the same performance LAD achieves after 2 or 3 iterations. This behavior can be partially attributed to the fact that LAD is designed to directly maximize the decrease in loss while our algorithm maximizes a lower bound on the decrease in loss. However, despite its initial performance, LDA seems to "get stuck" rather quickly and the final regressor it obtains has substantially higher loss on both data sets compared to our algorithm, whether it is trained with the log-additive update or the additive one. The improved generalization performance may be attributed to the following behavior that is common to boosting algorithms: as more base regressors are added, the regression error obtained on most of the examples is rather small. Thus, the weights q^+_i and q^-_i for most of the examples are also small and do not contribute too much to further

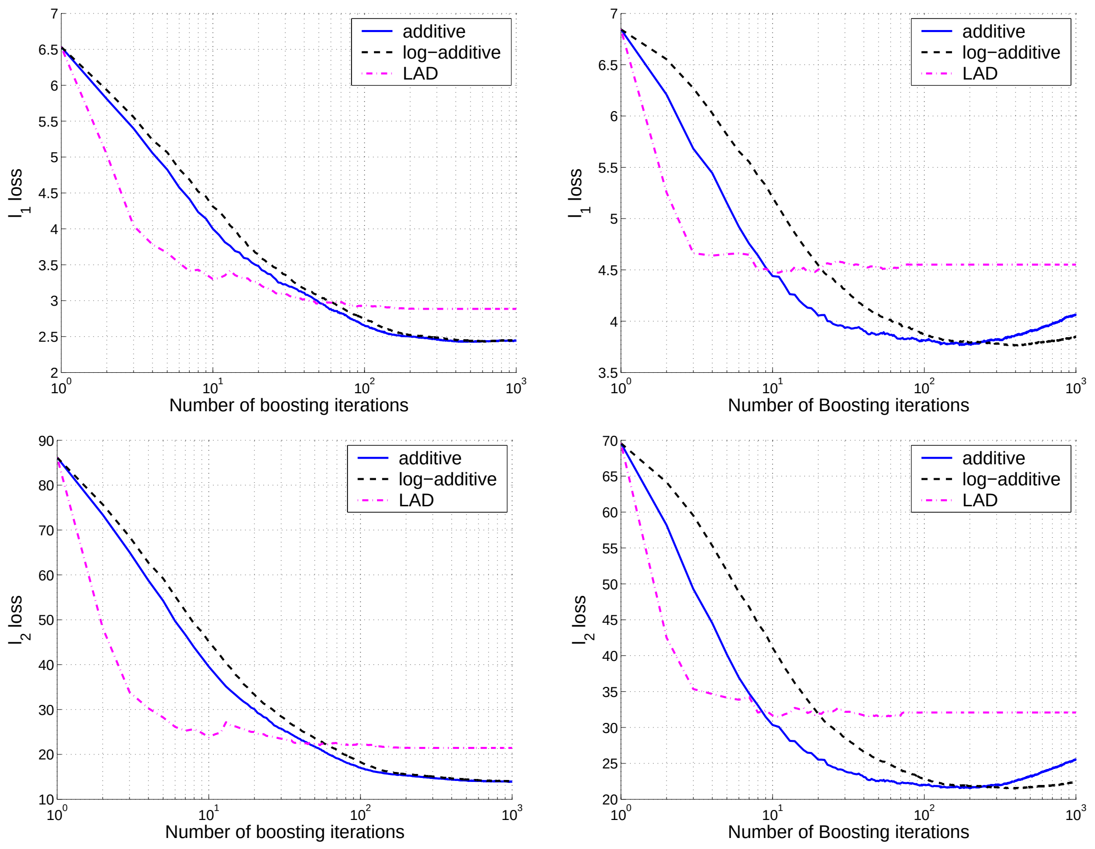


Figure 9: A comparison of the ℓ_1 and MSE losses obtained by the regression algorithm with stumps and the LAD algorithm (see text) on Boston housing data set (left) and body-fat (right) data sets.

decreases in the loss. Thus, even simple regressors such as decision stumps can further reduce the loss on the remaining examples for which the loss is still high. Indeed, we see that some over-fitting takes place when our regression algorithm is run for more than 200 iterations.

Comparing the performance of the additive and the log-additive updates in this experiment, it is apparent that the former seems more effective in reducing the loss but is also more susceptible to over-fitting. The accuracy of each update strategy seems to be problem dependent. We leave further theoretical and empirical research on the generalization properties of the two updates to future research.

8.4 Examining the Effect of Regularization

So far we have focused on experiments which illustrate the different facets of empirical loss minimization using different update schemes. In this section we shift our focus to the effects of the regularization technique discussed in Section 4. The role of regularization is to control the complexity of the final regressor. Indeed, as we demonstrate empirically, proper

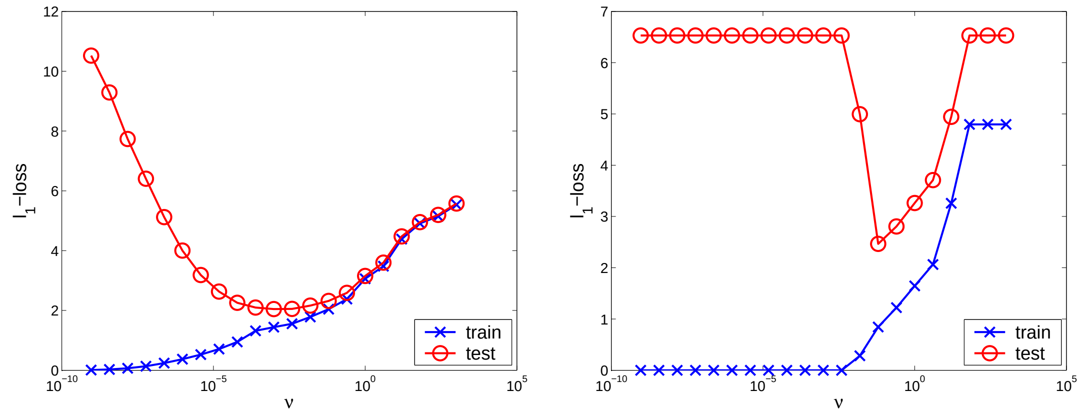


Figure 10: The training and test losses as a function of the regularization parameter (ν) for the log-loss (left) and Support Vector Regression (right).

regularization can ensure that the generalization loss (i.e. the regression loss suffered on test examples) would not greatly exceed the loss obtained on the training set. The feature space we use in this experiment is based on kernel operators. Concretely, the regressors we construct take the form

$$f_{\boldsymbol{\lambda}}(\mathbf{x}) = \sum_{j=1}^{m} \lambda_j k(\mathbf{x}_j, \mathbf{x}),$$

where $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ are the instances in the training set and k is a kernel function. We used the log-additive and additive update algorithms to minimize the following regularized loss,

$$\mathrm{Loss}(\boldsymbol{\lambda}, \nu, S) \;=\; \sum_{i=1}^{m} L_{\log}(f_{\boldsymbol{\lambda}}(\mathbf{x}_i) - y_i\,;\,\varepsilon) \;+\; \nu \sum_{j=1}^{m} L_{\log}(\lambda_j). \tag{20}$$

As illustrated in Figure 1, the log-loss can be interpreted as a smooth approximation to the ε-insensitive hinge loss used by Support Vector Regression (SVR). SVR is a technique for non-linear regression which uses kernel functions. For a thorough review of SVR, see for instance (Smola and Schölkopf, 1998). In our setting, the regularized loss from Eq. (20) can be viewed as a smooth approximation to the hinge-loss with l_1 regularization that is used in Linear Programming Support Vector Regression (LP-SVR), namely,

$$\sum_{i=1}^{m} |f_{\boldsymbol{\lambda}}(\mathbf{x}_i) - y_i|_{\varepsilon} \;+\; \nu \sum_{j=1}^{m} |\lambda_j|. \tag{21}$$

We ran experiments using the *Boston Housing* data set from the UCI Machine Learning Repository. Following Bi and Bennett (2003), we chose to use a Gaussian kernel with $2\sigma^2 = 3.9$. We ran experiments with ϵ set to $0, 1, 2, 3$. The preprocessing we performed consisted of shifting and scaling the input variables to the unit hypercube. Specifically, let $r_j = \min_i x_{i,j}$ and $s_j = \max_i x_{i,j}$, then $x_{i,j}$ was transformed to $(x_{i,j} - r_j)/(s_j - r_j)$. As in the

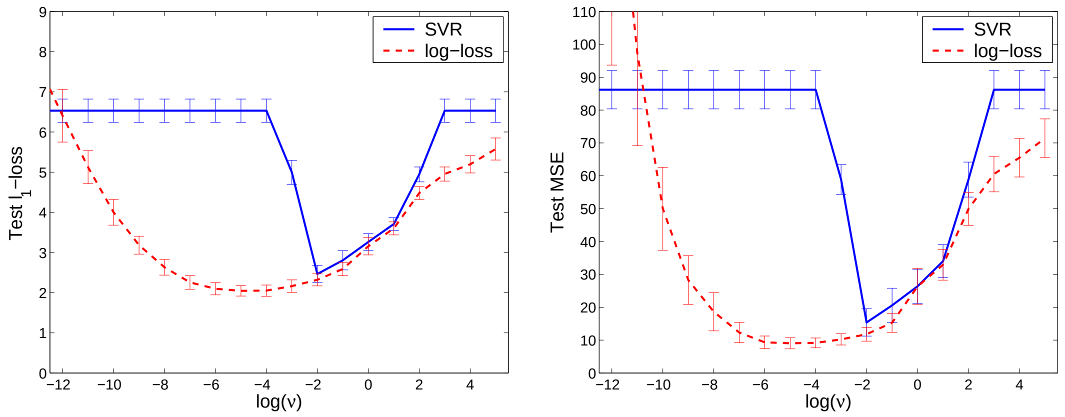


Figure 11: Comparisons of the test losses obtained by the regularized log-loss minimization procedure and by SVR as a function of ν. The losses used for evaluation are the ℓ_1 loss (left) and the mean squared error (right). The standard deviation of over the cross validation fold is depicted as error bars.

previous experiment, we used 10-fold cross validation to evaluate the results and measured the hinge-loss and the mean-squared error (MSE).

The train and test hinge losses for different values of the regularization parameter ν are depicted in Figure 10. As anticipated, the training loss for both algorithms is monotonically increasing in the regularization parameter ν while the difference between the test and training loss is monotonically decreasing in ν. This behavior is typical of regularization techniques. On the left hand side of Figure 11 we directly compare the test error obtained by the two algorithms. The standard deviation over the ten folds is shown using error bars. The MSE of the algorithms is given on the right hand side of Figure 11. As can be seen form the figure, the lowest test loss attained by SVR is very close to the value attained by the log-loss (with a slight advantage to the latter). However, the regressors obtained by the log-loss seem to be less sensitive to the particular choice of ν than the regressors obtained by SVR. Indeed, for ν in $[10^{-5}, 1]$, the discrepancy between the losses of the regressors found by the log-loss is less than 1 while in the same range the losses of SVR can be as much as 3 units apart. This behavior suggests that regression methods which use the smooth log-loss function may give a viable alternative to SVR as they are less sensitive to the particular choice of the regression parameter.

8.5 Online Experiments

We conclude the experiments section with two experiments which use our online algorithms. Theorem 4 states that the GD online algorithm attains a cumulative log-loss which is at most twice the loss of any fixed regressor $\boldsymbol{\mu}$, up to a constant additive factor. For any finite number of online rounds T, the theorem in particular holds for $\boldsymbol{\mu} = \boldsymbol{\lambda}_T^\star$, the regressor which attains the minimal log-loss on the first T examples in the sequence. In practice, however, we have found that GD performs much better than the theoretical guarantee in Theorem 4.

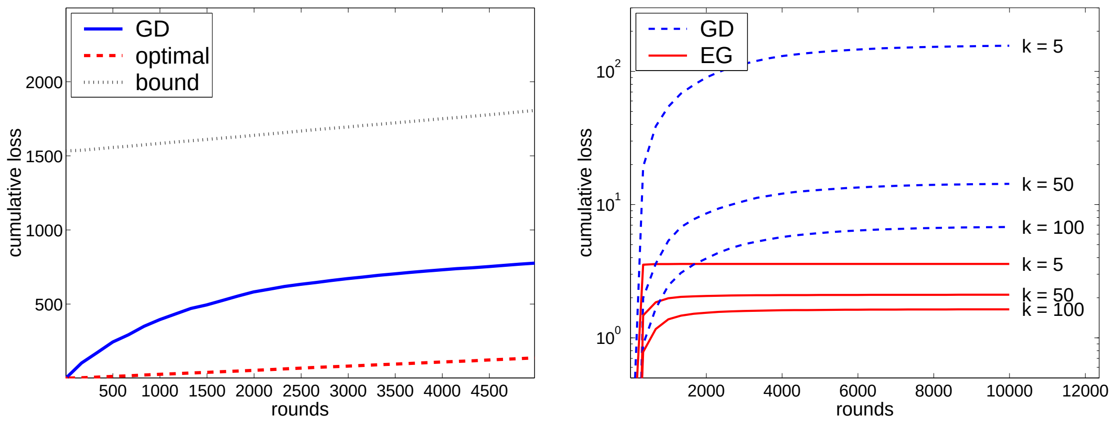


Figure 12: (Left) Cumulative loss of the GD online algorithm compared with the cumulative loss of the optimal fixed regressor and the worst case bound in Theorem 4. (Right) The cumulative loss of EG and GD for different numbers of relevant features.

To demonstrate this, we randomly generated instances and selected the target value for each instance according to a predefined linear function. We added random Gaussian noise to the target values and presented the sequence to the GD algorithm. The cumulative loss of the GD algorithm is depicted on the left hand side of Figure 12, along with the cumulative loss of $\boldsymbol{\lambda}_T^\star$ and the worst-case guarantee attained from Theorem 4 with $\boldsymbol{\mu} = \boldsymbol{\lambda}_T^\star$. Clearly, the cumulative loss of the GD algorithm lies significantly below the worst case bound. Moreover, despite the simplicity of the algorithm, its de facto performance is competitive with the best regressor on each round.

Our last experiment compares the performance of the EG and GD online algorithms. We randomly generated instances in $\{-1,1\}^{1000}$, and generated noise-free targets according to a linear function with only k non-zero components (for $k = 5, 50, 100$). For each value of k, the first k elements of the target function were set to be $1/k$ while the rest of the elements were set to zero. The results are depicted in Figure 12 (right). The cumulative loss curves indicate that the EG algorithm is faster to converge than GD. In fact, for $k = 5$ it takes less than 10 iterations for EG to cease making any significant regression errors. Indeed, simple calculations yield that the excess loss as given in the analyses of the online algorithms is $O(\log(n/k))$ for EG while for GD it is $O(n/k)$. This type of behavior is clearly observed on the right hand side of Figure 12: the higher k becomes the smaller the difference between GD and EG.

9. Discussion

We described a framework for solving regression problems by a symmetrization of margin-based loss functions commonly used in boosting techniques. Our approach naturally lent itself to a shifted and symmetric loss function which is approximately zero in a pre-specified interval and can thus be used as a smooth alternative to the ε-insensitive hinge loss. We presented both batch and online algorithms for solving the resulting regression problems. The updates of the batch algorithms we presented take either a log-additive or an additive form. Our framework also results in a new and very simple to implement regularization scheme for regression and classification boosting algorithms. As a byproduct, we obtained a new additive algorithm for boosting-style classification, which can be used in conjunction with the newly introduced regularization scheme. There are numerous extensions of this work. One of them is the application of Thms. 1 and 2 as splitting criteria for regression-tree learning algorithms. Another interesting direction is the marriage of the loss symmetrization technique with other boosting related techniques such as drifting games (Schapire, 1999; Freund and Opper, 2002).

Acknowledgments

We are in debt to Rob Schapire for making the connection to regularization and for numerous comments. We also would like to thank the anonymous reviewers for their constructive comments. This research was funded by NSF ITR Award 0205594 and by EU PASCAL Network of Excellence.

Appendix A. Technical Proofs

Proof of Lemma 5: Recall that we denote the discrepancy between the target predicted by the online algorithm and the true target by $\delta_t = \boldsymbol{\lambda}_t \cdot \mathbf{x}_t - y_t$. For brevity, let $\bar{\delta}_t$ denote $\boldsymbol{\mu}\cdot\mathbf{x}_t - y_t$. Given the form of the update rule of GD, we can expand $\boldsymbol{\lambda}_{t+1}$ to get the following lower bound on the progress given in Lemma 5,

$$\begin{aligned} & R\left(\|\boldsymbol{\lambda}_t - \boldsymbol{\mu}\|_2^2 - \ \|\boldsymbol{\lambda}_{t+1} - \boldsymbol{\mu}\|_2^2\right) \\ &= \ R\left(2(\boldsymbol{\lambda}_t - \boldsymbol{\mu})\cdot \frac{1}{R} L'_{\log}(\delta_t)\ \mathbf{x}_t - \left\|\frac{1}{R} L'_{\log}(\delta_t)\ \mathbf{x}_t\right\|^2\right) \\ &= \ 2L'_{\log}(\delta_t)(\boldsymbol{\lambda}_t - \boldsymbol{\mu})\cdot\mathbf{x}_t - \frac{1}{R}\left(L'_{\log}(\delta_t)\right)^2 \|\mathbf{x}_t\|_2^2 \\ &\geq \ 2L'_{\log}(\delta_t)(\boldsymbol{\lambda}_t - \boldsymbol{\mu})\cdot\mathbf{x}_t - \frac{1}{2}\left(L'_{\log}(\delta_t)\right)^2, \end{aligned} \tag{22}$$

where $L'_{\log}$ denotes the derivative of $L_{\log}$ and we used the fact that $2\|\mathbf{x}_t\|_2^2 \leq R$ in the last inequality. Since $(\boldsymbol{\lambda}_t - \boldsymbol{\mu})\cdot\mathbf{x}_t = (\boldsymbol{\lambda}_t\cdot\mathbf{x}_t - y_t) + (y_t - \boldsymbol{\mu}\cdot\mathbf{x}_t) = \delta_t - \bar{\delta}_t$, we can rewrite Eq. (22) as

$$2L'_{\log}(\delta_t)(\delta_t - \bar{\delta}_t) - \frac{1}{2}(L'_{\log}(\delta_t))^2 .$$

Therefore, it is sufficient to prove that the following function is non-negative

$$F(\delta,\bar{\delta}) = 2L'_{\log}(\delta)(\delta - \bar{\delta}) - \frac{1}{2}(L'_{\log}(\delta))^2 - L_{\log}(\delta) + 2L_{\log}(\bar{\delta}).$$

The partial derivative of F with respect to $\bar{\delta}$ is

$$\frac{\partial F(\delta,\bar{\delta})}{\partial\bar{\delta}} \ = \ -2L'_{\log}(\delta) + 2L'_{\log}(\bar{\delta}).$$

The only assignment of $\bar{\delta}$ for which this derivative equals 0 is $\bar{\delta} = \delta$. It is straightforward to verify that the second derivative of F with respect to $\bar{\delta}$ is positive since $L_{\log}(\cdot)$ is a convex function. Therefore, fixing δ, F attains its minimum at $\bar{\delta} = \delta$. Put another way, we have shown that for any $\delta, \bar{\delta} \in \mathbb{R}$, $F(\delta, \delta) \le F(\delta, \bar{\delta})$. Denoting $K(\delta) = F(\delta, \delta)$ and simplifying K we get,

$$K(\delta) = -\frac{1}{2}\left(L'_{\log}(\delta)\right)^2 + L_{\log}(\delta).$$

It is left to show that $K(\delta)$ is non-negative. We prove this by showing that for all δ $K(\delta) \ge K(0) = 0$. The derivative of K with respect to δ is $\frac{dK(\delta)}{d\delta} = L'_{\log}(\delta)(1 - L''_{\log}(\delta))$, where $L''_{\log}(\delta)$ is the second derivative of $L_{\log}(\delta)$, namely,

$$\begin{aligned} L''_{\log}(\delta) &= \frac{e^{-\delta+\varepsilon}}{(1+e^{-\delta+\varepsilon})^2} + \frac{e^{\delta+\varepsilon}}{(1+e^{\delta+\varepsilon})^2} \\ &= \left(1 - \frac{1}{1+e^{-\delta+\varepsilon}}\right)\frac{1}{1+e^{-\delta+\varepsilon}} + \left(1 - \frac{1}{1+e^{\delta+\varepsilon}}\right)\frac{1}{1+e^{\delta+\varepsilon}}. \end{aligned}$$

The above equation implies that $L''_{\log}(\delta)$ is the sum of two numbers, each of which is in $[0, 1/4]$ and therefore $0 \le L''_{\log}(\delta) \le \frac{1}{2}$ for all $\delta \in \mathbb{R}$. Therefore, $1 - L''_{\log}(\delta) \ge 0$. We complete the proof by noticing that $L'_{\log}(\delta)$ is a monotonically increasing function and $L'_{\log}(0) = 0$. Therefore, $\delta = 0$ is the single extreme point of $K(\delta)$ and since $K(1) > K(0) = 0$ we get that for all δ $K(\delta) \ge K(0) = 0$.

■

Proof of Lemma 7: Recall that the EG update rule is

$$\lambda_{t+1,j} = \frac{\lambda_{t,j} e^{-\beta_t x_{t,j}}}{\sum_k \lambda_{t,k} e^{-\beta_t x_{t,k}}}, \tag{23}$$

where $\beta_t = L'(\boldsymbol{\lambda}_t \cdot \mathbf{x}_t - y_t)/R$ and $R \ge (\max_j x_{t,j} - \min_j x_{t,j})^2$. First note that, without loss of generality, we can assume that $\min_j x_{t,j} = 0$. This is true since we can replace each instance-target pair $(\mathbf{x}_t, y_t)$ with the pair $(\mathbf{x}_t - (\min_j x_{t,j})\ ,\ y_t - (\min_j x_{t,j}))$. Since we consider only regressors in the probability simplex this transformation does not change discrepancy values. In addition, it is simple to verify that the update given in Eq. (23) is invariant to this shifting.

We now prove the bound in the Lemma, starting with the left-hand side of Eq. (16). Using the definition of the relative entropy we get

$$D_{\mathrm{RE}}(\boldsymbol{\mu}, \boldsymbol{\lambda}_t) - D_{\mathrm{RE}}(\boldsymbol{\mu}, \boldsymbol{\lambda}_{t+1}) = -\beta \boldsymbol{\mu} \cdot \mathbf{x}_t - \log\left(\sum_{j=1}^{n} \lambda_{t,j} e^{-\beta x_{t,j}}\right).$$

We employ the inequality $\alpha^z \le 1 - z(1-\alpha)$ which holds for every $\alpha \ge 0$ and $z \in [0,1]$. Applying this inequality with $\alpha = e^{-\beta\sqrt{R}}$ and $z = x_{t,j}/\sqrt{R}$ yields that

$$e^{-\beta x_{t,j}} \le 1 - \frac{x_{t,j}}{\sqrt{R}}\left(1 - e^{-\beta\sqrt{R}}\right),$$

and summing over j results in the bound,

$$\sum_{j=1}^{n} \lambda_{t,j} e^{-\beta x_{t,j}} \leq 1 - \frac{\boldsymbol{\lambda}_t \cdot \mathbf{x}_t}{\sqrt{R}} \left(1 - e^{-\beta\sqrt{R}}\right).$$

Hence,

$$D_{\text{RE}}(\boldsymbol{\mu}, \boldsymbol{\lambda}_t) - D_{\text{RE}}(\boldsymbol{\mu}, \boldsymbol{\lambda}_{t+1}) \geq -\beta \boldsymbol{\mu} \cdot \mathbf{x}_t - \log\left(1 - \frac{\boldsymbol{\lambda}_t \cdot \mathbf{x}_t}{\sqrt{R}}(1 - e^{-\beta\sqrt{R}})\right).$$

Therefore, to prove Eq. (16) it suffices to show that $F(y, \boldsymbol{\lambda} \cdot \mathbf{x}, \boldsymbol{\mu} \cdot \mathbf{x}) \geq 0$ where

$$F(y,p,r) = \frac{4}{3} R \left(-\beta r - \log\left(1 - \frac{p}{\sqrt{R}}(1 - e^{-\beta\sqrt{R}})\right)\right) + \frac{4}{3} L_{\log}(r-y) - L_{\log}(p-y).$$

We now show that for any $p, r \in \mathbb{R}$ we have $F(y,p,r) \leq F(y,p,p)$. The partial derivative of F with respect to the variable r is

$$\frac{\partial F(y,p,r)}{\partial r} = \frac{4}{3}\left(-R\beta + L_{\log}^{'}(r-y)\right) = \frac{4}{3}\left(-L_{\log}^{'}(p-y) + L_{\log}^{'}(r-y)\right) .$$

The only assignment of r for which this derivative is equal to 0 is $r = p$. The second derivative of F with respect to r is $4/3 L_{\log}^{''}(r-y)$ which is non-negative (see the end of the proof of Lemma 5). Hence, given p, F attains a global minimum at $r = p$. We thus have shown that for any $p, r \in \mathbb{R}$, $F(y,p,r) \leq F(y,p,p)$. $F(y,p,p)$ reduces to

$$\begin{aligned} F(y,p,p) &= \frac{4}{3}\left(-L_{\log}^{'}(p-y)p - R\log\left(1 - \frac{p}{\sqrt{R}}(1 - e^{-L_{\log}^{'}(p-y)/\sqrt{R}})\right)\right) \\ &+ \frac{1}{3} L_{\log}(p-y). \end{aligned}$$

It is left to show that $F(y,p,p)$ is non-negative. Let $z = p/\sqrt{R}$ and $\delta = p - y$. The definition of $\sqrt{R}$ implies that $z \in [0,1]$. Therefore, it is sufficient to prove that the function $G(z,\delta)$ is non-negative for all $z \in [0,1]$ and $\delta \in \mathbb{R}$ where,

$$G(z,\delta) = L_{\log}(\delta) - 4\sqrt{R}\left(z L_{\log}^{'}(\delta) + \sqrt{R}\log\left(1 - z\left(1 - e^{-L_{\log}^{'}(\delta)/\sqrt{R}}\right)\right)\right).$$

We now apply the inequality $\log(1 - z(1 - e^p)) \leq zp + p^2/8$, which holds for $z \in [0,1]$ and $p \in \mathbb{R}$. We get

$$G(z,\delta) \geq L_{\log}(\delta) - \frac{1}{2}\left(L_{\log}^{'}(\delta)\right)^2 .$$

The term lower-bounding $G(z,\delta)$ is equal to $K(\delta)$ where $K(\delta)$ was defined in Lemma 5. In that lemma we proved that $K(\delta) \geq 0$. Therefore, $G(z,\delta) \geq 0$ as required. ∎

References

J. Bi and K.P. Bennett. A geometric approach to support vector regression. In *Neurocomputing, special issue on support vector machines*, volume 55, pages 79–108, September 2003.

N. Cesa-Bianchi. Analysis of two gradient-based algorithms for on-line regression. *Journal of Computer and System Sciences*, 59(3):392–411, 1999.

M. Collins, R.E. Schapire, and Y. Singer. Logistic regression, AdaBoost and Bregman distances. *Machine Learning*, 47(2/3):253–285, 2002.

N. Duffy and D. Helmbold. Leveraging for regression. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory.* ACM, 2000.

Y. Freund and M. Opper. Drifting games and Brownian motion. *Journal of Computer and System Sciences*, 64:113–132, 2002.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28(2):337–374, April 2000.

J.H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, 2001.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning.* Springer, 2001.

P.J. Huber. *Robust Statistics.* John Wiley and Sons, New York, 1981.

J. Kivinen and M.K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–64, January 1997.

G. Lebanon and J. Lafferty. Boosting and maximum likelihood for exponential models. In *Advances in Neural Information Processing Systems 14*, 2001.

K.W. Penrose, A.G. Nelson, and A.G. Fisher. Generalized body composition prediction equation for men using simple measurement techniques. *Medicine and Science in Sports and Exercise*, 17(2):189, 1985.

T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9), 1990.

R.E. Schapire, M. Rochery, M. Rahim, and N. Gupta. Incorporating prior knowledge into boosting. In *Machine Learning: Proceedings of the Nineteenth International Conference*, 2002.

R.E. Schapire. Drifting games. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, 1999.

A. Smola and B. Schölkopf. A tutorial on support vector regression. Technical Report NC2-TR-1998-030, NeuroCOLT2, 1998.

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

Journal of Machine Learning Research 6 (2005) 743–781 Submitted 5/04; Revised 10/04; Published 5/05

Quasi-Geodesic Neural Learning Algorithms Over the Orthogonal Group: A Tutorial

Simone Fiori FIORI@UNIPG.IT
Facoltà di Ingegneria, Università di Perugia
Polo Didattico e Scientifico del Ternano
Località Pentima bassa, 21, I-05100 Terni, Italy

Editor: Yoshua Bengio

Abstract

The aim of this contribution is to present a tutorial on learning algorithms for a single neural layer whose connection matrix belongs to the orthogonal group. The algorithms exploit geodesics appropriately connected as piece-wise approximate integrals of the exact differential learning equation. The considered learning equations essentially arise from the Riemannian-gradient-based optimization theory with deterministic and diffusion-type gradient. The paper aims specifically at reviewing the relevant mathematics (and at presenting it in as much transparent way as possible in order to make it accessible to readers that do not possess a background in differential geometry), at bringing together modern optimization methods on manifolds and at comparing the different algorithms on a common machine learning problem. As a numerical case-study, we consider an application to non-negative independent component analysis, although it should be recognized that Riemannian gradient methods give rise to general-purpose algorithms, by no means limited to ICA-related applications.

Keywords: differential geometry, diffusion-type gradient, Lie groups, non-negative independent component analysis, Riemannian gradient

1. Introduction

From the scientific literature, it is known that a class of learning algorithms for artificial neural networks may be formulated in terms of matrix-type differential equations of network's learnable parameters, which give rise to learning flows on parameters' set. Often, such differential equations are defined over parameter spaces that may be endowed with a specific geometry, such as the general linear group, the compact Stiefel manifold, the orthogonal group, the Grassman manifold and the manifold of FIR filters[1] (Amari, 1998; Fiori, 2001, 2002; Liu et al., 2004; Zhang et al., 2002), that describes the constraints that the network parameters should fulfill and that is worth taking into account properly. From a practical viewpoint, the mentioned differential equations should be integrated (solved) properly through an appropriate numerical integration method that allows us to preserve the underlying structure (up to reasonable precision). This may be viewed as defining a suitable discretization method in the time domain that allows converting a differential learning equation into a discrete-time algorithm.

1. Roughly speaking, the manifold of FIR filters may be regarded as the set of rectangular matrices whose entries are polynomials in a complex-valued variable.

With the present contribution, we aim at studying and illustrating learning algorithms for a single neural layer whose connection matrix belongs to the orthogonal group, that is the group of square orthogonal matrices. As an appropriate approximation of the exact learning flows, the algorithms exploit approximate geodesics suitably glued together, as formerly proposed by Fiori (2002) and Nishimori (1999).

As a case-study, we consider an application to geodesic-learning-based non-negative independent component analysis, as proposed by Plumbley (2003). We present three different learning algorithms that are based on gradient-type optimization of a non-negative independent component analysis criterion over the group of orthogonal matrices. The first two algorithms arise from the direct application of Riemannian gradient optimization without and with geodesic line search, as proposed by Plumbley (2003). The third algorithm relies on a randomized gradient optimization based on diffusion-type Riemannian gradient, as proposed by Liu et al. (2004).

The contribution of the present tutorial may be summarized via the following key points:

- It provides a clear and well-motivated introduction to the mathematics needed to present the geometry-based learning algorithms.
- It clearly states and illustrates the idea that, when we wish to implement a gradient-based algorithm on a computer, it is necessary to discretize the differential learning equations in some suitable way (the ‘gradient flow’ simply cannot be computed exactly in practice).
- In order to effect such discretization, we may not employ standard discretization methods (such as the ones based on Euler forward-backward discretization), that do not work as they stand on curved manifolds. We should therefore resort to more sophisticated integration techniques such as the one based on geodesics.
- In order to improve the numerical performances of the learning algorithm, we might tentatively try adding some stochasticity to the standard gradient (through annealed-MCMC method) and try a geodesic search. It is not guaranteed that the above-mentioned improvement works on a concrete application, therefore it is worth testing them on ICA$^+$ problem. The results on this sides are so far disappointing, because numerical simulations shown that standard Riemannian gradient with no geodesic search nor stochasticity added outperforms the other methods on the considered ICA$^+$ problem.

Although in the machine learning community the presented differential geometry-based learning algorithms have so far been primarily invoked in narrow contexts such as principal/independent component analysis (interested readers might want to consult, for example, Fiori (2001), Celledoni and Fiori (2004) and Plumbley (2003) for a wide review), it should be recognized that differential-geometrical methods provide a general-purpose way of designing learning algorithms, which is profitable in those cases where a learning problem may be formulated mathematically as an optimization problem over a smooth manifold. Some recent advances and applications of these methods are going to be described in the journal special issue whose content is summarized in the editorial by Fiori and Amari (2005).

The paper is organized as follows. The purpose of Section 2 is to briefly recall some concepts from algebra and differential geometry, which are instrumental in the development of the presented learning algorithms. In particular, the concepts of algebraic groups, differential manifolds and Lie groups are recalled, along with the concepts of right-translation, Riemannian gradient and geodesic curves. Then, these results are customized to the case of the orthogonal group of concern in the present paper. Geodesic-based approximations of gradient-type learning differential equations over the orthogonal group are also explained. The Section 2 also presents some notes on the stability of such learning equations as well as on the relationship between the presented learning theory and the well-known natural-gradient theory and information geometry theory. Section 3 presents two deterministic-gradient learning algorithms, one of which is based on the optimization of the learning stepsize via 'geodesic search'. Next, the concept of diffusion-type gradient on manifolds is recalled in details and a third learning algorithm based on it is presented. Such learning algorithm also takes advantage of simulated annealing optimization technique combined with Markov-Chain Monte-Carlo sampling method, which are also recalled in the Section 3, along with some of their salient features. Section 4 deals with non-negative independent component analysis. Its definition and main properties are recalled and the orthogonal-group Riemannian-gradient of the associated cost function is computed. Such computation allows customizing the three generic Riemannian-gradient-based geodesic algorithms to the non-negative independent component analysis case. Also, a fourth projection-based algorithm is presented for numerical comparison purpose. The details of algorithms implementation and the results of computer-based experiments performed on non-negative independent component analysis of gray-level image mixtures are also illustrated in the Section 4. Section 5 concludes the paper.

2. Learning Over the Orthogonal Group: Gradient-Based Differential Systems and Their Integration

The aims of the present section are to recall some basic concepts from differential geometry and to derive the general form of gradient-based learning differential equations over the orthogonal group. We also discuss the fundamental issue of solving numerically such learning differential equations in order to obtain a suitable learning algorithm.

2.1 Basic Differential Geometry Preliminaries

In order to better explain the subsequent issues, it would be beneficial to recall some basic concepts from differential geometry related to the orthogonal group $O(p)$.

An algebraic group (G, m, i, e) is a set G that is endowed with an internal operation $m : G \times G \to G$, usually referred to as group multiplication, an inverse operation $i : G \to G$, and an identity element e with respect to the group multiplication. These objects are related in the following way. For every elements $x, y, z \in G$, it holds that

$$m(x, i(x)) = m(i(x), x) = e,\ m(x, e) = m(e, x) = x \\ \text{and } m(x, m(y, z)) = m(m(x, y), z).$$

Note that, in general, the group multiplication is not commutative, that is, given two elements $x, y \in G$, it holds $m(x, y) \neq m(y, x)$.

Two examples of algebraic groups are $(\mathbb{Z}, +, -, 0)$ and $(Gl(p), \cdot, ^{-1}, \mathbf{I}_p)$. The first group is the set of all integer numbers endowed with the standard addition as group multiplication. In this case, the inverse is the subtraction operation and the identity is the null element. In the second example, we considered the set of non-singular matrices:

$$Gl(p) \overset{\text{def}}{=} \{\mathbf{X} \in \mathbb{R}^{p \times p} | \det(\mathbf{X}) \neq 0\}, \tag{1}$$

endowed with standard matrix multiplication '$\cdot$' as group multiplication operation. In this case, the inverse is the standard matrix inverse and the identity is the identity matrix $\mathbf{I}_p$. It is easy to show that both groups operations/identity satisfy the above general conditions. As a counterexample, the set of the non-negative integer numbers $\mathbb{Z}_0^+ (\equiv \mathbb{N})$ does not form a group under standard addition/subtraction. A remarkable difference between the two groups above is that the first one is a discrete group while the second one is a continuous group.

A useful concept for the economy of the paper is the one of differential manifold. The formal definition of a differential manifold is quite involved, because it requires precise definitions from mathematical topology theory and advanced calculus (Olver, 2003). More practically, a manifold may be essentially regarded as a generalization of curves and surfaces in high-dimensional space, that is endowed with the noticeable property of being locally similar to a flat (Euclidean) space. Let us consider a differential manifold $\mathcal{M}$ and a point ξ on it. From an abstract point of view, ξ is an element of a set $\mathcal{M}$ and does not necessarily possess any particular numerical feature. In order to be able to make computations on manifolds, it is convenient to 'coordinatize' it. To this aim, a neighborhood (open set) $U \subset \mathcal{M}$ is considered, which ξ belongs to, and a coordinate map $\psi : U \to \mathcal{E}$ is defined, where $\mathcal{E}$ denotes a Euclidean space (as for example, $\mathbb{R}^p$ – the set of p-dimensional real-valued vectors – or $\mathbb{R}^{p \times p}$ – the set of the $p \times p$ real-valued matrices –). The function ψ needs to be a one-to-one map (homeomorphism). In this way, we attach a coordinate $x = \psi(\xi)$ to the point ξ. As ψ is a homeomorphism, there is a one-to-one correspondence between a point on a manifold and its corresponding coordinate-point, therefore normally the two concepts may be confused and we may safely speak of a point $x \in \mathcal{M}$. About these concepts, two short notes are in order:

- Borrowing terms from maritime terminology, a triple (ψ, U, p) is termed *coordinate chart* associated to the manifold $\mathcal{M}$. Such notation evidences that the elements ψ and $U \subset \mathcal{M}$ are necessary to coordinatize a point on the manifold and that the coordinate space has dimension p. If the dimension is clear from the context, the indication of p may of course be dispensed of.

- A main concept of differential geometry is that *every geometrical property is independent of the choice of the coordinate system*. As a safety note, it is important to remark that, when we choose to express geometrical relationships in coordinates (as it is implicitly assumed by the above-mentioned 'confusion' between a point $\xi \in \mathcal{M}$ and its coordinate $x \in \mathcal{E}$) we are by no means abandoning this fundamental principle, but we are obeying to the practical need of algorithm implementation on a computer that requires – of necessity – some explicit representation of the quantities of interest.

In general, it is impossible to cover a whole manifold with a unique coordinate map. Therefore, the procedure for coordinatizing a manifold generally consists in covering it with a convenient number of neighborhoods U_k, each of which is endowed with a coordinate map $\psi_k : U_k \to \mathcal{E}_k$, with $\mathcal{E}_k$ being an Euclidean space of dimension p, which, by definition, denotes the dimension of the manifold itself. Technically, the set $\{U_k\}$ is termed a *basis* for the manifold and it does not need to be finite (but it is inherently countable). It is important to note that, in general, the neighborhoods U_k may be overlapping. In this case, the maps ψ_k need to satisfy some constraints termed 'compatibility conditions' which formalize the natural requirement that there should be a one-to-one smooth correspondence between any two different coordinate systems. Technically, if $U_k \cap U_h \neq \emptyset$ then the maps $\psi_k^{-1} \circ \psi_h$ and $\psi_h^{-1} \circ \psi_k$, which are termed 'transition functions' and represent coordinate changes, should be diffeomorphisms, that is, C^∞ functions endowed with C^∞ inverse.

A smooth manifold is by nature a continuous object. A simple example is the unit hyper-sphere $S^p \stackrel{\text{def}}{=} \{\mathbf{x} \in \mathbb{R}^{p+1} | \mathbf{x}^T\mathbf{x} = 1\}$. This is a smooth manifold of dimension p embedded in the Euclidean space $\mathbb{R}^{p+1}$, in fact with only p coordinates we can identify any point on the sphere. Olver (2003) shows how to coordinatize such manifold through for example, the stereographic projection, which requires two coordinate maps applied to two convenient neighborhoods on the sphere.

An interesting object we may think to on a differential manifold $\mathcal{M}$ is a smooth curve $\gamma\colon [a,b] \to \mathcal{M}$. In coordinates,[2] $x = \gamma(t)$ describes a curve on the manifold $\mathcal{M}$ delimited by the endpoints $\gamma(a)$ and $\gamma(b)$. Here, the manifold is supposed to be immersed in a suitable ambient Euclidean space $\mathcal{A}$ of suitable dimension (for instance, the sphere S^p may be though of as immersed in the ambient space $\mathcal{A} = \mathbb{R}^{p+1}$).

Let us now suppose $0 \in [a,b]$ and let us consider a curve γ passing by a given point $x \in \mathcal{M}$, namely $x = \gamma(0)$. The smooth curve admits a tangent vector $\mathbf{v}_x$ at the point x on the manifold, which is defined by

$$\mathbf{v}_x \stackrel{\text{def}}{=} \lim_{t\to 0} \frac{\gamma(t) - \gamma(0)}{t} \in \mathcal{A}.$$

Clearly, the vector $\mathbf{v}_x$ does not belong to the curved manifold $\mathcal{M}$ but is tangent to it in the point x. Let us imagine to consider every possible smooth curve on a manifold of dimension p passing through the point x and to compute the tangent vectors to these curves in the point x. The collection of these vectors span a linear space of dimension p, which is referred to as *tangent space* to the manifold $\mathcal{M}$ at the point x, and is denoted with $T_x\mathcal{M} \subseteq \mathcal{A}$.

As a further safety note, it might deserve to recall that, in differential geometry, the main way to regard for example, tangent spaces and vector fields is based on differential operators (Olver, 2003). This means, for instance, that a tangent vector $\mathbf{v} \in T_x\mathcal{M}$ of some smooth manifold $\mathcal{M}$ is defined in such a way that if $\mathcal{F}$ denotes a smooth functional space then for instance $\mathbf{v} : \mathcal{F} \to \mathbb{R}$, namely $\mathbf{v}(f)$ is a scalar for $f \in \mathcal{F}$. In this paper we chose not to invoke such notation. The reason is that we are interested in a special matrix-type Lie group (the orthogonal group), whose geometry may be conveniently expressed in terms of matrix-type quantities/operations. The theoretical bridge between the differential-operator-based representation and the matrix-based representation is given by the observation that every differential operator in $T_x\mathcal{M}$ may be written as a linear combination of elementary differential operators, that form a basis for the tangent space, through some coefficients. The structure of the tangent space is entirely revealed by the relationships among these coefficients. Therefore, we may choose to represent tangent vectors as algebraic vectors/matrices of coefficients, that is exactly what is implicitly done here.

It is now possible to give the definition of Riemannian manifold, which is a pair $(\mathcal{M}, g)$ formed by a differential manifold $\mathcal{M}$ and an inner product $g_x(\mathbf{v}_x, \mathbf{u}_x)$, locally defined in every point x of the manifold as a bilinear function from $T_x\mathcal{M} \times T_x\mathcal{M}$ to $\mathbb{R}$. It is important to remark that the inner product $g_x(\cdot,\cdot)$ acts on elements from the tangent space to the manifold at some given point, it therefore depends (smoothly) on the point x.

On a Riemannian manifold $(\mathcal{M}, g)$, we can measure the length of a vector $\mathbf{v} \in T_x\mathcal{M}$ as

$$\|\mathbf{v}\| \stackrel{\text{def}}{=} \sqrt{g_x(\mathbf{v},\mathbf{v})}.$$

2. It is worth remarking that a curve may interest different coordinate charts (ψ_k, U_k, p), therefore, it is generally necessary to split a curve in as many branches (or segments) as coordinate charts it bypasses.

Also, a remarkable property of Riemannian manifolds is that we can measure the length of a curve $\gamma : [a,b] \to \mathcal{M}$ on the manifold through the local metric on its tangent spaces. In fact, the length of the curve $\gamma(\cdot)$ is, by definition, $L_\gamma \overset{\text{def}}{=} \int_a^b ds$, where ds is the infinitesimal arc length. From geometry we know that $ds = \|\dot{\gamma}(t)\| dt$, therefore we have

$$L_\gamma = \int_a^b \sqrt{g_{\gamma(t)}(\dot{\gamma}(t), \dot{\gamma}(t))}\, dt. \tag{2}$$

The net result of this argument is that, through a definition of an inner product on the tangent spaces to a Riemannian manifold, we are able to measure the length of paths in the manifold itself, and this *turns the manifold into a metric space*.

A vector field $\mathbf{v}_x$ on manifold $\mathcal{M}$ specifies a vector belonging to the tangent space $T_x\mathcal{M}$ to the manifold at every point x.

With the notions of vector field and curve on a manifold, we may define the important concept of *geodesics*. A geodesic on a smooth manifold may be intuitively looked upon in at least three different ways:

- On a general manifold, the concept of geodesic extends the concept of straight line on a flat space to a curved space. An informal interpretation of this property is that a geodesic is a curve on a manifold that would resemble a straight line in an infinitesimal neighborhood of any of its points. The formal counterpart of this interpretation is rather involved because it requires the notion of covariant derivative of a vector field with respect to another vector field and leads to a second-order differential equation involving the Christoffel structural functions of the manifold (Amari, 1989).

- On a Riemannian manifold, a geodesic among two points is locally defined as the *shortest curve* on the manifold connecting these endpoints. Therefore, once a metric $g(\cdot,\cdot)$ is specified, the equation of the geodesic arises from the minimization of the functional (2) with respect to γ. In general, the obtained equation is difficult to solve in closed form.

- Another intuitive interpretation is based on the observation that a geodesic emanating from a point x on the manifold coincides to the path followed by a particle sliding on the manifold itself with constant scalar speed specified by the norm of the vector $\mathbf{v}_x$. For a manifold embedded in a Euclidean space, this is equivalent to require that the acceleration of the particle is either zero or perpendicular to the tangent space to the manifold in every point.

The concept of geodesic and geodesic equation are recalled here only informally. Appendix A provides a detailed account of these and related concepts just touched here, such as the Christoffel functions (or affine-connection coefficients).

An important vector field often considered in the literature of function optimization over manifolds is the *gradient vector field*. If we consider a smooth function $f : \mathcal{M} \to \mathbb{R}$ and define its gradient $\text{grad}_x^{\mathcal{M}} f$, then a oft-considered differential equation is:

$$\frac{dx}{dt} = \pm\, \text{grad}_x^{\mathcal{M}} f, \tag{3}$$

where the signs $+$ or $-$ denote maximization or minimization of the function f over the manifold. The solution of the above differential equation is referred to as *gradient flow* of f on $\mathcal{M}$.

Formally, the concept of *gradient* on a Riemannian manifold may be defined as follows. Let us consider a Riemannian manifold $(\mathcal{M}, g)$ and, for every point x, the tangent space $T_x\mathcal{M}$. Let us also consider a smooth function $f : \mathcal{M} \to \mathbb{R}$, the standard Euclidean inner product $g^{\mathcal{E}}$ in $T_x\mathcal{M}$ and the Jacobian $\mathrm{grad}_x^{\mathcal{E}} f = \frac{\partial f}{\partial x}$ of the function f with respect to x. The Riemannian gradient $\mathrm{grad}_x^{\mathcal{M}} f$ of the function f over the manifold $\mathcal{M}$ in the point x is uniquely defined by the following two conditions:

- Tangency condition. For every $x \in \mathcal{M}$, $\mathrm{grad}_x^{\mathcal{M}} f \in T_x\mathcal{M}$.
- Compatibility condition. For every $x \in \mathcal{M}$ and every $\mathbf{v} \in T_x\mathcal{M}$, $g_x(\mathrm{grad}_x^{\mathcal{M}} f, \mathbf{v}) = g^{\mathcal{E}}(\mathrm{grad}_x^{\mathcal{E}} f, \mathbf{v})$.

The tangency condition expresses the fact that a gradient vector is always tangent to the base-manifold, while the compatibility condition states that the inner product, under a metric on a manifold, of a gradient vector with any other tangent vector is invariant with the chosen metric. However, note that the gradient *does* depend on the metric. The 'reference' inner product is assumed as the Euclidean inner product that a flat space may be endowed with. For instance, if the base manifold $\mathcal{M}$ has dimension p, then it may be assumed $\mathcal{E} = T_x\mathcal{E} = \mathbb{R}^p$ in every point x and $g^{\mathcal{E}}(\mathbf{u}, \mathbf{v}) = \mathbf{v}^T\mathbf{u}$. It is worth noting that such special metric is *uniform*, in that it does not actually depend on the point x.

In order to facilitate the use of the compatibility condition for gradient computation, it is sometimes useful to introduce the concept of *normal space* of a Riemannian manifold in a given point under a chosen metric $g^{\mathcal{A}}$:

$$N_x\mathcal{M} \overset{\text{def}}{=} \{\mathbf{n} \in \mathcal{A} | g_x^{\mathcal{A}}(\mathbf{n}, \mathbf{v}) = 0 \ , \ \forall \mathbf{v} \in T_x\mathcal{M}\}.$$

It represents the orthogonal complement of the tangent space with respect to an Euclidean ambient space $\mathcal{A}$ that the manifold $\mathcal{M}$ is embedded within.

With the notion of algebraic group and smooth manifold, we may now define a well-known object of differential geometry, that is the *Lie group*. A Lie group conjugates the properties of an algebraic group and of a smooth manifold, as it is a set endowed with both group properties and manifold structure. An example of a Lie group that we are interested in within the paper is the *orthogonal group*:

$$O(p) \overset{\text{def}}{=} \{\mathbf{X} \in \mathbb{R}^{p\times p} | \mathbf{X}^T\mathbf{X} = \mathbf{I}_p\}. \tag{4}$$

It is easy to verify that it is a group (under standard matrix multiplication and inversion) and it is also endowed with the structure of a smooth manifold.

Consequently, we may for instance consider the tangent space T_xG of a Lie group G at the point x. A particular tangent space is T_eG, namely the tangent at identity, which, properly endowed with a binary operator termed *Lie bracket*, has the structure of a *Lie algebra* and is denoted with $\mathfrak{g}$.

An essential peculiarity of the Lie groups (G, m, i, e) is that the whole group may be always brought back to a convenient neighborhood of the identity e and the same holds for every tangent space T_xG, $\forall x \in G$, that may be brought back to the algebra $\mathfrak{g}$. Let us consider, for instance, a curve $\gamma(t) \in G$ passing through the point x, with $t \in [a, b]$ such that $0 \in [a, b]$ and $x = \gamma(0)$. We may define the new curve $\tilde{\gamma}(t) \overset{\text{def}}{=} m(\gamma(t), i(x))$ that enjoys the property $\tilde{\gamma}(0) = e$; conversely, $\gamma(t) = m(\tilde{\gamma}(t), x)$. This operation closely resembles a translation of a curve into a convenient neighborhood of the group identity, so that we can define a special operator referred to as *right translation* as

$$R_x : G \to G \ , R_x(\gamma) \overset{\text{def}}{=} m(\gamma, i(x)).$$

It is clear that every tangent vector $\mathbf{v}_x$ to the curve γ at x is also translated to a tangent vector $\tilde{\mathbf{v}}$ of the curve $\tilde{\gamma}(t)$ by a conveniently defined operator:

$$dR_x : T_xG \to T_eG \ , \tilde{\mathbf{v}} = dR_x(\mathbf{v}),$$

which is commonly referred to as *tangent map* associated to the (right) translation R_x. Such map is invertible and allows us to translate a vector belonging to a tangent space of a group to a vector belonging to its algebra (and vice-versa).[3]

From the above discussion, it is straightforward to see that, if the structure of $\mathfrak{g}$ is known for a group G, it might be convenient to coordinatize a neighborhood of the identity of G through elements of the associated algebra with the help of a conveniently-selected homeomorphism. Such homeomorphism is known in the literature as *exponential map* and is denoted with $\exp : \mathfrak{g} \to G$. It is important to recall that 'exp' is only a symbol and, even for matrix-type Lie groups, *does not necessarily denote matrix exponentiation*.

2.2 Gradient Flows on the Orthogonal Group

As mentioned, the orthogonal group $O(p)$ is a Lie group, therefore it is endowed with a manifold structure. Consequently, we may use the above-recalled instruments in order to define gradient-based learning equations of the kind (3) over $O(p)$ and to approximately solve them.

Some useful facts about the geometrical structure of the orthogonal group $O(p)$ are:

- The standard group multiplication on $O(p)$ is non-commutative (for $p \geq 3$).
- The group $O(p)$ manifold structure has dimension $\frac{p(p-1)}{2}$. In fact, every matrix in $O(p)$ possesses p^2 entries which are constrained by $\frac{p(p+1)}{2}$ orthogonality/normality restrictions.
- The inverse operation $i(\mathbf{X}) = \mathbf{X}^{-1}$ coincides with the transposition, namely $i(\mathbf{X}) = \mathbf{X}^T$.
- The tangent space of the Lie group $O(p)$ has the structure $T_{\mathbf{X}}O(p) = \{\mathbf{V} \in \mathbb{R}^{p\times p} | \mathbf{V}^T\mathbf{X} + \mathbf{X}^T\mathbf{V} = 0_p\}$. This may be proven by differentiating a generic curve $\gamma(t) \in O(p)$ passing by $\mathbf{X}$ for $t = 0$. Every such curve satisfies the orthogonal-group characteristic equation (4), namely $\gamma^T(t)\gamma(t) = \mathbf{I}_p$, therefore, after differentiation, we get $\dot{\gamma}^T(0)\gamma(0) + \gamma^T(0)\dot{\gamma}(0) = \mathbf{0}_p$. By recalling that the tangent space is formed by the velocity vectors $\dot{\gamma}(0)$, the above-mentioned result is readily achieved.
- The Lie algebra associated to the orthogonal group is the set of skew-symmetric matrices $\mathfrak{so}(p) \overset{\text{def}}{=} \{\tilde{\mathbf{V}} \in \mathbb{R}^{p\times p} | \tilde{\mathbf{V}} + \tilde{\mathbf{V}}^T = \mathbf{0}_p\}$. In fact, at the identity ($\mathbf{X} = \mathbf{I}_p$), we have $T_{\mathbf{I}_p}O(p) = \mathfrak{so}(p)$.
- The Lie algebra $\mathfrak{so}(p)$ is a vector space of dimension $\frac{p(p-1)}{2}$.

First, it is necessary to compute the gradient of a function $f : O(p) \to \mathbb{R}$ over the group $O(p)$ in view of computing the geodesic that emanates from a point $\mathbf{X} \in O(p)$ with velocity proportional to $\text{grad}_{\mathbf{X}}^{O(p)} f$. In this derivation, we essentially follow the definition of Riemannian gradient given in Section 2.1.

3. This is the reason for which the Lie algebra of a Lie group is sometimes termed the 'generator' of the group.

Let the manifold $O(p)$ be equipped with the canonical induced metric $g^{O(p)}$, that is $g^{O(p)}_{\mathbf{X}}(\mathbf{U},\mathbf{V}) \stackrel{\text{def}}{=} \text{tr}[\mathbf{U}^T\mathbf{V}]$, for every $\mathbf{X} \in O(p)$ and every $\mathbf{U},\mathbf{V} \in T_{\mathbf{X}}O(p)$. This metric coincides with the standard Euclidean metric $g^{\mathbb{R}^{p\times p}}$ in $\mathbb{R}^{p\times p}$. Having endowed the manifold $O(p)$ with a metric, it is possible to describe completely its normal space, provided the ambient space $\mathcal{A}$ is endowed with the canonical Euclidean metric. In fact, we have

$$N_{\mathbf{X}}O(p) = \{\mathbf{N} = \mathbf{X}\mathbf{S} \in \mathbb{R}^{p\times p} | \text{tr}[\mathbf{N}^T\mathbf{V}] = 0 \ , \ \forall \mathbf{V} \in T_{\mathbf{X}}O(p)\}.$$

The matrix $\mathbf{S}$ should have a particular structure. In fact, the normality condition, in this case, writes $0 = \text{tr}[\mathbf{V}^T(\mathbf{X}\mathbf{S})] = \text{tr}[\mathbf{S}\mathbf{V}^T\mathbf{X}] = \text{tr}[(\mathbf{X}^T\mathbf{V})\mathbf{S}^T]$. The latter expression, thanks to the structure of tangent vectors, is equivalent to $-\text{tr}[(\mathbf{V}^T\mathbf{X})\mathbf{S}^T]$, therefore the normality condition may be equivalently rewritten as $\text{tr}[(\mathbf{V}^T\mathbf{X})(\mathbf{S}-\mathbf{S}^T)] = 0$. In order for this to be true, in the general case, it is necessary and sufficient that $\mathbf{S} = \mathbf{S}^T$. Thus:

$$N_{\mathbf{X}}O(p) = \{\mathbf{X}\mathbf{S} | \mathbf{S}^T = \mathbf{S} \in \mathbb{R}^{p\times p}\}.$$

Let $\text{grad}^{O(p)}_{\mathbf{X}} f$ be the gradient vector of f at $\mathbf{X} \in O(p)$ derived from the metric $g^{O(p)}$. According to the compatibility condition for the Riemannian gradient:

$$g^{\mathbb{R}^{p\times p}}_{\mathbf{X}}(\mathbf{V}, \text{grad}^{\mathbb{R}^{p\times p}}_{\mathbf{X}} f) = g^{O(p)}_{\mathbf{X}}(\mathbf{V}, \text{grad}^{O(p)}_{\mathbf{X}} f),$$

for every tangent vector $\mathbf{V} \in T_{\mathbf{X}}O(p)$, therefore:

$$g^{\mathbb{R}^{p\times p}}_{\mathbf{X}}(\mathbf{V}, \text{grad}^{\mathbb{R}^{p\times p}}_{\mathbf{X}} f - \text{grad}^{O(p)}_{\mathbf{X}} f) = 0,$$

for all $\mathbf{V} \in T_{\mathbf{X}}O(p)$. This implies that the quantity $\text{grad}^{\mathbb{R}^{p\times p}}_{\mathbf{X}} f - \text{grad}^{O(p)}_{\mathbf{X}} f$ belongs to $N_{\mathbf{X}}O(p)$. Explicitly:

$$\text{grad}^{\mathbb{R}^{p\times p}}_{\mathbf{X}} f = \text{grad}^{O(p)}_{\mathbf{X}} f + \mathbf{X}\mathbf{S}. \tag{5}$$

In order to determine the symmetric matrix $\mathbf{S}$, we may exploit the tangency condition on the Riemannian gradient, namely $(\text{grad}^{O(p)}_{\mathbf{X}} f)^T\mathbf{X} + \mathbf{X}^T(\text{grad}^{O(p)}_{\mathbf{X}} f) = \mathbf{0}_p$. Let us first pre-multiply both sides of the equation (5) by $\mathbf{X}^T$, which gives

$$\mathbf{X}^T \text{grad}^{\mathbb{R}^{p\times p}}_{\mathbf{X}} f = \mathbf{X}^T \text{grad}^{O(p)}_{\mathbf{X}} f + \mathbf{S}.$$

The above equation, transposed hand-by-hand, becomes

$$(\text{grad}^{\mathbb{R}^{p\times p}}_{\mathbf{X}} f)^T\mathbf{X} = (\text{grad}^{O(p)}_{\mathbf{X}} f)^T\mathbf{X} + \mathbf{S}.$$

Hand-by-hand summation of the last two equations gives

$$(\text{grad}^{\mathbb{R}^{p\times p}}_{\mathbf{X}} f)^T\mathbf{X} + \mathbf{X}^T(\text{grad}^{\mathbb{R}^{p\times p}}_{\mathbf{X}} f) = 2\mathbf{S},$$

that is:

$$\mathbf{S} = \frac{(\text{grad}^{\mathbb{R}^{p\times p}}_{\mathbf{X}} f)^T\mathbf{X} + \mathbf{X}^T(\text{grad}^{\mathbb{R}^{p\times p}}_{\mathbf{X}} f)}{2}. \tag{6}$$

By plugging the expression (6) into expression (5), we get the form of the Riemannian gradient in the orthogonal group, which is:

$$\text{grad}^{O(p)}_{\mathbf{X}} f = \frac{\text{grad}^{\mathbb{R}^{p\times p}}_{\mathbf{X}} f - \mathbf{X}(\text{grad}^{\mathbb{R}^{p\times p}}_{\mathbf{X}} f)^T\mathbf{X}}{2}.$$

About the expression of the geodesic, as mentioned, in general it is not easy to obtain in closed form. In the present case, with the assumptions considered, the geodesic on $O(p)$ departing from the identity with velocity $\tilde{\mathbf{V}} \in \mathfrak{so}(p)$ has expression $\tilde{\gamma}(t) = \exp(t\tilde{\mathbf{V}})$. (It is immediate to verify that $\tilde{\gamma}(0) = \mathbf{I}_p$ and $\left.\frac{d\tilde{\gamma}(t)}{dt}\right|_{t=0} = \tilde{\mathbf{V}}$.) It might be useful to verify such essential result by the help of the following arguments.

As already recalled in Section 2.1, when a manifold is embedded in a Euclidean space, the second derivative of the geodesic with resepct to the parameter is either zero or perpendicular to the tangent space to the manifold in every point (see Appendix A). Therefore, a geodesic $\tilde{\gamma}(t)$ on the Riemannian manifold $(O(p), g^{O(p)})$ embedded in the Euclidean ambient space $(\mathbb{R}^{p\times p}, g^{\mathbb{R}^{p\times p}})$, departing from the identity $\mathbf{I}_p$, should be such that $\ddot{\tilde{\gamma}}(t) \in N_{\mathbf{I}_p}O(p)$, therefore it should hold:

$$\ddot{\tilde{\gamma}}(t) = \tilde{\gamma}(t)\mathbf{S}(t) \text{ , with } \mathbf{S}^T(t) = \mathbf{S}(t). \tag{7}$$

Also, we known that any geodesic branch belongs entirely to the base manifold, therefore $\tilde{\gamma}^T(t)\tilde{\gamma}(t) = \mathbf{I}_p$. By differentiating two times such expression with respect to the parameter t it is easily gotten:

$$\ddot{\tilde{\gamma}}^T(t)\tilde{\gamma}(t) + 2\dot{\tilde{\gamma}}^T(t)\dot{\tilde{\gamma}}(t) + \tilde{\gamma}^T(t)\ddot{\tilde{\gamma}}(t) = \mathbf{0}_p. \tag{8}$$

By plugging equation (7) into equation (8), we find $\mathbf{S}(t) = -\dot{\tilde{\gamma}}^T(t)\dot{\tilde{\gamma}}(t)$, which leads to the second-order differential equation on the orthogonal group:

$$\ddot{\tilde{\gamma}}(t) = -\tilde{\gamma}(t)(\dot{\tilde{\gamma}}^T(t)\dot{\tilde{\gamma}}(t)),$$

to be solved with initial conditions $\tilde{\gamma}(0) = \mathbf{I}_p$ and $\dot{\tilde{\gamma}}(0) = \tilde{\mathbf{V}}$. It is a straightforward task to verify that the solution to this second-order differential equation is given by the one-parameter curve $\tilde{\gamma}(t) = \exp(t\tilde{\mathbf{V}})$, where $\exp(\cdot)$ denotes matrix exponentiation.

The expression of the geodesic in the position of interest may be made explicit by taking advantage of the Lie-group structure of the orthogonal group endowed with the canonical metric. In fact, let us consider the pair $\mathbf{X} \in O(p)$ and $\text{grad}_{\mathbf{X}}^{O(p)} f \in T_{\mathbf{X}}O(p)$ as well as the geodesic $\gamma(t)$ that emanates from $\mathbf{X}$ with velocity $\mathbf{V}$ proportional to $\text{grad}_{\mathbf{X}}^{O(p)} f$, and let us suppose for simplicity that $\gamma(0) = \mathbf{X}$. Let us now consider the right-translated curve $\tilde{\gamma}(t) = \gamma(t)\mathbf{X}^T$. The new curve enjoys the following properties:

1. It is such that $\tilde{\gamma}(0) = \mathbf{I}_p$, therefore it passes through the identity of the group $O(p)$.
2. The tangent vector $\mathbf{V}$ to the curve at $\mathbf{X}$ is 'transported' into the tangent vector:
$$\tilde{\mathbf{V}} = \mathbf{V}\mathbf{X}^T, \tag{9}$$
at the identity, so $\tilde{\mathbf{V}} \in \mathfrak{so}(p)$.
3. As the right-translation is an isometry, the curve $\tilde{\gamma}(t)$ is still a geodesic departing from the identity matrix with velocity proportional to $\tilde{\mathbf{V}} = (\text{grad}_{\mathbf{X}}^{O(p)} f)\mathbf{X}^T$.

From these observations, we readily obtain the geodesic in the position of interest $\mathbf{X} \in O(p)$, namely $\gamma(t) = \exp(t\tilde{\mathbf{V}})\mathbf{X}$.

As this is an issue of prime importance, we deem it appropriate to verify that the curve $\gamma(t)$ just defined belongs to the orthogonal group at any time. This may be proven by computing the quantity $\gamma^T(t)\gamma(t)$ and taking into account the identity $\exp^T(t\tilde{\mathbf{V}}) = \exp(-t\tilde{\mathbf{V}})$. Then we have

$$\gamma^T(t)\gamma(t) = \mathbf{X}^T\exp^T(t\tilde{\mathbf{V}})\exp(t\tilde{\mathbf{V}})\mathbf{X} = \mathbf{X}^T\exp(-t\tilde{\mathbf{V}})\exp(t\tilde{\mathbf{V}})\mathbf{X} = \mathbf{X}^T\mathbf{X} = \mathbf{I}_p.$$

2.3 Comments on Stability and the Relationship with Natural Gradient Theory

Some comments on the questions of the stability of gradient-based learning algorithms on the orthogonal group and on the relationship of Riemannian gradient-based learning algorithms on the orthogonal group with the well-known 'natural' gradient-based optimization theory are in order.

When applied to the manifold $\mathcal{M} = O(p)$, the general gradient-based learning equation (3) has the inherent property of keeping the connection matrix $\mathbf{X}$ within the group $O(p)$ at any time. It is very important to note that the discrete-time version of this learning equation, described in Section 3, also enjoys this noticeable property. When for example, learning algorithms based on the manifold $Gl(p)$, defined in equation (1), are dealt with, one of the theoretical efforts required to prove their stability consists in showing that there exists a compact sub-manifold that is an attractor for the learning system. The above observations reveal that the problem of the existence of an orthogonal-group-attractor for discrete-time learning systems based on the orthogonal Lie group does not arise when a proper integration algorithm is exploited. Moreover, as opposed to the Euclidean space $\mathbb{R}^p$ and the general-linear group $Gl(p)$, the orthogonal group $O(p)$ is a compact space. This means that no diverging trajectories exist for the learning system (3) or its discrete-time counterpart. Such effect may be easily recognized in the two-dimensional ($p = 2$) case, through the parameterization $\psi^{-1} : [-\pi, \pi[\to SO(2)$:

$$\mathbf{X} = \begin{bmatrix} \cos\beta & -\sin\beta \\ \sin\beta & \cos\beta \end{bmatrix}. \tag{10}$$

It is worth noting that $\det(\mathbf{X}) = 1$, while, in general, the determinant of an orthonormal matrix may be either -1 or $+1$, in fact $1 = \det(\mathbf{X}^T\mathbf{X}) = \det^2(\mathbf{X})$ for $\mathbf{X} \in O(p)$. This means that the above parameterization spans one of the two components of the orthogonal group termed *special orthogonal* group and denoted by $SO(p)$. (In the above notation, we easily recognize a coordinate chart $(\psi, SO(2), 1)$ associated to $O(2)$.) Now, by singling out the columns of the matrix $\mathbf{X} = [\mathbf{x}_1\ \mathbf{x}_2]$, we easily see that $\|\mathbf{x}_1\| = \|\mathbf{x}_2\| = 1$, which proves the space $SO(2)$ is compact. The same reasoning may be repeated for the remaining component of $O(2)$.

In its general formulation, the widely-known 'natural gradient' theory for learning may be summarized as follows. The base-manifold for learning is the group of non-singular matrices $Gl(p)$ that is endowed with a metrics based on the Fisher metric tensor which, in turn, derives from a truncated expansion of the Kullback-Leibler informational divergence (KLD) (Amari, 1998). The latter choice derives from the possibility – offered by the KLD – to induce a metrics in the abstract space of neural networks having same topology but different connection parameters, which is referred to as *neural manifold*.

In the independent component analysis case, a special structure was envisaged by Yang and Amari (1997) for the natural gradient by imposing a Riemannian structure on the Lie group of non-singular matrices $Gl(p)$. We believe it could be useful to briefly recall this intuition here by using the language of Lie groups recalled in Section 2.1. First, the tangent space at identity $\mathbf{I}_p$ to $Gl(p)$ is denoted by $\mathfrak{gl}(p)$, as usual. Such Lie algebra may be endowed with a scalar product $g_{\mathbf{I}_p}^{Gl(p)}(\cdot,\cdot) : \mathfrak{gl}(p) \times \mathfrak{gl}(p) \to \mathbb{R}$. As there is no reason to weight in a different way the components of the matrices in $\mathfrak{gl}(p)$, it is assumed $g_{\mathbf{I}_p}^{Gl(p)}(\tilde{\mathbf{U}}, \tilde{\mathbf{V}}) \stackrel{\text{def}}{=} \text{tr}[\tilde{\mathbf{U}}^T\tilde{\mathbf{V}}]$. The question is now how to define the scalar product in a generic tangent space $T_{\mathbf{X}}Gl(p)$, with $\mathbf{X} \in Gl(p)$. Let us consider, to this purpose, a curve $\gamma(t) \in Gl(p)$ passing by the point $\mathbf{X}$ at $t = 0$, namely $\gamma(0) = \mathbf{X}$. This curve may always be translated into a neighborhood of the identity of the group by the left-translation $\tilde{\gamma}(t) \stackrel{\text{def}}{=} \mathbf{X}^{-1}\gamma(t)$, in fact, the inverse $\mathbf{X}^{-1}$ surely exists because $Gl(p)$ is the set of all invertible $p \times p$ matrices by definition and now $\tilde{\gamma}(0) = \mathbf{I}_p$. Therefore, if $\mathbf{V} \in T_{\mathbf{X}}Gl(p)$ denotes the tangent vector to the curve $\gamma(t)$ at $t = 0$ and $\tilde{\mathbf{V}} \in \mathfrak{gl}(p)$ denotes the tangent vector to the curve $\tilde{\gamma}(t)$ at $t = 0$, they are related by the corresponding tangent map $\mathbf{V} \to \tilde{\mathbf{V}} = \mathbf{X}^{-1}\mathbf{V}$. This observation may be exploited to define an inner product on the tangent spaces of $Gl(p)$ by *imposing* the Riemannian-structure invariance property:

$$g_{\mathbf{X}}^{Gl(p)}(\mathbf{U},\mathbf{V}) \stackrel{\text{def}}{=} g_{\mathbf{I}_p}^{Gl(p)}(\mathbf{X}^{-1}\mathbf{U},\mathbf{X}^{-1}\mathbf{V}) = \text{tr}[\mathbf{U}^T(\mathbf{X}^T)^{-1}\mathbf{X}^{-1}\mathbf{V}].$$

Having defined a general (non-uniform) metric in the tangent spaces to $Gl(p)$, we may now compute the Riemannian (natural) gradient on it, by invoking the tangency and compatibility conditions as stated in Section 2.1. Actually, the tangency condition does not provide any constraint in the present case, because every $T_{\mathbf{X}}Gl(p)$ is ultimately isomorphic to $\mathbb{R}^{p\times p}$. The compatibility condition, instead, writes, for a smooth function $f : Gl(p) \to \mathbb{R}$:

$$g_{\mathbf{X}}^{Gl(p)}(\text{grad}_{\mathbf{X}}^{Gl(p)} f, \mathbf{V}) = \text{tr}\left[\left(\frac{\partial f}{\partial \mathbf{X}}\right)^T \mathbf{V}\right] \ , \ \forall \mathbf{V} \in T_{\mathbf{X}}Gl(p).$$

This condition implies:

$$\begin{aligned} & \text{tr}\left[\left\{(\text{grad}_{\mathbf{X}}^{Gl(p)} f)^T(\mathbf{X}^T)^{-1}\mathbf{X}^{-1} - \left(\frac{\partial f}{\partial \mathbf{X}}\right)^T\right\}\mathbf{V}\right] = 0 \ , \ \forall \mathbf{V} \in T_{\mathbf{X}}Gl(p) \\ \Rightarrow \quad & (\text{grad}_{\mathbf{X}}^{Gl(p)} f)^T(\mathbf{X}^T)^{-1}\mathbf{X}^{-1} = \left(\frac{\partial f}{\partial \mathbf{X}}\right)^T \\ \Rightarrow \quad & \text{grad}_{\mathbf{X}}^{Gl(p)} f = (\mathbf{X}\mathbf{X}^T)\frac{\partial f}{\partial \mathbf{X}}. \end{aligned}$$

Of course, a different form for the natural gradient may be obtained by choosing the right-translation $\tilde{\gamma}(t) \stackrel{\text{def}}{=} \gamma(t)\mathbf{X}^{-1}$ as a basis for invariance, as done for example, by Yang and Amari (1997). The 'natural gradient' theory for $Gl(p)$ and the Riemannian-gradient-theory for the group $O(p)$ are thus somewhat unrelated, even if ultimately the 'natural gradient' is a Riemannian gradient on the group $Gl(p)$ arising from a specific metric. Some further details on the optimization problem over the general linear group (about for example, using the exponential map on $Gl(p)$) have been presented by Akuzawa (2001).

Another interesting comparison is with the information-geometry theory for learning.[4] In the spirit of information geometry, the natural gradient works on a manifold of parameterized likelihood. Now, in two dimensions, the Riemannian geometry of the orthogonal group, defined by the parameterization (10) above, may be clearly related to the information geometry of the binomial distribution defined by the variables r, q such that $r + q = 1$, via the transform $r = \cos^2(\beta)$, $q = \sin^2(\beta)$. Whether such link exists in any dimension ($p \geq 3$) is not known to the author and would be worth investigating in future works. The same holds for the relationship with second order (Newton) method, which is known for the natural gradient (see, for example, Park et al. (2000) and references therein) but whose relationship with general Riemannian gradient theory is to be elucidated.

4. This interesting connection was suggested by a reviewer.

3. Learning Over the Orthogonal Group: Three Algorithms

In order to numerically integrate a continuous learning differential equation on a manifold, a proper discretization method should be exploited. On a flat space, a possible discretization method is line approximation based on Euler's or trapezoidal technique (or some more sophisticated techniques such as the Runge-Kutta method). However, if applied to differential equations based on curved manifolds, such ordinary discretization methods produce updating rules that do not satisfy the manifold constraints. Following the general differential-geometric knowledge, two possible ways to tackle the problem are:

- The projection method. It consists in projecting the updated value to the manifold after each iteration step. More formally, this method consists in embedding the manifold $\mathcal{M}$ of interest into a Euclidean space of proper dimension $\mathcal{A}$ and to discretize the differential equation whose variable is regarded as belonging to $\mathcal{A}$ through any suitable ordinary method. Then, in each iteration, the newly found approximated solution is projected back to the manifold through a suitable *projector* $\Pi : \mathcal{A} \to \mathcal{M}$. The next iteration starts from the projected putative solution.

- The geodesic method. The principle behind the geodesic method is to replace the line approximation to the original differential equation by the geodesic approximation in the manifold. From a geometrical point of view, this seems a natural approximation because a geodesic on a manifold is a counterpart of a line in the Euclidean space. Furthermore, a geodesic on a Riemannian manifold is a length-minimizing curve between two points, which looks quite appealing if we regard an optimization process as connecting an initial solution to a stationary point of a criterion function through the shortest path.

The viewpoint adopted in the present contribution is that the geodesic-based approach is the most natural one from a geometric perspective and the most capable of future extensions to different base-manifolds. The projection method will also be considered, for comparison purposes only, in the section devoted to simulation results.

In particular, we suppose to approximate the flow of the differential learning equation (3) through geodesic arcs properly connected, so as to obtain a piece-wise geodesic-type approximation of the exact gradient flow. If we denote by $\mathbf{W} \in O(p)$ the pattern to be learnt (for instance the connection matrix of a one-layer neural network), the considered geodesic-based learning algorithm corresponding to the exact Riemannian gradient flow is implemented by considering learning steps of the form:

$$\mathbf{W}_{n+1} = \exp(\eta_n((\mathrm{grad}_{\mathbf{W}_n}^{\mathbb{R}^{p\times p}} f)\mathbf{W}_n^T - \mathbf{W}_n(\mathrm{grad}_{\mathbf{W}_n}^{\mathbb{R}^{p\times p}} f)))\mathbf{W}_n, \tag{11}$$

where the index $n \in \mathbb{N}$ denotes a learning step counter and η_n denotes an integration or learning stepsize (the factor $\frac{1}{2}$ may be safely absorbed in η_n) usually termed *(learning) schedule* or step-size. It deserves underlining that the integration step-size may change across iterations because it may be beneficial to vary the step-size according to the progress of learning. The initial solution $\mathbf{W}_0$ should be selected in $O(p)$. It should be noted that the matrix $\mathbf{W}$ plays now the role of the general matrix $\mathbf{X}$ used in the previous section.

The aim of the present section is to consider three Riemannian gradient algorithms over the Lie group of orthogonal matrices. All three algorithms ensure that the current network-state matrix remain within the orthogonal group:

- Algorithm 1 uses a fixed step-size in the general geodesic-based learning equation (11).

- Algorithm 2 uses a geodesic line search for optimizing the step-size in the general geodesic-based learning equation (11).
- Algorithm 3 introduces stochasticity in the Algorithm 1, using a Markov-Chain Monte-Carlo method, jointly with an annealing procedure.

3.1 Deterministic Algorithms

A learning algorithm based on the findings of Section 2 may be stated as follows, where it is supposed that a constant learning step-size is employed.

$\diamond$ **Learning algorithm 1**:

1. Set $n = 0$, generate an initial solution $\mathbf{W}_0$ and set $f_0 = f(\mathbf{W}_0)$ and define a constant step-size η.
2. Compute a candidate solution $\mathbf{W}_{n+1}$ through the equation (11), increment n and return to 2, unless n exceeds the maximum number of iteration permitted: In this case, exit.

Formally, as mentioned in Section 2.1, the concept of geodesic is essentially local, therefore the discrete steps (11) on the orthogonal group should be extended for small values of η_n. Instead of keeping η_n constant or letting it progressively decreases through some 'cooling scheme', as it is customary in classical learning algorithms, it could allegedly be convenient to optimize it during learning. It is worth underlining at this point that the numerical evaluation of the geodesic curve through the exponential map, as well as the effective movement along a geodesic, are computationally expensive operations.

Step-size adaptation may be accomplished through a proper 'line search', as explained in what follows. Let us first define the following quantities for the sake of notation conciseness:

$$\tilde{\mathbf{V}}_n \overset{\text{def}}{=} (\text{grad}^{\mathbb{R}^{p\times p}}_{\mathbf{W}_n} f)\mathbf{W}_n^T - \mathbf{W}_n(\text{grad}^{\mathbb{R}^{p\times p}}_{\mathbf{W}_n} f)^T,\ \mathbf{E}_n(t) \overset{\text{def}}{=} \exp(t\tilde{\mathbf{V}}_n). \tag{12}$$

Starting from a point $\mathbf{W}_n$ at iteration step n, according to equation (11), the next point would be $\mathbf{E}_n(t)\mathbf{W}_n$, therefore the learning criterion function would descend from $f(\mathbf{W}_n)$ to $f_n(t) \overset{\text{def}}{=} f(\mathbf{E}_n(t)\mathbf{W}_n)$. From the definition of f, which is continuous and defined on a compact manifold, it follows that the function $f_n(t)$ admits a point of minimum for $t \in \mathbb{T} \subset \mathbb{R}^-$, that may be denoted as $t_\star$. If we are able to find $t_\star$ in a computationally convenient way, we may then select $\eta_n = t_\star$. The operation of searching for a convenient value as close as possible to $t_\star$ is termed *geodesic search* as it closely resembles the familiar concept of 'line search'.

Basically, we may perform a geodesic search in two different ways:

- By sampling the interval $\mathbb{T}$ through a sequence of discrete indices t_k, computing the value of $f_n(t_k)$ and selecting the value that grants the smallest cost.
- By computing the derivative $\frac{df_n(t)}{dt}$ and looking for the value of the index t for which it is equal (or sufficiently close) to zero. This approach would look advantageous if the expression of such equation could be handled analytically in an straightforward way. We found it is not the case and that this approach looks excessively cumbersome from a computational viewpoint, therefore it will not be adopted in this paper.

A second learning algorithm based on the above considerations may be stated as follows.

$\diamond$ **Learning algorithm 2**:

1. Set $n = 0$, generate an initial solution $\mathbf{W}_0$ and set $f_0 = f(\mathbf{W}_0)$.
2. Compute the quantity $\tilde{\mathbf{V}}_n$ in the equations (12).
3. Perform a geodesic-search for the optimal step-size η_n.
4. Compute a candidate solution $\mathbf{W}_{n+1}$ through the equation (11) and evaluate $f_{n+1} = f(\mathbf{W}_{n+1})$.
5. If $f_{n+1} < f_n$ then accept the candidate solution, increment n and return to 2, unless n exceeds the maximum number of iteration permitted: In this case, exit. If $f_{n+1} \geq f_n$, then proceed to 6.
6. Generate a small random step-size η_n.
7. Compute the candidate solution $\mathbf{W}_{n+1}$ through the equation (11), evaluate $f_{n+1} = f(\mathbf{W}_{n+1})$, increment n and return to 2.

The steps 6 and 7 in the above algorithm have been introduced in order to tackle the case in which the geodesic search gives rise to a candidate solution that causes the network's connection pattern to ascend the cost function f instead of making it descend. In this case, moving along the geodesic of a small random quantity does not ensure monotonic decreasing of the cost function, but it might help moving to another zone of the parameter space in which the geodesic learning might be effective.

3.2 Diffusion-Type Gradient Algorithm

In order to mitigate the known numerical convergence difficulties associated to the plain gradient-based optimization algorithms, it might be beneficial to perturb the standard Riemannian gradient to obtain a randomized gradient. In particular, following Liu et al. (2004), we may replace the gradient-based optimization steps with joint *simulated annealing* and *Markov-Chain Monte-Carlo* (MCMC) optimization technique, which gives rise to a so-termed *diffusion-type optimization process*. The Markov-Chain Monte-Carlo method was proposed and developed in the classical papers by Hastings (1970) and Metropolis et al. (1953).

It is worth recalling that, in classical algorithms, perturbations are easily introduced by sampling each network input one by one and by exploiting only such instantaneous information at a time. When used in conjunction with gradient-based learning algorithms, this inherently produces a stochastic gradient optimization based on a random walk on the parameters space. The two main reasons for which such choice is not adopted here are:

- When statistical expectations are replaced by one-sample mean, as it is customarily done, for example, in on-line signal processing, part of the information content pertaining to past samples is discarded from the learning system, and this might be a serious side effect on learning capability.
- The annealed MCMC method offers the possibility of actually *controlling* the amount of stochasticity introduced in the learning system by properly setting the method's free parameters such as the annealing temperature. Classical random-walk learning algorithms – as the one based on sampling each network input one by one – do not seem to offer such possibility.

A general discussion on the possible benefits owing to the introduction of stochasticity in gradient-based learning systems has been presented by Wilson and Martinez (2003).

It is understood that in a learning process having a Euclidean space as base-manifold, each step is simply proportional to the gradient computed in the departing point, therefore the learning steps may be directly perturbed in order to exploit randomized parameter-space search. In the present context, however, the base manifold $O(p)$ is curved, therefore it is sensible to perturb the gradient in the Lie algebra and then apply the formulas explained in the Section 2.2 to compute the associated step in the base-group.

In short, simulated annealing consists in adding to the deterministic gradient a random component whose amplitude is proportional to a parameter referred to as *temperature*. This mechanism may help the optimization algorithm to escape local solutions, but it has the drawback of occasionally leading to changes of the variable of interest toward the wrong direction (that is, it may lead to intermediate solutions with higher values of the criterion function when its minimum is sought for or vice-versa). Such drawback may be gotten rid of by adopting a MCMC-type simulated annealing optimization strategy where the diffusion-type gradient is exploited to generate a possible candidate for the next intermediate solution which is accepted/rejected on the basis of an appropriate probability distribution.

According to Liu et al. (2004), the diffusion-type gradient on the algebra $\mathfrak{so}(p)$ may be assumed as

$$\tilde{\mathbf{V}}_{\text{diff}}(t) = \tilde{\mathbf{V}}(t) + \sqrt{2\Theta} \sum_{k=1}^{p(p-1)/2} \mathbf{L}_k \frac{d\mathcal{W}_k}{dt}, \tag{13}$$

where $\tilde{\mathbf{V}}(t)$ is the gradient (9), $\{\mathbf{L}_k\}$ is a basis of the Lie algebra $\mathfrak{so}(p)$, orthogonal with respect to the metric $g^{O(p)}_{\mathbf{I}_p}$, the $\mathcal{W}_k(t)$ are real-valued, independent standard Wiener processes and the parameter $\Theta > 0$ denotes the aforementioned temperature, which proves useful for simulating annealing during learning. It is worth recalling that a Wiener process is a continuous-time stochastic process $\mathcal{W}(t)$ for $t \geq 0$, that satisfies the following conditions (Higham, 2001):

- $\mathcal{W}(0) = 0$ with probability 1.
- For $0 \leq \tau < t$ the random variable given by the increment $\mathcal{W}(t) - \mathcal{W}(\tau)$ is normally distributed with mean zero and variance $t - \tau$. Equivalently, $\mathcal{W}(t) - \mathcal{W}(\tau) \sim \sqrt{t-\tau}\mathcal{N}(0,1)$, where $\mathcal{N}(0,1)$ denotes a normally distributed random variable with zero mean and unit variance.
- For $0 \leq \tau < t < u < v$, the increments $\mathcal{W}(t) - \mathcal{W}(\tau)$ and $\mathcal{W}(v) - \mathcal{W}(u)$ are statistically independent.

The learning differential equation on the orthogonal group associated to the gradient (13) reads

$$\frac{d\mathbf{W}}{dt} = -\tilde{\mathbf{V}}_{\text{diff}}(t)\mathbf{W}(t), \tag{14}$$

is a *Langevin-type stochastic differential equation* (LDE).

By analogy with physical phenomena described by this equation, such as the Brownian motion of particles, the solution to the LDE is termed a *diffusion process*. Under certain conditions on the criterion function f, the solution of equation (14) is a Markov process endowed with a *unique stationary* probability density function (Srivastava et al., 2002), described by

$$\pi_{\mathrm{LDE}}(\mathbf{W}) \stackrel{\mathrm{def}}{=} \frac{1}{Z(\Theta)} \exp(-f(\mathbf{W})/\Theta), \tag{15}$$

where $Z(\Theta)$ denotes the density-function normalizer (partition function).[5] In other terms, the LDE 'samples' from the distribution $\pi_{\mathrm{LDE}}(\mathbf{W})$: This is a main concept in the method of using the LDE to generate random samples according to a given energy/cost function.

The choice of assuming the probability π_{LDE} inversely proportional to the value of $f(\mathbf{W})$ serves at discouraging network states corresponding to high values of the learning cost function. Also, it deserves to note that care should be taken of the problem related to the consistency of the above definition: The problem of the existence of π_{LDE}, that is connected to the existence of the partition function $Z(\Theta)$, must be dealt with. To this aim, it is worth noting that $f(\mathbf{W})$ is a continuous function of the argument which belongs to a compact space, we may therefore argue that $f(\mathbf{W})$ is bounded from above and from below. Thus, the function $\exp(-f(\mathbf{W}))$ is bounded and its integral over the whole orthogonal group through a coordinate-invariant measure of volume, such as the Haar measure (Srivastava et al., 2002), is surely existent.

In order to practically perform statistical sampling via the LDE, we can distinguish between *rejection* and *MCMC* methods:

1. The rejection algorithm is designed to give an exact sample from the distribution. Let us denote by $\pi(x)$ a density to sample from a set $\mathcal{X}$: We can sample from another distribution $\mu(x)$ (instrumental distribution) such that sampling from it is practically easier than actually sampling from $\pi(x)$. Then, it is possible to generate x^* from $\mu(x)$ and accept it with probability

$$\alpha \stackrel{\mathrm{def}}{=} \frac{\pi(x^*)}{\mu(x^*)M},$$

where M is a constant such that $\pi(x)/\mu(x) \leq M$ for all $x \in \mathcal{X}$. If the generated sample is not accepted, rejection is performed until acceptance. When accepted, it is considered to be an exact sample from $\pi(x)$. A consequence of adopting this method is that the number of necessary samplings from $\mu(x)$ is unpredictable.

2. In MCMC, a Markov chain is formed by sampling from a conditional distribution $\mu(x|y)$: The algorithm starts from x_0 and proceeds iteratively as follows: At step n, sample x^* from $\mu(x|x_n)$ and compute the acceptation (Metropolis-Hastings) probability as

$$\alpha_n \stackrel{\mathrm{def}}{=} \min\left\{1, \frac{\pi(x^*)\mu(x_n|x^*)}{\pi(x_n)\mu(x^*|x_n)}\right\}, \tag{16}$$

then accept x^* with probability α_n. This means letting $x_{n+1} = x^*$ with probability α_n, otherwise $x_{n+1} = x_n$. This is the main difference with rejection method: If the candidate sample is not accepted, then the previous value is retained.

5. The theory presented by Srivastava et al. (2002) deals with the special case in which the base-manifold is $O(3)$. This result is not related to the dimension of the orthogonal group of interest, indeed, therefore it may be extended without difficulty to the general case $O(p)$ of concern in the present paper.

In the MCMC method, the quantity $\mu(x|y)$ denotes a *transition probability* as it describes the probability of 'jumping' from state y to state x. The total probability of transition from state x_n to state x_{n+1} is given by the combination of the instrumental distribution $\mu(x|y)$ and the Metropolis-Hastings acceptation probability: The transition kernel $K(x_{n+1}|x_n)$ is, in fact:

$$K(x_{n+1}|x_n) \overset{\text{def}}{=} \alpha_n \mu(x_{n+1}|x_n) + (1-\alpha_n)\delta(x_{n+1}-x_n).$$

In order to gain a physical interpretation of the instrumental probability $\mu(x|y)$, it pays to take for example a symmetric instrumental $\mu(x|y)$. Under this hypothesis, the ratio in the definition (16) would become $\pi(x^*)/\pi(x_n)$: The chain jumps to the state x^* if it is more plausible ($\alpha_n = 1$) than the previous state x_n, otherwise (case $\alpha_n < 1$), the chain jumps to the generated state according to the probability α_n. As an example of symmetric instrumental conditional probability, $\mu(x|y)$ may be assumed as Gaussian in x with mean y.

If the Markov chain $\{x_n\}_{n=1,\ldots,N}$ converges to the true probability $\pi(x)$, then x_n is asymptotically drawn from $\pi(x)$, so x_n is not an exact sample as in the rejection method. However, there is a powerful mathematical result that warrants that the empirical average (ergodic sum) $\sum_n \ell(x_n)/N$, for a regular function $\ell : \mathcal{X} \to \mathbb{R}$, converges to $\mathbb{E}[\ell(x)]$ if the chain converges asymptotically to the true distribution. For example, if x is a zero-mean scalar random variable and $\mathcal{X} = \mathbb{R}$, then $\ell(x) = x^2$ for the variance and $\ell(x) = x^4$ for the kurtosis of the variable. For this reason, MCMC methods are considered to be preferable over rejection method because in this latter only one exact sample is obtained, while with the former we obtain a chain and are thus able to approximate expectations. In order to perform MCMC, there is a great flexibility in choosing the instrumental probability density $\mu(x|y)$.

For a recent review of the MCMC method, interested readers may consult for instance the surveys by Kass et al. (1998) and Warnes (2001).

In order to numerically integrate the learning LDE, it is necessary to discretize the Wiener random process. Let us denote again by η the chosen (constant) step-size: A time-discretization of the stochastic gradient (13) may be written as

$$\tilde{\mathbf{V}}_{\text{diff},n} = \tilde{\mathbf{V}}_n + \sqrt{\frac{2\Theta}{\eta}} \sum_{k=1}^{p(p-1)/2} \mathbf{L}_k \nu_k, \tag{17}$$

where each ν_k is a independent, identically distributed normal random variable (Higham, 2001) and the gradient $\tilde{\mathbf{V}}_n$ is given in equation (12).

Having defined the new diffusion-type gradient (and its time-discretized version), the associated stochastic flow may be locally approximated through the geodesic learning algorithm explained in Section 2.2. Also, at every learning step n, the temperature Θ_n may be decreased in order to make the diffusive disturbance term peter out after the early stages of learning. This gives rise to the following simulated-annealing/MCMC learning scheme.

$\diamond$ **Learning algorithm 3**:

1. Set $n = 0$, generate an initial solution $\mathbf{W}_0$ and set $f_0 = f(\mathbf{W}_0)$, select a constant learning step-size η, select a temperature value Θ_0 and select a $g^{O(p)}$-orthonormal base $\mathbf{L}_k$ of the Lie algebra $\mathfrak{so}(p)$.

2. Generate a set of identically-distributed, independent standard Gaussian random variables ν_k.

3. Compute the diffusive gradient (17), compute a candidate solution $\mathbf{W}_{n+1}$ through the equation (11), where the deterministic gradient is replaced by the diffusive gradient, and evaluate $f_{n+1} = f(\mathbf{W}_{n+1})$.

4. Compute the MCMC probability $\pi_{MCMC} \overset{\text{def}}{=} \min\{1, \exp(-(f_{n+1} - f_n)/\Theta_n)\}$.

5. Accept the candidate solution with probability π_{MCMC} (or reject the candidate solution with probability $1 - \pi_{MCMC}$). Rejection corresponds to assuming $\mathbf{W}_{n+1} = \mathbf{W}_n$.

6. Decrease the temperature Θ_n to Θ_{n+1} following a pre-defined cooling scheme.

7. Increment n and return to 2, unless n exceeds the maximum number of iteration permitted: In this case, exit.

4. Application to Non-Negative Independent Component Analysis: Algorithms Implementation and Numerical Experiments

The aims of the present section are to recall the concept of non-negative independent component analysis (ICA$^+$) and the basic related results, to customize the general learning algorithms on the orthogonal group to the case of ICA$^+$, and to present and discuss some numerical cases related to non-negative ICA applied to the separation of gray-level images.

4.1 Non-Negative Independent Component Analysis

Independent component analysis (ICA) is a signal/data processing technique that allows to recover independent random processes from their unknown combinations (Cichocki and Amari, 2002; Hyvärinen et al., 2001). In particular, standard ICA allows the decomposition of a random process $\mathbf{x}(t) \in \mathbb{R}^p$ into the affine instantaneous model:

$$\mathbf{x}(t) = \mathbf{A}\mathbf{s}(t) + \mathbf{n}(t), \tag{18}$$

where $\mathbf{A} \in \mathbb{R}^{p\times p}$ is the *mixing* operator, $\mathbf{s}(t) \in \mathbb{R}^p$ is the *source stream* and $\mathbf{n}(t) \in \mathbb{R}^p$ denotes the disturbance affecting the measurement of $\mathbf{x}(t)$ or some nuisance parameters that are not taken into account by the linear part of the model.

The classical hypotheses on the involved quantities are that the mixing operator is full-rank, that at most one among the source signals exhibit Gaussian statistics, and that the source signals are statistically independent at any time. The latter condition may be formally stated through the complete factorization principle, which ensures that the joint probability density function of statistically independent random variables factorizes into the product of their marginal probability density functions. We also add the technical hypothesis that the sources do not have degenerate (that is, point-mass-like) joint probability density function. This implies that for example, the probability that the sources are simultaneously exactly zero is null. Under these hypotheses, it is possible to recover the sources up to (usually unessential) re-ordering and scaling, as well as the mixing operator.

Neural ICA consists in training an artificial neural network described by $\mathbf{y}(t) = \mathbf{W}(t)\mathbf{x}(t)$, with $\mathbf{y}(t) \in \mathbb{R}^p$ and $\mathbf{W}(t) \in \mathbb{R}^{p\times p}$, so that the network output signals become as statistically independent as possible.

Due to the difficulty of measuring the statistical independence of the network's output signals, several different techniques have been developed in order to perform ICA. The most common approaches to ICA are those based on working out the fourth-order statistics of the network outputs and to the minimization of the (approximate) mutual information among the network's outputs. The existing approaches invoke some approximations or assumptions in some stage of ICA-algorithm development, most of which concern the (unavailable) structure of the source's probability distribution.

As it is well-known, a linear, full-rank, *noiseless* and instantaneous model may be always replaced by an orthogonal model, in which the mixing matrix $\mathbf{A}$ is supposed to belong to $O(p)$. This result may be obtained by pre-whitening the observed signal $\mathbf{x}$, which essentially consists in removing second-order statistical information from the observed signals. When the mixture is orthogonal, the separating network's connection matrix must also be orthogonal, so we may restrict the learning process to searching the proper connection matrix within $O(p)$.

An interesting variant of standard ICA may be invoked when the additional knowledge on the non-negativity of the source signals is considered. In some signal processing situations, in fact, it is *a priori* known that the sources to be recovered have non-negative values (Plumbley, 2002, 2003). This is the case, for instance, in image processing, where the values of the luminance or the intensity of the color in the proper channel are normally expressed through non-negative integer values. Another interesting potential application is spectral unmixing in remote sensing (Keshava and Mustard, 2002). The evolution of passive remote sensing has witnessed the collection of measurements with great spectral resolution, with the aim of extracting increasingly detailed information from pixels in a scene for both civilian and military applications. Pixels of interest are frequently a combination of diverse components: In hyper-spectral imagery, pixels are a mixture of more than one distinct substance. In fact, this may happen if the spatial resolution of a sensor is so low that diverse materials can occupy a single pixel, as well as when distinct materials are combined into a homogeneous mixture. Spectral demixing is the procedure with which the measured spectrum is decomposed into a set of component spectra and a set of corresponding abundances, that indicate the proportion of each component present in the pixels. The theoretical foundations of the *non-negative independent component analysis* (ICA$^+$) have been given by Plumbley (2002), and then Plumbley (2003) proposed an optimization algorithm for non-negative ICA based on geodesic learning and applied it to the blind separation of three gray-level images. Further recent news on this topic have been published by Plumbley (2004). In our opinion, non-negative ICA as proposed by Plumbley (2003) is an interesting task and, noticeably, it also gives rise to statistical-approximation-free and parameter-free learning algorithms.

Under the hypotheses motivated by Plumbley (2002), a way to perform non-negative independent component analysis is to construct a cost function $f(\mathbf{W})$ of the network connection matrix that is identically zero if and only if the entries of network's output signal $\mathbf{y}$ are non-negative with probability 1. The criterion function chosen by Plumbley (2003) is $f : O(p) \to \mathbb{R}_0^+$ defined by

$$f(\mathbf{W}) \stackrel{\text{def}}{=} \frac{1}{2}\mathbb{E}_{\mathbf{x}}[\|\mathbf{x} - \mathbf{W}^T \rho(\mathbf{W}\mathbf{x})\|_2], \tag{19}$$

where $\mathbb{E}_{\mathbf{x}}[\cdot]$ denotes statistical expectation with respect to the statistics of $\mathbf{x}$, $\|\cdot\|_2$ denotes the standard L_2 vector norm and the function $\rho(\cdot)$ denotes the 'rectifier':

$$\rho(u) \stackrel{\text{def}}{=} \begin{cases} u \text{ , if } u \geq 0 \text{ ,} \\ 0 \text{ , otherwise.} \end{cases}$$

In the definition (19), the rectifier acts component-wise on vectors. From the definition (19), it is clear that when all the network output signals have positive values, it results $f = 0$, otherwise $f \neq 0$. The described cost function closely resembles a non-linear principal component analysis criterion designed on the basis of the minimum reconstruction error principle (Hyvärinen et al., 2001). This observation would be beneficial for future extensions to complex-weighted neural networks, as suggested by Fiori (2004).

In this case, learning a ICA$^+$ network may thus be accomplished by minimizing the criterion function f.

In order to design a gradient-based learning algorithm over the orthogonal group according to the general theory developed in the Section 2.2, it is necessary to compute the Euclidean gradient of the function (19) with respect to the connection matrix $\mathbf{W}$. After rewriting the learning criterion function as

$$2f(\mathbf{W}) = \mathbb{E}_{\mathbf{x}}[\|\mathbf{x}\|_2 + \|\rho(\mathbf{y})\|_2 - 2\mathbf{y}^T\rho(\mathbf{y})],$$

some lengthy but straightforward computations lead to the expression:

$$\text{grad}_{\mathbf{W}}^{\mathbb{R}^{p\times p}} f = \mathbb{E}_{\mathbf{x}}[((\rho(\mathbf{y}) - \mathbf{y}) \diamond \rho'(\mathbf{y}))\mathbf{x}^T - \rho(\mathbf{y})\mathbf{x}^T],$$

where the symbol $\diamond$ denotes component-wise (Hadamard) product of two vectors and $\rho'(\cdot)$ denotes the derivative of the rectifier, that is, the unit-step function. This is undefined in the origin. From a practical point of view, this is a minor difficulty: In fact, thanks to the hypothesis of non-degeneracy of the joint probability density function of the source, the probability that the components of the networks output vector vanish to zero simultaneously is equal to zero. It is now easy to recognize that the vector $(\rho(\mathbf{y}) - \mathbf{y}) \diamond \rho'(\mathbf{y})$ is identically zero (where it is defined), therefore the above gradient reduces to the simple expression:

$$\text{grad}_{\mathbf{W}}^{\mathbb{R}^{p\times p}} f = -\mathbb{E}_{\mathbf{x}}[\rho(\mathbf{y})\mathbf{x}^T].$$

Following the notation introduced by Plumbley (2003), we find it convenient to define the rectified network output:

$$\mathbf{y}_n^+ \overset{\text{def}}{=} \rho(\mathbf{y}_n) \text{ , where } \mathbf{y}_n \overset{\text{def}}{=} \mathbf{W}_n\mathbf{x}. \tag{20}$$

With this convention, the Riemannian gradient and the associate learning algorithm (valid for example, for the versions of Algorithms 1 and 2) write, respectively:

$$\begin{aligned}
&2\,\text{grad}_{\mathbf{W}_n}^{O(p)} f = \mathbb{E}_{\mathbf{x}}[\mathbf{y}_n(\mathbf{y}_n^+)^T\mathbf{W}_n] - \mathbb{E}_{\mathbf{x}}[(\mathbf{y}_n^+)\mathbf{x}^T],\\
&\mathbf{W}_{n+1} = \exp(\eta_n(\mathbb{E}_{\mathbf{x}}[\mathbf{y}_n(\mathbf{y}_n^+)^T] - \mathbb{E}_{\mathbf{x}}[\mathbf{y}_n^+\mathbf{y}_n^T]))\mathbf{W}_n \text{ ,}\\
&n = 1,\ 2,\ 3,\ \ldots
\end{aligned}$$

The initial connection matrix $\mathbf{W}_0$ may be randomly picked in $O(p)$. Another practical choice is $\mathbf{W}_0 = \mathbf{I}_p$.

4.2 Details on the Used Data and on Algorithms Implementation

The gray-level images used in the experiments are illustrated in the Figure 1. It is important to note that, in general, real-world images are not completely statistically independent. For instance,

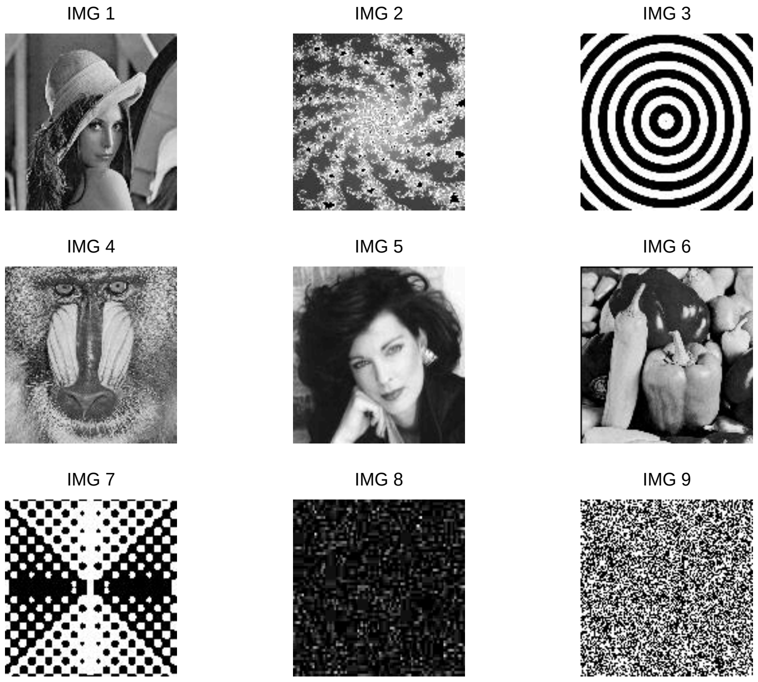


Figure 1: The nine gray-level images used in the experiments.

the images used in the present experiments are slightly statistically correlated, as can be seen by computing their 9×9 covariance matrix (approximated to two decimal digits) $\mathbf{C}_{\mathbf{s}} =$

$$10^3 \times \begin{bmatrix} 2.81 & 0.07 & 0.1 & -0.05 & -0.33 & -0.55 & 0.29 & -0.04 & -0.12 \\ 0.07 & 4.52 & 0 & -0.04 & 0.61 & 0.49 & -0.16 & 0.01 & -0.02 \\ 0.1 & 0 & 15.05 & 0.33 & 0.14 & 0.06 & -0.37 & -0.09 & 0.01 \\ -0.05 & -0.04 & 0.33 & 2.32 & 0.17 & 0.38 & -0.43 & 0 & -0.09 \\ -0.33 & 0.61 & 0.14 & 0.17 & 5.49 & 0.67 & 0.8 & 0.01 & 0.02 \\ -0.55 & 0.49 & 0.06 & 0.38 & 0.67 & 5.69 & -0.63 & -0.04 & -0.04 \\ 0.29 & -0.16 & -0.37 & -0.43 & 0.8 & -0.63 & 15.3 & -0.01 & 0.12 \\ -0.04 & 0.01 & -0.09 & 0 & 0.01 & -0.04 & -0.01 & 0.89 & -0.01 \\ -0.12 & -0.02 & 0.01 & -0.09 & 0.02 & -0.04 & 0.12 & -0.01 & 15.33 \end{bmatrix},$$

which is not diagonal, but diagonal-dominated.

It is now necessary to explain in details the pre-whitening algorithm. We distinguish between the noiseless and noisy case.

- In the noiseless case (namely, $\mathbf{n}(t) \equiv 0$), the pre-whitening stage is based on the observation that in the model (18) the square matrix $\mathbf{A}$ may be written through the singular value decomposition (SVD) as $\mathbf{F}_1\mathbf{D}\mathbf{F}_2^T$, where $\mathbf{F}_1, \mathbf{F}_2 \in O(p)$ and $\mathbf{D} \in \mathbb{R}^{p \times p}$ is diagonal invertible. Then, it is readily verified that $\mathbf{C}_{\mathbf{x}} \overset{\text{def}}{=} \mathbb{E}_{\mathbf{x}}[\overline{\mathbf{x}}\overline{\mathbf{x}}^T] = \mathbf{A}\mathbb{E}_{\mathbf{s}}[\overline{\mathbf{s}}\overline{\mathbf{s}}^T]\mathbf{A}^T$, where the overline denotes centered signals (for example, $\overline{\mathbf{x}} \overset{\text{def}}{=} \mathbf{x} - \mathbb{E}_{\mathbf{x}}[\mathbf{x}]$.) In the (non-restrictive) hypothesis that $\mathbb{E}_{\mathbf{s}}[\overline{\mathbf{s}}\overline{\mathbf{s}}^T] = \mathbf{I}_p$, we thus have $\mathbf{C}_{\mathbf{x}} = \mathbf{A}\mathbf{A}^T = \mathbf{F}_1\mathbf{D}^2\mathbf{F}_1^T$. The factors $\mathbf{F}_1$ and $\mathbf{D}$ may thus be computed through the standard eigenvalue decomposition of the covariance $\mathbf{C}_{\mathbf{x}}$. The whitened observation signal is then

$$\hat{\mathbf{x}} \stackrel{\text{def}}{=} \mathbf{D}^{-1}\mathbf{F}_1^T\mathbf{x} = \mathbf{F}_2^T\mathbf{s}.$$

It is now clear that the last rotation $\mathbf{F}_2^T$ of the source signals cannot be removed by second-order statistics, while orthogonal non-negative ICA may be effective to separate out the independent/non-negative components.

- In the noisy case, when the model of the observed signal is given by (18), the noise component cannot be filtered out by using pre-whitening nor independent component analysis itself. However, pre-whitening still makes it possible to use orthogonal ICA$^+$, provided *the additive noise affecting the observations is not too strong*. In fact, by hypothesizing the noise component $\mathbf{n}(t)$ is a zero-mean multivariate random sequence with covariance matrix $\sigma^2\mathbf{I}_p$, termed 'spherical' noise, the covariance of the observations writes $\mathbf{C}_{\mathbf{x}} = \mathbf{A}\mathbf{A}^T + \sigma^2\mathbf{I}_p$. In case of strong disturbance, it is therefore clear that, in general, pre-whitening cannot rely on eigenvalue decomposition of $\mathbf{C}_{\mathbf{x}}$. In any case, the difficulty due to the presence of strong additive noise is theoretically unavoidable, even if pre-whitening is dispensed of and ICA algorithms that search in $Gl(p)$ are employed.

In order to compute a separation performance index, we consider that, at convergence, the separation product $\mathbf{P}_n \stackrel{\text{def}}{=} \mathbf{W}_n\mathbf{D}^{-1}\mathbf{F}_1^T\mathbf{A} \in \mathbb{R}^{p\times p}$ should ideally exhibit only one entry per row (or column) different from zero, while the magnitude of non-zero values does not care. In a real-word situation, of course some residual interference should be tolerated. Therefore, a valid separation index is

$$Q_n \stackrel{\text{def}}{=} \frac{1}{p}\|\mathbf{P}_n\mathbf{P}_n^T - \operatorname{diag}(\mathbf{P}_n\mathbf{P}_n^T)\|_{\mathrm{F}}, \tag{21}$$

where $\|\cdot\|_{\mathrm{F}}$ denotes the Frobenius norm. The index above is based on the fact that ideally the matrix $\mathbf{P}_n\mathbf{P}_n^T$ should be diagonal, therefore Q_n measures the total off-diagonality averaged over the total number of network's outputs. (As normally the index Q_n assumes very low values, it is worth normalizing it to its initial value, namely by Q_n/Q_0.)

Another valid network-performance index is the criterion function (19) itself. For easy computation of the index, we note that by defining $\mathbf{y}_n^- \stackrel{\text{def}}{=} \mathbf{W}_n\mathbf{x} - \rho(\mathbf{W}_n\mathbf{x})$, the value of the cost function at the n-th learning step computes as $f_n = \frac{1}{2}\mathbb{E}_{\mathbf{x}}[\|\mathbf{y}_n^-\|_2]$. (The learning algorithm seeks for a neural transformation that minimizes the negativity of its outputs, in fact.)

With regard to the computational complexity analysis of the described algorithms, we consider the number of floating-point operations (flops) per iteration and the average run-time per iteration. The codes were implemented in MATLAB on a 600 MHz, 128 MB platform.

With regard to the selection of the schedule η_n, in the experiments we found it convenient to write first the learning step-size η_n as $\tilde{\eta}_n/\|\tilde{\mathbf{V}}_n\|_{\mathrm{F}}$, where $\tilde{\mathbf{V}}_n$ denotes again the gradient on the Lie algebra of $O(p)$ defined in the equations (12) and then to optimize the normalized step-size $\tilde{\eta}_n$. This convention keeps valid throughout the remaining part of the paper, so we can continue to use the notation η_n even for the normalized step-size without confusion.

In order to establish a numerically efficient geodesic search for the Algorithm 2, we seek for the optimal η_n in a suitable interval by sampling this interval at sub-intervals of proper size. The details on these quantities are given in the section dedicated to the numerical experiments for each category of experiment.

About the cooling scheme for the simulated-annealing/MCMC algorithm, according to Liu et al. (2004), we adopted the schedule $\Theta_{n+1} = \Theta_n/1.025$.

As a general note, the ensemble average denoted by the statistical expectation operator $\mathbb{E}[\cdot]$ is replaced everywhere by sample (empirical) mean.

4.3 Results of Numerical Experiments

The present part of the paper aims at presenting some numerical results obtained with the above-described learning algorithms applied to non-negative independent component analysis. The numerical analysis begins with the illustration of some toy experiments that aim at showing the consistency of the adopted 'non-negativity' optimization principle. Then, the analysis continues with an investigation and a comparison of the behavior of the three algorithms described in the previous sections.

4.3.1 Preliminary Experiments

As a case study, we consider the mixing of two images with a randomly generated mixing matrix $\mathbf{A} \in \mathbb{R}^{2\times 2}$. As the orthogonal separation matrix $\mathbf{W}$ is of size 2×2, it may be easily parameterized, as in equation (10), by

$$\mathbf{W}(\beta) = \begin{bmatrix} \cos\beta & -\sin\beta \\ \sin\beta & \cos\beta \end{bmatrix},$$

with $\beta \in [-\pi, \pi[$ being the separation angle. As already underlined in Section 2.3, this parameterization does not cover the whole group $O(2)$, but this problem is unessential for ICA purpose. By properly sampling the interval $[-\pi, \pi[$, it is possible to give a graphical representation of the behavior of the non-negative independent component analysis criterion $f(\mathbf{W}(\beta))$ defined in equation (19) and of the separation index $Q(\beta)$ defined by equation (21) (which depends on variable β through the separation product $\mathbf{P}$).

The results of this analysis for a randomly generated mixing matrix, with source images number 1 and 2 of Figure 1, are shown in the Figure 2. The Figure 2 shows the two-image mixtures, the behavior of the cost function f and of the separation index Q as well as the separated images obtained with the optimal separation angle, which is defined as the angle corresponding to the *minimal criterion function value*. As it clearly emerges from the above figure, the cost function has a only minimum, which coincides with one of the minima of the separation index. The minimum of the cost function corresponds to a pair of well-separated network outputs.

The result of the analysis with source images number 3 and 4 of Figure 1 are shown in the Figure 3. The Figure 3 shows the mixtures, the behavior of the cost function and of the separation index as well as the separated images. Again, the cost function exhibits a only minimum that coincides with one of the minima of the separation index, which, in turn, corresponds to a pair of well-separated non-negative independent components. This second result, compared with the previous one, illustrates the dependency of the shape of the cost function on the mixing matrix as well as on the mixed components.

To end the series of preliminary experiments, we consider here again the mixing of images 1 and 2 with a randomly generated mixing matrix $\mathbf{A}$ in the *noisy mixture case*. In particular, as anticipated in the Section 4.1, 'spherical' additive white Gaussian noise is supposed to contaminate the observations as in the original ICA model (18). The quantity that describes the relative weight of the noise in the mixture is the signal-to-noise ratio (SNR), which, in this particular case, may be

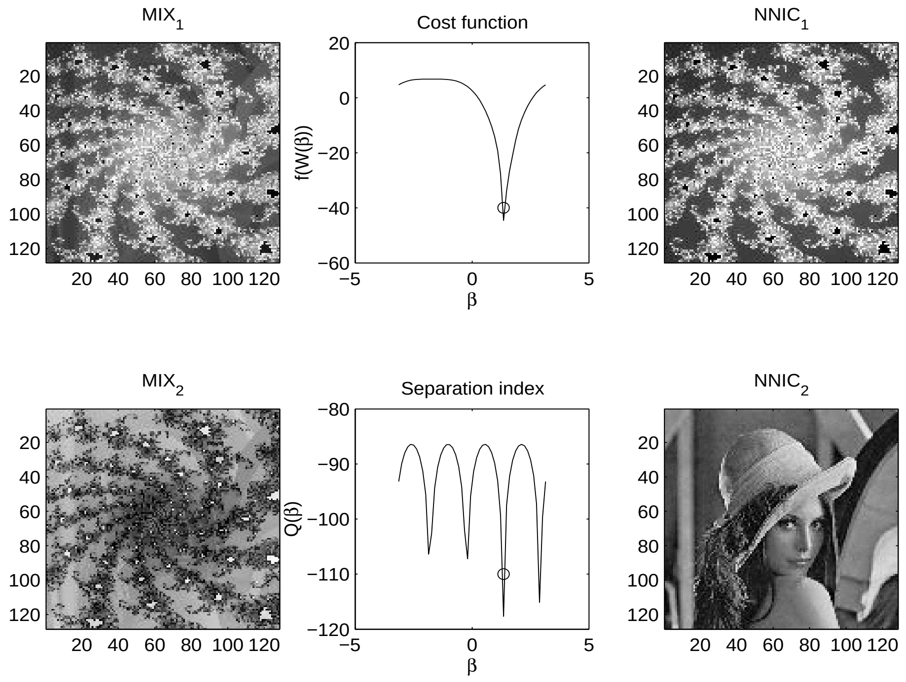


Figure 2: Images 1 and 2 mixtures (MIX$_1$ and MIX$_2$), behavior of cost function f and separation index Q (shown in dB scales) and separated images (NNIC$_1$ and NNIC$_2$) obtained with the optimal separation angle. The open circle denotes the value of the the parameter β corresponding to the minimum criterion $f(\mathbf{W}(\beta))$ value.

compactly defined as

$$SNR \overset{\text{def}}{=} 10\log_{10}\sqrt{\exp(\text{trace}\{\log[(\text{diag}(\mathbf{C_m})\text{diag}(\mathbf{C_n})^{-1})]\})},$$

where $\text{diag}(\mathbf{C_m})$ denotes the diagonal part of the 2×2 covariance matrix of the noiseless observation (term $\mathbf{As}(t)$) while $\text{diag}(\mathbf{C_n})$ denotes the diagonal part of the covariance matrix of the noise term $\mathbf{n}(t)$, referred to the ICA model (18).

The results of this analysis are shown in the Figures 4 and 5, which illustrate the behavior of the cost function f and of the separation index Q as well as the separated images obtained with the optimal separation angle, for two different noisy mixtures. In the experiment illustrated in the Figure 4, the value of the signal-to-noise ratio was $SNR = 11.64$ dB. The Figure shows that the cost function exhibits a only minimum that is quite close to one of the minima of the separation index, which, in turn, corresponds to a pair of well-separated non-negative independent components. Of course, the mixtures *as well as the recovered components* look a little noisy. In the experiment illustrated in the Figure 5, the value of the signal-to-noise ratio was $SNR = 4.18$ dB. In this experiment, the power of the disturbance is close to the power of the source-images, therefore the mixture may be considered as rather noisy. The Figure 5 shows that the cost function exhibits a only minimum that is quite far from the minima of the separation index. The neural network outputs look very noisy and do not resemble the original independent components. This result confirms the observations of Section 4.1

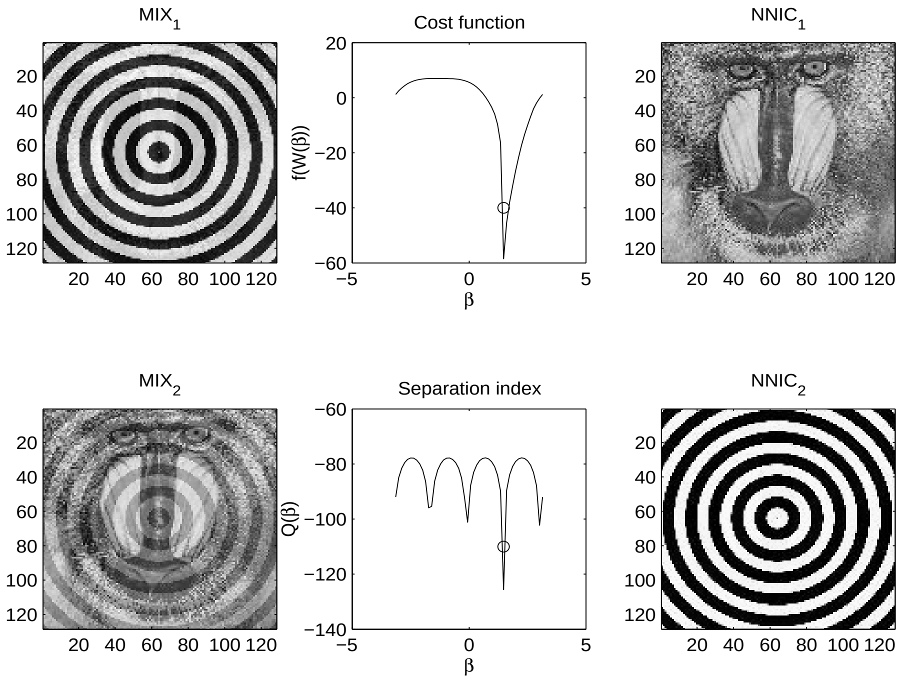


Figure 3: Images 3 and 4 mixtures (MIX$_1$ and MIX$_2$), behavior of cost function f and separation index Q (shown in dB scales) and separated images (NNIC$_1$ and NNIC$_2$) obtained with the optimal separation angle. The open circle denotes the value of the the parameter β corresponding to the minimum criterion $f(\mathbf{W}(\beta))$ value.

about the unavoidability of the problems related to the presence of strong noise in the mixture by plain ICA.

In the next sections, we shall therefore take into account noiseless mixtures, which also illustrate the behavior of the algorithm in presence of *weak* disturbances. It is in fact to be recognized that the pre-whitening/sphering issue is a different problem from optimization on $O(p)$: Noisy mixtures cannot be pre-whitened, but if the noise is weak, its presence has negligible effects on the separation performances.

4.3.2 A Further 'Conventional' Algorithm for Numerical Comparison Purposes

In order to gain incremental knowledge on the advantages offered by Lie-group methods via numerical comparisons, it would be beneficial to consider a 'conventional' learning algorithm in which the ordinary gradient and explicit orthogonalization are employed.[6] To this aim, we defined the following non-Lie-group algorithm:

$$\tilde{\mathbf{W}}_{n+1} = \mathbf{W}_n - \eta \mathbb{E}[\mathbf{y}_n^+ \mathbf{x}^T], \tag{22}$$

$$\mathbf{W}_{n+1} = (\tilde{\mathbf{W}}_{n+1} \tilde{\mathbf{W}}_{n+1}^T)^{-\frac{1}{2}} \tilde{\mathbf{W}}_{n+1}, \tag{23}$$

6. This comparison was suggested by a reviewer.

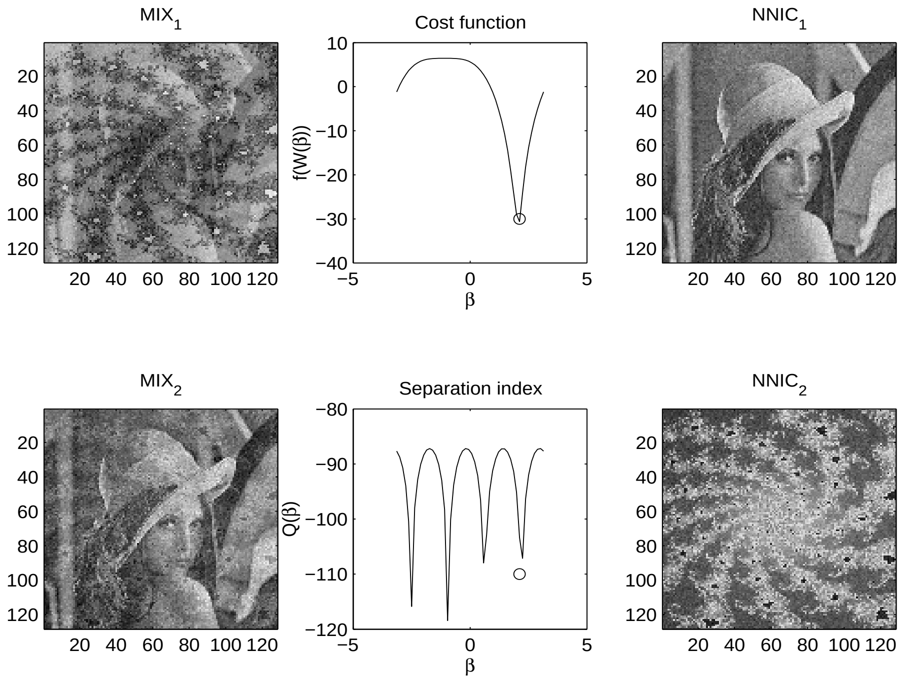


Figure 4: Images 1 and 2 weakly-noisy mixtures (MIX_1 and MIX_2), behavior of cost function f and separation index Q (shown in dB scales) and separated images (NNIC_1 and NNIC_2) obtained with the optimal separation angle. The open circle denotes the value of the the parameter β corresponding to the minimum criterion $f(\mathbf{W}(\beta))$ value.

where the rectified network output is defined as in equation (20) and with the initial connection pattern $\mathbf{W}_0 \in O(p)$ and the learning step-size $\eta < 0$ being chosen according to the same rules used with the Algorithms 1, 2 and 3. It is worth remarking that we again consider the normalization $\eta = \tilde{\eta}/\|\mathbb{E}[\mathbf{y}_n^+ \mathbf{x}^T]\|$, so the actual step-size to be selected is $\tilde{\eta}$, as previously assumed for the Algorithms 1, 2 and 3.

The first line of the above algorithm moves the connection pattern at step n from the matrix $\mathbf{W}_n$ over the orthogonal group toward the direction of the Euclidean gradient of the ICA^+ cost function to the new point $\tilde{\mathbf{W}}_{n+1}$. However, the matrix $\tilde{\mathbf{W}}_{n+1}$ does not belong to the orthogonal group so it is necessary to project it back to the group with the help of a suitable projector (according to what granted in Section 2.1). In this case, it is assumed $\Pi : \mathbb{R}^{p\times p} \to O(p)$ as

$$\Pi(\mathbf{X}) \overset{\text{def}}{=} (\mathbf{X}\mathbf{X}^T)^{-\frac{1}{2}}\mathbf{X}. \tag{24}$$

(It is straightforward to verify that $\Pi^T(\mathbf{X})\Pi(\mathbf{X}) = \mathbf{I}_p$ for all $\mathbf{X} \in Gl(p)$.) In the case of the orthogonal-group projector, the ambient space was assumed as $\mathcal{A} = \mathbb{R}^{p\times p}$. It is worth underlining that, from a theoretical point of view, there is no guarantee that the partially updated matrix $\tilde{\mathbf{W}}_{n+1}$ belongs to $Gl(p) \subset \mathbb{R}^{p\times p}$ and, therefore, there is no guarantee that the projector Π may be computed at every iteration.

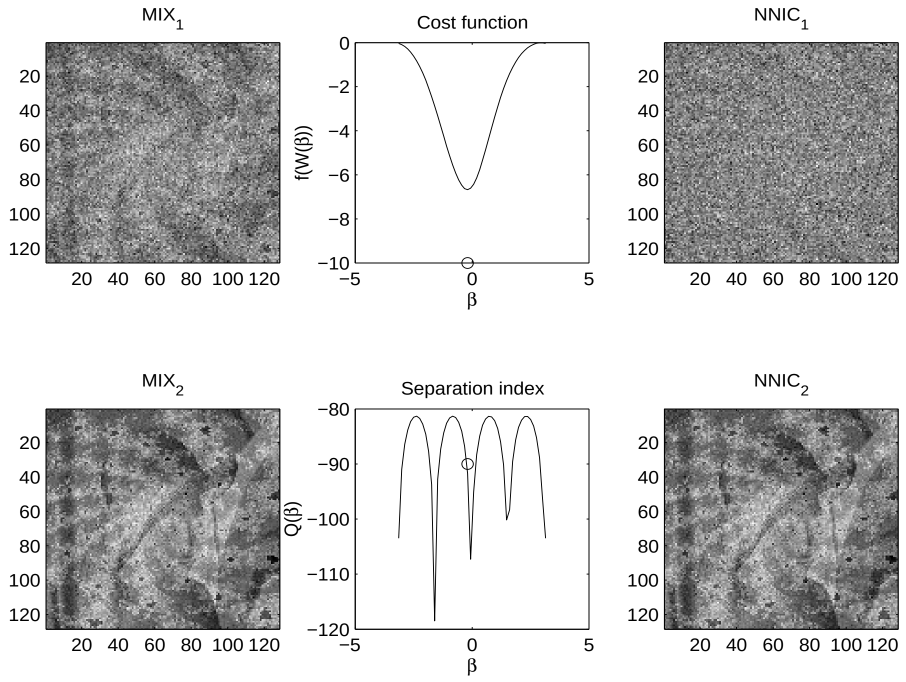


Figure 5: Images 1 and 2 strongly-noisy mixtures (MIX$_1$ and MIX$_2$), behavior of cost function f and separation index Q (shown in dB scales) and separated images (NNIC$_1$ and NNIC$_2$) obtained with the optimal separation angle. The open circle denotes the value of the the parameter β corresponding to the minimum criterion $f(\mathbf{W}(\beta))$ value.

4.3.3 NUMERICAL ANALYSIS AND COMPARISON OF THE ICA$^+$ ALGORITHMS

The first experiment of this section aims at investigating a 4×4 ICA$^+$ case tackled with the help of the deterministic-gradient-based algorithm endowed with geodesic search (Algorithm 2). In particular, in this case the optimal step-size is searched for within the interval $[-1, -0.1]$ partitioned into 10 bins and the random step-size generated in case of non-acceptance is a small random number uniformly picked in $[-0.1, 0[$. The maximum number of iterations has been fixed to 100 and the used images are number 1, 2, 3 and 4 of Figure 1.

The results of this experiment are shown in the Figures 6, 7 and 8.

In particular, the Figure 6 shows the behavior of the (normalized) separation index Q_n/Q_0 and of the cost function f_n versus the iteration index n. As these panels show, the separation index as well as the cost function values decrease from initial values to lower values, confirming that the separation behavior is good, in this experiment. The same Figure also shows the Frobenius norm of the Riemannian gradient $\tilde{\mathbf{V}}_n$ defined in equation (12), which decreases to low values during iteration, as well as the value of the ‘optimal’ learning step-size η_n selected at each iteration.

The Figure 7 shows a picture of the cost function as seen by the ‘geodesic search’ procedure: It shows, at each iteration, the shape of the cost function $f_n(\eta)$ as a function of the step-size η and shows the numerical minimal value to be selected as ‘optimal’ learning step-size. As explained in the description of the Algorithm 2, such value is actually selected only if the corresponding value

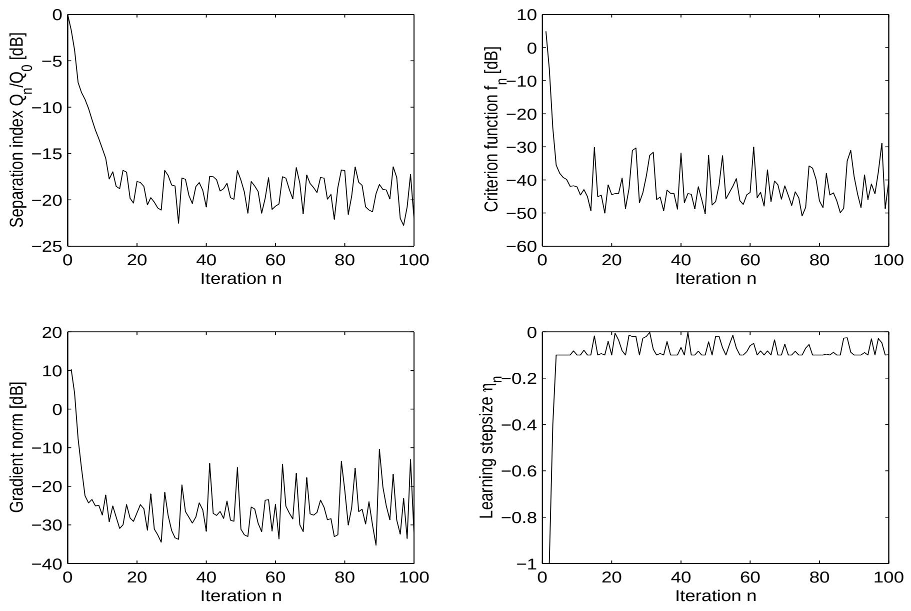


Figure 6: Four-source problem. Top-left: Normalized separation index versus the iteration index n. Top-right: Cost function f_n versus the iteration index n. Bottom-left: Norm of the Riemannian gradient of the ICA$^+$ cost function versus the iteration index n. Bottom-right: 'Optimal' learning step-size η_n selected at each iteration.

of the cost function is smaller than the value of the cost function achieved in the previous iteration, otherwise the result of the geodesic search is ignored and a small random step-size is selected. From the picture, it clearly emerges that the function $f_n(\eta)$ exhibits a only minimum in the interval of interest for η. Also, as the learning procedure progresses, the minimal value is almost always located at relatively low values of η because of the sharpness of the cost function around the optimal separating solution evidenced by the Figures 2 and 3.

The Figure 8 shows the result of this analysis for a randomly generated 4×4 mixing matrix with four source images. The de-mixing matrix is the optimal one as obtained by the learning procedure. The visual appearance of the obtained components confirms the quality of the blind recovering procedure.

The second experiment of this section aims at investigating a 9×9 ICA$^+$ case tackled with the help of the deterministic-gradient-based algorithm endowed with geodesic search (Algorithm 2). In particular, in this case the optimal step-size is searched for within the interval $[-2, -0.1]$ partitioned into 10 bins and the random step-size generated in case of non-acceptance is a small random number uniformly picked in $[-0.1, 0[$. The maximum number of iterations has been fixed to 200. The results of this experiment are shown in the Figures 9 and 10. In this experiment, the separated images have been recovered sufficiently faithfully.

The same separation problem was also tackled through the deterministic-gradient-based algorithm without geodesic search (Algorithm 1). From the previous experiment, it emerges that the 'optimal' value of the step-size is almost always selected within the interval $[-0.1, 0[$. Therefore, in this experiment, the learning step-size was fixed to -0.05 and the number of iterations was set

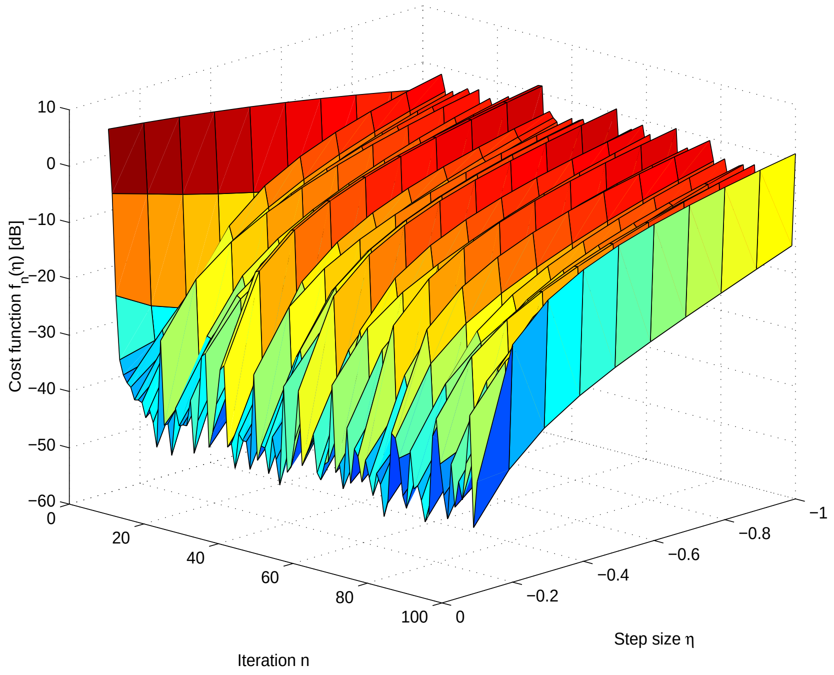


Figure 7: Four-source problem. Shape of the ICA$^+$ cost function as seen by the 'geodesic search' procedure.

to 400. It is worth noting that, in this case, not only the learning step-size was set to a constant value, but every move in the parameter manifold is accepted without checking if it actually leads to a decrease of the value of the learning criterion. The objective results of this experiment are shown in the Figure 11, while the resulting recovered components are not shown because they are similar to those illustrated in the Figure 10.

The nine-source separation problem was also tackled through the diffusion-type gradient-based algorithm (Algorithm 3). In this case, the learning step-size was set to -0.1, the initial temperature was set to $\Theta_0 = 0.5$ and the number of iterations was set to 400. The objective results of this experiment are shown in the Figure 12, while the resulting components are not shown because they are similar to those illustrated in the Figure 10.

As mentioned in Section 4.3.2, the behavior of Algorithms 1, 2 and 3 may be compared to the behavior of a non-Lie-group algorithm based on explicit orthogonalization via projection. Therefore, the nine-source separation problem was also tackled through the projection-based learning algorithm. In this case, the number of iterations was set to 400. The obtained results are not comforting about the suitability of this algorithm to non-negative independent component analysis. In spite that several values of the learning step-size were tried (ranging from -0.5 to -0.005), no good results were obtained, in this case. Two possible explanations of the observed behavior are that:

- The projection operation wastes the most part of the time in canceling out the component of the Euclidean gradient that is normal to the manifold instead of advancing the solution toward the most appropriate direction.

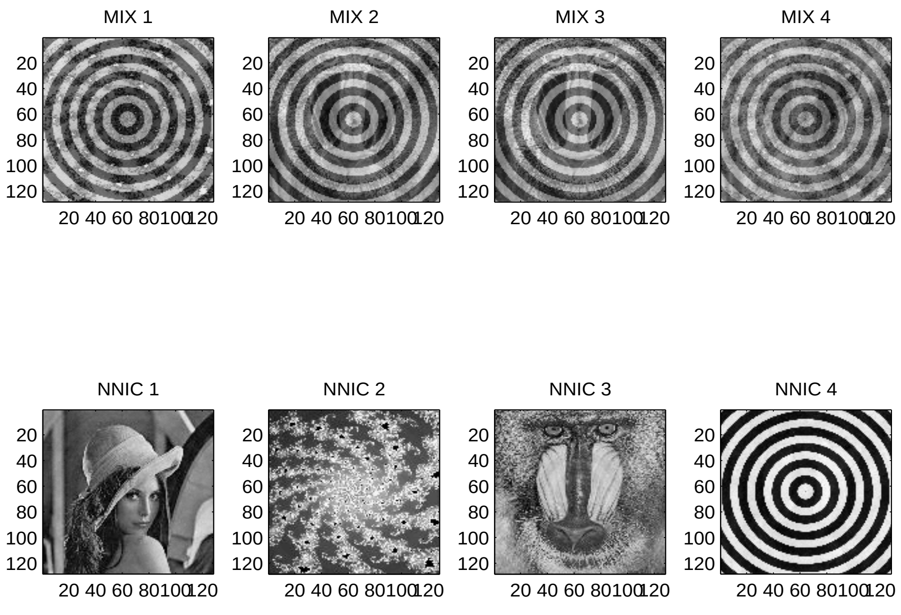


Figure 8: Four-source problem. Mixtures and separated images.

ALGORITHM	AVERAGE RUN-TIME (SEC.S)	FLOPS PER ITERATION
Algorithm 1	0.27	5.46×10^6
Algorithm 2	1.83	3.69×10^7
Algorithm 3	0.27	4.76×10^6
Projection	0.29	5.48×10^6

Table 1: Nine-source problem. Computational complexity comparison of Algorithms 1, 2, 3 and the projection-based learning algorithm (in terms of flops and run-time per iteration).

- The algorithm described by equations (22) and (23) looks essentially as fixed-point algorithm: Such kind of algorithms may easily get trapped in non-converging or very-slowly-converging cycles if the operator that describes the fixed-point iteration is not contractive. However, proving (or forcing) the convergence of such algorithms is far from being an easy task. A short discussion on this topic has been recently presented by Fiori (2002).

With regard to the computational complexity comparison of the algorithms on the nine-source separation problem, the number of flops per iteration and the average run-times per iteration are reported in the Table 1. It is worth underlining that both run-times and flop-counts depend on the platform and on the specific implementation of the algorithms, therefore only differences than span one or more magnitude orders should be retained as meaningful.

The conclusion of the above numerical analysis pertaining to the nine-source problem is quite straightforward: In the present problem, the adoption of the diffusion-type gradient is not beneficial as the initial ‘burn-in’ stage due to MCMC is quite long and the final achieved result is completely comparable to those exhibited by the other two algorithms. Among Algorithms 1 and 2, they achieve

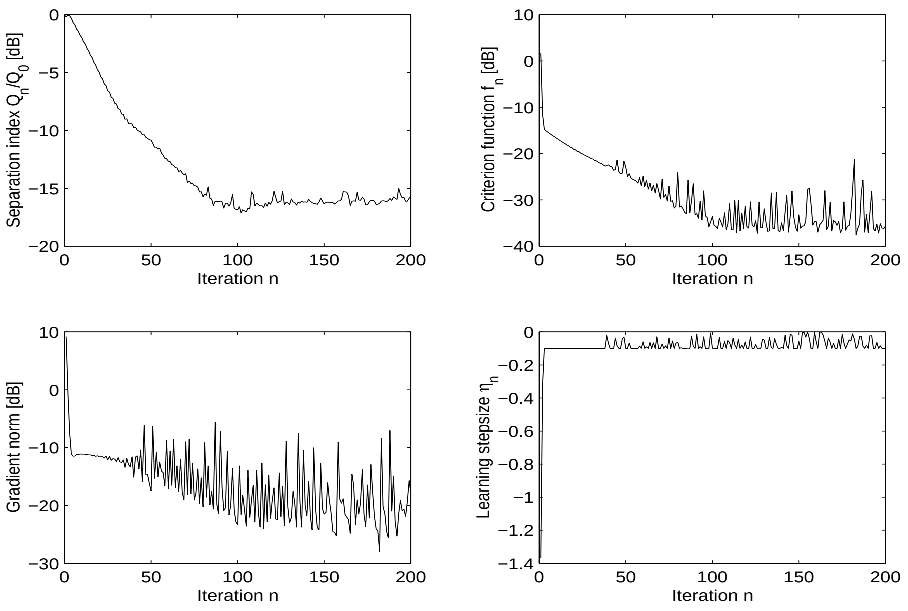


Figure 9: Nine-source problem, Algorithm 2. Top-left: Normalized separation index versus the iteration index n. Top-right: Cost function f_n versus the iteration index n. Bottom-left: Norm of the Riemannian gradient of the ICA$^+$ cost function versus the iteration index n. Bottom-right: 'Optimal' learning step-size η_n selected at each iteration.

comparable separation results, but the Algorithm 1 is definitely lighter, in terms of computational complexity, than the Algorithm 2. The computational complexity pertaining to the projection-based algorithm is comparable to the complexity exhibited by Algorithms 1 and 3.

5. Conclusion

The aim of the present tutorial was to illustrate learning algorithms based on Riemannian-gradient-based criterion optimization on the Lie group of orthogonal matrices. Although the presented differential-geometry-based learning algorithms have so far been mainly exploited in narrow contexts they may aid the design of general-purpose learning algorithms in those cases where a learning task may be formulated as an optimization one over a smooth manifold. The considered algorithms have been applied to non-negative independent component analysis both in the standard version equipped with geodesic-line search and in the diffusion-type gradient version.

The analytical developments evidenced the following advantages and similarities of the $O(p)$-type learning algorithm with respect to the existing $Gl(p)$-type algorithms:

- In the general case, the search for a connection pattern should be performed in the Lie group $Gl(p)$, while in the second case the search is performed in the orthogonal Lie group $O(p)$. The group $O(p)$ is compact (that is, closed and limited) therefore the stability of a $O(p)$-type learning algorithm is inherently ensured (up to machine precision), while this is not true for the $Gl(p)$-type learning algorithms.

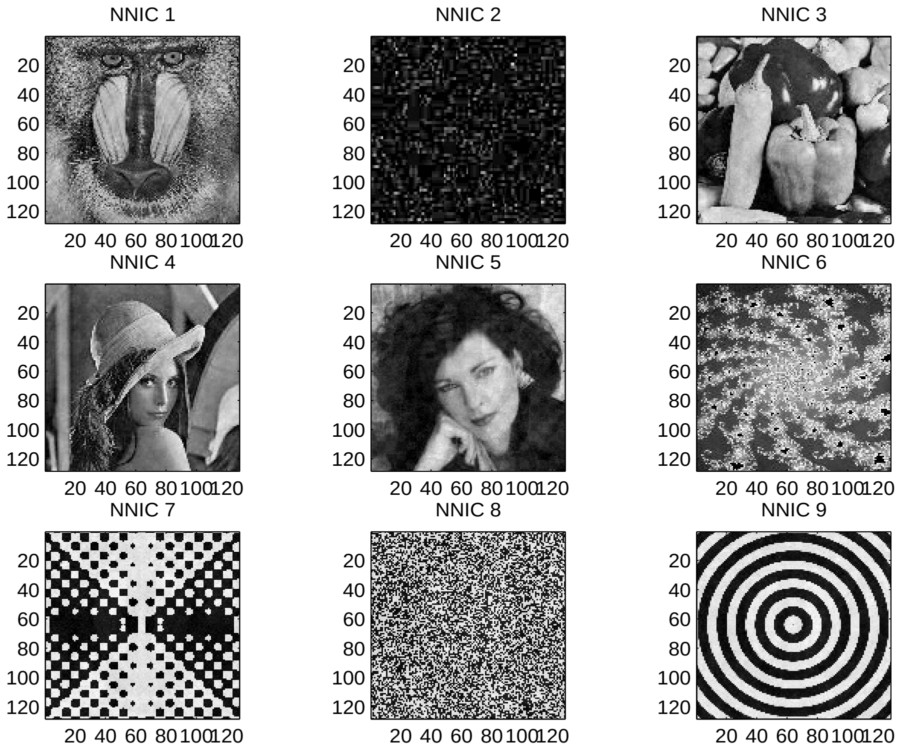


Figure 10: Nine-source problem, Algorithm 2. Separated images.

- In general, the $Gl(p)$-type learning algorithms cannot avoid quasi-degeneracy of the neural network, that is the case in which more than one neuron nearly happen to encode the same feature. In the context of $O(p)$-type learning algorithms, this case is inherently impossible.

- The possible amplification of the additive noise in the noisy ICA case is not avoided by the $O(p)$-type learning algorithms, even if care should be taken in this context of properly computing the pre-whitening operator. Even the $Gl(p)$-type learning algorithms, that do not require pre-whitening, cannot avoid the amplification of the disturbance on the observations.

The conclusions of the comparative analysis pertaining to the nine-source ICA problem are quite straightforward: The simple gradient adaptation, with a properly chosen learning step-size, is sufficient to achieve good separation performance at low computational burden. It deserves to remark, however, that the ‘geodesic search’ procedure automatically provides a suitable value of the learning step-size, which should be manually selected in absence of any tuning procedure.

It is worth underlining that the Algorithm 1, which appears to be the solution of choice in the context of ICA problem, as well as Algorithms 2 and 3, has been derived in a framework that is more general than ICA, but has only been applied it to ICA in the present manuscript. In the ICA$^+$ context, and with the chosen metric for the orthogonal group, the Algorithms 1 and 2 essentially coincide to the algorithms presented by Plumbley (2003). With respect to the work of Plumbley (2003), the conclusion we draw from the presented numerical analysis on ICA$^+$ problems is that, for general high-dimensional ICA$^+$ problems, the introduction of geodesic-search is not beneficial.

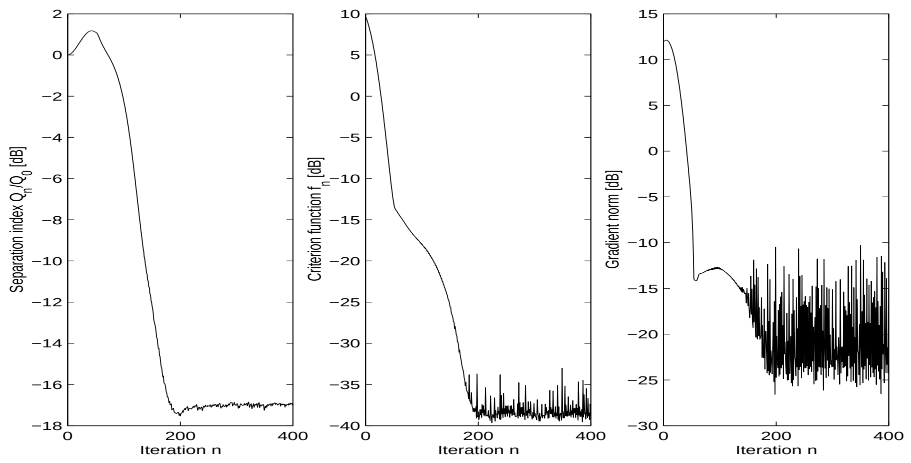


Figure 11: Nine-source problem, Algorithm 1. Left panel: Normalized separation index versus the iteration index *n*. Middle panel: Cost function f_n versus the iteration index *n*. Right: Norm of the Riemannian gradient of the ICA$^+$ cost function versus the iteration index *n*.

The same holds for the introduction of stochasticity under the form of annealed MCMC, that does not helped speeding up network learning convergence in the considered analysis.

About further and future efforts, we believe the following notes are worth mentioning:

- As a general remark on the computational complexity of the discussed algorithms, it is worth noting that the most burdensome operation is the computation of the exponential map in the updating rule (11). In the present paper we employed MATLAB's 'expm' primitive but, of course, several ways are known in the scientific literature to compute exponential maps. Two examples are the Cayley transform and the canonical coordinates of the first kind (interested readers might consult, for example, Celledoni and Fiori (2004) and references therein). A promising alternative solution would be to exploit the latest advancements in the field of numerical calculus on manifold for exponential maps computation, which should allegedly lead to a considerable saving of computational effort without detriment of separation effectiveness.

- As mentioned in the Section 2.1, learning algorithms based on the ordinary gradient and explicit orthogonalization (projection) are known in the scientific literature. The issue whether Lie-group methods are more advantageous, compared to methods based on the projection to the feasible set by orthogonalization, is currently being investigated.

- As it also emerges from Section 2.1, all the learning equations/algorithms developed in this manuscript are based on a particular choice of the metric that turns the Lie-algebra associated to the Lie group of orthogonal matrices into a metric space. Although, in principle, the choice of the metric may be shown not to affect the final result of learning, nor should it affect the learning path over the base-manifold, preliminary experiments suggest that the choice of metric indeed affects the behavior of discrete-time algorithms when implemented on a computer due to accumulation of numerical errors (Fiori, 2005).

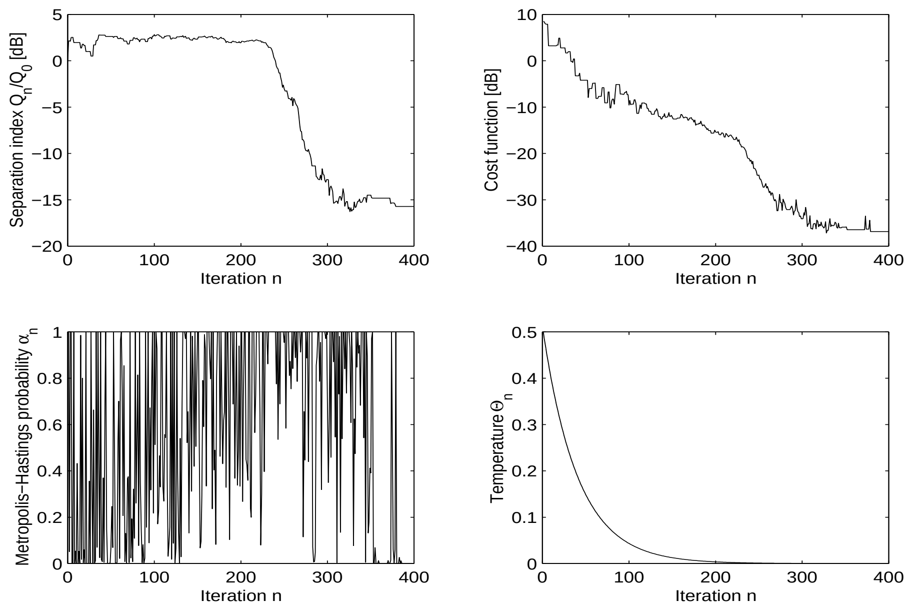


Figure 12: Nine-source problem, Algorithm 3. Top-left: Normalized separation index versus the iteration index n. Top-right: Cost function f_n versus the iteration index n. Bottom-left: Metropolis-Hastings probability α_n versus the iteration index n. Bottom-right: Simulated-annealing temperature Θ_n versus the iteration index n.

Acknowledgments

The author wishes to gratefully thank Elena Celledoni and Yasunori Nishimori for many insightful discussions on the differential geometry of the orthogonal group, Andrzej Cichocki for an insightful discussion on the noisy-mixture ICA case, Toshihisa Tanaka for kindly making it available the images used in the presented experiments, Mark Plumbley for kindly sharing viewpoints and codes on ICA$^+$ and Hichem Snoussi for the insightful discussions and useful suggestions about the theory and implementation of the MCMC sampling method.

The author also wishes to gratefully thank the JMLR Action Editor who handled the submission of this paper, Yoshua Bengio, and the anonymous reviewers, for providing careful and detailed comments that helped improving the clarity and thoroughness of the presented scientific material.

The penultimate version of this manuscript was prepared while the author was a visiting researcher at the Faculty of Information Technology, Mathematics and Electrical Engineering of Norwegian University of Science and Technology (Trondheim, Norway) during January-February 2005. The author wishes to gratefully thank Elena Celledoni for making this fruitful visit be possible.

The last version of this manuscript was prepared while the author was a visiting professor at the Laboratory for Signal and Image Processing of the Tokyo University of Agriculture and Technology (Tokyo, Japan) during March-April 2005. The author wishes to gratefully thank Toshihisa Tanaka for making this fruitful visit be possible.

Appendix A. Geodesic Equation and Relevant Properties

In the present appendix, we consider the problem of constructing a geodesic curve on a Riemannian manifold $(\mathcal{M}, g)$ and illustrate some relevant properties of geodesics on Riemannian manifolds embedded in a Euclidean ambient space $\mathbb{R}^p$. The result of the following calculation will be a second-order differential equation in the components x_k of x ($k = 1,\ 2,\ \cdots,\ p$).[7]

Before considering the problem of geodesic calculation, it is instrumental to consider the general variational problem of minimizing the functional:

$$A \stackrel{\text{def}}{=} \int_{t_0}^{t_1} H(x, \dot{x}) dt, \tag{25}$$

where $H : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ is a potential function, $x = x(t)$ is a curve on $\mathcal{M}$ with parameter $t \in [t_0\ ,\ t_1]$ and A is an integral functional of $x(t)$ (sometimes termed *action*). In the above equation and thereafter, overdots denote derivation with respect to the parameter t.

It is know that, under proper conditions, the solution of the above variational problem is given by the solution of the Euler-Lagrange equation:

$$\frac{\partial H}{\partial x_k} - \frac{d}{dt}\frac{\partial H}{\partial \dot{x}_k} = 0\ ,\ k = 1,\ 2,\ \ldots,\ p.$$

By comparing the equation (25) and the curve-length equation (2), it is readily seen that, in order to set a curve-length minimization problem into an action minimization problem, it suffices to set $H(x, \dot{x}) = \sqrt{g_x(\dot{x}, \dot{x})}$ in the above setting. To this purpose, it is worth noting that, thanks to the bi-linearity of the scalar product and according to the decomposition $\dot{x} = \sum_i \dot{x}_i e_i$, where $\{e_i\}$ denotes whatever basis of $\mathbb{R}^p$, it holds $g_x(\dot{x}, \dot{x}) = \sum_i \sum_j g_{ij} \dot{x}_i \dot{x}_j$, where the functions $g_{ij} \stackrel{\text{def}}{=} g_x(e_i, e_j)$ denote the components of the so-termed *metric tensor* and specify completely the metric properties of the manifold $\mathcal{M}$. The components of the metric tensor are functions of the coordinates $x_1,\ \cdots,\ x_p$. The metric tensor is symmetric, that is, $g_{ij} = g_{ji}$ for every $i, j \in \{1,\ 2,\ \ldots,\ p\}$ and non-singular, that is its inverse exists everywhere.

By replacing the above expression of the potential into the Euler-Lagrange equation and calculating the required derivatives, we get

$$\sum_i \sum_j \frac{\partial g_{ij}}{\partial x_k} \dot{x}_i \dot{x}_j - 2 \sum_i \frac{d g_{ik}}{dt} \dot{x}_i - 2 \sum_i g_{ik} \ddot{x}_i = 0.$$

Now, the following identities are of use:

$$\sum_i \frac{d g_{ik}}{dt} \dot{x}_i = \sum_i \sum_\ell \frac{\partial g_{ik}}{\partial x_\ell} \dot{x}_i \dot{x}_\ell = \sum_i \sum_\ell \frac{\partial g_{\ell k}}{\partial x_i} \dot{x}_i \dot{x}_\ell,$$

because the indices i and ℓ may be swapped in the second-last expression. Then the equation of minimizing curve becomes:

$$\sum_i g_{ik} \ddot{x}_i + \frac{1}{2} \sum_i \sum_j \frac{\partial g_{ik}}{\partial x_j} \dot{x}_i \dot{x}_j + \frac{1}{2} \sum_i \sum_j \frac{\partial g_{jk}}{\partial x_i} \dot{x}_i \dot{x}_j - \frac{1}{2} \sum_i \sum_j \frac{\partial g_{ij}}{\partial x_k} \dot{x}_i \dot{x}_j = 0.$$

7. In the present paper, we do not make use of the standard covariant/contra-variant notation for tensor indices nor of the Einstein convention for summations.

It is now worth introducing the inverse of the metric tensor, whose elements are denoted by g^{ab}, defined by the equations $\sum_b g^{ab} g_{bc} = \delta^a_c$, where δ^a_c denotes the fundamental tensor (and may be regarded as a Kronecker 'delta'). By multiplying both sides of the above equation by $g^{\ell k}$ and summing with respect to k, the result is

$$\sum_k \sum_i g_{ik} g^{k\ell} \ddot{x}_i + \frac{1}{2} \sum_k \sum_i \sum_j g^{k\ell} \left(\frac{\partial g_{ik}}{\partial x_j} + \frac{\partial g_{jk}}{\partial x_i} - \frac{\partial g_{ij}}{\partial x_k} \right) \dot{x}_i \dot{x}_j = 0.$$

Let us further define the Christoffel (or affine connection) coefficients as

$$\Gamma^k_{ij} \stackrel{\text{def}}{=} \frac{1}{2} \sum_\ell g^{k\ell} \left(\frac{\partial g_{i\ell}}{\partial x_j} + \frac{\partial g_{j\ell}}{\partial x_i} - \frac{\partial g_{ij}}{\partial x_\ell} \right),$$

through which the geodesic equation assumes the classical expression:

$$\ddot{x}_k + \sum_i \sum_j \Gamma^k_{ij} \dot{x}_i \dot{x}_j = 0 \, , \; k = 1 \, , 2 \, , \, \cdots \, , \; p. \tag{26}$$

As anticipated, it appears under the form of a set of second-order differential equations in the coordinates x_k and needs therefore two boundary conditions. These may specify the geodesic endpoints: $x(t_0) = x_0 \in \mathcal{M}$ and $x(t_1) = x_1 \in \mathcal{M}$, or the initial position and initial velocity: $x(t_0) = x_0 \in \mathcal{M}$ and $\dot{x}(t_0) = \mathbf{v}_0 \in T_{x_0}\mathcal{M}$.

A result we make use of in the paper is that, when a Riemannian manifold is embedded into an Euclidean space, the second derivative of the geodesic ($\ddot{x}$) belongs to the normal space to the embedded manifold at x. Let us begin the proof of this important property by proving that, along a geodesic, the quantity $g_x(\dot{x}, \dot{x})$ is constant with respect to the parameter t or, equivalently, that $\frac{d}{dt} g_x(\dot{x}, \dot{x}) = 0$. We have

$$\begin{aligned}
\frac{d}{dt} g_x(\dot{x}, \dot{x}) &= \frac{d}{dt} \sum_a \sum_b g_{ab} \dot{x}_a \dot{x}_b \\
&= \sum_a \sum_b \left(g_{ab} \ddot{x}_a \dot{x}_b + g_{ab} \dot{x}_a \ddot{x}_b + \frac{dg_{ab}}{dt} \dot{x}_a \dot{x}_b \right) \\
&= 2 \sum_a \sum_b g_{ab} \ddot{x}_a \dot{x}_b + \sum_a \sum_b \frac{dg_{ab}}{dt} \dot{x}_a \dot{x}_b.
\end{aligned}$$

By replacing the expression of $\ddot{x}_a$ from the geodesic equation (26) into the last expression, we get

$$\frac{d}{dt} g_x(\dot{x}, \dot{x}) = -2 \sum_a \sum_b \sum_i \sum_j g_{ab} \Gamma^a_{ij} \dot{x}_b \dot{x}_i \dot{x}_j + \sum_a \sum_b \frac{dg_{ab}}{dt} \dot{x}_a \dot{x}_b.$$

Now, the following identity holds:

$$\sum_b g_{ab} \Gamma^b_{ij} = \frac{1}{2} \left(\frac{\partial g_{ia}}{\partial x_j} + \frac{\partial g_{ja}}{\partial x_i} - \frac{\partial g_{ij}}{\partial x_a} \right),$$

thus it may be further written:

$$\begin{aligned}
\frac{d}{dt} g_x(\dot{x}, \dot{x}) &= -\sum_a \sum_i \sum_j \frac{\partial g_{ia}}{\partial x_j} \dot{x}_j \dot{x}_i \dot{x}_a - \sum_a \sum_i \sum_j \frac{\partial g_{ja}}{\partial x_i} \dot{x}_i \dot{x}_j \dot{x}_a \\
&+ \sum_a \sum_i \sum_j \frac{\partial g_{ij}}{\partial x_a} \dot{x}_a \dot{x}_i \dot{x}_j + \sum_a \sum_b \frac{dg_{ab}}{dt} \dot{x}_a \dot{x}_b.
\end{aligned}$$

It is readily recognized that, e.g., $\sum_j \frac{\partial g_{ia}}{\partial x_j}\dot{x}_j = \frac{dg_{ia}}{dt}$, therefore, all the sums in the above expression are equal up to proper index re-ordering/renaming. As a consequence, the four terms sum up to zero.

The last step consists in recalling that the manifold has been supposed to be embedded in a Euclidean ambient space and we assume $g_x(\dot{x},\dot{x}) \overset{\text{def}}{=} \dot{x}^T\dot{x}$. Its derivative is thus $\frac{d}{dt}g_x(\dot{x},\dot{x}) = 2\ddot{x}^T\dot{x} = 0$, which proves that, under the specified conditions, the second derivative $\ddot{x}$ is orthogonal to the first derivative $\dot{x}$ in any point of the embedded geodesic.

References

S.-i. Amari. *Differential-Geometrical Methods in Statistics*. Lecture Notes in Statistics 28, Springer-Verlag, 1989.

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

T. Akuzawa. New fast factorization method for multivariate optimization and its realization as ICA algorithm. In *Proceedings of the 3rd International Conference on Independent Component Analysis and Blind Signal Separation* pages 114–119, San Diego, California, USA, 2001

E. Celledoni and S. Fiori. Neural learning by geometric integration of reduced ‘rigid-body’ equations. *Journal of Computational and Applied Mathematics*, 172(2):247–269, 2004.

A. Cichocki and S.-i. Amari. *Adaptive Blind Signal and Image Processing*, J. Wiley & Sons, 2002

S. Fiori. A theory for learning by weight flow on Stiefel-Grassman manifold. *Neural Computation*, 13(7):1625–1647, 2001.

S. Fiori. A theory for learning based on rigid bodies dynamics. *IEEE Trans. on Neural Networks*, 13(3):521–531, 2002.

S. Fiori. A fast fixed-point neural blind deconvolution algorithm. *IEEE Trans. on Neural Networks*, 15(2):455–459, 2004.

S. Fiori. Non-linear complex-valued extensions of Hebbian learning: An essay. *Neural Computation*, 17(4):779–838, 2005.

S. Fiori. Formulation and integration of learning differential equations on the Stiefel manifold. *IEEE Trans. on Neural Networks*, forthcoming.

S. Fiori and S.-i. Amari. Editorial: Special issue on “Geometrical Methods in Neural Networks and Learning”, *Neurocomputing*, forthcoming.

W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57:97–109, 1970.

A. Hyvärinen, J. Karhunen and E. Oja. *Independent Component Analysis,* John Wiley & Sons, 2001.

D. J. Higham. An algorithmic introduction to numerical simulation of stochastic differential equations. *SIAM Review*, 43(3):525–546, 2001.

R. E. Kass, B. P. Carlin, A. Gelman and R. M. Neal. Markov Chain Monte Carlo in practice: A roundtable discussion. *The American Statistician*, 52(2):93–100, 1998.

N. Keshava and J. F. Mustard. Spectral unmixing. *IEEE Signal Processing Magazine*, 19(1):44–57, 2002.

X. Liu, A. Srivastava and K. Gallivan. Optimal linear representation of images for object recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26(5):662–666, 2004.

N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1091, 1953.

Y. Nishimori. Learning algorithm for ICA by geodesic flows on orthogonal group. In *Proc. of the International Joint Conference on Neural Networks* pages 1625–1647, 1999.

P. J. Olver. Applications of Lie groups to differential equations. *Graduate Texts in Mathematics 107*, Second Edition, Springer, 2003.

H. Park, S.-i. Amari and K. Fukumizu. Adaptive Natural Gradient Learning Algorithms for Various Stochastic Models. *Neural Networks*, 13:755–764, 2000.

M. D. Plumbley. Conditions for non-negative independent component analysis. *IEEE Signal processing Letters*, 9(6):177–180, 2002.

M. D. Plumbley. Algorithms for nonnegative independent component analysis. *IEEE Trans. on Neural Networks*, 14(3):534–543, 2003.

M. D. Plumbley. Lie group methods for optimization with orthogonality constraints. In *Proceedings of the International Conference on Independent Component Analysis and Blind Signal Separation*, pages 1245–1252, Granada, Spain, 2004.

A. Srivastava, U. Grenander, G. R. Jensen and M. I. Miller. Jump-diffusion Markov processes on orthogonal groups for object recognition. *Journal of Statistical Planning and Inference*, 103(1/2):15–37, 2002.

H. H. Yang and S.-i. Amari. Adaptive online learning algorithms for blind separation: Maximum entropy and minimum mutual information. *Neural Computation*, 9:1457–1482, 1997.

G. R. Warnes. The normal kernel coupler: An adaptive MCMC method for efficiently sampling from multi-modal distributions. Technical Report 39, Dept. of Statistics, University of Washington, 2001.

D. R. Wilson and T. R. Martinez. The general inefficiency of batch training for gradient descent learning. *Neural Networks*, 16(10):1429–1451, 2003.

L.-Q. Zhang, A. Cichocki and S.-i. Amari. Geometrical structures of FIR manifold and multichannel blind deconvolution. *Journal of VLSI for Signal Processing Systems*, 31:31–44, 2002.

Journal of Machine Learning Research 6 (2005) 783–816 Submitted 11/03; Revised 12/04; Published 5/05

Machine Learning Methods for Predicting Failures in Hard Drives: A Multiple-Instance Application

Joseph F. Murray JFMURRAY@JFMURRAY.ORG
Electrical and Computer Engineering, Jacobs Schools of Engineering
University of California, San Diego
La Jolla, CA 92093-0407 USA

Gordon F. Hughes GFHUGHES@UCSD.EDU
Center for Magnetic Recording Research
University of California, San Diego
La Jolla, CA 92093 USA

Kenneth Kreutz-Delgado KREUTZ@ECE.UCSD.EDU
Electrical and Computer Engineering, Jacobs Schools of Engineering
University of California, San Diego
La Jolla, CA 92093-0407 USA

Editor: Dale Schuurmans

Abstract

We compare machine learning methods applied to a difficult real-world problem: predicting computer hard-drive failure using attributes monitored internally by individual drives. The problem is one of detecting rare events in a time series of noisy and nonparametrically-distributed data. We develop a new algorithm based on the multiple-instance learning framework and the naive Bayesian classifier (mi-NB) which is specifically designed for the low false-alarm case, and is shown to have promising performance. Other methods compared are support vector machines (SVMs), unsupervised clustering, and non-parametric statistical tests (rank-sum and reverse arrangements). The failure-prediction performance of the SVM, rank-sum and mi-NB algorithm is considerably better than the threshold method currently implemented in drives, while maintaining low false alarm rates. Our results suggest that nonparametric statistical tests should be considered for learning problems involving detecting rare events in time series data. An appendix details the calculation of rank-sum significance probabilities in the case of discrete, tied observations, and we give new recommendations about when the exact calculation should be used instead of the commonly-used normal approximation. These normal approximations may be particularly inaccurate for rare event problems like hard drive failures.

Keywords: hard drive failure prediction, rank-sum test, support vector machines (SVM), exact nonparametric statistics, multiple instance naive-Bayes

1. Introduction

We present a comparison of learning methods applied to a difficult real-world pattern recognition problem: predicting impending failure in hard disk drives. Modern hard drives are reliable devices, yet failures can be costly to users and many would benefit from a warning of potential problems that would give them enough time to backup their data. The problem can be characterized as one of detecting rare events from a time series of noisy and nonparametrically-distributed attributes.

Hard drive manufacturers have been developing self-monitoring technology in their products since 1994, in an effort to predict failures early enough to allow users to backup their data (Hughes et al., 2002). This Self-Monitoring and Reporting Technology (SMART) system uses attributes collected during normal operation (and during off-line tests) to set a failure prediction flag. The SMART flag is a one-bit signal that can be read by operating systems and third-party software to warn users of impending drive failure. Some of the attributes used to make the failure prediction include counts of track-seek retries, read errors, write faults, reallocated sectors, head fly height too low or high, and high temperature. Most internally-monitored attributes are error count data, implying positive integer data values, and a pattern of increasing attribute values (or their rates of change) over time is indicative of impending failure. Each manufacturer develops and uses its own set of attributes and algorithm for failure prediction. Every time a failure warning is triggered the drive can be returned to the factory for warranty replacement, so manufacturers are very concerned with reducing the false alarm rates of their algorithms. Currently, all manufacturers use a threshold algorithm which triggers a SMART flag when any single attribute exceeds a predefined value. These thresholds are set conservatively to avoid false alarms at the expense of predictive accuracy, with an acceptable false alarm rate on the order of 0.1% per year (that is, one drive in 1000). For the SMART algorithm currently implemented in drives, manufacturers estimate the failure detection rate to be 3-10%. Our previous work has shown that by using nonparametric statistical tests, the accuracy of correctly detected failures can be improved to as much as 40-60% while maintaining acceptably low false alarm rates (Hughes et al., 2002; Hamerly and Elkan, 2001).

In addition to providing a systematic comparison of prediction algorithms, there are two main novel algorithmic contributions of the present work. First, we cast the hard drive failure prediction problem as a multiple-instance (MI) learning problem (Dietterich et al., 1997) and develop a new algorithm termed multiple-instance naive Bayes (mi-NB). The mi-NB algorithm adheres to the strict MI assumption (Xu, 2003) and is specifically designed with the low false-alarm case in mind. Our second contribution is to highlight the effectiveness and computational efficiency of nonparametric statistical tests in failure prediction problems, even when compared with powerful modern learning methods. We show that the rank-sum test provides good performance in terms of achieving a high failure detection rate with low false alarms at a low computational cost. While the rank-sum test is not a fully general learning method, it may prove useful in other problems that involve finding outliers from a known class. Other methods compared are support vector machines (SVMs), unsupervised clustering using the Autoclass software of Cheeseman and Stutz (1995) and the reverse-arrangements test (another nonparametric statistical test) (Mann, 1945). The best performance overall was achieved with SVMs, although computational times were much longer and there were many more parameters to set.

The methods described here can be used in other applications where it is necessary to detect rare events in time series including medical diagnosis of rare diseases (Bridge and Sawilowsky, 1999; Rothman and Greenland, 2000), financial forecasting such as predicting business failures and personal bankruptcies (Theodossiou, 1993), and predicting mechanical and electronic device failure (Preusser and Hadley, 1991; Weiss and Hirsh, 1998).

1.1 Previous Work in Hard Drive Failure Prediction

In our previous work (Hughes et al., 2002) we studied the SMART failure prediction problem, comparing the manufacturer-selected decision thresholds to the rank-sum statistical test. The data set used was from the Quantum Corporation, and contained data from two drive models. The data set used in the present paper is from a different manufacturer, and includes many more attributes (61 vs. 14), which is indicative of the improvements in SMART monitoring that have occurred since the original paper. An important observations made by Hughes et al. (2002) was that many of the SMART attributes are *nonparametrically distributed*, that is, their distributions cannot be easily characterized by standard parametric statistical model (such as normal, Weibull, chi-squared, etc.). This observation led us to investigate nonparametric statistical tests for comparing the distribution of a test drive attribute to the known distribution of good drives. Hughes et al. (2002) compared single-variate and multivariate rank-sum tests with simple thresholds. The single-variate test was combined for multiple attributes using a logical OR operation, that is, if any of the single attribute tests indicated that the drive was not from the good population, then the drive was labeled failed. The OR-ed test performed slightly better than the multivariate for most of the region of interest (low false alarms). In the present paper we use only the single-variate rank-sum test (OR-ed decisions) and compare additional machine learning methods, Autoclass and support vector machines. Another method for SMART failure prediction, called *naive Bayes EM* (expectation-maximization), using the original Quantum data was developed by Hamerly and Elkan (2001). The naive Bayes EM is closely related to the Autoclass unsupervised clustering method used in the present work. Using a small subset of the features provided better performance than using all the attributes. Some preliminary results with the current SMART data were presented in Murray et al. (2003).

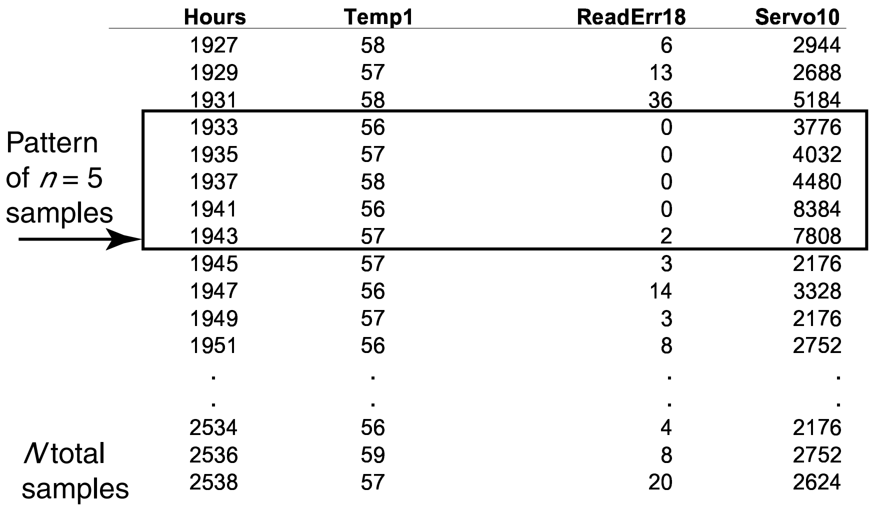
1.2 Organization

This paper is organized as follows: In Section 2, we describe the SMART data set used here, how it differs from previous SMART data and the notation used for drives, patterns, samples, etc. In Section 3, we discuss feature selection using statistical tests such as reverse arrangements and z-scores. In Section 4, we describe the multiple instance framework, our new algorithm multiple-instance naive-Bayes (mi-NB), the failure prediction algorithms, including support vector machines, unsupervised clustering and the rank-sum test. Section 5 presents the experimental results comparing the classifiers used for failure prediction and the methods of preprocessing. A discussion of our results is given in Section 6 and conclusions are presented in Section 7. An Appendix describes the calculation of rank-sum significance levels for the discrete case in the presence of tied values, and new recommendations are given as to when the exact test should be used instead of the standard approximate calculation.

2. Data Description

The data set consists of time series of SMART attributes from a single drive model, and is a different data set than that used in Hughes et al. (2002); Hamerly and Elkan (2001).[1] Data from 369 drives were collected, and each drive was labeled *good* or *failed*, with 178 drives in the good class and 191 drives in the failed class. Drives labeled as good were from a reliability test, run in a controlled environment by the manufacturer. Drives labeled as failed were returned to the manufacturer from users after a failure. It should be noted that since the good drive data were collected in a controlled uniform environment and the failed data come from drives that were operated by users, it is reasonable to expect that there will be differences between the two populations due to the different

1. The SMART data set used in this paper is available at `http://cmrr.ucsd.edu/smart`.


Pattern
of $n = 5$
samples
N total
samples

Hours	Temp1	ReadErr18	Servo10
1927	58	6	2944
1929	57	13	2688
1931	58	36	5184
1933	56	0	3776
1935	57	0	4032
1937	58	0	4480
1941	56	0	8384
1943	57	2	7808
1945	57	3	2176
1947	56	14	3328
1949	57	3	2176
1951	56	8	2752
.	.	.	.
.	.	.	.
2534	56	4	2176
2536	59	8	2752
2538	57	20	2624

Figure 1: Selected attributes from a single good drive. Each row of the table represents a sample (all attributes recorded for a single time interval). The box shows the n selected consecutive samples in each pattern $\mathbf{x}_j$ used to make a failure prediction at the time pointed at by the arrow. The first sample available in the data set for this drive is from Hours = 1927, as only the most recent 300 samples are stored in drives of this model.

manner of operation. Algorithms that attempt to learn the difference between the good and failed populations may in fact be learning this difference and not the desired difference between good and nearly-failing drive samples. We highlight this point to emphasize the importance of understanding the populations in the data and considering alternative reasons for differences between classes.

A *sample* is all the attributes for a single drive for a single time interval. Each SMART sample was taken at two hour intervals in the operating drives, and the most recent 300 samples are saved on the disk. The number of available valid samples for each drive i is denoted N_i, and N_i may be less than 300 for those drives that did not survive 600 hours of operation. Each sample contains the drive's serial number, the total power-on-hours, and 60 other performance-monitoring attributes. Not all attributes are monitored in every drive, and the unmonitored attributes are set to a constant, non-informative value. Note that there is no fundamental reason why only 300 samples were collected; this was a design choice made by the drive manufacturer. Methods exist by which all samples over the course of the drive's life can be recorded for future analysis. Figure 1 shows some selected attributes from a single good drive, and examples of samples (each row) and patterns (the boxed area). When making a failure prediction a *pattern* $\mathbf{x}_j \in \mathbb{R}^{n \cdot a}$ (where a is the number of attributes) is composed of the n consecutive samples and used as input to a classifier. In our experiments n was a design parameter which varied between 1 and 100. The pair $(X_i, \mathcal{Y}_i)$ represents the data in each drive, where the set of patterns is $X_i = [\mathbf{x}_1, \ldots, \mathbf{x}_{N_i}]$ and the classification is $\mathcal{Y}_i \in \{0, 1\}$. For drives labeled good, $\mathcal{Y}_i = 0$ and for failed drives $\mathcal{Y}_i = 1$.

Hughes et al. (2002) used a data set from a different manufacturer which contained many more drives (3744 vs. 369) but with fewer failed drives (36 vs. 191). The earlier data set contained fewer attributes (14 vs. 61), some of which are found in the new data set but with different names and possibly different methods of measurement. Also, all good and failed drive data were collected during a single reliability test (whereas in the current set, the failed drives were returns from the field).

A preliminary examination of the current set of SMART data was done by plotting the histograms of attributes from good and failed drives. Figure 2 shows histograms of some representative attributes. As was found with earlier SMART data, for many of the attributes the distributions are difficult to describe parametrically as they may be multimodal (such as the Temp4 attribute) or very heavy tailed. Also noteworthy, many attributes have large numbers of zero values, and these zero-count bins are truncated in the plots. These highly non-Gaussian distributions initially lead us to investigate nonparametric statistical tests as a method of failure prediction. For other pattern recognition methods, special attention should be paid to scaling and other preprocessing.

3. Feature Selection

The process of feature selection includes not only deciding which attributes to use in the classifier, but also the number of time samples, n, used to make each decision, and whether to perform a preprocessing transformation on these input time series. Of course, these choices depend strongly on which type of classifier is being used, and issues of feature selection will also be discussed in the following sections.

As will be demonstrated below, some attributes are not strongly correlated with future drive failure and including these attributes can have a negative impact on classifier performance. Because it is computationally expensive to try all combinations of attribute values, we use the fast nonparametric reverse-arrangements test and attribute z-scores to identify potentially useful attributes. If an attribute appeared promising with either method it was considered for use in the failure detection algorithms (see Section 4).

3.1 Reverse Arrangements Test

The *reverse arrangements test* is a nonparametric test for trend which is applied to each attribute in the data set (Mann, 1945; Bendat and Piersol, 2000). It is used here based on the idea that a pattern of increasing drive errors is indicative of failure. Suppose we have a time sequence of observations of a random variable, $x_i, i = 1...N$. In our case x_i could be, for example, the seek error count of the most recent sample. The test statistic, $A = \sum_{i=1}^{N-1} A_i$, is the sum of all *reverse arrangements*, where a reverse arrangement is defined as an occurrence of $x_i > x_j$ when $i < j$. To find A we use the intermediate sums A_i and the indicator function h_{ij},

$$A_i = \sum_{j=i+1}^{N} h_{ij} \quad \text{where} \quad h_{ij} = I(x_i > x_j) \ .$$

We now give an example of calculating A for the case of $N = 10$. With data $\mathbf{x}$ (which is assumed to be a permutation of the ranks of the measurements),

$$\mathbf{x} = [x_1, \ldots, x_{10}] = [1, 4, 3, 7, 2, 8, 6, 10, 9, 5] \ ,$$

the values of A_i for $i = 1 \ldots 9$ are found,

$$A_1 = \sum_{j=2}^{10} h_{1j} = 0\,, \quad A_2 = \sum_{j=3}^{10} h_{2j} = 2\,, \quad \ldots \quad A_9 = \sum_{j=9}^{10} h_{9j} = 1\ ,$$

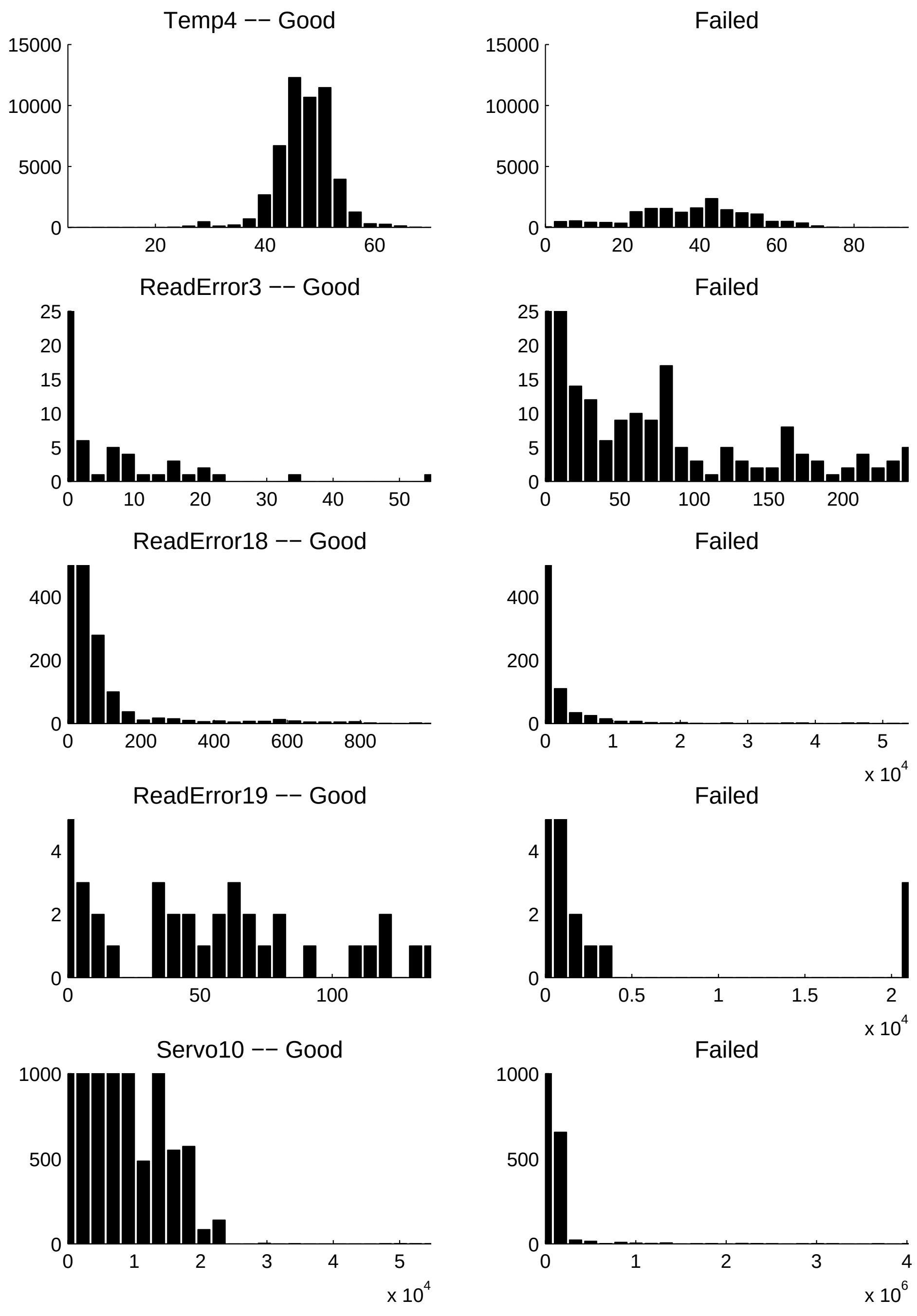


Figure 2: Histograms of representative attributes from good and failed drives, illustrating the non-parametric nature of many of the attributes. Axis scales are different for each plot to emphasize features of their distributions. Zero-count bins are much larger than plotted and the count-axis is shortened accordingly.

with the values $[A_i] = [0,2,1,3,0,2,1,2,1]$. The test statistic A is the sum of these values, $A = 12$.

For large values of N, the test statistic A is normally distributed under the null hypothesis of no trend (all measurements are random with the same distribution) with mean and variance (Mann, 1945),

$$\mu_A = \frac{N(N-1)}{4}\ , \quad \sigma_A^2 = \frac{2N^3 + 3N^2 - 5N}{72}\ .$$

For small values of N, the distribution can be calculated exactly by a recursion (Mann, 1945, eq. 1). First, we find the count $C_N(A)$ of permutations of $\{1,2,\ldots,N\}$ with A reverse arrangements,

$$C_N(A) = \sum_{i=A-N+1}^{A} C_{N-1}(i)\ ,$$

where $C_N(A) = 0$ for $A < 0$ and $C_0(A) = 0$. Since every permutation is equally likely with probability $\frac{1}{n!}$ under the null hypothesis, the probability of A is $\frac{C_N(A)}{n!}$.

Tables of the exact significance levels of A have been made. For significance level α, Appendix Table A.6 of Bendat and Piersol (2000) gives the acceptance regions,

$$A_{N;1-\alpha/2} < A \leq A_{N;\alpha/2}\ ,$$

for the null hypothesis of no trend in the sequence x_i (that is, that x_i are independent observations of the same underlying random variable).

The test is formulated assuming that the measurements are drawn from a continuous distribution, so that the ranks $\mathbf{x}$ are distinct (no ties). SMART error count data values are discrete and allow the possibility of ties. It is conventional in rank-based methods to add random noise to break the ties, or to use the *midrank* method described in Section 4.6.

3.2 Z-scores

The *z-score* compares the mean values of each attribute in either class (good or failed). It is calculated over all samples,

$$z = \frac{m_f - m_g}{\sqrt{\frac{\sigma_f^2}{n_f} + \frac{\sigma_g^2}{n_g}}}\ ,$$

where m_f and σ_f^2 are the mean and variance of the attribute in failed drives, m_g and σ_g^2 are the mean and variance in good drives, n_f and n_g are the total number of samples of failed and good drives. Large positive z-scores indicate the attribute is higher in the population of failed drive samples, and that there is likely a significant difference in the means between good and failed samples. However, it should be noted that the z-score was developed in the context of Gaussian statistics, and may be less applicable to nonparametric data (such as the error count attributes collected by hard drives).

3.3 Feature Selection for SMART Data

To apply the reverse arrangements test to the SMART data for the purpose of feature extraction, the test is performed on a set of 100 samples taken at the end of the time series available. To break ties, uniform random noise within the range $[-0.1, 0.1]$ is added to each value (which are initially non-negative integers). The percentage of drives for which the null hypothesis of no trend is rejected is calculated for good and failed drives. Table 3.3 lists attributes and the percent of drives that have significant trends for the good and failed populations. The null hypothesis (no trend) was accepted for $1968 \leq A \leq 2981$, for a significance level higher than 99%. We are interested in attributes that have both a high percentage of failed drives with significant trends and a low percentage of good drives with trends, in the belief that an attribute that increases over time in failed drives while remaining constant in good drives is likely to be informative in predicting impending failure.

From Table 3.3 we can see that attributes such as Servo2, ReadError18 and Servo10 could be useful predictors. Note that these results are reported for a test of one group of 100 samples from each drive using a predefined significance level, and no learning was used. This is in contrast to the way a failure prediction algorithm must work, which must test each of many (usually N) consecutive series of samples, and if any fail, then the drive is predicted to fail (see Section 4.1 for details).

Some attributes (for example CSS) are *cumulative*, meaning that they report the number of occurrences since the beginning of the drive's life. All cumulative attributes either will have no trend (nothing happens) or have a positive trend. Spin-ups is the number of times the drive motors start the platters spinning, which happens every time the drive is turned on, or when it reawakens from a low-power state. It is expected that most drives will be turned on and off repeatedly, so it is unsurprising that both good and failed drives show increasing trends in Table 1. Most attributes (for example ReadError18) report the number of occurrences during the two-hour sample period.

Table 3.3 lists selected attributes sorted by descending z-score. Attributes near the top are initially more interesting because of more significant differences in the means, that is, the mean value of an attribute (over all samples) for failed drives was higher than for good drives. Only a few of the attributes had negative z-scores, and of these even fewer were significant. Some attributes with negative z-scores also appeared to be measured improperly for some drives.

From the results of the reverse arrangements and z-score tests, a set of 25 attributes[2] was selected by hand from those attributes which appear to be promising due to increasing attribute trends in failed drives and large z-score values. The tests also help eliminate attributes that are not measured correctly, such as those with zero or very high variance.[3] This set of attributes was used in the SVM, mi-NB and clustering algorithms (see the next section). Individual attributes in this set were tried one at a time with the rank-sum test. Attributes that provided good failure detection with low false alarms in the classifiers were then used together (see Section 5).

We note that the feature selection process is not a black-box automatic method, and required trial-and-error testing of attributes and combinations of attributes in the classifiers. Many of the attributes that appeared promising from the z-score and reverse-arrangements tests did not actually work well for failure prediction, while other attributes (such as ReadError19) were known to be important from our previous work and from engineering and physics knowledge of the problem gained from discussions with the manufacturers. While an automatic feature selection method would be ideal, it would likely involve a combinatorial optimization problem which would be computationally expensive.

2. Attributes in the set of 25 are: GList1, PList, Servo1, Servo2, Servo3, Servo5, ReadError1, ReadError2, ReadError3, FlyHeight5, FlyHeight6, FlyHeight7, FlyHeight8, FlyHeight9, FlyHeight10, FlyHeight11, FlyHeight12, ReadError18, ReadError19, Servo7, Servo8, ReadError20, GList2, GList3, Servo10.
3. Attributes that were not used because all measurements were zero are: Temp2, Servo4, ReadErr13-16. Also excluded are other attributes that appear to be measured improperly for certain drives are FlyHeight13-16, Temp5, and Temp6.

Attribute	% Good	% Failed
Temp1	11.8%	48.2%
Temp3	34.8%	42.9%
Temp4	8.4%	58.9%
GList1	0.6%	10.7%
PList	0.6%	3.6%
Servo1	0.0%	0.0%
Servo2	0.6%	30.4%
Servo3	0.6%	0.0%
CSS	97.2%	92.9%
ReadError1	0.0%	0.0%
ReadError1	0.6%	5.4%
ReadError3	0.0%	0.0%
WriteError	1.1%	0.0%
ReadError18	0.0%	41.1%
ReadError19	0.0%	0.0%
Servo7	0.6%	0.0%
ReadError20	0.0%	0.0%
GList3	0.0%	8.9%
Servo10	1.7%	39.3%

Table 1: Percent of drives with significant trends by the reverse arrangements test for selected attributes, which indicates potentially useful attributes. Note that this test is performed only on the last $n = 100$ samples of each drive, while a true failure prediction algorithm must test each pattern of n samples taken throughout the drives' history. Therefore, these results typically represent an upper bound on the performance of a reverse-arrangements classifier. CSS are cumulative and are reported over the life of the drive, so it is unsurprising that most good and failed drives show increasing trends (which simply indicate that the drive has been turned on and off).

The z-scores for each attribute were calculated using the entire data set, which may lead to questions about training on the test set. (The reverse-arrangements test was calculated using only about 1/3 of the data). In practical terms, z-scores obtained using random subsets are similar and lead to the same conclusions about attribute selection. Conceptually, however, the issue remains: is it correct to use data that has been used in the feature selection process in the test sets used for estimating performance? Ideally, the reuse of data should be avoided, and the *double-resampling* method should be used to estimate performance (Cherkassky and Mulier, 1998). In double-resampling, the data is divided into a *training* set and a *prediction* set, with the prediction set used only once to measure error, and the training set further divided into *learning* and *validation* sets that are used for feature selection and parameter tuning (by way of cross-validation). Double-resampling produces an unbiased estimate of error, but for finite data sets the estimate can be highly dependent on the initial choice of training and prediction sets, leading to high variance estimates. For the hard-drive failure problem, the number of drives is limited, and the variance of the classification error (see Section 5) is already quite high. Further reducing the data available by creating a separate prediction set would likely lead to high-variance error estimates (the variance of which cannot be estimated). We note that for all the classification error results in Section 5, the test set was not seen during the training process. The issue just discussed relates to the question of whether we have biased the results by having performed statistical tests on the complete data set and used those results to inform our (mostly manual) feature and attribute selection process. The best solution is to collect more data from drives to validate the false alarm and detection rates, which a drive manufacturer would do in any case to test the method and set the operating curve level before actual implementation of improved SMART algorithms in drives.

Attribute	z-score
Servo5	45.4
Servo10	29.5
Writes	28.1
FlyHeight6	24.8
FlyHeight8	23.7
FlyHeight9	22.7
FlyHeight7	22.5
Reads	22.3
FlyHeight10	21.3
FlyHeight11	19.8
FlyHeight13	19.8
FlyHeight12	19.6
Servo2	16.2
ReadError18	15.1
FlyHeight1	12.4
ReadError1	11.2
ReadError3	10.2
ReadError1	9.5
PList	8.3

Table 2: Attributes with large positive z-score values.

4. Failure Detection Algorithms

We describe how the pattern recognition algorithms and statistical tests are applied to the SMART data set for failure prediction. First, we discuss the preprocessing that is done before the data is presented to some of the pattern recognition algorithms (SVM and Autoclass); the rank-sum and reverse-arrangements test require no preprocessing. Next, we develop a new algorithm called multiple-instance naive-Bayes (mi-NB) based on the multiple-instance framework and especially suited to low-false alarm detection. We then describe how the SVM and unsupervised clustering (Autoclass) algorithms are applied. Finally we discuss the nonparametric statistical tests, rank-sum and reverse-arrangements.

Some notation and methods are common among all the pattern recognition algorithms. A vector $\mathbf{x}$ of n consecutive samples (out of the N total samples from each drive) of each selected attribute is used to make the classification, and every vector of n consecutive samples in the history of the drive is used (see Figure 1). The length of $\mathbf{x}$ is $(n \times a)$ where a is the number of attributes. There are N vectors $\mathbf{x}$ created, with zeros prepended to those $\mathbf{x}$ in the early history of the drive. Results are not significantly different if the early samples are omitted (that is, $N - n$ vectors are created) and this method allows us to make SMART predictions in the very early history of the drive. If any $\mathbf{x}$ is classified as failed, then the drive is predicted to fail. Since the classifier is applied repeatedly to all N vectors from the same drive, each test must be very resistant to false alarms.

4.1 Preprocessing: Scaling and Binning

Because of the nonparametric nature of the SMART data, two types of preprocessing were considered: binning and scaling. Performance comparison of the preprocessing is given in Section 5.

The first type of preprocessing is *binning* (or discretization), which takes one of two forms: *equal-frequency* or *equal-width* (Dougherty et al., 1995). In equal-frequency binning, an attributes' values are converted into discrete levels such that the number of counts at each level is the same (the discrete levels are percentile groups). In equal-width binning, each attribute's range is divided into a fixed number of equal magnitude bins and values are converted into bin numbers. In both cases, the levels are set based on the training set. In both the equal-width and equal-frequency cases, the rank-order with respect to bin is preserved (as opposed to converting the attribute into multiple binary nominal attributes, one for each bin). Because there are a large number of zeros for some attributes in the SMART data (see Figure 2), a special zero-count bin is used with both equal-width and equal-frequency binning. The two types of binning were compared using the Autoclass and SVM classifiers. For the SVM, the default attribute scaling in the algorithm implementation (MySVM) was also compared to binning (see 4.4).

Binning (as a form of discretization) is a common type of preprocessing in machine learning and can provide certain advantages in performance, generalization and computational efficiency (Frank and Witten, 1999; Dougherty et al., 1995; Catlett, 1991). As shown by Dougherty et al. (1995), discretization can provide performance improvements for certain classifiers (such as naive Bayes), and that while more complex discretization methods (such as those involving entropy) did provide improvement over binning, the difference in performance between binning and the other methods was much smaller than that between discretization and no discretization. Also, binning can reduce overfitting resulting in a simpler classifier which may generalize better (Frank and Witten, 1999). Preserving the rank-order of the bins so that the classifier may take into account the ordering information (which we do) has been shown to be an improvement over binning into independent nominal bins (Frank and Witten, 1999). Finally, for many algorithms, it is more computationally efficient to train using binned or discretized attributes rather than numerical values. Equal-width binning into five bins (including the zero-count bin) was used successfully by Hamerly and Elkan (2001) on the earlier SMART data set, and no significant difference was found using up to 20 bins.

4.2 The Multiple-Instance Framework

The hard drive failure prediction problem can be cast as a *multiple-instance learning* problem, which is a two-class semi-supervised problem. In multiple-instance (MI) learning, we have a set of objects which generate many *instances* of data. All the data from one object is known as a *bag*. Each bag has a single label $\{0,1\}$, which is assumed to be known (and given during training), while each instance also has a true label $\{0,1\}$ which is hidden. The label of a bag is related to the correct labeling of the instances as follows: if the label of each instance is 0, then the bag label is 0; if *any* of the instances is labeled 1, then the bag label is 1. This method of classifying a bag as 1 if any of its instances is labeled 1 is known as the *MI assumption*. Because the instance labels are unknown, the goal is to learn the labels, knowing that at least one of the instances in each 1 bag has label 1, and all the instance labels in each 0 bag should be 0.

The hard drive problem can be fit naturally into the MI framework. Each pattern $\mathbf{x}$ (composed of n samples) is an instance, and the set of all patterns for a drive i is the bag X_i. The terms *bag label* and *drive label* are interchangeable, with failed drives labeled $\mathcal{Y}_i = 1$ and good drives labeled $\mathcal{Y}_i = 0$. The hidden instance (pattern) labels are $y_j, j = 1 \dots N_i$ for the N_i instances in each bag (drive). Figure 3 show a schematic of the MI problem.

The multiple-instance framework was originally proposed by Dietterich et al. (1997) and applied to a drug activity prediction problem; that of discovering which molecules (each of which may exist in a number of different shapes, the group of all shapes for a specific molecule comprising a bag) bind to certain receptors, specifically that of smell receptors for the scent of musk. The instances consist of 166 attributes that represent the shape of one possible configuration of a molecule from X-ray crystallography, and the class of each molecule (bag) is 1 if the molecule (any instance) smells like musk as determined by experts. The so-called "musk" data sets have become the standard benchmark for multiple-instance learning.

The algorithm developed by Dietterich et al. (1997) is called axis-parallel-rectangles, and other algorithms were subsequently developed based on many of the paradigms in machine learning such as support vector machines (Andrews et al., 2003), neural networks, expectation-maximization, nearest-neighbor (Wang and Zucker, 2000), as well as special purpose algorithms like the diverse-density algorithm. An extended discussion of many of these is given by Xu (2003), who makes the important distinction between two classes of MI algorithms: those which adhere to the MI assumption (as described above) and those which make other assumptions, most commonly that the label for each positive bag is determined by some other method than simply if one instance has a positive label. Algorithms that violate the MI assumption usually assume that the data from all instances in a bag is available to make a decision about the class. Such algorithms are difficult to apply to the hard drive problem, as we are interested in construction on-line classifiers that make a decision based on each instance (pattern) as it arrives. Algorithms that violate the MI-assumption include Citation-k-Nearest-Neighbors (Wang and Zucker, 2000), SVMs with polynomial minimax kernel, and the statistical and wrapper methods of Xu (2003), and these will not be considered further for hard drive failure prediction.

4.3 Multiple Instance Naive Bayes (mi-NB)

We now develop a new multiple instance learning algorithm using naive Bayes (also known as the *simple Bayesian classifier*) and specifically designed to allow control of the false alarm rate. We call this algorithm mi-NB (multiple instance-naive Bayes) because of its relation to the mi-SVM algorithm of Andrews et al. (2003). The mi-SVM algorithm does adhere to the MI assumption and so could be used for the hard drive task, but since it requires repeated relearning of an SVM, it is presently too computationally intensive. By using the fast naive Bayes algorithm as the base classifier, we can create an efficient multiple-instance learning algorithm.

The mi-NB algorithm begins by assigning a label y_j to each pattern: for good drives, all patterns are assigned $y_j = 0$; for failed drives, all patterns except for the last one in the time series are assigned $y_j = 0$, with the last one assigned to the failed class, $y_{N_i} = 1$. Using these class labels, a

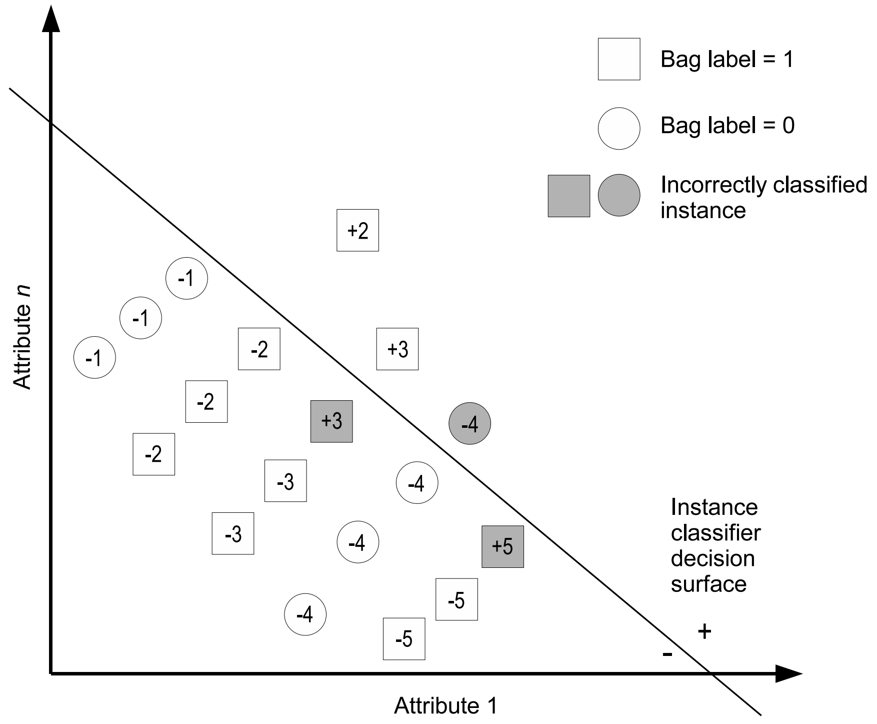


Figure 3: Multiple-instance learning. The numbers are bag (drive) numbers, and each circle or square represents an instance (pattern). Instances from class +1 (failed drives) are squares, while instances from class 0 are circles. The + or - in each instance represents the hidden underlying class of each instance, 1 or 0 respectively. The decision surface represents the classification boundary induced by a classifier. Grayed instances are those misclassified by the decision surface. Bag 1: All - instances are classified correctly, and the bag is correctly classified as 0 (good drive). Bag 2: One instance is classified as +, so the bag is correctly classified as 1 (failed drive). Bag 3: One instance of the failed drive is classified as -, but another is classified as +, so the bag is correctly classified (failed). Bag 4: An instance with true class - is labeled +, so the bag is misclassified as 1 (false alarm). Bag 5: All instances of the + bag (failed drive) are classified as -, so the bag is misclassified as 0 (missed detection).

naive Bayes model is trained (see below). Using the NB model, each pattern in the training set is assigned to a class $\widehat{y}_j \in \{0,1\}$. Because nearly all patterns are assigned to the good class $y_j = 0$, this initial condition insures that the algorithm will start with a low false alarm rate. In each iteration of the mi-NB algorithm, for every failed drive $\mathcal{Y}_i = 1$ that was misclassified (that is, all patterns were classified as good, $\widehat{y}_j = 0$), the pattern $j*$ (with current label $y_j = 0$) that is most likely to be from the failed class, $j* = \underset{j \in \{1 \ldots N_i | y_j = 0\}}{\arg\max} \; f_1(\mathbf{x}_j)$, is relabeled to the failed class $y_{j*} = 1$, where $f_1(\mathbf{x})$ is the log-posterior of class 1 (see Equation 1 below). The NB model is updated using the new class labels (which can be done very efficiently). Iterations continue until the false alarm rate on the training set increases to over the target level, $FA > FA_{\text{target}}$. The mi-NB algorithm is detailed in Algorithm 1. The procedure given in Algorithm 1 may be applied with different base classifiers other than naive Bayes, although the resulting algorithm may be computationally expensive unless there is an efficient way to update the model without retraining from scratch. Other stopping conditions could also be used, such as detection rate greater than a certain value or number of iterations.

Algorithm 1 mi-NB Train (for SMART failure prediction)

```
Input: $\mathbf{x}, \mathcal{Y}, FA_{\text{desired}}$ (desired false alarm rate)
Initialize:
        Good drives: For drives with $\mathcal{Y}_i = 0$ initialize $y_j = 0$ for $j = 1 \ldots N_i$
        Failed drives: For drives with $\mathcal{Y}_i = 1$ initialize $y_j = 0$ for $j = 1 \ldots N_i - 1$, and $y_{N_i} = 1$
Learn NB model
$\widehat{y}_j = \arg\max_{c \in \{0,1\}} f_c(\mathbf{x}_j)$    Classify each pattern using the NB model
Find $FA$ and $DET$ rate
while $FA < FA_{\text{target}}$ do
  for all Misclassified failed drives, $\widehat{y}_j = 0 \ \forall \ j = 1 \ldots N_i$ do
    $j* = \arg\max_{j \in \{1 \ldots N_i | y_j = 0\}} f_1(\mathbf{x}_j)$    Find pattern closest to decision surface with label $y_j = 0$
    $y_{j*} \leftarrow 1$    Reclassify the pattern as failed
    Update NB model
  end for
  $\widehat{y}_j = \arg\max_{c \in \{0,1\}} f_c(\mathbf{x}_j)$    Reclassify each pattern using the NB model
  Find $FA$ and $DET$ rate
end while
Return: NB model
```

In Bayesian pattern recognition, the *maximum a posterior* (MAP) method is used to estimate the class $\widehat{y}$ of a pattern $\mathbf{x}$,

$$\begin{aligned}\widehat{y} &= \arg\max_{c \in \{0,1\}} p(y = c|\mathbf{x}) \\ &= \arg\max_{c \in \{0,1\}} p(\mathbf{x}|y = c)p(y = c) \ .\end{aligned}$$

The "naive" assumption in naive Bayes is that the class-conditional distribution $p(\mathbf{x}|y = c)$ is factorial (independent components), $p(\mathbf{x}|y = c) = \prod_{m=1}^{n \cdot a} p(x_m|y = c)$ where $n \cdot a$ is the size of $\mathbf{x}$ (see Section 2). The class estimate becomes,

$$\begin{aligned}f_c(\mathbf{x}) &= \sum_{m=1}^{n \cdot a} \log \widehat{p}(x_m|y = c) + \log \widehat{p}(y = c) \\ \widehat{y} &= \arg\max_{c \in \{0,1\}} f_c(\mathbf{x}) \ ,\end{aligned} \tag{1}$$

where we have used estimates $\widehat{p}$ of the probabilities. Naive Bayes has been found to work well in practice even in cases where the components x_m are not independent, and a discussion of this is given by Domingos and Pazzani (1997). Assuming discrete distributions for x_m, counts of the number elements $\#\{\cdot\}$ can be found. Training a naive Bayes classifier is then a matter of finding the smoothed empirical estimates,

$$\begin{aligned}\widehat{p}(x_m = k|y = c) &= \frac{\#\{x_m = k, y = c\} + \ell}{\#\{y = c\} + 2\ell} \\ \widehat{p}(y = c) &= \frac{\#\{y = c\} + \ell}{\#\{\text{patterns}\} + 2\ell} \,, \end{aligned} \tag{2}$$

where ℓ is a smoothing parameter, which we set to $\ell = 1$ corresponding to Laplace smoothing (Orlitsky et al. (2003), who also discuss more recent methods for estimating probabilities, including those based on the Good-Turing estimator). Ng and Jordan (2002) show that naive Bayes has a higher asymptotic error rate (as the amount of training data increases) but that it approaches this rate more quickly than other classifiers and so may be preferred in small-sample problems. Since each time we have to switch a pattern in the mi-NB iteration, we only have to change a few of the counts in (2), updating the model after relabeling certain patterns is very fast.

Next, we show that the mi-NB algorithm has non-decreasing detection and false alarm rates over the iterations.

Lemma 1 *At each iteration t, the mi-NB algorithm does not decrease the detection and false alarm rates (as measured on the training set) over the previous iteration $t-1$,*

$$\begin{aligned}&f_1^{(t-1)}(\mathbf{x}_j) \le f_1^{(t)}(\mathbf{x}_j) \\ &f_0^{(t-1)}(\mathbf{x}_j) \ge f_0^{(t)}(\mathbf{x}_j) \qquad \forall j = 1 \ldots N \,. \end{aligned} \tag{3}$$

Proof At iteration $t-1$ the probability estimates for a certain k are,

$$\widehat{p}_{t-1}(x_m = k|y = 1) = \frac{b + \ell}{d + 2\ell} \,,$$

where $b = \#\{x_m = k, y = c\}, d = \#\{y = c\}$, and of course $b \le d$. Since class estimates are always switched from $y_j = 0$ to 1, for some k

$$\widehat{p}_t(x_m = k|y = 1) = \frac{b + \ell + 1}{d + 2\ell + 1}$$

(and for other k it will remain constant). It is now shown that the conditional probability estimates are non-decreasing,

$$\begin{aligned}\widehat{p}_{t-1}(x_m = k|y = 1) &\le \widehat{p}_t(x_m = k|y = 1) \\ (b + \ell)(d + 2\ell + 1) &\le (d + 2\ell)(b + \ell + 1) \\ b &\le d + \ell \,, \end{aligned}$$

with equality only in the case of $b = d, \ell = 0$. Similarly, the prior estimate is also non-decreasing, $\widehat{p}_{t-1}(y = 1) \le \widehat{p}_t(y = 1)$. From (1) this implies that $f_1^{(t-1)}(\mathbf{x}) \le f_1^{(t)}(\mathbf{x})$.

For class $y = 0$, it can similarly be shown that $\widehat{p}_{t-1}(x_m = k|y = 0) \ge \widehat{p}_t(x_m = k|y = 0)$ and $\widehat{p}_{t-1}(y = 0) \ge \widehat{p}_t(y = 0)$, implying $f_0^{(t-1)}(\mathbf{x}_j) \ge f_0^{(t)}(\mathbf{x}_j)$ and completing the proof. ■

Note that Algorithm 1 never relabels a failed pattern as a good pattern, as this might reduce the detection rate (and invalidate the proof of Lemma 1 in Section 4.3). The initial conditions of the algorithm ensure a low false alarm rate, and the algorithm proceeds (in a greedy fashion) to pick patterns that are mostly likely representatives of the failed class without re-evaluating previous choices. A more sophisticated algorithm could be designed that moves patterns back to the good class as they become less likely failed candidates, but this requires a computationally expensive combinatorial search.

4.4 Support Vector Machines (SVMs)

The support vector machine (SVM) is a popular modern pattern recognition and regression algorithm. First developed by Vapnik (1995), the principle of the SVM classifier is to project the data into a higher dimensional space where the classes are separated by a linear hyperplane which is defined by a small set of support vectors. For an introduction to SVMs for pattern recognition, see Burges (1998). The hyperplane is found by a quadratic optimization problem, which can be formulated for either the case where the patterns are linearly separable, or the non-linearly separable case which requires the use of slack variables ξ_i for each pattern and a parameter C that penalizes the slack. We use the non-linearly separable case and in addition use different penalties L^+, L^- for incorrectly labeling each class. The hyperplane is found by solving,

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\left(\sum_{\forall i|y_i=+1} L^+\xi_i + \sum_{\forall i|y_i=-1} L^-\xi_i\right) \\ \text{subject to:} \quad & y_i(\mathbf{w}^T\phi(\mathbf{x}_i)+b) \geq 1-\xi_i \\ & \xi_i \geq 0 \end{aligned}$$

where $\mathbf{w}$ and b are the parameters of the hyperplane $\widehat{y} = \mathbf{w}^T\phi(\mathbf{x}) + b$ and $\phi(\cdot)$ is the mapping to the high-dimensional space implicit in the kernel $k(\mathbf{x}_j, \mathbf{x}_k) = \phi(\mathbf{x}_j)^T\phi(\mathbf{x}_k)$ (Burges, 1998). In the hard-drive failure problem, L^+ penalizes false alarms, and L^- penalizes missed detections. Since C is multiplied by both L^+ and L^-, there are only two independent parameters and we set $L^- = 1$ and adjust C, L^+ when doing a grid search for parameters.

To apply the SVM to the SMART data set, drives are randomly assigned into training and test sets for a single trial. For validation, means and standard deviations of detection and false alarm rates are found over 10 trials, each with different training and test sets. Each pattern is assigned to the same label as the drive (all patterns in a failed drive $\mathcal{Y} = 1$ are assigned to the failed class, $y_i = +1$, and all patterns in good drives $\mathcal{Y} = 0$ are set to $y_i = -1$). Multiple instance learning algorithms like mi-SVM (Andrews et al., 2003) could be used to find a better way of assigning pattern classes, but these add substantial extra computation to the already expensive SVM training.

We use the MySVM[4] package developed by Ruping (2000). Parameters for the MySVM software are set as follows: $epsilon = 10^{-2}$, $max\_iterations = 10000$, $convergence\_ epsilon = 10^{-3}$. When equal-width or equal-frequency binning is used (see Section 4.1), *no_scale* is set; otherwise, the default attribute scaling in MySVM is used. The parameters C and L^+ (with $L^- = 1$) are varied to adjust the tradeoff between detection and false alarms. Kernels tested include dot product, polynomials of degree 2 and 3, and radial kernels with width parameter γ.

4. MySVM is available at: `http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVM`.

4.5 Clustering (Autoclass)

Unsupervised clustering algorithms can be used for anomaly detection. Here, we use the Autoclass package (Cheeseman and Stutz, 1995) to learn a probabilistic model of the training data from only good drives. If any pattern is an anomaly (outlier) from the learned statistical model of good drives, then that drive is predicted to fail. The *expectation maximization (EM)* algorithm is used to find the highest-likelihood mixture model that fits the data. A number of forms of the probability density function (pdf) are available, including Gaussian, Poisson (for integer count data) and nominal (unordered discrete, either independent or covariant). For the hard drive problem, they are all set to independent nominal to avoid assuming a parametric form for any attribute's distribution. This choice results in an algorithm very closely related to the *naive Bayes EM* algorithm (Hamerly and Elkan, 2001), which was found to perform well on earlier SMART data.

Before being presented to Autoclass the attribute values are discretized into either equal-frequency bins or equal-width bins (Section 4.1), where the bin range is determined by the maximum range of the attribute in the training set (of only good drives). An additional bin was used for zero-valued attributes. The training procedure attempts to find the most likely mixture model to account for the good drive data. The number of clusters can also be determined by Autoclass, but here we have restricted it to a small fixed number from 2 to 10. Hamerly and Elkan (2001) found that for the naive Bayes EM algorithm, 2 clusters with 5 bins (as above) worked best. During testing, the estimated probability of each pattern under the mixture model is calculated. A failure prediction warning is triggered for a drive if the probability of any of its samples is below a threshold (which is a parameter of the algorithm). To increase robustness, the input pattern contained between 1 and 15 consecutive samples n of each attribute (as described above for the SVM). The Autoclass threshold parameter was varied to adjust tradeoff between detection and false alarm rates.

4.6 Rank-sum Test

The Wilcoxon-Mann-Whitney rank-sum test is used to determine if the two random data sets arise from the same probability distribution (Lehmann and D'Abrera, 1998, pg. 5). One set T comes from the drive under test and the other R is a *reference set* composed of samples from good drives. The use of this test requires some assumptions to be made about the distributions underlying the attribute values and the process of failure. Each attribute has a *good distribution* G and an *about-to-fail distribution* F. For most of the life of the drive, each attribute value is chosen from the G, and then at some time before failure, the values begin to be chosen from F. This model posits an abrupt change from G to F, however, the test should still be expected to work if the distribution changes gradually over time, and only give a warning when it has changed significantly from the reference set.

The test statistic W_S is calculated by ranking the elements of R (of size m) and T (of size n) such that each element of R and T has a rank $S \in [1, n+m]$ with the smallest element assigned $S = 1$. The rank-sum W_S is the sum of the ranks S of the test set.

The rank-sum test is often presented assuming continuous data. The attributes in the SMART data are discrete which creates the possibility of ties. Tied values are ranked by assigning identical values to their *midrank* (Lehmann and D'Abrera, 1998, pg. 18), which is the average rank that the values would have if they were not tied. For example, if there were three elements tied at the smallest value, they would each be assigned the midrank $\frac{1+2+3}{3} = 2$.

If the set sizes are large enough (usually, if the smaller set $n > 10$ or $m+n > 20$), the rank-sum statistic W_S is normally distributed under the null hypothesis (T and R are from the same population) due to the central limit theorem, with mean and variance:

$$\begin{aligned} E(W_S) &= \frac{1}{2}n(m+n+1) \\ Var(W_S) &= \frac{mn(m+n+1)}{12} - C_T \ , \end{aligned}$$

where C_T is the ties correction, defined as

$$C_T = \frac{mn \sum_{i=1}^{e} (d_i^3 - d_i)}{12(m+n)(m+n-1)} \ ,$$

where e is the number of distinct values in R and T, and d_i is the number of tied elements at each value (see Appendix A for more details). The probability of a particular W_S can be found using the standard normal distribution, and a critical value α can be set at which to reject the null hypothesis. In cases of smaller sets where the central limit theorem does not apply (or where there are many tied values), an exact method of calculating the probability of the test statistic is used (see Appendix A, which also gives examples of calculating the test statistic).

For application to the SMART data, the reference set R for each attribute (size $m = 50$ for most experiments) is chosen at random from the samples of good drives. The test set T (size $n = 15$ for most experiments) is chosen from consecutive samples of the drive under test. If the test set for any attribute over the history of the drive is found to be significantly different from the reference set R then the drive is predicted to fail. The significance level α is adjusted in the range $[10^{-7}, 10^{-1}]$ to vary the tradeoff between false alarms and correct detections. We use the one-sided test of T coming from a larger distribution than R, against the hypothesis of identical distributions.

Multivariate nonparametric rank-based tests that exploit correlations between attribute values have been developed (Hettmansperger, 1984; Dietz and Killeen, 1981; Brunner et al., 2002). A different multivariate rank-sum test was successfully applied to early SMART data (Hughes et al., 2002). It exploits the fact that error counts are always positive. Here, we use a simple OR test to use two or more attributes: if the univariate rank-sum test for any attribute indicates a different distribution from the reference set, then that pattern is labeled failed. The use of the OR test is motivated by the fact that very different significance level ranges (per-pattern) for each attribute were needed to achieve low false alarm rates (per-drive).

4.7 Reverse Arrangements Tests

The reverse arrangements test described above for feature selection can also be used for failure prediction. No training set is required, as the test is used to determine if there is a significant trend in the time series of an attribute. For use with the SMART data, 100 samples are used in each test, and every consecutive sequence of samples is used. For each drive, if any test of any attribute shows a significant trend, then the drive is predicted to fail. As with the rank-sum test, the significance level α controls the tradeoff between detection and false alarm rates.

5. Results

In this section we present results from a representative set of experiments conducted with the SMART data. Due to the large number of possible combinations of attributes and classifier parameters, we could not exhaustively search this space, but we hope to have provided some insight into the hard drive failure prediction problem and a general picture of which algorithms and preprocessing methods are most promising. We also can clearly see that some methods are significantly better than the current industry-used SMART thresholds implemented in hard drives (which provide only an estimated 3-10% detection rate with 0.1% false alarms).

5.1 Failure Prediction Using 25 Attributes

Figure 4 shows the failure prediction results in the form of a Receiver Operating Characteristic (ROC) curve using the SVM, mi-NB, and Autoclass classifiers with the 25 attributes selected because of promising reverse arrangements test or z-score values (see Section 3.3). One sample per pattern was used, and all patterns in the history of each test drive were tested. (Using more than one sample per pattern with 25 attributes proved too computationally expensive for the SVM and Autoclass implementations, and did not significantly improve the mi-NB results.) The detection and false alarm rates were measured per drive: if any pattern in the drive's history was classified as failed, the drive was classified as failed. The curves were created by performing a grid search over the parameters of the algorithms to adjust the trade-off between false alarms and detection. For the SVM, the radial kernel was used with the parameters adjusted as follows: kernel width $\gamma \in [0.01, 0.1, 1]$, capacity $C \in [0.001, 0.01, 0.1, 1]$, the cost penalty $L^+ \in [1, 10, 100]$. Table 5.3 shows the parameters used in all SVM experiments. For Autoclass, the threshold parameter was adjusted in $[99.99, 99.90, 99.5, 99.0, 98.5]$ and the number of clusters was adjusted in $[2, 3, 5, 10]$.

Although all three classifiers appear to have learned some aspects of the problem, the SVM is superior in the low false-alarm region, with 50.6% detection and no measured false alarms. For all the classifiers, it was difficult to find parameters that yielded low enough false alarm rates compared with the low 0.3-1.0% annual failure rate of hard drives. For mi-NB, even at the initial condition (which includes only the last sample from each failed drive in the failed class) there is a relatively high false alarm rate of 1.0% at 34.5% detection.

For the 25 attributes selected, the SVM with the radial kernel and default scaling provided the best results. Results using the linear kernel with the binning and scaling are shown in Figure 5. The best results with the linear kernel were achieved with the default scaling, although it was not possible to adjust to false alarm rate to 0%. Equal-width binning results in better performance than equal-frequency binning for SVM and Autoclass. The superiority of equal-width binning is consistent with other experiments (not shown) and so only equal-width binning will be considered in the remaining sections. Using more bins (10 vs. 5) for the discretization did not improve performance, confirming the results of Hamerly and Elkan (2001).

The good performance of the SVM comes at a high computational price as shown in Figure 6. The bars represent the average time needed to train each algorithm for a given set of parameters. The total training time includes the time needed for the grid search to find the best parameters. For SVMs with the radial kernel (Figure 4), training took 497 minutes for each set of parameters, and 17893 minutes to search all 36 points on the parameter grid. The mi-NB algorithm was much quicker, and only had one parameter to explore, taking 17 minutes per point and 366 minutes for the grid search.

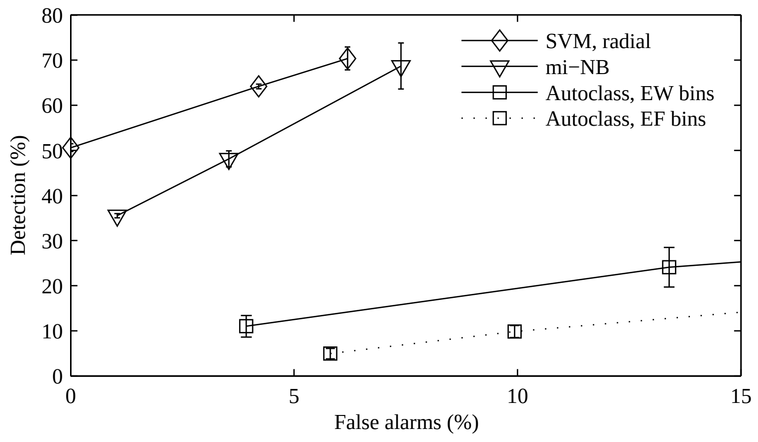


Figure 4: Failure prediction performance of SVM, mi-NB and Autoclass using 25 attributes (one sample per pattern) measured per drive. For mi-NB, the results shown are for equal-width binning. Autoclass is tested using both equal-width (EW) and equal-frequency (EF) binning (results with 5 bins shown). Error bars are ±1 standard error in this and all subsequent figures.

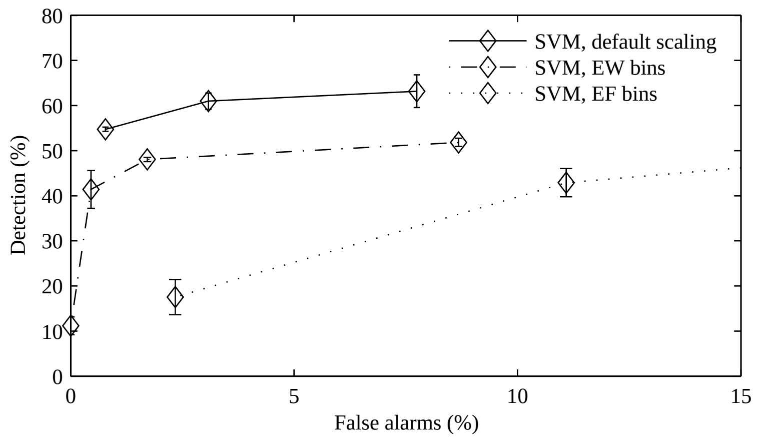


Figure 5: Comparison of preprocessing with the SVM using 25 attributes (one sample per pattern). A linear kernel is used, and the default attribute scaling is compared with equal-width and equal-frequency binning.

Also of interest is how far in advance we are able to predict an imminent failure. Figure 7 shows a histogram of the time before actual failure that the drives are correctly predicted as failing, plotted for SVM at the point 50.6% detection, 0.0% false alarms. The majority of detected failures are predicted within 100 hours (about 4 days) before failure, which is a long enough period to be reasonable for most users to backup their data. A substantial number of failures were detected over 100 hours before failure, which is one of the motivations for initially labeling all patterns from failed drives as being examples of the failed class (remembering that our data only includes the last 600 hours of SMART samples from each drive).

5.2 Single-attribute Experiments

In an effort to understand which attributes are most useful in predicting imminent hard-drive failure, we tested the attributes individually using the non-parametric statistical methods (rank-sum and reverse arrangements). The results of the reverse arrangements test on individual attributes (Section 3 and Table 3.3) indicate that attributes such as ReadError18 and Servo2 could have high sensitivity. The ReadError18 attribute appears promising with 41.1% of failed drives and 0 good drives showing significant increasing trends. Figure 8 shows the failure prediction results using only the ReadError18 attribute with the rank-sum, reverse arrangements, and SVM classifiers. Reducing the number of attributes from 25 to 1 increases the speed of all classifiers, and this increase is enough so that more samples can be used per pattern, with 5 samples per pattern used in Figure 8. The rank-sum test provided the best performance, with 24.3% detection with false alarms too low to measure, and 33.2% detection with 0.5% false alarms. The mi-NB and Autoclass algorithms using the ReadError18 (not shown in Figure 8 for clarity) perform better than the reverse-arrangements test and slightly worse than the SVM.

Single attribute tests using rank-sum were run on all 25 attributes selected in Section 3.3 with 15 samples per pattern. Of these 25, only 8 attributes (Figure 9) were able to detect failures at sufficiently low false alarm rates: ReadError1, ReadError2, ReadError3, ReadError18, ReadError19, Servo7, GList3 and Servo10. Confirming the observations of the feature selection process, ReadError18 was the best attribute, with 27.6% detection at 0.06% false alarms.

For the rank-sum test, the number of samples to use in the reference set (samples from good drives) is an adjustable parameter. Figure 10 shows the effects of using reference set sizes 25, 50 and 100 samples, with no significant improvement for 100 samples over 50. For all other rank-sum test results 50 samples were used in the reference set.

5.3 Combinations of Attributes

Using combinations of attributes in the rank-sum test can lead to improved results over single-attribute classifiers (Figure 11). The best single attributes from Figure 9 were ReadError1, ReadError3, ReadError18 and ReadError19. Using these four attributes and 15 samples per pattern, the rank-sum test detected 28.1% of the failures, with no measured false alarms. Higher detection rates (52.8%) can be had if more false alarms are allowed (0.7%). These four attributes were also tested with the SVM classifier (using default scaling). Interestingly, the linear kernel provided better performance than the radial, illustrating the need to evaluate different kernels for each data set.

All the ROC curves plotted in this section include error bars at ± 1 standard error. We also note that the number of good drives is relatively small (178) and with up to 40% of these used in the training set, measuring low false alarm rates is imprecise. When results are reported with false alarm

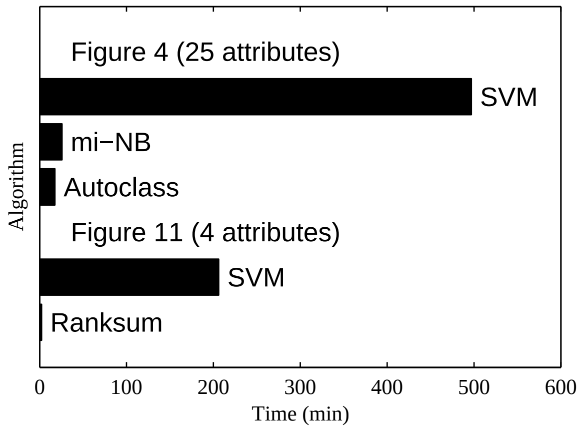


Figure 6: Training times (in minutes) for each of the algorithms used in Figures 4 and 11. The training times shown are averaged over a set of parameters. The total training time includes a search over multiple parameters. For example, the SVM used in Figure 4 required a grid search over 36 points which took a total of 17893 minutes for training with parameter selection. For the rank-sum test, only one parameter needs to be adjusted, and the training time for each parameter value was 2.2 minutes, and 21 minutes for the search through all parameters.

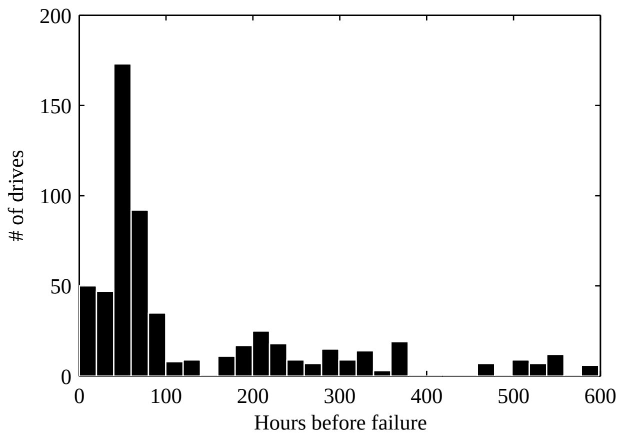


Figure 7: Histogram of time (hours) before failure that a correct failure prediction was made. Counts are summed over ten trials of the SVM algorithm (radial kernel with 25 attributes) from the point in Figure 4 at 50.6% detection, no false alarms.

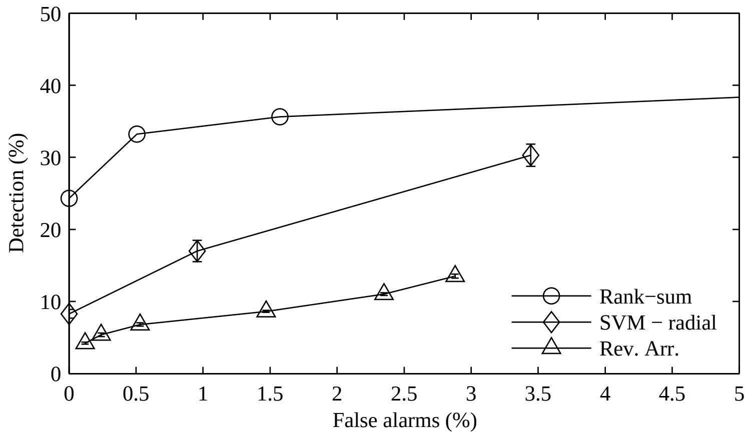


Figure 8: Failure prediction performance of classifiers using a single attribute, ReadError18, with 5 input samples per pattern. For rank-sum and reverse arrangements, error bars are smaller than line markers. For this attribute, the SVM performed best using the radial kernel and default attribute scaling (no binning).

rates of $< 1\%$, this means that some of the trials had no false alarm drives while other trials had very few (1 or 2). Because some drives are inherently more likely to be predicted as false alarms, whether these drives are included in the test or training sets can lead to a variance from trial to trial, causing large error bars at some of the points.

6. Discussion

We discuss the results of our findings and their implications for hard-drive failure prediction and machine learning in general.

While the SVM provided the best overall performance (50.6% detection with no measured false-alarms, see Figure 4), a few caveats should be noted. Using the radial kernel, three parameters must be searched to find the optimum performance (kernel width γ, capacity C and cost penalty $L+$) which was very computationally expensive and provides no guarantee as to optimality. After examining the SVM classifiers, it was found that a large number of the training examples were chosen as support vectors. For example, in a typical experiment using the radial kernel with 25 attributes, over 26% of the training examples were support vectors (6708 of 25658). This indicates that the classifier is likely overfitting the data and using outliers as support vectors, possibly causing errors on unseen data. Other researchers have noticed this property of SVMs and have developed algorithms that create smaller sets of support vectors, such as the relevance vector machine (Tipping, 2001), kernel matching pursuit (Vincent and Bengio, 2002) and Bayesian neural networks (Liang, 2003). The SMART failure prediction algorithms (as currently implemented in hard-drives) run on the internal CPU's of the drive and have rather limited memory and processing to devote to SMART. To implement the SVM classifiers learned here, they would have to evaluate the kernel with each support vector for every new sample, which may be prohibitive.

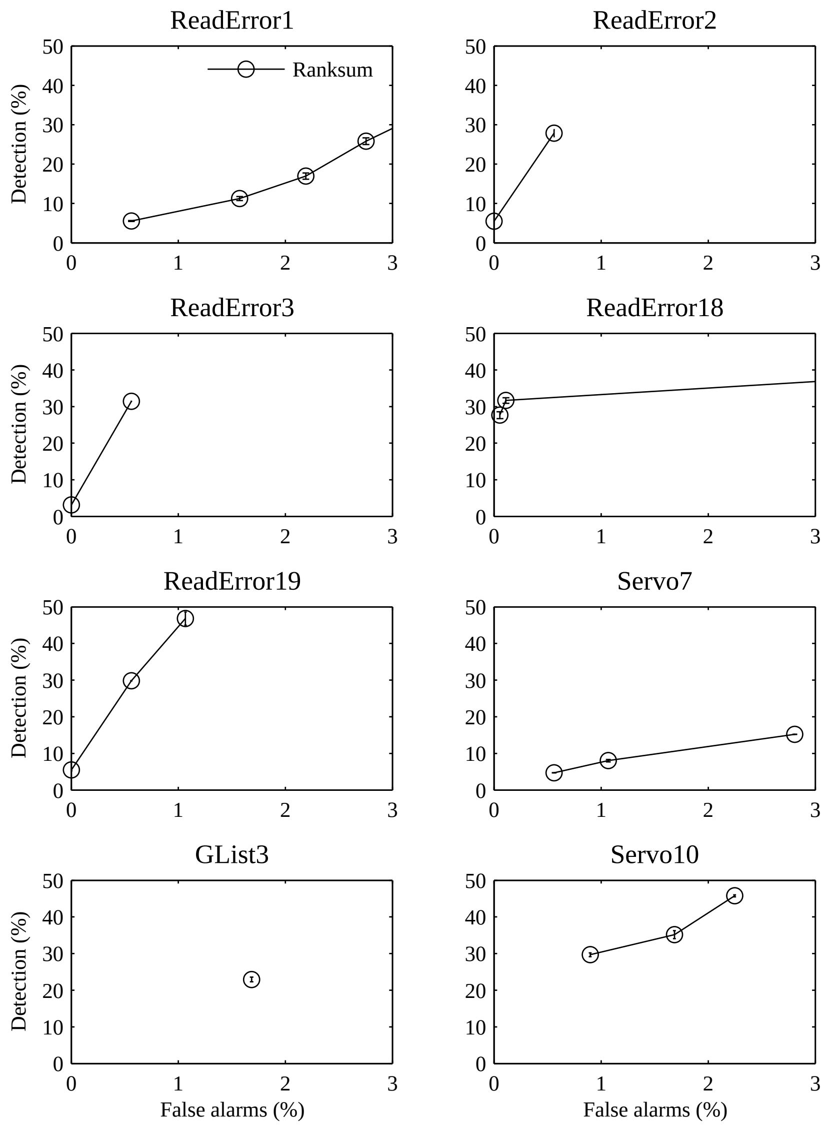


Figure 9: Failure prediction performance of rank-sum using the best single attributes. The number of samples per pattern is 15, with 50 samples used in the reference set.

The rank-sum test provided the second-best detection rate (on a set of 4 attributes, Figure 11), 28.1% with no measured false-alarms, and while lower than the best SVM result, it is still much higher than the currently implemented SMART threshold algorithms. At higher false alarm rates, the rank-sum detection rate is 52.8% with 0.7% false alarms, which means (due to the small number of good drives) that only 1 drive at most triggered a false alarm in the test set. A larger sample of good drives would be desirable for a more accurate measure of the false alarm rate. The rank-sum test has a number of advantages over the SVM: faster training time (about 100 times), faster testing of new samples, fewer parameters, and lower memory requirements. These advantages may

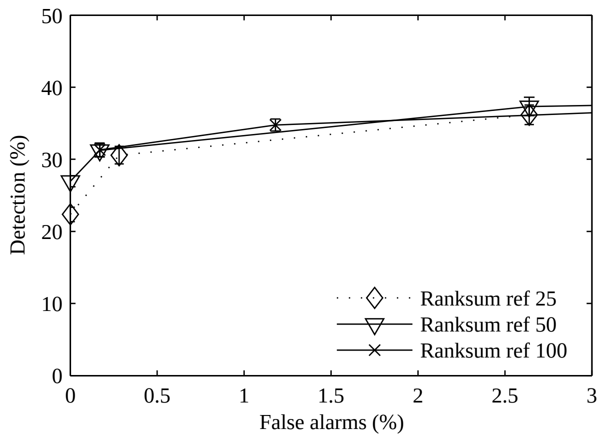


Figure 10: Rank-sum test with reference set sizes 25, 50 and 100 using ReadError18 attribute and 15 test samples. There is no improvement in performance using 100 samples in the reference set instead of 50 (as in all other rank-sum experiments).

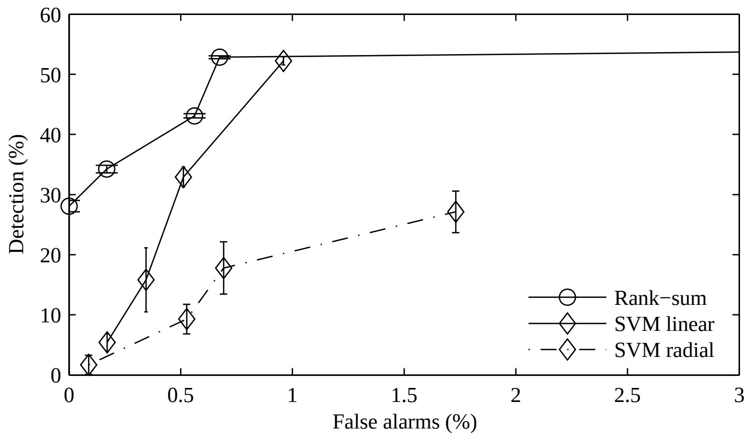


Figure 11: Failure prediction performance of rank-sum and SVM classifiers using four attributes: ReadError1, ReadError3, ReadError18 and ReadError19.

make it more suitable for implementation in hard drive firmware. For offline situations where more processing power is available (such as when the failure prediction algorithm is run on the host CPU), the SVM may be practical. For some machine learning problems, the rank-sum test may be superior to SVMs as shown in Figure 11. In this case the four attributes were selected because of good performance in the rank-sum test, and so of course it is not an entirely fair comparison but in some situations the only attributes available may be those that favor rank-sum. From a drive reliability perspective, the rank-sum test indicates that attributes that measure read errors (in this case, ReadError1, ReadError3, ReadError18 and ReadError19) were the most useful in predicting

Figure 4

Point	Detection	False Alarm	Kernel	gamma	C	L+	L-
1	50.60	0.00	radial	0.100	0.010	100.0	1.0
2	64.18	4.21	radial	0.010	0.100	100.0	1.0
3	70.38	6.20	radial	0.010	1.000	100.0	1.0

Figure 5

Point	Detection	False Alarm	Kernel	C	L+	L-
1 default scaling	54.73	0.78	linear	0.001	1000.0	1.0
2	60.97	3.09	linear	0.100	5.0	1.0
3	63.17	7.75	linear	0.010	5.0	1.0
1 EW bins	11.18	0.00	linear	0.001	100.0	1.0
2	41.40	0.46	linear	0.001	5.0	1.0
3	48.05	1.72	linear	0.001	1.0	1.0
4	51.83	8.68	linear	0.001	0.5	1.0
1 EF bins	17.54	2.34	linear	0.001	5.0	1.0
2	42.90	11.09	linear	0.100	5.0	1.0
3 (off graph)	70.22	35.40	linear	0.100	10.0	1.0

Figure 8

Point	Detection	False Alarm	Kernel	gamma	C	L+	L-
1	8.28	0.00	radial	0.010	0.010	100.0	1.0
2	17.01	0.96	radial	0.100	0.010	1.0	1.0
3	30.29	3.45	radial	1.000	0.010	1.0	1.0

Figure 11

Point	Detection	False Alarm	Kernel	gamma	C	L+	L-
1 linear	5.43	0.17	linear		0.001	1000.0	1.0
2	15.82	0.35	linear		0.010	1000.0	1.0
3	32.92	0.51	linear		0.010	1.0	1.0
4	52.23	0.96	linear		0.100	1.0	1.0
1 radial	1.68	0.09	radial	0.100	0.001	100.0	1.0
2	9.29	0.53	radial	0.001	0.010	100.0	1.0
3	17.79	0.69	radial	1.000	1.000	1000.0	1.0
4	27.13	1.73	radial	0.100	0.100	100.0	1.0

Table 3: Parameters for SVM experiments in Figures 4, 5, 8 and 11.

imminent failure. Also of interest, although with less selectivity, are attributes that measure seek errors.

Our new mi-NB algorithm demonstrated promising initial performance, which although less successful than the SVM was considerably better than the unsupervised Autoclass algorithm which was also based on naive Bayesian models (Figure 4). The multiple instance framework addresses the problem of which patterns in the time series should be labeled as failed during learning. In order to reduce false alarms, our algorithm begins with the assumption that only the last pattern in each failed drive's history should be labeled failed, and during subsequent iterations, it switches the labels of those good samples mostly likely to be from the failed distribution. This semi-supervised approach can be contrasted with the unsupervised Autoclass and the fully supervised SVM, where all patterns from failed drives were labeled failed.

The reverse-arrangements test performed more poorly than expected, as we believed that the assumption of increasing trend made by this test was well suited for hard drive attributes (like read-error counts) that would presumably increase before a failure. The rank-sum test makes no assumptions about trends in the sets, and in fact all time-order information is removed in the ranking process. The success of the rank-sum method led us to speculate that this removal of time-order over the sample interval was important for failure prediction. There are physical reasons in drive technology why impending failure need not be associated with an increasing trend in error counts. The simplest example is sudden stress from a failing drive component which causes a sudden increase in errors, followed by drive failure.

It was also found that a small number of samples (from 1 to 15) in the input patterns was sufficient to predict failure accurately, this indicates that the drive's performance can degrade quickly, and only a small window of samples is needed to make an accurate prediction. Conversely, using too many samples may dilute the weight of an important event that occurs within a short time frame.

One of the difficulties in conducting this research was the need to try many combinations of attributes and classifier parameters in order to construct ROC curves. ROC curves are necessary to compare algorithm performance because the cost of misclassifying one class (in this case, false alarms) is much higher than for the other classes. In many other real world applications such as the examples cited in Section 1, there will also be varying costs for misclassifying different classes. Therefore, we believe it is important that the machine learning community develop standardized methods and software for the systematic comparison of learning algorithms that include cycling through ranges of parameters, combinations of attributes and number of samples to use (for time series problems). An exhaustive search may be prohibitive even with a few parameters, so we envision an intelligent method that attempts to find the broad outline of the ROC curve by exploring the limits of the parameter space, and gradually refines the curve estimate as computational time allows. Another important reason to create ROC curves is that some algorithms (or parameterizations) may perform better in certain regions of the curve than others, with the best algorithm dependent on the actual costs involved (which part of the curve we wish to operate in).

7. Conclusions

We have shown that both nonparametric statistical tests and machine learning methods can significantly improve over the performance of the hard drive failure-prediction algorithms which are currently implemented. The SVM achieved the best performance of 50.6% detection/0% false alarms, compared with the 3-10% detection/0.1-0.3% false alarms of the algorithms currently implemented in hard drives. However, the SVM is computationally expensive for this problem and has many free parameters, requiring a time-consuming and non-optimal grid search.

We developed a new algorithm (mi-NB) in the multiple-instance framework that uses naive Bayesian learning as its base classifier. The new algorithm can be seen as semi-supervised in that it adapts the class label for each pattern based on whether it is likely to come from a failed drive. The mi-NB algorithm performed considerably better than an unsupervised clustering algorithm (Autoclass) that also makes the naive Bayes assumption. Further increases in performance might be achieved with base classifiers other than naive Bayes, for example, the mi-SVM algorithm (Andrews et al., 2003) could be suitably adapted but probably remains computationally prohibitive.

We also showed that the nonparametric rank-sum test can be useful for pattern recognition and that it can have higher performance than SVMs for certain combinations of attributes. The best performance was achieved using a small set of attributes: the rank-sum test with four attributes predicted 28.1% of failures with no false alarms (and 52.8% detection/0.7% false alarms). Attributes useful for failure prediction were selected by using z-scores and the reverse arrangements test for increasing trend.

Improving the performance of hard drive failure prediction will have many practical benefits. Increased accuracy of detection will benefit users by giving them an opportunity to backup their data. Very low false alarms (in the range of 0.1%) will reduce the number of returned good drives, thus lowering costs to manufacturers of implementing improved SMART algorithms. While we believe the algorithms presented here are of high enough quality (relative to the current commercially-used algorithms) to be implemented in drives, it is still important to test them on larger number of drives (on the order of thousands) to measure accuracy to the desired precision of 0.1%. We also note that each classifier has many free parameters and it is computationally prohibitive to exhaustively search the entire parameter space. We choose many parameters by non-exhaustive grid searches; finding more principled methods of exploring the parameter space is an important topic of future research.

We hope that the insights we have gained in employing the rank-sum test, multiple-instance framework and other learning methods to hard drive failure prediction will be of use in other problems where rare events must be forecast from noisy, nonparametric time series, such as in the prediction of rare diseases, electronic and mechanical device failures, and bankruptcies and business failures (see references in Section 1).

Acknowledgments

This work is part of the UCSD Intelligent Disk Drive Project funded by the Information Storage Industry Center (a Sloan Foundation Center), and by the UCSD Center for Magnetic Recording Research (CMRR). J. F. Murray gratefully acknowledges support by the ARCS Foundation. We wish to thank the anonymous reviewers work for their detailed and insightful comments and suggestions, particularly regarding the use of the multiple instance framework. We also thank the corporate sponsors of the CMRR for providing the data sets used in our work.

Appendix A: Exact and Approximate Calculation of the Wilcoxon-Mann-Whitney Significance Probabilities

The Wilcoxon-Mann-Whitney test is a widely used statistical procedure for comparing two sets of single-variate data (Wilcoxon, 1945; Mann and Whitney, 1947). The test makes no assumptions about the parametric form of the distributions each set is drawn from and so belongs to the class of nonparametric or distribution-free tests. It tests the null hypothesis that the two distributions are equal against the alternative that one is stochastically larger than the other (Bickel and Doksum, 1977, pg. 345). For example, two populations identical except for a shift in mean is sufficient but not necessary for one to be stochastically larger than the other.

Following Klotz (1966), suppose we have two sets $X = [x_1, x_2, \ldots, x_n]$, $Y = [y_1, y_2, \ldots, y_m]$, $n \leq m$, drawn from distributions F and G. The sets are concatenated and sorted, and each x_i and y_i

X	74	59	63	64							n = 4
Y	65	55	58	67	53	71					m = 6
$[X,Y]$ sorted	53	55	58	59	63	64	65	67	71	74	
Ranks	1	2	3	4	5	6	7	8	9	10	
X ranks	10	4	5	6							W_X = 25
Y ranks	7	2	3	8	1	9					W_Y = 30

Table 4: Calculating the Wilcoxon statistic W_X and W_Y without ties

is assigned a rank according to its place in the sorted list. The Wilcoxon statistic W_X is calculated by summing the ranks of each x_i, hence the term rank-sum test. Table 7 gives a simple example of how to calculate W_X and W_Y. If the two distributions are discrete, some elements may be tied at the same value. In most practical situations the distributions are either inherently discrete or effectively so due to the finite precision of a measuring instrument. The tied observations are given the rank of the average of the ranks that they would have taken, called the *midrank*. Table 7 gives an example of calculating the Wilcoxon statistic in the discrete case with ties. There are five elements with the value ‘0’ which are all assigned the average of their ranks: $(1+2+3+4+5)/5 = 3$.

To test the null hypothesis H_0 that the distributions F and G are equal against the alternative H_a that $F(x) \leq G(x) \forall x$, $F \neq G$ we must find the probability $p_0 = P(W_X > w_x)$ that under H_0 the true value of the statistic is greater than the observed value, now called w_x (Lehmann and D’Abrera, 1998, pg. 11). If we were interested in the alternative that $F \leq G$ or $F \geq G$, a two-sided test would be needed. The generalization to the two-sided case is straightforward and will not be considered here, see Lehmann and D’Abrera (1998, pg. 23). Before computers were widely available, values of p_0 (the significance probability) were found in tables if the set sizes were small (usually m and $n < 10$) or calculated from a normal approximation if the set sizes were large. Because of the number of possible combinations of tied elements, the tables and normal approximation were created for the simplest case, namely continuous distributions (no tied elements).

X	0	0	0	1	3			n = 5
Y	0	0	1	2	2	3	4	m = 7
X ranks	3	3	3	6.5	10.5			W_X = 26
Y ranks	3	3	6.5	8.5	8.5	10.5	12	W_Y = 52
		z_1	z_2	z_3	z_4	z_5		
Discrete values:		0	1	2	3	4		
		t_1	t_2	t_3	t_4	t_5		
Ties configuration:		5	2	2	2	1		

Table 5: Calculating the Wilcoxon statistic W_X and W_Y with ties

Lehman (1961) and Klotz (1966) report on the discrepancies between the exact value of p_0 and its normal approximation, which can be over 50%, clearly large enough to lead to an incorrect decision. Unfortunately, many introductory texts do not discuss these errors nor give algorithms for computing the exact probabilities. Here we outline how to calculate the exact value of p_0 but keep in mind there are other more efficient (but more complicated) algorithms (Mehta et al., 1988a,b; Pagano and Tritchler, 1983). Each element in X and Y can take one of c values, $z_1 < z_2 < \cdots < z_c$. The probability that x_i will take on a value z_k is p_k:

$$P(x_i = z_k) = p_k \quad i = 1..n, \quad k = 1..c\ .$$

Similarly for y_i,

$$P(y_j = z_k) = r_k \quad j = 1..m, \quad k = 1..c\ .$$

Under H_0, $p_k = r_k \forall k$. The count of elements in X that take on a value z_k is given by u_k and the count of elements in Y that are equal to z_k is given by v_k so that

$$\begin{array}{ll} u_k = \#\{X = z_k\} & v_k = \#\{Y = z_k\} \\ \sum_{k=1}^{c} u_k = n & \sum_{k=1}^{c} v_k = m\ . \end{array}$$

The vectors $U = [u_1, u_2, \ldots, u_c]$ and $V = [v_1, v_2, \ldots, v_c]$ give the ties configuration of X and Y. The vector $T = [t_1, t_2, \ldots, t_c] = U + V$ gives the ties configuration of the concatenated set. See Table 7 for an example of how to calculate T. Under the null hypothesis H_0, the probability of observing ties configuration U is given by (Klotz, 1966),

$$P(U|T) = \frac{\binom{t_1}{u_1}\binom{t_2}{u_2}\cdots\binom{t_c}{u_c}}{\binom{n+m}{n}}\ .$$

To find p_0, we must find all the U such that $W_U > W_x$, where W_U is the rank sum of a set with ties configuration U,

$$\begin{aligned} p_0 &= \sum_{U_i \in U_g} P(U_i|T) \qquad \text{Exact significance probability} \\ U_g &= \{U | W_U > W_X\}\ . \end{aligned} \tag{4}$$

Equation (4) gives us the exact probability of observing a set with a rank sum W greater than W_X. Because of the number of U to be enumerated, each requiring many factorial calculations, the algorithm is computationally expensive but still possible for sets as large as $m = 50$ and $n = 20$. We can compare the exact p_0 to the widely-used normal approximation and find the conditions when the approximation is valid and when the exact algorithm is needed.

The normal approximation to the distribution of the Wilcoxon statistic W can also be used to find p_0. Because W is the sum of identical, independent random variables, the central limit theorem states that its distribution will be normal asymptotically. The mean and variance of W are given by Lehmann and D'Abrera (1998),

$$\begin{aligned} E(W) &= \frac{1}{2}n(m+n+1) \\ \mathrm{Var}(W) &= \frac{mn(m+n+1)}{12} - \frac{mn\sum_{i=1}^{c}(t_i^3 - t_i)}{12(m+n)(m+n-1)}\ . \end{aligned} \tag{5}$$

		m (Large)								
		10	15	20	25	30	35	40	45	50
	5	12.298	5.332	6.615	8.480	2.212	0.947	1.188	0.527	0.630
n (Small)	10	4.057	3.482	2.693	0.595	0.224	0.14	0.064	0.091	0.042
	15		1.648	0.306	0.069	0.081	0.026	0.019	0.010	0.009
	20			0.082	0.048	0.016	0.014	0.006	0.005	0.006

Table 6: Mean-square error between exact and normal approximate to the distribution of W. All z_k are equally likely. Averages are over 20 trials at each set size

		m (Large)								
		10	15	20	25	30	35	40	45	50
	5	31.883	25.386	28.300	26.548	14.516	16.654	19.593	9.277	11.380
n (Small)	10	3.959	4.695	3.594	1.884	1.058	1.657	0.427	0.735	0.369
	15		1.984	0.733	0.311	0.336	0.230	0.245	0.317	0.205
	20			0.303	0.146	0.123	0.059	0.045	0.071	0.034

Table 7: Mean-square error between exact and normal approximate to the distribution of W. One discrete value, z_1 is much more likely than the other z_k. Averages are over 20 trials at each set size

Using the results of (5) we can find p_0 by using a table of normal curve area or common statistical software. Note that $\text{Var}(W)$ takes into account the configuration of ties $T = [t_1, t_2, \ldots, t_c]$ defined above. The second term on the right in the expression for $\text{Var}(W)$ is known as the ties correction factor.

The exact and approximate distributions of W were compared for set sizes ranging from $10 \leq m \leq 50$ and $5 \leq n \leq 20$ with tied observations. For each choice of m and n the average error between the exact and normal distributions is computed for $0 \leq p_0 \leq 0.20$ which is the range that most critical values will fall into. The mean-square error (mse) is computed over 20 trials for each set size. Table 7 gives the results of this comparison for the case where each discrete value z_k is equally likely, $p_k = r_k =$ constant $\forall k$. As expected, the accuracy improves as the set size increases, but it should be noted that these are only averages; that accuracy of p_0 for any particular experiment may be worse than suggested by Table 7. To illustrate this, Table 7 compares the distributions in the case when the first value z_1 is much more likely ($p_1 = 60\%$) than the other z_k which are equally likely. When $n < 10$, the normal approximation is too inaccurate to be useful even when $m = 50$. This is the situation when using the Wilcoxon test with the hard drive failure-prediction data, and motivated our investigation into the exact calculation of p_0. Again, Tables 7 and 7 should be used only to observe the relative accuracies of the normal approximation under various set sizes and distributions; the accuracy in any particular problem will depend on the configuration of ties T, the actual value of p_0, and the set size. The inaccuracies of normal approximations in small sample data size situations is a known aspect of the central limit theorem. It is particularly weak for statistics dependent on extreme values (Kendall, 1969).

Recommendations Based on the results of the comparisons between the exact calculation of p_0 and the normal approximation (Tables 7 and 7), we offer recommendations on how to perform the Wilcoxon-Mann-Whitney test in the presence of tied observations:

1. If $n \leq 10$ and $m \leq 50$, the exact calculation should always be used.
2. The normal approximation loses accuracy if one of the values is much more likely than the others. If this is the case, values of $n \leq 15$ will require the exact calculation.
3. The exact calculation is no longer prohibitively slow for $n \leq 20$ and $m \leq 50$, and should be considered if the significance probability p_0 is close to the desired critical value.

These recommendations are stronger than those given in Emerson and Moses (1985). A number of software packages can perform the exact test, including StatXact (http://www.cytel.com), the SAS System (http://www.sas.com) and SPSS Exact Tests (http://www.spss.com). We hope that an increased awareness of exact procedures will lead to higher quality statistical results.

References

Stuart Andrews, Ioannis Tsochantaridis, and Thomas Hofmann. Support vector machines for multiple-instance learning. In *Advances in Neural Information Processing Systems 15 (NIPS*2002)*, pages 1–8, Cambridge, MA, 2003. MIT Press.

Julius S. Bendat and Allan G. Piersol. *Random Data*. Wiley, New York, 3rd edition, 2000.

P. J. Bickel and K. A. Doksum. *Mathematical Statistics*. Holden-Day, San Francisco, 1977.

P. D. Bridge and S. S. Sawilowsky. Increasing physicians' awareness of the impact of statistics on research outcomes: Comparative power of the t-test and Wilcoxon rank-sum test in small samples applied research. *Journal Of Clinical Epidemiology*, 52(3):229–235, March 1999.

Edgar Brunner, Ullrich Munzel, and Madan L. Puri. The multivariate nonparametric Behrens-Fisher problem. *Journal of Statistical Planning and Inference*, 108:37–53, 2002.

Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

J. Catlett. On changing continuous attributes into ordered discrete attributes. In Y. Kodratoff, editor, *Proceedings of the European Working Session on Learning*, pages 164–178, Berlin, 1991. Springer-Verlag.

P. Cheeseman and J. Stutz. *Advances in Knowledge Discovery and Data Mining*, chapter Bayesian Classification (AutoClass), pages 158–180. AAAI Press, Menlo Park, CA, 1995.

Vladimir Cherkassky and Filip Mulier. *Learning from Data: Concepts, Theory, and Methods*. Wiley, New York, 1998.

Thomas G. Dietterich, Richard H. Lathrop, and Tomas Lozano-Perez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89:31–71, 1997.

E. Jacquelin Dietz and Timothy J. Killeen. A nonparametric multivariate test for monotone trend with pharmaceutical applications. *Journal of the American Statistical Association*, 76(373):169–174, March 1981.

Pedro Domingos and Michael Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29:103–130, 1997.

J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *Proceedings of the 12th International Conference on Machine Learning*, pages 194–202. Morgan Kaufmann, 1995.

J. D. Emerson and L. E. Moses. A note on the Wilcoxon-Mann-Whitney test for 2 x k ordered tables. *Biometrics*, 41:303–309, March 1985.

Eibe Frank and Ian H. Witten. Making better use of global discretization. In I. Bratko and S. Dzeroski, editors, *Proceedings of the Sixteenth International Conference on Machine Learning, Bled, Slovenia*, pages 115–123, San Francisco, CA, 1999. Morgan Kaufmann Publishers.

Greg Hamerly and Charles Elkan. Bayesian approaches to failure prediction for disk drives. In *Eighteenth International Conference on Machine Learning*, pages 1–9, 2001.

T. P. Hettmansperger. *Statistical Inference Based on Ranks*. Wiley, New York, 1984.

Gordon F. Hughes, Joseph F. Murray, Kenneth Kreutz-Delgado, and Charles Elkan. Improved disk-drive failure warnings. *IEEE Transactions on Reliability*, 51(3):350–357, September 2002.

Maurice G. Kendall. *The Advanced Theory of Statistics*, volume 1. Hafner, New York, 1969.

J. H. Klotz. The Wilcoxon, ties, and the computer. *Journal of the American Statistical Association*, 61(315):772–787, September 1966.

S. Y. Lehman. Exact and approximate distributions for the Wilcoxon statistic with ties. *Journal of the American Statistical Association*, 56(294):293–298, 1961.

E. L. Lehmann and H. J. M. D'Abrera. *Nonparametrics: Statistical Methods Based on Ranks*. Prentice Hall, Upper Saddle River, NJ, 1998.

Faming Liang. An effective Bayesian neural network classifier with a comparison study to support vector machine. *Neural Computation*, 15:1959–1989, 2003.

Henry B. Mann. Nonparametric tests against trend. *Econometrica*, 13(3):245–259, 1945.

Henry B. Mann and D. R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *Annals of Mathematical Statistics*, 19:50–60, 1947.

C. R. Mehta, N. R. Patel, and P. Senchaudhuri. Importance sampling for estimating exact probabilities in permutational inference. *Journal of the American Statistical Association*, 83(404): 999–1005, December 1988a.

C. R. Mehta, N. R. Patel, and L. J. Wei. Constructing exact significance tests with restricted randomization rules. *Biometrika*, 75:295–302, 1988b.

Joseph F. Murray, Gordon F. Hughes, and Kenneth Kreutz-Delgado. Hard drive failure prediction using non-parametric statistical methods. In *Proceedings of the International Conference on Artificial Neural Networks ICANN 2003*, Istanbul, Turkey, June 2003.

A. Y. Ng and M. I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. In *Advances in Neural Information Processing Systems (NIPS)*, volume 14, 2002.

Alon Orlitsky, Narayana P. Santhanam, and Junan Zhang. Always Good Turing: Asymptotically optimal probability estimation. *Science*, 302:427–431, October 2003.

M. Pagano and D. Tritchler. On obtaining permutation distributions in polynomial time. *Journal of the American Statistical Association*, 78:435–441, 1983.

B. E. Preusser and G. L. Hadley. Motor current signature analysis as a predictive maintenance tool. In *Proceedings of the American Power Conference, Illinois Institute of Technology*, pages 286–291, April 1991.

K. Rothman and S. Greenland. *Modern Epidemiology*. Lippencott-Raven, Philadelphia, 2nd ed. edition, 2000.

Stefan Ruping. mySVM manual. Technical report, University of Dortmund, CS Department, AI Unit, October 2000.

P. T. Theodossiou. Predicting shifts in the mean of a multivariate time series process: an application in predicting business failures. *Journal of the American Statistical Association*, 88(422):441–449, 1993.

Michael E. Tipping. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, 2001.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1995.

Pascal Vincent and Yoshua Bengio. Kernal matching pursuit. *Machine Learning*, 48:165–187, 2002.

Jun Wang and Jean-Daniel Zucker. Solving multiple-instance problem: A lazy learning approach. In Pat Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 1119–1125, Stanford, CA, 2000. Morgan Kaufmann.

G. M. Weiss and H. Hirsh. Learning to predict rare events in event sequences. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*. AAAI Press, August 1998.

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

Xin Xu. Statistical learning in multiple instance problems. Master's thesis, University of Waikato, Hamilton, New Zealand, June 2003.

Journal of Machine Learning Research 6 (2005) 817–850 Submitted 2/05; Revised 5/05; Published 5/05

Multiclass Classification with Multi-Prototype Support Vector Machines

Fabio Aiolli AIOLLI@MATH.UNIPD.IT
Alessandro Sperduti SPERDUTI@MATH.UNIPD.IT
Dip. di Matematica Pura e Applicata
Università di Padova
Via G. Belzoni 7
35131 Padova, Italy

Editor: Yoram Singer

Abstract

Winner-take-all multiclass classifiers are built on the top of a set of prototypes each representing one of the available classes. A pattern is then classified with the label associated to the most 'similar' prototype. Recent proposal of SVM extensions to multiclass can be considered instances of the same strategy with one prototype per class.

The multi-prototype SVM proposed in this paper extends multiclass SVM to multiple prototypes per class. It allows to combine several vectors in a principled way to obtain large margin decision functions. For this problem, we give a compact constrained quadratic formulation and we propose a greedy optimization algorithm able to find locally optimal solutions for the non convex objective function.

This algorithm proceeds by reducing the overall problem into a series of simpler convex problems. For the solution of these reduced problems an efficient optimization algorithm is proposed. A number of pattern selection strategies are then discussed to speed-up the optimization process. In addition, given the combinatorial nature of the overall problem, stochastic search strategies are suggested to escape from local minima which are not globally optimal.

Finally, we report experiments on a number of datasets. The performance obtained using few simple linear prototypes is comparable to that obtained by state-of-the-art kernel-based methods but with a significant reduction (of one or two orders) in response time.

Keywords: multiclass classification, multi-prototype support vector machines, kernel machines, stochastic search optimization, large margin classifiers

1. Introduction

In multiclass classification, given a set of labelled examples with labels selected from a finite set, an inductive procedure builds a function that (hopefully) is able to map unseen instances to their appropriate classes. In this work, we exclusively focus on the *single-label* version of the multiclass classification problem in which instances are associated with *exactly one* element of the label set. However, throughout this paper, we will refer to this problem simply as multiclass problem. Binary classification can be considered a particular instance of the multiclass setting where the cardinality of the label set is two.

Multiclass classifiers are often based on the *winner-take-all* (WTA) rule. WTA based classifiers define a set of prototypes, each associated with one of the available classes from a set $\mathcal{Y}$. A scoring function $f : \mathcal{X} \times \mathcal{M} \rightarrow \mathbb{R}$ is then defined, measuring the similarity of an element in $\mathcal{X}$ with prototypes defined in a space $\mathcal{M}$. For simplicity, in the following, we assume $\mathcal{M} \equiv \mathcal{X}$. When new instances are presented in input, the label that is returned is the one associated with the most 'similar' prototype:

$$H(\mathbf{x}) = \mathcal{C}\left(\arg\max_{r \in \Omega} f(\mathbf{x}, M_r)\right) \tag{1}$$

where Ω is the set of prototype indexes, the M_r's are the prototypes and $\mathcal{C} : \Omega \rightarrow \mathcal{Y}$ the function returning the class associated to a given prototype. An equivalent definition can also be given in terms of the minimization of a distance or loss (these cases are often referred to as *distance-based* and *loss-based* decoding respectively).

1.1 Motivations and Related Work

Several well-known methods for binary classification, including neural networks (Rumelhart et al., 1986), decision trees (Quinlan, 1993), k-NN (see for example (Mitchell, 1997)), can be naturally extended to the multiclass domain and can be viewed as instances of the WTA strategy. Another class of methods for multiclass classification are the so called *prototype based methods*, one of the most relevant of which is the *learning vector quantization* (LVQ) algorithm (Kohonen et al., 1996). Although different versions of the LVQ algorithm exist, in the more general case these algorithms quantize input patterns into codeword vectors c_i and use these vectors for 1-NN classification. Several codewords may correspond to a single class. In the simplest case, also known as LVQ1, at each step of the codewords learning, for each input pattern $\mathbf{x}_i$, the algorithm finds the element c_k closest to $\mathbf{x}_i$. If that codeword is associated to a class which is the same as the class of the pattern, then c_k is updated by $c_k \leftarrow c_k + \eta(t)(\mathbf{x}_i - c_k)$ thus making the prototype get closer to the pattern, otherwise it is updated by $c_k \leftarrow c_k - \eta(t)(\mathbf{x}_i - c_k)$ thus making the prototype farther away. Other more complicated versions exist. For example, in the LVQ2.1, let y be the class of the pattern, at each step the closest codeword of class $c \neq y$ and the closest codeword of class y are updated simultaneously. Moreover, the update is done only if the pattern under consideration falls in a "window" which is defined around the midplane between the selected codewords.

When the direct extension of a binary method into a multiclass one is not possible, a general strategy to build multiclass classifiers based on a set of binary classifiers is always possible, the so called *error correcting output coding* (ECOC) strategy, originally proposed by Dietterich and Bakiri in (Dietterich and Bakiri, 1995). Basically, this method codifies each class of the multiclass problem as a fixed size binary string and then solves one different binary problem for each bit of the string. Given a new instance, the class whose associated string is most 'similar' to the output of the binary classifiers on that instance is returned as output. Extensions to codes with values in $\{-1, 0, +1\}$ (Allwein et al., 2000) and continuous codes (Crammer and Singer, 2000) have been recently proposed.

Recently, large margin kernel-based methods have shown state-of-the-art performance in a wide range of applications. They search for a large margin linear discriminant model in a typically very high dimensional space, the *feature space*, where examples are implicitly mapped via a function $\mathbf{x} \mapsto \phi(\mathbf{x})$. Since kernel-based algorithms use only dot products in this space, it is possible to resort to the 'kernel trick' when dot products can be computed efficiently by means of a kernel function $k(\mathbf{x},\mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$ defined in terms of the original patterns. Examples of kernel functions are the polynomial kernel

$$k(\mathbf{x},\mathbf{y}) = (\langle \mathbf{x},\mathbf{y} \rangle + u)^d, u \geq 0, d \in \mathbb{N}$$

of which the linear case is just an instance ($d = 1$) and the radial basis function (RBF) kernel

$$k(\mathbf{x},\mathbf{y}) = \exp(-\lambda ||\mathbf{x}-\mathbf{y}||^2), \lambda \geq 0.$$

Kernel machines, and the SVM in particular, has been initially devised for the binary setting. However, extensions to the multiclass case have been promptly proposed (e.g. Vapnik, 1998; Weston and Watkins, 1999; Guermeur et al., 2000; Crammer and Singer, 2000).

The discriminant functions generated by general kernel-based methods are implicitly defined in terms of a subset of the training patterns, the so called *support vectors*, on the basis of a linear combination of kernel products $f(\mathbf{x}) = \sum_{i \in SV} \alpha_i k(\mathbf{x}_i, \mathbf{x})$. In the particular case of the kernel function being linear, this sum can be simplified in a single dot product. When this is not the case, the implicit form allows to elegantly deal with non linear decision functions obtained by using non linear kernels. In this last case, the efficiency with respect to the time spent for classifying new vectors tends to be low when the number of support vectors is large. This has motivated some recent works, briefly discussed in the following, whose aim was at building kernel-based machines with a minimal number of support vectors.

The *relevance vector machine* (RVM) in (Tipping, 2001) is a model used for regression and classification exploiting a probabilistic Bayesian learning framework. It introduces a prior over the weights of the model and a set of hyperparameters associated to them. The form of the RVM prediction is the same as the one used for SVM. Sparsity is obtained because the posterior distributions of many of the weights become sharply peaked around the zero. Other interesting advantages of the RVM are that it produces probabilistic predictions and that it can be applied to general functions and not only to kernel functions satisfying the Mercer's condition. The *minimal kernel classifier* (MKC) in (Fung et al., 2002) is another model theoretically justified by linear programming perturbation and a bound on the leave-one-out error. This model uses a particular loss function measuring both the presence and the magnitude of an error. Finally, quite different approaches are those in (Schölkopf et al., 1999; Downs et al., 2001) that try to reduce the number of support vectors after the classifiers have been constructed.

The approach we propose here gives an alternative method to combine simple predictors together to obtain large margin multiclass classifiers. This can be extremely beneficial for two main reasons. First, adding prototypes can produce higher margin decision functions without dramatically increasing the complexity of the generated model. This can be trivially shown by considering that the single-prototype margin is a lower bound on the margin for multi-prototype since it can be obtained when all the prototypes of the same class coincide. Second, combining several simple models can be advisable when no a priori knowledge is available about the task at hand. In the following, we will study only the linear version of the algorithm without exploring more complex kernel functions, the rationale being that adding linear prototypes in the original space allows to increase the expressiveness of the decision functions without requiring the (computationally expensive) use of kernels. Moreover, linearity makes easier the interpretation of the produced models, which can be useful in some particular tasks, and allows for an easier extension to the on-line setting since the explicit representation for the models can be used.

In Section 2 we give some preliminaries and the notation we adopt along the paper. Then, in Section 3 we derive a convex quadratic formulation for the easier problem of learning one prototype per class. The obtained formulation can be shown to be equivalent, up to a change of variables and constant factors, to the one proposed by Crammer and Singer (2000). When multiple prototypes are introduced in Section 4, the problem becomes not convex in general. However, in Section 5 we will see that once fixed an appropriate set of variables, the reduced problem is convex. Moreover, three alternative methods are given for this optimization problem and heuristics for the ”smart” selection of patterns in the optimization process are proposed and compared. Then, in Section 6 we give a greedy procedure to find a locally optimal solution for the overall problem and we propose an efficient stochastic-search based method to improve the quality of the solution. In Section 7 we give theoretical results about the generalization ability of our model. Specifically, we present an upper bound on the leave-one-out error and upper bounds on the generalization error. Finally, the experimental work in Section 8 compares our linear method with state-of-the-art methods, with respect to the complexity of the generated solution and with respect to the generalization error.

This paper substantially extends the material contained in other two conference papers. Namely, (Aiolli and Sperduti, 2003) which contains the basic idea and the theory of the multi-prototype SVM together with preliminary experimental work and (Aiolli and Sperduti, 2002a) which proposes and analyzes selection heuristics for the optimization of multiclass SVM.

2. Preliminaries

Let us start by introducing some definitions and the notation that will be used in this paper. We assume to have a labelled training set $\mathcal{S} = \{(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_n, c_n)\}$ of cardinality n, where $\mathbf{x}_i \in \mathcal{X}$ are the examples in a inner-product space $\mathcal{X} \subseteq \mathbb{R}^d$ and $c_i \in \mathcal{Y} = \{1, \ldots, m\}$ the corresponding class or label. To keep the notation clearer, we focus on the linear case where kernels are not used. However, we can easily consider the existence of a feature mapping $\phi : I \rightarrow \mathcal{X}$. In this case, it is trivial to extend the derivations we will obtain to non-linear mappings $\mathbf{x} \mapsto \phi(\mathbf{x})$ of possibly non vectorial patterns by substituting dot products $\langle \mathbf{x}, \mathbf{y} \rangle$ with a suited kernel function $k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$ and the squared 2-norm $||\mathbf{x}||^2$ with $k(\mathbf{x}, \mathbf{x})$ consequently. The kernel matrix $K \in \mathbb{R}^{n \times n}$ is the matrix containing the kernel products of all pairs of examples in the training set, i.e. $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$.

We consider dot-product based WTA multiclass classifiers having the form

$$H_M(\mathbf{x}) = \mathcal{C}\left(\arg\max_{r \in \Omega} \langle M_r, \mathbf{x} \rangle\right) \tag{2}$$

where Ω is the set of prototype indices and the prototypes are arranged in a matrix $M \in \mathbb{R}^{|\Omega| \times d}$ and $\mathcal{C} : \Omega \rightarrow \mathcal{Y}$ the function that, given an index r, returns the class associated to the r-th prototype. We also denote by y_i^r, $1 \leq i \leq n$, $r \in \Omega$, the constant that is equal to 1 if $\mathcal{C}(r) = c_i$ and -1 otherwise. Moreover, for a given example $\mathbf{x}_i$, $\mathcal{P}_i = \{r \in \Omega : y_i^r = 1\}$ is the set of ’positive’ prototypes for the example $\mathbf{x}_i$, i.e. the set of prototype indices associated to the class of $\mathbf{x}_i$, while $\mathcal{N}_i = \Omega \setminus \mathcal{P}_i = \{r \in \Omega : y_i^r = -1\}$ is the set of ’negative’ prototypes, i.e. the set of prototype indices associated to classes different from the class of $\mathbf{x}_i$. The dot product $f_r(\mathbf{x}) = \langle M_r, \mathbf{x} \rangle$ is referred to as the *similarity score* (or simply *score*) of the r-th prototype vector for the instance $\mathbf{x}$. Finally, symbols in bold represent vectors and, as particular case, the symbol $\mathbf{0}$ represents the vector with all components set to 0.

3. Single-Prototype Multi-Class SVM

One of the most effective multi-class extension of SVM has been proposed by Crammer and Singer (2000). The resulting classifier is of the same form of Eq. (2) where each class has associated exactly one prototype, i.e. $\Omega \equiv \mathcal{Y}$ and $\forall r \in \Omega,\ \mathcal{C}(r) = r$. The solution is obtained through the minimization of a convex quadratic constrained function. Here, we derive a formulation that, up to a change of variables, can be demonstrated to be equivalent to the one proposed by Crammer and Singer (see Aiolli and Sperduti (2002a)). This will serve to introduce a uniform notation useful for presenting the multi-prototype extension in the following sections.

In multiclass classifiers based on Eq. (2), in order to have a correct classification, the prototype of the correct class is required to have a score greater than the maximum among the scores of the prototypes associated to incorrect classes. The multiclass margin for the example $\mathbf{x}_i$ is then defined by

$$\rho(\mathbf{x}_i, c_i|M) = \langle M_{y_i}, \mathbf{x}_i\rangle - \max_{r \neq y_i} \langle M_r, \mathbf{x}_i\rangle,$$

where y_i such that $\mathcal{C}(y_i) = c_i$, is the index of the prototype associated to the correct label for the example $\mathbf{x}_i$. In the single prototype case, with no loss of generality, we consider a prototype and the associated class indices to be coincident, that is $y_i = c_i$. Thus, a correct classification of the example $\mathbf{x}_i$ with a margin greater or equal to 1 requires the condition

$$\langle M_{y_i}, \mathbf{x}_i\rangle \geq \theta_i + 1 \text{ where } \theta_i = \max_{r \neq y_i} \langle M_r, \mathbf{x}_i\rangle. \tag{3}$$

to be satisfied. Note that, the condition above is implied by the existence of a matrix $\hat{M}$ such that $\forall r \neq y_i,\ \langle \hat{M}_{y_i}, \mathbf{x}_i\rangle > \langle \hat{M}_r, \mathbf{x}_i\rangle$. In fact, the matrix M can always be obtained by an opportune re-scaling of the matrix $\hat{M}$. With these premises, a set of examples is said to be *linearly separable* by a multiclass classifier if there exists a matrix M able to fulfill the above constraints for every pattern in the set.

Unfortunately, the examples in the training set can not always be separated and some examples may violate the margin constraints. We consider these cases by introducing soft margin slack variables $\xi_i \geq 0$, one for each example, such that

$$\xi_i = [\theta_i + 1 - \langle M_{y_i}, \mathbf{x}_i\rangle]_+,$$

where the symbol $[z]_+$ corresponds to the soft-margin loss that is equal to z if $z > 0$ and 0 otherwise. Note that the value ξ_i can also be seen as an upper bound on the binary loss for the example $\mathbf{x}_i$, and consequently its average value over the training set is an upper bound on the empirical error.

Motivated by the *structural risk minimization* (SRM) principle in (Vapnik, 1998; Schölkopf and C. Burges and V. Vapnik, 1995), we search for a matrix M with small norm such to minimize the empirical error over the training set. We use the 2-norm of the matrix M. We thus formulate the problem in a SVM style by requiring a set of small norm prototypes to fulfill the soft constraints given by the classification requirements. Specifically, the single-prototype version of multiclass

SVM (SProtSVM in the following) will result in:

$$\begin{array}{l} \min_{M,\xi,\theta} \frac{1}{2}||M||^2 + C\sum_i \xi_i \\ \text{subject to: } \begin{cases} \forall i, r \neq y_i,\ \langle M_r, \mathbf{x}_i \rangle \leq \theta_i, \\ \forall i,\ \langle M_{y_i}, \mathbf{x}_i \rangle \geq \theta_i + 1 - \xi_i, \\ \forall i,\ \xi_i \geq 0 \end{cases} \end{array} \tag{4}$$

where the parameter C controls the amount of regularization applied to the model.

It can be observed that, at the optimum, θ_i will be set to the maximum value among the negative scores for the instance $\mathbf{x}_i$ (in such a way to minimize the corresponding slack variables) consistently with Eq. (3).

The problem in Eq. (4) is convex and it can be solved in the standard way by resorting to the optimization of the Wolfe dual problem. In this case, the Lagrangian is:

$$\begin{array}{rcl} L(M,\xi,\theta,\alpha,\lambda) & = & \frac{1}{2}||M||^2 + C\sum_i \xi_i + \\ & & \sum_{i,r\neq y_i} \alpha_i^r(\langle M_r, \mathbf{x}_i\rangle - \theta_i) + \\ & & \sum_i \alpha_i^{y_i}(\theta_i + 1 - \xi_i - \langle M_{y_i}, \mathbf{x}_i\rangle) - \\ & & \sum_i \lambda_i \xi_i \\ & = & \frac{1}{2}||M||^2 - \sum_{i,r} y_i^r \alpha_i^r(\langle M_r, \mathbf{x}_i\rangle - \theta_i) + \\ & & \sum_i \alpha_i^{y_i} + \sum_i (C - \alpha_i^{y_i} - \lambda_i)\xi_i, \end{array} \tag{5}$$

subject to the constraints $\alpha_i^r, \lambda_i \geq 0$.

By differentiating the Lagrangian with respect to the primal variables and imposing the optimality conditions we obtain a set of constraints that the variables have to fulfill in order to be an optimal solution:

$$\begin{array}{lcl} \frac{\partial L(M,\xi,\theta,\alpha,\lambda)}{\partial M_r} = 0 & \Leftrightarrow & M_r = \sum_i y_i^r \alpha_i^r \mathbf{x}_i \\ \frac{\partial L(M,\xi,\theta,\alpha,\lambda)}{\partial \xi_i} = 0 & \Leftrightarrow & C - \alpha_i^{y_i} - \lambda_i = 0 \Leftrightarrow \alpha_i^{y_i} \leq C \\ \frac{\partial L(\mathbf{w},\xi,\theta,\alpha,\lambda)}{\partial \theta_i} = 0 & \Leftrightarrow & \alpha_i^{y_i} = \sum_{r\neq y_i} \alpha_i^r \end{array} \tag{6}$$

By using the facts $\alpha_i^{y_i} = \frac{1}{2}\sum_r \alpha_i^r$ and $||M(\alpha)||^2 = \sum_{i,j,r} y_i^r y_j^r \alpha_i^r \alpha_j^r \langle \mathbf{x}_i, \mathbf{x}_j \rangle$, substituting equalities from Eq. (6) into Eq. (5) and omitting constants that do not change the solution, the problem can be restated as:

$$\begin{array}{l} \max_\alpha \sum_{i,r} \alpha_i^r - ||M(\alpha)||^2 \\ \text{subject to: } \begin{cases} \forall i, r,\ \alpha_i^r \geq 0 \\ \forall i, \alpha_i^{y_i} = \sum_{r\neq y_i} \alpha_i^r \leq C \end{cases} \end{array}$$

Notice that, when kernels are used, by the linearity of dot-products, the scoring function for the r-th prototype and a pattern $\mathbf{x}$ can be conveniently reformulated as

$$f_r(\mathbf{x}) = \langle M_r, \phi(\mathbf{x}) \rangle = \sum_{i=1}^n y_i^r \alpha_i^r k(\mathbf{x}, \mathbf{x}_i).$$

The next section includes an efficient optimization procedure for the more general multi-prototype setting that includes the single-prototype case as an instance.

4. Multi-Prototype Multi-Class SVM

The SProtSVM model presented in the previous section is here extended to learn more than one prototypes per class. This is done by generalizing Eq. (3) to multiple prototypes. In this setting, one instance is correctly classified if and only if *at least* one of the prototypes associated to the correct class has a score greater than the maximum of the scores of the prototypes associated to incorrect classes.

A natural extension of the definition for the margin in the multi-prototype case is then

$$\rho(\mathbf{x}_i, c_i|M) = \max_{r\in\mathcal{P}_i}\langle M_r, \mathbf{x}_i\rangle - \max_{r\in\mathcal{N}_i}\langle M_r, \mathbf{x}_i\rangle.$$

and its value will result greater than zero if and only if the example $\mathbf{x}_i$ is correctly classified.

We can now give conditions for a correct classification of an example $\mathbf{x}_i$ with a margin greater or equal to 1 by requiring that:

$$\exists r \in \mathcal{P}_i : \langle M_r, \mathbf{x}_i\rangle \geq \theta_i + 1 \text{ and } \theta_i = \max_{r\in\mathcal{N}_i} \langle M_r, \mathbf{x}_i\rangle. \tag{7}$$

To allow for margin violations, for each example $\mathbf{x}_i$, we introduce soft margin slack variables $\xi_i^r \geq 0$, one for each positive prototype, such that

$$\forall r \in \mathcal{P}_i,\ \xi_i^r = [\theta_i + 1 - \langle M_r, \mathbf{x}_i\rangle]_+.$$

Given a pattern $\mathbf{x}_i$, we arrange the soft margin slack variables ξ_i^r in a vector $\xi_i \in \mathbb{R}^{|\mathcal{P}_i|}$. Let us now introduce, for each example $\mathbf{x}_i$, a new vector having a number of components equal to the number of positive prototypes for $\mathbf{x}_i$, $\pi_i \in \{0,1\}^{|\mathcal{P}_i|}$, whose components are all zero except one component that is 1. In the following, we refer to π_i as the *assignment* of the pattern $\mathbf{x}_i$ to the (positive) prototypes. Notice that the dot product $\langle \pi_i, \xi_i\rangle$ is always an upper bound on the binary loss for the example $\mathbf{x}_i$ independently from its assignment and, similarly to the single-prototype case, the average value over the training set represents an upper bound on the empirical error.

Now, we are ready to formulate the general multi-prototype problem by requiring a set of prototypes of small norm and the best assignment for the examples able to fulfill the soft constraints given by the classification requirements. Thus, the MProtSVM formulation can be given as:

$$\begin{array}{l} \min_{M,\xi,\theta,\pi} \frac{1}{2}||M||^2 + C\sum_i \langle \pi_i, \xi_i\rangle \\ \text{subject to: } \begin{cases} \forall i, r\in\mathcal{N}_i,\ \langle M_r, \mathbf{x}_i\rangle \leq \theta_i, \\ \forall i, r\in\mathcal{P}_i,\ \langle M_r, \mathbf{x}_i\rangle \geq \theta_i + 1 - \xi_i^r, \\ \forall i, r\in\mathcal{P}_i, \xi_i^r \geq 0 \\ \forall i, \pi_i \in \{0,1\}^{|\mathcal{P}_i|}. \end{cases} \end{array} \tag{8}$$

Unfortunately, this is a mixed integer problem that is not convex and it is a difficult problem to solve in general. However, as we will see in the following, it is prone to an efficient optimization procedure that approximates a global optimum. At this point, it is worth noticing that, since this formulation is itself an (heuristic) approximation to the structural risk minimization principle where the parameter C rules the trade-off between keeping the VC-dimension low and minimizing the training error, a good solution of the problem in Eq. (8), even if not optimal, can nevertheless give good results in practice. As we will see, this claim seems confirmed by the results obtained in the experimental work.

In the following section we demonstrate that when the assignment is fixed for each pattern, the problem results tractable and we are able to give an efficient procedure to solve the associated problem.

5. Optimization with Static Assignments

Let suppose that the assignments are kept fixed. In this case, the reduced problem becomes convex and it can be solved as described above by resorting to the optimization of the Wolfe dual problem. In this case, the Lagrangian is:

$$\begin{array}{rcl} L_\pi(M,\xi,\theta,\alpha,\lambda) & = & \frac{1}{2}||M||^2 + C\sum_i \langle \pi_i, \xi_i \rangle + \\ & & \sum_{i,r\in\mathcal{P}_i} \alpha_i^r(\theta_i + 1 - \xi_i^r - \langle M_r, \mathbf{x}_i \rangle) - \\ & & \sum_{i,r\in\mathcal{P}_i} \lambda_i^r \xi_i^r + \\ & & \sum_{i,r\in\mathcal{N}_i} \alpha_i^r(\langle M_r, \mathbf{x}_i \rangle - \theta_i), \end{array} \tag{9}$$

subject to the constraints $\alpha_i^r, \lambda_i^r \geq 0$.

As above, by differentiating the Lagrangian of the reduced problem and imposing the optimality conditions, we obtain:

$$\begin{array}{lcl} \frac{\partial L_\pi(M,\xi,\theta,\alpha,\lambda)}{\partial M_r} = 0 & \Leftrightarrow & M_r = \sum_i y_i^r \alpha_i^r \mathbf{x}_i \\ \frac{\partial L_\pi(M,\xi,\theta,\alpha,\lambda)}{\partial \xi_i^r} = 0 & \Leftrightarrow & C\pi_i^r - \alpha_i^r - \lambda_i^r = 0 \Leftrightarrow \alpha_i^r \leq C\pi_i^r \\ \frac{\partial L_\pi(M,\xi,\theta,\alpha,\lambda)}{\partial \theta_i} = 0 & \Leftrightarrow & \sum_{r\in\mathcal{P}_i} \alpha_i^r = \sum_{r\in\mathcal{N}_i} \alpha_i^r \end{array} \tag{10}$$

Notice that the second condition requires the dual variables associated to (positive) prototypes not assigned to a pattern to be 0. By denoting now as y_i the unique index $r \in \mathcal{P}_i$ such that $\pi_i^r = 1$, once using the conditions of Eq. (10) in Eq. (9) and omitting constants that do not change the obtained solution, the reduced problem can be restated as:

$$\begin{array}{l} \max_\alpha \sum_{i,r} \alpha_i^r - ||M(\alpha)||^2 \\ \text{subject to: } \begin{cases} \forall i, r,\ \alpha_i^r \geq 0 \\ \forall i,\ \alpha_i^{y_i} = \sum_{r\in\mathcal{N}_i} \alpha_i^r \leq C \\ \forall i, r \in \mathcal{P}_i \setminus \{y_i\},\ \alpha_i^r = 0. \end{cases} \end{array} \tag{11}$$

It can be trivially shown that this formulation is consistent with the formulation of the SProtSVM dual given above. Moreover, when kernels are used, the score function for the r-th prototype and a pattern $\mathbf{x}$ can be formulated as in the single-prototype case as

$$f_r(\mathbf{x}) = \langle M_r, \phi(\mathbf{x}) \rangle = \sum_{i=1}^n y_i^r \alpha_i^r k(\mathbf{x}, \mathbf{x}_i).$$

Thus, when patterns are statically assigned to the prototypes via constant vectors π_i, the convexity of the associated MProtSVM problem implies that the optimal solution for the primal problem in Eq. (8) can be found through the maximization of the Lagrangian as in problem in Eq. (11). Assuming an equal number q of prototypes per class, the dual involves $n \times m \times q$ variables which leads to a very large scale problem. Anyway, the independence of constraints among the different patterns allows for the separation of the variables in n disjoint sets of $m \times q$ variables.

The algorithms we propose for the optimization of the problem in Eq. (11) are inspired by the ones already presented in (Crammer and Singer, 2000, 2001) consisting in iteratively selecting patterns from the training set and greedily optimizing with respect to the variables associated to that pattern. In particular, the authors propose a fixed-point procedure for the optimization of the reduced problem.

In the following, we first show that the pattern related problem can be further decomposed until the solution for a minimal subset of two variables is required. This is quite similar to the SMO procedure for binary SVM. Then, a training algorithm for this problem can be defined by iterating this basic step.

5.1 The Basic Optimization Step

In this section the basic step corresponding to the simultaneous optimization of a subset of variables associated to the same pattern is presented. Let pattern $\mathbf{x}_p$ be fixed. Since we want to enforce the linear constraint $\sum_{r\in\mathcal{N}_p}\alpha_p^r+\lambda_p=C$, $\lambda_p\geq 0$, from the second condition in Eq. (10), two elements from the set of variables $\{\alpha_p^r, r\in\mathcal{N}_p\}\cup\{\lambda_p\}$ will be optimized in pair while keeping the solution inside the feasible region. In particular, let ζ_1 and ζ_2 be the two selected variables, we restrict the updates to the form $\zeta_1\leftarrow\zeta_1+\nu$ and $\zeta_2\leftarrow\zeta_2-\nu$ with optimal choices for ν.

In order to compute the optimal value for ν we first observe that an additive update ΔM_r to the prototype r will affect the squared norm of the prototype vector M_r of an amount

$$\Delta||M_r||^2=||\Delta M_r||^2+2\langle M_r,\Delta M_r\rangle.$$

Then, we examine separately the two ways a pair of variables can be selected for optimization.

(Case 1) We first show how to analytically solve the problem associated to an update involving a single variable α_p^r, $r\in\mathcal{N}_p$ and the variable $\alpha_p^{y_p}$. Note that, since λ_p does not influence the value of the objective function, it is possible to solve the associated problem with respect to the variable α_p^r and $\alpha_p^{y_p}$ in such a way to keep the constraint $\alpha_p^{y_p}=\sum_{r\in\mathcal{N}_p}\alpha_p^r$ satisfied and afterwards to enforce the constraints $\lambda_p=C-\sum_{s\in\mathcal{N}_p}\alpha_p^s\geq 0$. Thus, in this case we have:

$$\alpha_p^r\leftarrow\alpha_p^r+\nu \text{ and } \alpha_p^{y_p}\leftarrow\alpha_p^{y_p}+\nu.$$

Since $\Delta M_r=-\nu\mathbf{x}_p$, $\Delta M_{y_p}=\nu\mathbf{x}_p$ and $\Delta M_s=0$ for $s\notin\{r,y_p\}$, we obtain

$$\Delta||M||^2=\Delta||M_r||^2+\Delta||M_{y_p}||^2=2\nu^2||\mathbf{x}_p||^2+2\nu(f_{y_p}(\mathbf{x}_p)-f_r(\mathbf{x}_p))$$

and the difference obtained in the Lagrangian value will be

$$\Delta L(\nu)=2\nu(1-f_{y_p}(\mathbf{x}_p)+f_r(\mathbf{x}_p)-\nu||\mathbf{x}_p||^2).$$

Since this last formula is concave in ν, it is possible to find the optimal value when the first derivative is null, i.e.

$$\hat{\nu}=\arg\max_{\nu}\Delta L(\nu)=\frac{1-f_{y_p}(\mathbf{x}_p)+f_r(\mathbf{x}_p)}{2||\mathbf{x}_p||^2} \tag{12}$$

If the values of α_p^r and $\alpha_p^{y_p}$, after being updated, turn out to be not feasible for the constraints $\alpha_p^r\geq 0$ and $\alpha_p^{y_p}\leq C$, we select the unique value for ν such to fulfill the violated constraint bounds at the limit ($\alpha_p^r+\nu=0$ or $\alpha_p^{y_p}+\nu=C$ respectively).

(Case 2) Now, we show the analytic solution of the associated problem with respect to an update involving a pair of variables $\alpha_p^{r_1},\alpha_p^{r_2}$ such that $r_1,r_2\in\mathcal{N}_p$ and $r_1\neq r_2$. Since, in this case, the update must have zero sum, we have:

$$\alpha_p^{r_1}\leftarrow\alpha_p^{r_1}+\nu \text{ and } \alpha_p^{r_2}\leftarrow\alpha_p^{r_2}-\nu$$

In this case, $\Delta M_{r_1} = -\nu \mathbf{x}_p$, $\Delta M_{r_2} = \nu \mathbf{x}_p$ and $\Delta M_s = 0$ for $s \notin \{r_1, r_2\}$, thus

$$\Delta||M||^2 = \Delta||M_{r_1}||^2 + \Delta||M_{r_2}||^2 = 2\nu^2||\mathbf{x}_p||^2 + 2\nu(f_{r_2}(\mathbf{x}_p) - f_{r_1}(\mathbf{x}_p))$$

leading to an Lagrangian improvement equals to

$$\Delta L(\nu) = 2\nu(f_{r_1}(\mathbf{x}_p) - f_{r_2}(\mathbf{x}_p) - \nu||\mathbf{x}_p||^2).$$

Since also this last formula is concave in ν, it is possible to find the optimal value

$$\hat{\nu} = \arg\max_{\nu} \Delta L(\nu) = \frac{f_{r_1}(\mathbf{x}_p) - f_{r_2}(\mathbf{x}_p)}{2||\mathbf{x}_p||^2} \tag{13}$$

Similarly to the previous case, if the values of the $\alpha_p^{r_1}$ and $\alpha_p^{r_2}$, after being updated, turn out to be not feasible for the constraints $\alpha_p^{r_1} \geq 0$ and $\alpha_p^{r_2} \geq 0$, we select the unique value for ν such to fulfill the violated constraint bounds at the limit (in this case, considering $f_{r_1}(\mathbf{x}_p) \leq f_{r_2}(\mathbf{x}_p)$ and thus $\hat{\nu} \leq 0$ with no loss in generality, we obtain $\alpha_p^{r_1} + \nu = 0$ or $\alpha_p^{r_2} - \nu = C$ respectively).

Note that, when a kernel is used, the norm in the feature space can be substituted with the diagonal component of the kernel matrix, i.e. $||\mathbf{x}_p||^2 = k(\mathbf{x}_p, \mathbf{x}_p) = K_{pp}$ while the scores can be maintained in implicit form and computed explicitly when necessary.

To render the following exposition clearer, we try to compact the two cases in one. This can be done by defining the update in a slightly different way, that is, for each pair $(r_a, r_b) \in (\mathcal{P}_p \cup \mathcal{N}_p) \times \mathcal{N}_p$ we define:

$$\alpha_p^{r_a} \leftarrow \alpha_p^{r_a} + y_p^{r_a}\nu \text{ and } \alpha_p^{r_b} \leftarrow \alpha_p^{r_b} - y_p^{r_b}\nu$$

and hence the improvement obtained for the value of the Lagrangian is

$$V^p_{r_a,r_b}(\nu) = 2\nu\left(\frac{1}{2}(y_p^{r_a} - y_p^{r_b}) - f_{r_a}(\mathbf{x}_p) + f_{r_b}(\mathbf{x}_p) - \nu k(\mathbf{x}_p, \mathbf{x}_p)\right) \tag{14}$$

where the optimal value for the ν is

$$\hat{\nu} = \frac{\frac{1}{2}(y_p^{r_a} - y_p^{r_b}) - f_{r_a}(\mathbf{x}_p) + f_{r_b}(\mathbf{x}_p)}{2k(\mathbf{x}_p, \mathbf{x}_p)}$$

subject to the constraints

$$\alpha_p^{r_a} + y_p^{r_a}\nu > 0,\ \alpha_p^{r_b} - y_p^{r_b}\nu > 0,\ \alpha_{y_p} + \frac{1}{2}(y_p^{r_a} - y_p^{r_b})\nu \leq C.$$

The basic step algorithm and the updates induced in the scoring functions are described in Figure 1 and Figure 2, respectively.

5.2 New Algorithms for the Optimization of the Dual

In the previous section we have shown how it is possible to give an explicit optimal solution of the reduced problem obtained by fixing all the variables apart for the two variables under consideration.

In this section, we analyze different algorithms that are based on the step given above. The basic idea is the same as SMO for SVM (Platt, 1998), that is to repeat a process in which

```
BasicStep($p$, $r_a$, $r_b$)
    $\nu = \frac{\frac{1}{2}(y_p^{r_a} - y_p^{r_b}) - f_{r_a}(\mathbf{x}_p) + f_{r_b}(\mathbf{x}_p)}{2K_{pp}}$
    if ($\alpha_p^{r_a} + y_p^{r_a}\nu < 0$) then $\nu = -y_p^{r_a}\alpha_p^{r_a}$
    if ($\alpha_p^{r_b} - y_p^{r_b}\nu < 0$) then $\nu = y_p^{r_b}\alpha_p^{r_b}$
    if ($\alpha_p^{y_p} + \frac{1}{2}(y_p^{r_a} - y_p^{r_b})\nu > C$) then $\nu = 2\frac{C - \alpha_p^{y_p}}{y_p^{r_a} - y_p^{r_b}}$
    return $\nu$
```

Figure 1: The basic optimization step: explicit optimization of the reduced problem with two variables, namely $\alpha_p^{r_a}$ and $\alpha_p^{r_b}$.

```
BasicUpdate($p$, $r_a$, $r_b$, $\nu$)
    $\alpha_{r_a} = \alpha_{r_a} + y_p^{r_a}\nu$;   $\alpha_{r_b} = \alpha_{r_b} + y_p^{r_b}\nu$;
    $f_{r_a}(\mathbf{x}_p) = f_{r_a}(\mathbf{x}_p) + y_p^{r_a}\nu K_{pp}$;   $f_{r_b}(\mathbf{x}_p) = f_{r_b}(\mathbf{x}_p) - y_p^{r_b}\nu K_{pp}$;
```

Figure 2: Updates done after the basic optimization step has been performed and the optimal solution found. K_{pp} denotes the p-th element of the kernel matrix diagonal.

- a minimal subset of independent multipliers are selected
- the analytic solution of the reduced problem obtained by fixing all the variables but the ones we selected in the previous step is found.

In our case, a minimal set of two variables associated to the same example are selected at each iteration. As we showed in the last section, each iteration leads to an increase of the Lagrangian. This, together with the compactness of the feasible set guarantees the convergence of the procedure. Moreover, this optimization procedure can be considered incremental in the sense that the solution we have found at one step forms the initial condition when a new subset of variables are selected for optimization. Finally, it should be noted that for each iteration the scores of the patterns in the training set must be updated before to be used in the selection phase. The general optimization algorithm just described is depicted in Figure 3.

In the following, we present three alternative algorithms for the optimization of the problem in Eq. (11) which differ in the way they choose the pairs to optimize through the iterations, i.e. the `OptimizeOnPattern` procedure.

The first practical and very simple algorithm for solving the problem in Eq. (11) can be derived from the steps given above where at each iteration a pair of multipliers is selected and then optimized according to the analytic solution given in the previous section until some convergence criterion

```
OptimizeStaticProblem($\varphi_V$)
    repeat
        PatternSelection(p) // Heuristically choose an example $p$ based on Eq. (14)
        OptimizeOnPattern(p, $\varphi_V$)
    until converge.
```

Figure 3: High-level procedure for the optimization of a statically assigned multi-prototype SVM. The parameter φ_V is the tolerance when checking optimality in the OptimizeOnPattern procedure.

```
BasicOptimizeOnPattern(p, $\varphi_V$)
    Heuristically choose two indexes $r_a \neq r_b$ based on Eq. (14)
    $\nu =$ BasicStep$(p, r_a, r_b)$
    BasicUpdate$(p, r_a, r_b, \nu)$
```

Figure 4: SMO-like algorithm for the optimization of statically assigned multi-prototype SVM.

is fulfilled. Eq. (14) gives a natural method for the selection of the two variables involved, i.e. take the two indexes that maximize the value of that formula. Finally, once chosen two variables to optimize, the basic step in the algorithm in Figure 1 provides the optimal solution. This very general optimization algorithm will be referred to as `BasicOptimizeOnPattern` and it is illustrated in Figure 4.

A second method to solve the optimization problem in Eq. (11) is given in the following and can be also considered as an alternative method to the Crammer and Singer fixed-point algorithm for the optimization over a single example (Crammer and Singer, 2001). This method consists in fixing an example and iterating multiple times the basic step described above on pairs of variables chosen among that associated to the pattern into consideration until some convergence conditions local to the pattern under consideration are matched. Notice that this algorithm requires just a single step in the binary single-prototype case. In Figure 5 the pseudo-code of the proposed pattern optimization algorithm referred to as `AllPairsOptimizeOnPattern` is presented. At each step, the algorithm applies the basic step to the $m(m-1)/2$ pairs of variables associated with the pattern chosen for optimization until a certain condition on the value of the increment of the Lagrangian is verified. Iterating multiple times the basic step described above on pairs of variables chosen among that associated to a given pattern it is guaranteed to find the optimality condition for the pattern. The optimization step of this reduced problem can require the optimization over all the $q^2m(m-1)/2$ pairs of variables not constrained to 0 associated with the selected pattern. Thus the complexity of

```
AllPairsOptimizeOnPattern($p$, $\varphi_V$)
    $t = 0, V(0) = 0.$
    do
     $t \leftarrow t+1, V(t) = 0$
     For each $r_1 \neq r_2$
       $\nu =$ BasicStep($p, r_1, r_2$)
       $V(t) = V(t) + 2\nu \left(\frac{1}{2}(y_p^{r_1} - y_p^{r_2}) - f_{r_1}(\mathbf{x}_p) + f_{r_2}(\mathbf{x}_p) - \nu K_{pp}\right)$
       BasicUpdate($p, r_1, r_2, \nu$)
    until $(V(t) \leq \varphi_V)$
```

Figure 5: Algorithm for the incremental optimization of the variables associated with a given pattern of a staticallly assigned multi-prototype SVM

the optimization of the reduced problem is $O((mq)^2 I)$ where I is the number of iterations.

Now, we perform a further step by giving a third algorithm that is clearly faster than the previous versions having at each iteration a complexity $O(mq)$. For this we give an intuitive derivation of three optimality conditions. Thus, we will show that if a solution is such that all these conditions are not fulfilled, then this solution is just the optimal one since it verifies the KKT conditions.

First of all, we observe that for the variables $\{\alpha_p^r, \lambda_p\}$ associated to the pattern $\mathbf{x}_p$ to be optimal, the value ν returned by the basic step must be 0 for each pair. Thus, we can consider the two cases above separately. For the first case, in order to be able to apply the step, it is necessary for one of the following two conditions to be verified:

$$(\Psi_1) \qquad (\alpha_p^{y_p} < C) \wedge (f_{y_p}(\mathbf{x}_p) < \max_{r \in \mathcal{N}_p} f_r(\mathbf{x}_p) + 1)$$
$$(\Psi_2) \quad (\alpha_p^{y_p} > 0) \wedge (f_{y_p}(\mathbf{x}_p) > \max_{r \in \mathcal{N}_p, \alpha_p^r > 0} f_r(\mathbf{x}_p) + 1)$$

In fact, in Eq. (12), when there exists $r \in \mathcal{N}_p$ such that $f_{y_p}(\mathbf{x}_p) < f_r(\mathbf{x}_p) + 1$, the condition $\hat{\nu} > 0$ holds. In this case the pair $(\alpha_p^{y_p}, \alpha_p^r)$ can be chosen for optimization. Thus, it must be $\alpha_p^{y_p} < C$ in order to be possible to increase the values of the pair of multipliers. Alternatively, if $\alpha_p^{y_p} > 0$ and there exists an index r such that $\alpha_p^r > 0$ and $f_{y_p}(\mathbf{x}_p) > f_r(\mathbf{x}_p) + 1$ then $\hat{\nu} < 0$ and (at least) the pair $(\alpha_p^{y_p}, \alpha_p^k)$ where $k = \arg\max_{r \in \mathcal{N}_p, \alpha_p^r > 0} f_r(\mathbf{x}_p)$ can be chosen for optimization. Finally, from Eq. (13), we can observe that in order to have $\hat{\nu} \neq 0$, we need the last condition to be verified:

$$(\Psi_3) \quad (\alpha_p^{y_p} > 0) \wedge (\max_{r \in \mathcal{N}_p} f_r(\mathbf{x}_p) > \min_{r \in \mathcal{N}_p, \alpha_p^r > 0} f_r(\mathbf{x}_p))$$

In fact, in Eq. (13), if there exists a pair $(\alpha_p^{r_a}, \alpha_p^{r_b})$ such that $f_{r_a}(\mathbf{x}_p) > f_{r_b}(\mathbf{x}_p)$ and $\alpha_p^{r_b} > 0$, the condition $\hat{\nu} > 0$ holds for this pair and it can be chosen for optimization.

Note that in Ψ_2 and Ψ_3 the condition $\alpha_p^{y_p} > 0$ is redundant and serves to assure that the second condition makes sense. In fact, when the first condition is not verified, we would have $\alpha = \mathbf{0}$ and the second condition is undetermined.

Summarizing, we can give three conditions of non-optimality. This means that whenever at least one among these conditions is verified the solution is not optimal. They are

$$\begin{array}{ll} (a) & (\alpha_p^{y_p} < C) \wedge (f_{y_p}(\mathbf{x}_p) < \max_{r \in \mathcal{N}_p} f_r(\mathbf{x}_p) + 1) \\ (b) & (\alpha_p^{y_p} > 0) \wedge (f_{y_p}(\mathbf{x}_p) > \max_{r \in \mathcal{N}_p, \alpha_p^r > 0} f_r(\mathbf{x}_p) + 1) \\ (c) & (\alpha_p^{y_p} > 0) \wedge (\max_{r \in \mathcal{N}_p} f_r(\mathbf{x}_p) > \min_{r \in \mathcal{N}_p, \alpha_p^r > 0} f_r(\mathbf{x}_p)) \end{array} \tag{15}$$

Now we are able to demonstrate the following theorem showing that when no one of these conditions are satisfied the conditions of optimality (KKT conditions) are verified:

Theorem 1 *Let α be an admissible solution for the dual problem in Eq. (11) not satisfying any of the conditions in Eq. (15), then α is an optimal solution.*

Proof. We consider the *Kuhn-Tucker* theorem characterizing the optimal solutions of convex problems. We know from theoretical results about convex optimization that for a solution α to be optimal a set of conditions are both necessary and sufficient. These conditions are the one reported in Eq. (10) plus the so-called *Karush-Kuhn-Tucker* (KKT) complementarity conditions that in our case correspond to:

$$\begin{array}{lll} (a) & \forall p, r \in \mathcal{P}_p, & \alpha_p^r(\theta_p + 1 - \xi_p^r - f_r(\mathbf{x}_p)) = 0 \\ (b) & \forall p, r \in \mathcal{P}_p, & \lambda_p^r \xi_p^r = 0 \\ (c) & \forall p, v \in \mathcal{N}_p, & \alpha_p^v(f_v(\mathbf{x}_p) - \theta_p) = 0. \end{array} \tag{16}$$

Then, we want to show that these KKT complementary conditions are satisfied by the solution α_p for every $p \in \{1, \dots, n\}$. To this end let us fix an index p and consider a solution where all the conditions in Eq. (15) are not satisfied. We want to show that the KKT conditions in Eq. (16) are verified in this case.

First of all, we observe that for all the variables associated to a positive prototype $r \in \mathcal{P}_p$ not assigned to the pattern $\mathbf{x}_p$, that is such that $\pi_p^r = 0$, from Eq. (10) we trivially have $\alpha_p^r = 0$ and $\lambda_p^r = 0$ thus verifying all conditions in Eq. (16).

Let now consider the case $0 < \alpha_p^{y_p} < C$. In this case the non applicability of condition in Eq. (15)c says that $\theta_p = \max_{v \in \mathcal{N}_p} f_v(\mathbf{x}_p)$ and $\forall v \in \mathcal{N}_p$, $\alpha_p^v > 0 \Rightarrow f_v(\mathbf{x}_p) = \theta_p$ that is the condition in Eq. (16)c holds. Moreover, the condition in Eq. (15)a, if not satisfied, implies $f_{y_p}(\mathbf{x}_p) \geq \theta_p + 1$, thus $\alpha_p^{y_p} = 0$ and $\xi_p^{y_p} = 0$ thus satisfying the conditions in Eq. (16)a and Eq. (16)b.

Let now consider the case $\alpha_p^{y_p} = 0$. The conditions in Eq. (16)a and Eq. (16)c follow immediately. In this case, Eq. (15)b and Eq. (15)c are not satisfied. For what concerns the Eq. (15)a it must be the case $f_{y_p}(\mathbf{x}_p) \geq \max_{v \in \mathcal{N}_p} f_v(\mathbf{x}_p) + 1$ and so $\xi_p^{y_p} = 0$ thus verifying the condition in Eq. (16)b.

Finally, in the case $\alpha_p^{y_p} = C$, from Eq. (10) we have $\lambda_p = 0$ and hence the condition in Eq. (16)b is verified. Moreover, from the fact that Eq. (15)c is not satisfied $\forall v \in \mathcal{N}_p : \alpha_p^v > 0 \Rightarrow \theta_p = \max_{r \in \mathcal{N}_p} f_r(\mathbf{x}_p) \leq f_v(\mathbf{x}_p) \Rightarrow \theta_p = f_v(\mathbf{x}_p)$ and the condition in Eq. (16)c holds. Moreover, from condition in Eq. (15)b we obtain $f_{y_p}(\mathbf{x}_p) \leq \theta_p + 1$ and $\xi_p^{y_p} = \theta_p + 1 - f_{y_p}(\mathbf{x}_p)$ thus implying the truth of the condition in Eq. (16)a. $\square$

```
OptKKTOptimizeOnPattern($\mathbf{x}_p$,$\varphi_V$)
$\forall r,\ f_r := f_r(\mathbf{x}_p) = \sum_{i=1}^n y_i^r \alpha_i^r k(\mathbf{x}_i, \mathbf{x}_p),\ K_{pp} = k(\mathbf{x}_p, \mathbf{x}_p)$;
do
  if ($\alpha_p^{y_p} = 0$) then {
    $r_1 := \arg\max_{r \in \mathcal{N}_p} f_r$;
    $\nu_1$ := BasicStep($p, y_p, r_1$); $V_1 := 2\nu_1(1 - f_{y_p} + f_{r_1} - \nu_1 K_{pp})$;
    $k := 1$}
  else {
    $r_1 := \arg\max_{r \in \mathcal{N}_p} f_r$; $r_2 := \arg\max_{r \in \mathcal{N}_p, \alpha_p^r > 0} f_r$; $r_3 := \arg\min_{r \in \mathcal{N}_p, \alpha_p^r > 0} f_r$;
    $\nu_1$ := BasicStep($p, y_p, r_1$); $V_1 := 2\nu_1(1 - f_{y_p} + f_{r_1} - \nu_1 K_{pp})$;
    $\nu_2$ := BasicStep($p, y_p, r_2$); $V_2 := 2\nu_2(1 - f_{y_p} + f_{r_2} - \nu_2 K_{pp})$;
    $\nu_3$ := BasicStep($p, r_1, r_3$); $V_3 := 2\nu_3(f_{r_1} - f_{r_3} - \nu_3 K_{pp})$;
    $k := \arg\max_j V_j$; }
  case k of {
    1: BasicUpdate($p$,$y_p$,$r_1$,$\nu_1$);
    2: BasicUpdate($p$,$y_p$,$r_2$,$\nu_2$);
    3: BasicUpdate($p$,$r_1$,$r_3$,$\nu_3$); }
until ($V_k \leq \varphi_V$);
```

Figure 6: Algorithm for the optimization of the variables associated with a given pattern $\mathbf{x}_p$ and a tolerance φ_V.

All the conditions in Eq. (15) can be checked in time linear with the number of classes. If none of these conditions are satisfied, this means that the condition of optimality has been found. This consideration suggests an efficient procedure that is presented in Figure 6 that searches to greedily fulfill these conditions of optimality and it is referred to as `OptKKTOptimizeOnPattern`. Briefly, the procedure first checks if the condition $\alpha_p = \mathbf{0}$ holds. In this case, two out of the three conditions do not make any sense and the choice of the pair to optimize is mandatory. Otherwise, three indexes (r_1, r_2, r_3) are found defining the conditions in Eq. (15). Then for every pair associated to the condition a basic step is performed and the pair obtaining the larger improvement in the Lagrangian is chosen for the effective update.

5.3 Selection Criteria and Cooling Schemes

The efficiency of the general scheme in Figure 3 is tightly linked to the strategy based on which the examples are selected for optimization.

The algorithm proposed in (Crammer and Singer, 2001) is just an instance of the same scheme. In that work, by using the KKT conditions of the optimization problem, the authors derive a quantity $\psi_i \geq 0$ for each example and show that this value needs to be equal to zero at the optimum. Thus, they use this value to drive the optimization process. In the baseline implementation, the example that maximizes ψ_i is selected. Summarizing, their algorithm consists of a main loop which is composed of: (*i*) an example selection, via the ψ_i quantity, (*ii*) an invocation of a fixed-point algorithm that is able to approximate the solution of the reduced pattern-related problem and (*iii*) the computation of the new value of ψ_i for each example. At each iteration, most of the computation time is spent on the last step since it requires the computation of one row of the kernel matrix, that one relative to the pattern with respect to which they have just optimized. This is why it is so important a strategy that tries to minimize the total number of patterns selected for optimization. Their approach is to maintain an active set containing the subset of patterns having $\psi_i \geq \varepsilon$ where ε is a suitable accuracy threshold. Cooling schemes, i.e. heuristics based on the gradual decrement of this accuracy parameter, are used for improving the efficiency with large datasets.

In our opinion, this approach has however some drawbacks:

i) while $\psi_i \approx 0$ gives us the indication that the variables associated to the pattern $\mathbf{x}_i$ are almost optimal and it would be better not to change them, the actual value ψ_i does not give us information about the improvement we can obtain choosing those variables in the optimization;

ii) cooling schemes reduce the incidence of the above problem but, as we will see, they do not always perform well;

iii) at each iteration, the fixed point optimization algorithm is executed from scratch, and previously computed solutions obtained for an example can't help when the same example is chosen again in future iterations; in addition, it is able to find just an *approximated* solution for the associated pattern-related problem.

According to the above-mentioned considerations, it is not difficult to define a number of criteria to drive a 'good' pattern selection strategy, which seem to be promising. We consider the following three procedures which return a value V_p that we use for deciding if a pattern has to be selected for optimization. Namely:

i) Original KKT as defined in Crammer and Singer's work (here denoted KKT): in this case, the value of V_p corresponds to the ψ_p;

ii) Approximate Maximum Gain (here denoted AMG): in this case the value of V_p is computed as: $\max_{r_1 \neq r_2} V^p_{r_1,r_2}(\hat{\nu})$ as defined in Eq. (14). Notice that this is a lower bound of the total increment in the Lagrangian obtained when the pattern p is selected for optimization and the optimization on the variables associated to it is completed;

iii) True Maximum Gain (here denoted BMG): in this case the value is computed using iteratively Eq. (14) and it represents the actual increment in the Lagrangian obtained when the pattern p is selected for optimization.

At the begin of each iteration, a threshold for pattern selection θ_V is computed. For each example of the training set one of the above strategies is applied to it and the example is selected for optimization if the value returned is greater than the threshold. The definition of the threshold θ_V can be performed either by a cooling scheme that decreases its value as the iteration proceeds or in a data dependent way. In our case, we have used a logarithmic cooling scheme since this is the one that has shown the best results for the original Crammer and Singer approach. In addition, we propose two new schemes for the computation of the value θ_V: MAX where the threshold is computed as $\theta_V = \mu \cdot \max_p V_p$, $0 \leq \mu \leq 1$, and MEAN where the threshold is computed as $\theta_V = \frac{1}{n}\sum_{p=1}^{n} V_p$.

5.4 Experiments with Pattern Selection

Experiments comparing the proposed pattern selection approaches versus the Crammer and Singer one has been conducted using a dataset consisting of 10705 digits randomly taken from the NIST-3 dataset. The training set consisted of 5000 randomly chosen digits.

The optimization algorithm has been chosen among: *i)* The base-line Crammer and Singer original fixed-point procedure (here denoted CS); *ii)* `AllPairsOptimizeOnPatterns` (here denoted ALL); *iii)* `BasicOptimizeOnPatterns` (here denoted BAS). In the first experiments we used a cache for the kernel matrix of size 3000 that was able to contain all the matrix rows associated to the support vectors. For all the following experiments a AMD K6-II, 300MHz, with 64MB of memory has been used.

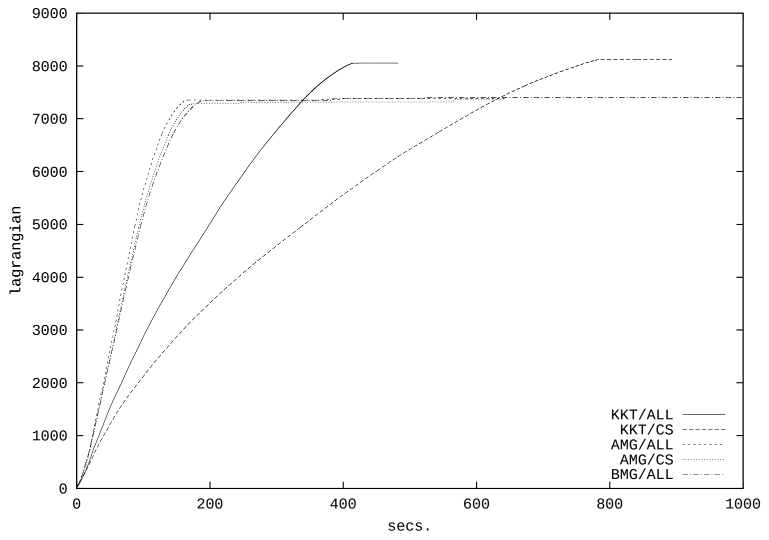


Figure 7: The effect of the logarithmic cooling scheme on different selection/optimization strategies.

In Figure 7 the effect of the application of the logarithmic scheme of cooling to the different selection/optimization strategies is shown. It is possible to note that even if the proposed selection strategies largely improve the convergence rate, the optimal solution can not be reached. This clearly shows how cooling schemes of the same family of that proposed in (Crammer and Singer, 2001) are not suitable for these new proposed selection strategies. This is mostly due to the fact that the

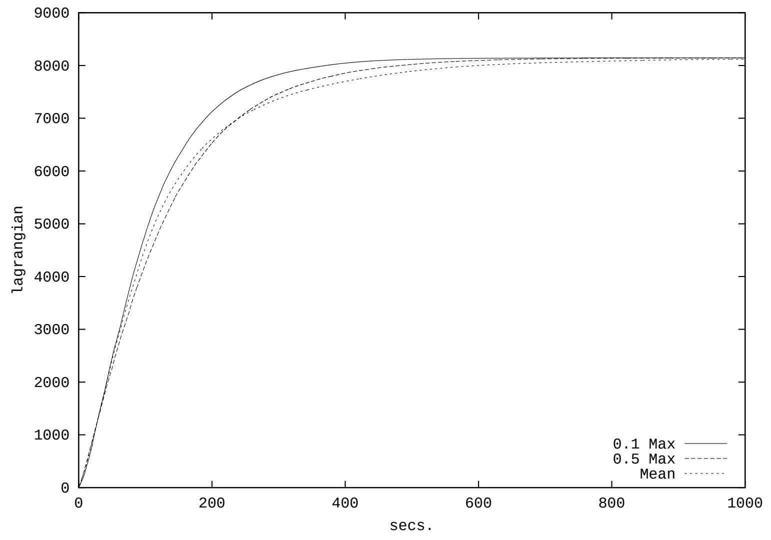


Figure 8: Comparison of different heuristics for the computation of the value θ_V for the SMO-like algorithm.

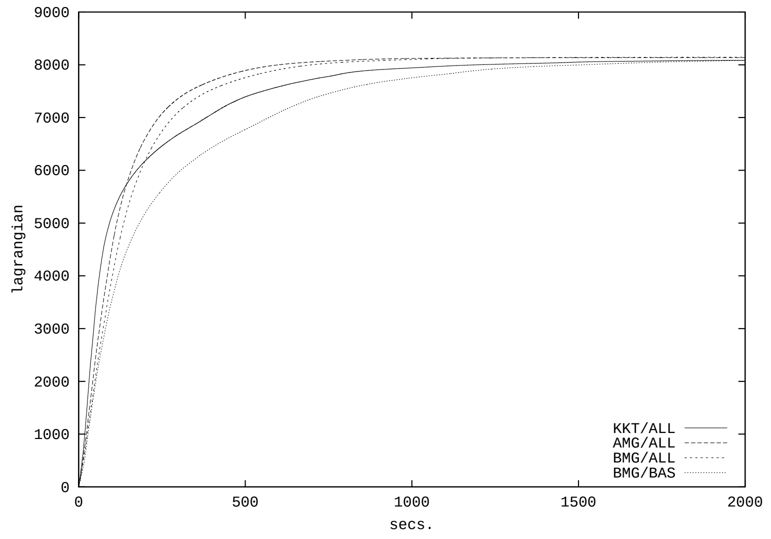
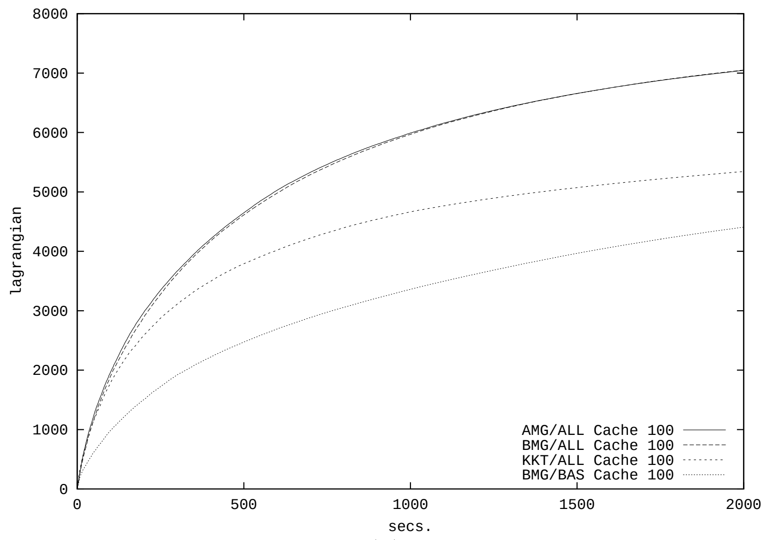
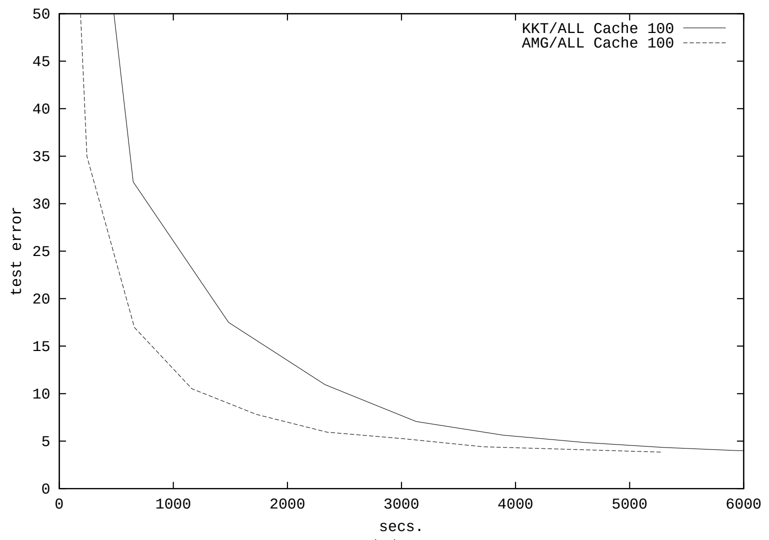


Figure 9: Comparison of different selection strategies using the heuristic MEAN.

logarithmic function is very slow to converge to zero, and because of that, the value returned by the strategies will be soon below the threshold. In particular the logarithmic function remains on a value of about 0.1 for many iterations. While this value is pretty good for the accuracy of the KKT solution, it is not sufficient for our selection schemes. In Figure 8 different heuristics for the computation of the value θ_V of the selection strategy of the SMO-like algorithm are compared. In this case the very simple heuristics MAX and MEAN reach similar performance, which is much better than the baseline scheme. In Figure 9, given the heuristic MEAN, different selection strategies



(a)



(b)

Figure 10: The effect of the cache limitation: (a) Lagrangian value versus time; (b) test performance versus time.

are compared. In this case, the new strategies slightly outperform the one based on Crammer and Singer's KKT conditions. Actually, as we will see in the following, this slight improvement is due to the big size of the cache of kernel matrix rows that prevents the algorithm suffering of the large amount of time spent in the computation of kernels that are not present in the cache.

In order to reproduce conditions similar to the ones occurring when dealing with large datasets, the size of the cache of kernel matrix rows has been reduced to 100 rows. As it is possible to see in figure 10-a a decrease in the performance is evident for each method, however, this decrease becomes more evident when KKT conditions are used as the pattern selection strategy. From the same figure we can see also a quite poor performance when the basic version of the SMO-like is used as a global optimization method. This demonstrates how important is to solve the overall problem one pattern at time. In fact, this leads to a decrease of the total number of patterns selected for optimization and consequently to a decrease of the number of kernel computations. This puts also in evidence the amount of time spent in kernel computation versus the amount of time spent in the optimization. Figure 10-b clearly shows that the same argument can be applied to the recognition accuracy.

5.5 Brief Discussion

The type of strategies we have analyzed in earlier sections are very similar to the ones used by SMO (Platt, 1998), modified SMO (Keerthi et al., 1999) and svmlight (Joachims, 1999) algorithms for binary SVM. In these cases, linear constraints involving dual variables which are related to different patterns (derived by KKT conditions over the bias term) are present. However, in our case, as in the Crammer and Singer's algorithm (Crammer and Singer, 2001), constraints involve dual variables which are related to the same pattern (but over different prototypes). This makes a difference in the analysis since it turns out that it is convenient to optimize as much as possible the reduced problem obtained for a single pattern as this optimization does not require the computation of new kernels. This claim is supported by our experimental results comparing BMG-ALL vs. BMG-BAS in Figure 10.

Also, we have shown experimentally that the use of heuristics based on the increase of the Lagrangian tend to be faster than KKT based ones, when used for pattern selection (compare BMG-ALL vs. KKT-ALL in Figure 10). This can be due to the fact that the number of different patterns selected along the overall optimization process tends to be smaller and this largely compensates the inefficiency derived by the computation of the increase of the Lagrangian and the thresholds. On the other hand, according to the same experimental analysis, KKT conditions help when used for the choice of pairs to optimize in the reduced problems obtained for a given pattern. According to these considerations, this mixed approach has been adopted in the experiments that follow.

6. Optimization of General MProtSVM

By now, we have analyzed the (static) problem obtained when the assignment is given. In this section, we describe methods for the optimization with respect to the assignments π as well. Naturally, the full problem is no longer convex. So, we first present an efficient procedure that guarantees to reach a stationary point of the objective function of the problem in Eq. (8) associated to MProtSVM. Then, we insert it in a stochastic search framework with the aim to improve the quality of the solutions we find.

6.1 Greedy Optimization of MProtSVM

In the following, an algorithm for the optimization of the problem in Eq. (8) is described. The algorithm consists of two steps: a step in which, fixed the values for the set of variables α, we select the assignments π's in such a way to minimize the primal value, followed by a step in which the optimization of the variables α is performed once fixed the assignments. Each of these steps will lead to an improvement of the objective function thus guaranteeing the convergence to a stationary point.

Let suppose to start by fixing an initial assignment $\pi(1)$ for the patterns. As we have already seen, the associated problem is then convex and can be efficiently solved for example by using the general scheme in Figure 3. Once that the optimal value for the primal, let say $P^*_{\pi(1)}$, has been reached, we can easily observe that the solution can be further improved by updating the assignments in such a way to associate each pattern $\mathbf{x}_i$ to a positive prototype having associated the minimal slack value, i.e. by setting the vector $\pi_i(2)$ so to have the unique 1 corresponding to the best performing positive prototype. However, with this new assignment $\pi(2)$, the variables α may no longer fulfill the second admissibility condition in Eq. (10). If this is the case, it simply means that the current solution $M(\alpha)$ is not optimal for the primal (although still admissible). Furthermore, α cannot be optimal for the dual given the new assignment since it not even admissible. Thus, a Lagrangian optimization, done by keeping the constraints dictated by the admissibility conditions in Eq. (10) satisfied for the new assignment, is guaranteed to obtain a new α with a better optimal primal value $P^*_{\pi(2)}$, i.e. $P^*_{\pi(2)} \leq P^*_{\pi(1)}$. For the optimization algorithm to succeed, however, KKT conditions on α have to be restored in order to return back to a feasible solution and then finally resuming the Lagrangian optimization with the new assignment $\pi(2)$. Admissibility conditions can be simply restored by setting $\alpha_i = \mathbf{0}$ whenever there exists any $r \in \mathcal{P}_i$ such that the condition $\alpha_i^r > 0 \wedge \pi_i^r = 0$ holds. Note that, when the values assigned to the slack variables allow to define a new assignment for π corresponding to a new problem with a better optimal primal value, then, because of convexity, the Lagrangian of the corresponding dual problem will have an optimal value that is strictly smaller than the optimal dual value of the previous problem.

Performing the same procedure over different assignments, each one obtained from the previous one by the procedure described above, implies the convergence of the algorithm to a fixed-point consisting of a stationary point for the primal problem when no improvements are possible and the KKT complementarity conditions are all fulfilled by the current solution.

One problem with this procedure is that it can result onerous when dealing with large datasets or when using many prototypes since, in this case, many complete Lagrangian optimizations have to be performed. For this, we can observe that for the procedure to work, at each step, it is sufficient to stop the optimization of the Lagrangian when we find a value for the primal which is better than the last found value and this is going to happen for sure since the last solution was found not optimal. This requires only a periodic check of the primal value when optimizing the Lagrangian.

6.2 Stochastic Modifications for MProtSVM Optimization

Another problem with the procedure given in the previous section is that it leads to a stationary point (either a local minima or a saddle point) that can be very far from the best possible solution. Moreover, it is quite easy to observe that the problem we are solving is combinatorial. In fact, since the induced problem is convex for each possible assignment, then there will exist a unique optimal primal value $P^*(\pi, \alpha^*(\pi))$ associated with optimal solutions $\alpha^*(\pi)$ for the assignments π. Thus, the overall problem can be reduced to find the best among all possible assignments. However, when assuming an equal number q of prototype vectors for each class, there are q^n possible solutions with many trivial symmetries.

Given the complexity of the problem we are trying to optimize, we propose to resort to stochastic search techniques. Specifically, the approach we suggest can be considered an instance of Iterated Local Search (ILS). ILS is a family of general purpose metaheuristics for finding good solutions of combinatorial optimization problems (Lourenco et al., 2002). These algorithms are based on building a sequence of solutions by first perturbing the current solution and then applying local search to that modified solution.

In the previous section, a way to perform approximated local search has been given. Let us now consider how to perturb a given solution. We propose to perform a perturbation that is variable with time and is gradually cooled by a simulated annealing like procedure.

For this, let us view the value of the primal as an energy function

$$E(\pi) = \frac{1}{2}||M||^2 + C\sum_i \langle \pi_i, \xi_i \rangle.$$

Let suppose to have a pattern $\mathbf{x}_i$ having slack variables ξ_i^r, $r \in \mathcal{P}_i$, and suppose that the probability for the assignment to be in the state of nature s (i.e. with the s-th component set to 1) follows the law

$$p_i(s) \propto e^{-\Delta E_s / T}$$

where T is the temperature of the system and $\Delta E_s = C(\xi_i^s - \xi_i^{y_i})$ the variation of the system energy when the pattern $\mathbf{x}_i$ is assigned to the s-th prototype. By multiplying every term $p_i(s)$ by the normalization term $e^{C(\xi_i^{y_i} - \xi_i^0)/T}$ where $\xi_i^0 = \min_{r \in \mathcal{P}_i} \xi_i^r$ and considering that probabilities over alternative states must sum to one, i.e. $\sum_{r \in \mathcal{P}_i} p_i(r) = 1$, we obtain

$$p_i(s) = \frac{1}{Z_i} e^{-\frac{C(\xi_i^s - \xi_i^0)}{T}} \tag{17}$$

with $Z_i = \sum_{r \in \mathcal{P}_i} e^{-C(\xi_i^r - \xi_i^0)/T}$ the partition function.

Thus, when perturbing the assignment for a pattern $\mathbf{x}_i$, each positive prototype s will be selected with probability $p_i(s)$. From Eq. (17) it clearly appears that, when the temperature of the system is low, the probability for a pattern to be assigned to a prototype different from the one having minimal slack value tends to 0 and we obtain a behavior similar to the deterministic version of the algorithm. The simulated annealing is typically implemented by decreasing the temperature, as the number of iterations increases, by a monotonic decreasing function $T = T(t, T_0)$.

Summarizing, an efficient realization of the ILS-based algorithm is obtained by substituting the true local optimization with one step of the algorithm in Section 6.1 and is given in Figure 11.

7. Generalization Ability of MProtSVM

In this section, we give a theoretical analysis of the generalization ability of the MProtSVM model. For simplicity, we consider MProtSVM with a fixed number q of prototypes per class. We first assume training data being separated by a MProtSVM model and we give a margin based upper bound on the error that holds with high probability on a independently generated set of examples. Then, we give a growth-function based bound on the error which do not assume linear separability of training data.

Margin based generalization bound Let us suppose that an *i.i.d.* sample $\mathcal{S}$ of n examples and a model M are given such that the condition in Eq. (7) holds for every example in $\mathcal{S}$, i.e.

$$\forall (\mathbf{x}_i, c_i) \in \mathcal{S}, \exists r \in \mathcal{P}_i : \langle M_r, \mathbf{x}_i \rangle \geq \theta_i + 1 \text{ and } \theta_i = \max_{r \in \mathcal{N}_i} \langle M_r, \mathbf{x}_i \rangle.$$

```
                    Annealed_MProtSVM()
$T := T_0$; randomly initialize $\pi(1)$;
compute the primal $E(1) := P_{\pi(1)}(0)$;
**for** $t = 1,..,t_{max}$
  **do** for all the examples $\mathbf{x}_p \in S$
       $\alpha_p$ = OptimizeOnPattern($\mathbf{x}_p$ , $\varepsilon$);
  **until** $P_{\pi(t)}(\alpha) < E(t)$;
  compute a new assignment $\pi(t+1)$ using $T(t, T_0)$ in Eq. (17);
  compute the new primal $E(t+1) := P_{\pi(t+1)}(\alpha)$;
  restore KKT conditions on $\alpha$ /*see Section 6*/
end;
```

Figure 11: Fast annealed algorithm for the optimization of MProtSVM.

With this assumption, fixing a pattern $\mathbf{x}_p$, there will be at least one slack variable ξ_p^r, associated with it, equal to zero. In fact, the condition $\xi_p^r = 0$ is true at least in the case $r = y_p$, where y_p is the positive prototype associated to the pattern $\mathbf{x}_p$, i.e. such that $\pi_p^{y_p} > 0$.

To give the margin-based bound on the generalization error, we use the same technique as in an(Platt et al., 2000) for general Perceptron DDAG[1] (and thus SVM-DAG also), i.e. we show how the original multiclass problem can be reduced into one made of multiple binary decisions. The structure of our proof resembles the one given in (Crammer and Singer, 2000) for single-prototype multiclass SVM.

A Perceptron DDAG is a rooted binary DAG with N leaves labelled by the classes where each of the $K = m(m-1)$ internal nodes is associated with a perceptron able to discriminate between two classes. The nodes are arranged in a triangle with the single root node at the top, two nodes in the second layer and so on until the final layer of m leaves. The i-th node in layer $j < m$ is connected to the i-th and $(i+1)$-st node in the $(j+1)$-st layer. A Perceptron DDAG based classification can also be though of as operating on a list of classes with associated a set of perceptrons, one for each different pair of classes in the list. The evaluation of new patterns is made by evaluating the pattern with the perceptron discriminating the classes in the first and in the last position of the list. The losing class between the two is eliminated from the list. This process is repeated until only one class remains in the list and this class is returned.

Similarly, MProtSVM classification can be thought of as operating on a list. Suppose the index of prototypes $r \in R = \{1,\ldots,mq\}$ are ordered according to their class $\mathcal{C}(r) \in \{1,\ldots,m\}$. Then,

1. Note that the term ”perceptron” here simply denotes a linear decision function which is not necessarily produced by the ”perceptron algorithm”.

given a new pattern, we compare the scores obtained by the two prototypes in the head and in the tail of the list and the loser is removed from the list. This is done until only prototypes of the same class remain on the list and this is the class returned for the pattern under consideration. It is easy to show that this procedure is equivalent to the rule in Eq. (2).

In the following, we will refer to the following theorem giving a bound on the generalization error of a Perceptron DDAG:

Theorem 2 *(Platt et al., 2000) Suppose we are able to classify a random sample of labelled examples using a Perceptron DDAG on m classes containing K decision nodes with margin γ_i at node i, then we can bound the generalization error with probability greater than $1-\delta$ to be less than*

$$\frac{1}{n}(130R^2D'\log(4en)\log(4n)+\log(\frac{2(2n)^K}{\delta}))$$

where $D'=\sum_{i=1}^K\gamma_i^{-2}$, and R is the radius of a ball containing the support of the distribution.

Note that, in this theorem, the margin γ_i for a perceptron $(\mathbf{w}_i,b_i)$ associated to the pair of classes (r,s) is computed as $\gamma_i=\min_{c_p\in\{r,s\}}|\langle\mathbf{w}_i,\mathbf{x}_p\rangle-b_i|$. Moreover, we can observe that the theorem depends only on the number of nodes (number of binary decisions) and does not depend on the particular architecture of the DAG.

Going back to MProtSVM, for the following analysis we define the hyperplane $\mathbf{w}_{rs}=M_r-M_s$ for each pair of prototypes indexes r,s such that $\mathcal{C}(r)<\mathcal{C}(s)$. and the *support* of the hyperplane $\mathbf{w}_{rs}$ as the subset of patterns

$$\Gamma_{rs}=\{i\in\{1,\dots,n\}:(r\in\mathcal{P}_i\wedge\pi_i^r>0)\vee(s\in\mathcal{P}_i\wedge\pi_i^s>0)\}.$$

Now, we can define the margin of the classifier $h_{rs}(\mathbf{x})=\langle\mathbf{w}_{rs},\mathbf{x}\rangle$ as the minimum of the (geometrical) margins of the patterns associated to it, i.e.

$$\gamma_{rs}=\min_{i\in\Gamma_{rs}}\frac{|h_{rs}(\mathbf{x}_i)|}{||\mathbf{w}_{rs}||} \tag{18}$$

Note that, from the hypothesis of separation of the examples and from the way we defined the margin, we have $|h_{rs}(\mathbf{x}_i)|\geq 1$ and hence the lower bound on the margin $\gamma_{rs}\geq||\mathbf{w}_{rs}||^{-1}$.

Now, we can show that the maximization of these margins leads to a small generalization error by demonstrating the following result.

Lemma 3 *Suppose we are able to classify a random sample of labelled examples using a MProtSVM with q prototypes for each of the m classes with margin γ_{rs} when $\mathcal{C}(r)<\mathcal{C}(s)$, then we can bound the generalization error with probability greater than $1-\delta$ to be less than*

$$\frac{1}{n}(130R^2D\log(4en)\log(4n)+\log(\frac{2(2n)^K}{\delta}))$$

where $D=\sum_{r,s:\ \mathcal{C}(r)<\mathcal{C}(s)}\gamma_{rs}^{-2}$, $K=\frac{1}{2}q^2m(m-1)$, and R is the radius of a ball containing the support of the distribution.

Proof. First of all, we show that an MProtSVM can be reduced to a Perceptron DDAG. Let be given prototype indices $r \in R = \{1,\dots,mq\}$ ordered according to their class $\mathcal{C}(r) \in \{1,\dots,m\}$. Consider two cursors r and s initially set to the first and the last prototype in R, respectively. Now, we build a DAG node for (r,s) based on the classifier h_{rs}. Then, recursively, left and right edges are built associated to nodes $(r,s-1)$ and $(r+1,s)$ respectively. This is made until the condition $\mathcal{C}(r) = \mathcal{C}(s) = t$ holds. When this is the case, a leaf node is built instead with label t. This construction is based on the fact that there is not need to compare the scores obtained by prototypes associated to the same class.

We show now that the number of nodes in the skeleton of a DAG D which is built in this way is exactly $K = \frac{1}{2}q^2m(m-1)$. In fact, consider the DAG D' obtained by keeping on constructing DAG nodes (r,s) until the condition $r = s$ holds, instead of just $\mathcal{C}(r) = \mathcal{C}(s)$. This graph would be the same that would have been obtained by considering mq classes with one prototype each. Note that D' is balanced and it consists of $\frac{1}{2}mq(mq-1)$ nodes. It follows that, to obtain the DAG D, for each class y, we subtract the subDAG constructed by considering all possible $k_y = q(q-1)/2$ pairs of prototypes associated to that class.

Summarizing, the number of nodes of the DAG D is the number of nodes of the balanced DAG D' minus the total number of mk_y subDAG nodes. That is we get:

$$K = \frac{1}{2}mq(mq-1) - m(\frac{1}{2}q(q-1)) = \frac{1}{2}q^2m(m-1).$$

Now, we can apply Theorem 2, by considering a Perceptron DDAG with K nodes associated to pairs $r,s:\ \mathcal{C}(r) < \mathcal{C}(s)$ and the margin for the node (r,s) defined as in Eq. (18).$\square$

By now, we have demonstrated that the minimization of the term $D = \sum_{r,s:\ \mathcal{C}(r)<\mathcal{C}(s)} \gamma_{rs}^{-2}$ proportional to the margin of the nodes of the Perceptron DDAG we have constructed, leads to a small generalization error. This result can then be improved by showing how these margins are linked to the norm of the MProtSVM matrix M and finally proving the following theorem.

Theorem 4 *Suppose we are able to classify a random sample of n labelled examples using a MProtSVM with q prototypes for each of the m classes and matrix M, then we can bound the generalization error with probability greater than* $1-\delta$ *to be less than*

$$\frac{1}{n}\left(130R^2q(m-1+q)||M||^2\log(4en)\log(4n) + \log(\frac{2(2n)^K}{\delta})\right)$$

where $K = \frac{1}{2}q^2m(m-1)$ *and R is the radius of a ball containing the support of the distribution.*

Proof. First of all, note that we have $\gamma_{rs}^{-2} \leq ||\mathbf{w}_{rs}||^2 = ||M_r - M_s||^2$ and $\sum_r M_r = 0$. The second condition can be easily verified. In fact, from conditions in Eq. (10), it follows

$$\sum_r M_r = \sum_r \sum_i y_i^r \alpha_i^r \mathbf{x}_i = \sum_i (\underbrace{\sum_r y_i^r \alpha_i^r}_{0})\mathbf{x}_i = 0.$$

Now, we have

$$\sum_{r,s:\ \mathcal{C}(r)<\mathcal{C}(s)} ||M_r - M_s||^2 \quad = \quad q(m-1)\sum_r ||M_r||^2 - 2\sum_{r,s:\mathcal{C}(r)<\mathcal{C}(s)} \langle M_r, M_s\rangle. \tag{19}$$

The second term of the equation above is

$$\begin{array}{rcl}\sum_{r,s:\ \mathcal{C}(r)<\mathcal{C}(s)}\langle M_r,M_s\rangle & = & \frac{1}{2}(\sum_{r,s}\langle M_r,M_s\rangle - \sum_{r,s,\ \mathcal{C}(r)=\mathcal{C}(s)}\langle M_r,M_s\rangle) \\ & = & -\frac{1}{2}\sum_{r,s:\ \mathcal{C}(r)=\mathcal{C}(s)}\langle M_r,M_s\rangle \\ & = & -\frac{1}{2}(\sum_r ||M_r||^2 + \sum_{r\neq s:\ \mathcal{C}(r)=\mathcal{C}(s)}\langle M_r,M_s\rangle).\end{array} \tag{20}$$

where the following inequality holds

$$\begin{array}{rcl}\sum_{r\neq s,\ \mathcal{C}(r)=\mathcal{C}(s)}\langle M_r,M_s\rangle & \leq & \sum_{j=1}^m \sum_{r\neq s:\ \mathcal{C}(r)=\mathcal{C}(s)=j}\langle M_r,M_s\rangle \\ & \leq & q^2\sum_{y=1}^m ||\tilde{M}_y||^2 \\ & \leq & q^2\sum_r ||M_r||^2\end{array} \tag{21}$$

once we set $\tilde{M}_y = \arg\max_{r:\ \mathcal{C}(r)=y}||M_r||$. Finally, substituting back Eq. (21) in Eq. (20) and Eq. (20) in Eq. (19) we obtain:

$$D \leq q(m-1+q)\sum_r ||M_r||^2$$

and the theorem easily follows. Note that this bound nicely generalizes the case of single prototype per class already shown in (Crammer and Singer, 2000).$\Box$

Growth function based generalization bound In the following, we give another kind of analysis of the generalization capability of our model based on the growth function. In order to do that, it is convenient to show that our multi-prototype model is equivalent to a three-layer network of perceptrons where the weights of the second and third layer are decided before learning. Thus, the free parameters of the network are only the weights of perceptrons in the first layer. As before, with no loss of generality, we assume to have q prototypes for each class $c \in \mathcal{Y}$.

Given a MProtSVM $H_M(\cdot)$, the corresponding network $\mathcal{N}_{H_M}$ is constructed as follows (we assume threshold perceptrons $(\mathbf{w},\theta)$ with output $o(\mathbf{x}) = \text{sign}(\langle\mathbf{w},\mathbf{x}\rangle - \theta)$):

First layer: $\forall r,s \in \Omega,\ \mathcal{C}(r) < \mathcal{C}(s)$, define the perceptron $h_{rs}^{(1)}$ with weight vector $\mathbf{w}_{rs}^{(1)} = M_r - M_s$ and $\theta_{rs}^{(1)} = 0$;

Second layer (AND): $\forall u \in \mathcal{Y} = \{1,\dots,m\}$, $\forall v \in \Omega : \mathcal{C}(v) = u$ define the perceptron $h_{uv}^{(2)}$, taking input from all $h_{rs}^{(1)}$ such that $r = v$ or $s = v$ and connection equal to 1 if $r = v$, -1 otherwise; set threshold to the value $\theta_{uv}^{(2)} = q(m-1) - 0.5$ ("on" if all the inputs are 1).

Third layer (OR): $\forall w\ \ such that\ \ w \in \mathcal{Y} = \{1,\dots,m\}$ define the perceptron $h_w^{(3)}$, taking input from all $h_{uv}^{(2)}$ $such that$ $w = u$ and connections all equal to 1; set the threshold to the value $\theta_w^{(3)} = 1/2$ ("on" if any input is 1).

See Figure 12 for an example of network construction when $q = 2$ and $m = 3$. Notice that, by construction, for any input there will be no two activated perceptrons $h_{uv}^{(2)}$ and $h_{\hat{u}\hat{v}}^{(2)}$ such that $u \neq \hat{u}$. So, only one out of the perceptrons at the third layer will be activated, and its index will correspond to the predicted class.

The constructed network has $\sigma = \frac{q^2m(m-1)}{2}$ perceptrons at the first layer. Since only these perceptrons have trainable weights, the VC-dimension of the network only depends on these free parameters (apart for the hard-threshold functions).

We are now ready to state the following result.

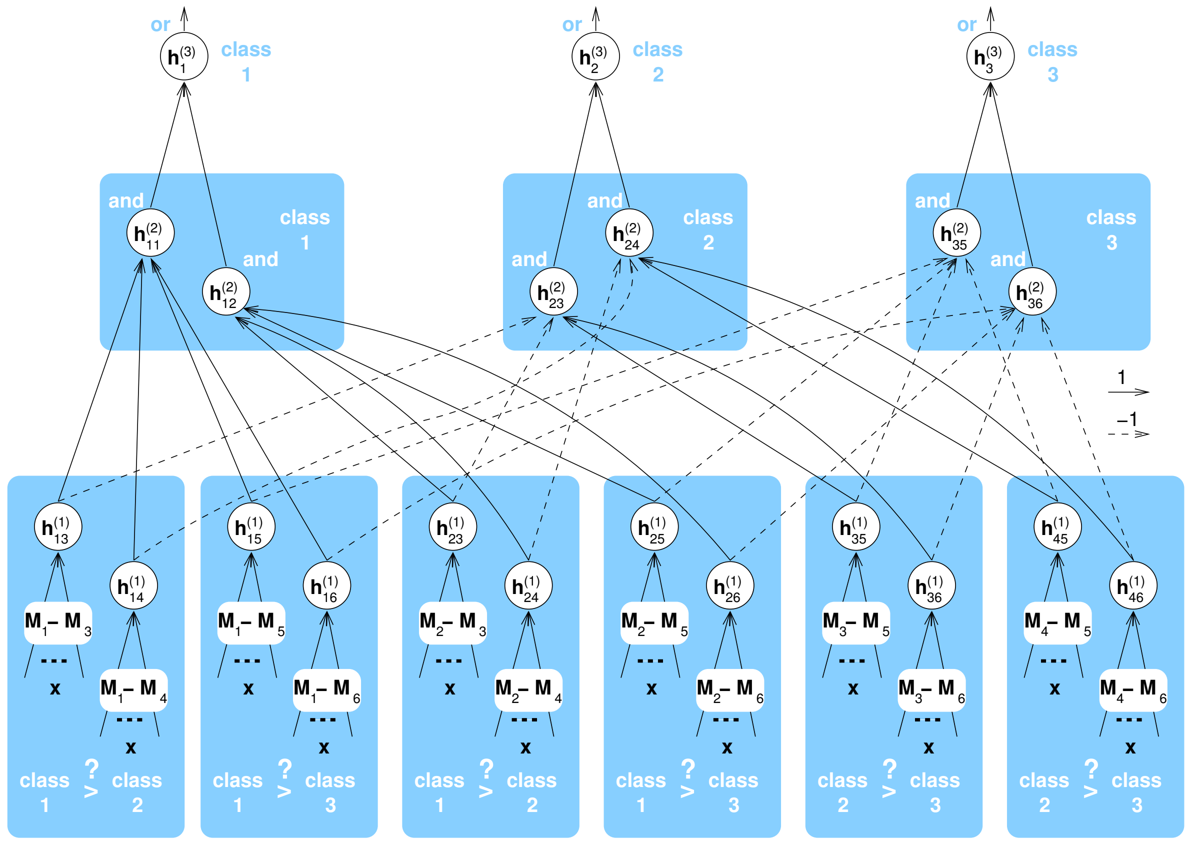


Figure 12: Example of network construction when $q = 2$ and $m = 3$. With this setting, prototypes M_1 and M_2 are associated with class 1, prototypes M_3 and M_4 are associated with class 2, and prototypes M_5 and M_6 are associated with class 3.

Theorem 5 *For any* $0 < \delta < 1$, *any MProtSVM* $H_M(\cdot)$ *with* q *prototypes for each of the* m *classes, given* $\mathcal{S}$ *a sample of size* n *drawn i.i.d. from* $\mathcal{D}_{\mathbb{R}^d \times \{1,\ldots,m\}}$, *with probability at least* $1-\delta$

$$err_{\mathcal{D}}(H_M) \leq err_{\mathcal{S}}(H_M) + \sqrt{4\frac{1+(qm+\frac{1}{2})ln(qm)+\frac{dq^2m(m-1)}{2}ln(2en/d)-ln(\delta/4)}{n}} + \frac{1}{n}.$$

Proof. By the above construction, the class of functions computable by a MProtSVM with q prototypes for each of the m classes is contained in the class of functions computable by three-layer perceptrons defined as above. This class of functions is completely characterized by the set of collective states that the σ perceptrons at the first layer can assume. It is well known (Kearns and Vazirani, 1994) that, by Sauer Lemma, the growth function of a single perceptron (with 0 threshold) is bounded from above by the quantity $(en/d)^d$, and so the growth function of our class of networks is bounded from above by the quantity $(en/d)^{d\sigma}$.

This bound, however, does not consider that not all the possible configurations of σ bits can be generated by the first layer. In fact, by construction of the network, we have that if $h^{(1)}_{rs}(\mathbf{x}) = 1$

and $h^{(1)}_{s\hat{s}}(\mathbf{x}) = 1$, then for sure $h^{(1)}_{r\hat{s}}(\mathbf{x}) = 1$, as well as, if $h^{(1)}_{rs}(\mathbf{x}) = 0$ and $h^{(1)}_{s\hat{s}}(\mathbf{x}) = 0$, then for sure $h^{(1)}_{r\hat{s}}(\mathbf{x}) = 0$. This is the result of the fact that, given an input vector $\mathbf{x}$, the outputs of the first layer perceptrons are fully determined by the total order over the MProtSVM prototypes induced by the score functions $f_r(\cdot)$. Thus we can compute an upper bound on the proportion of 'legal' configurations by considering all possible permutations of the qm prototypes divided by all possible configurations, i.e., 2^σ. Notice that this is an upper bound since when considering prototypes of the same class, we do not care about their relative order.

So we can bound the growth function of our class of networks by

$$\frac{(qm)!}{2^\sigma}(en/d)^{d\sigma} < \sqrt{2\pi qm}(qm/e)^{qm}e^{\frac{1}{12qm}}2^{-\sigma}(en/d)^{d\sigma},$$

where the last inequality has been obtained by using Stirling's formula.

Making explicit the value of σ, the right term of the above inequality can be written as

$$\sqrt{2\pi}(qm)^{qm+\frac{1}{2}}(n/d)^{\frac{dq^2m(m-1)}{2}}e^{\frac{6q^3m^2(m-1)(d-ln(2))-12q^2m^2+1}{12qm}}.$$

Now, we can apply Theorem 4.1 in (Vapnik, 1998) (involving the logarithm of the growth function for a sample of dimension $2n$) obtaining

$$\begin{aligned} err_{\mathcal{D}}(H_M(\cdot)) &\leq err_{\mathcal{S}}(H_M(\cdot)) + \\ &\quad + \sqrt{4\frac{ln(\sqrt{2\pi}(qm)^{qm+\frac{1}{2}}(\frac{2n}{d})^{\frac{dq^2m(m-1)}{2}}e^{\frac{6q^3m^2(m-1)(d-ln(2))-12q^2m^2+1}{12qm}}) - ln(\frac{\delta}{4})}{n}} + \frac{1}{n} \\ &\leq err_{\mathcal{S}}(H_M(\cdot)) + \sqrt{4\frac{1 + (qm+\frac{1}{2})ln(qm) + \frac{dq^2m(m-1)}{2}ln\frac{2en}{d} - ln\frac{\delta}{4}}{n}} + \frac{1}{n} \end{aligned}$$

□

8. Experimental Results

In the following, we report experiments we have done for testing the complexity and the generalization performance of the MProtSVM model with respect to other state-of-the-art algorithms. We choose to compare our model against results already published in literature on different datasets instead of doing experiments with those methods directly. This is because we have not available the code for all those methods and hence a re-implementation would be necessary. This can potentially introduce errors or uncorrect use of the methods and it is far more onerous for us. For this, we experimented on our model trying to replicate the initial conditions of published results as much as possible in such a way to obtain fair comparisons.

For all the following experiments, the linear kernel $K(\mathbf{x},\mathbf{y}) = (\langle\mathbf{x},\mathbf{y}\rangle + 1)$ has been used. Moreover, the annealing process required by MProtSVM has been implemented by decreasing the temperature of the system with the exponential law:

$$T(t,T_0) = T_0(1-\tau)^t$$

where t is the current iteration, $0 < \tau < 1$ and $T_0 > 0$ are external parameters. We used $T_0 = 10$ for all the following experiments. In addition, the only free parameter C of MProtSVM has been selected by performing validation of the model on a subset of the training set with values $C = \{10^k, k = -2,..,2\}$.

Initially, we tested our model against three multiclass datasets that we briefly describe in the following:

NIST: it consists of a 10-class task of 10705 digits randomly taken from the NIST-3 dataset. The training set consists of 5000 randomly chosen digits, while the remaining 5705 digits are used in the test set.

USPS: it consists of a 10-class OCR task (digits from 0 to 9) whose input are the pixels of a scaled digit image. There are 7291 training examples and 2007 test examples.

LETTER: it consists of a task with 26 classes consisting of alphabetic letters A-Z. Inputs are measures of the printed font glyph. The first 15000 examples are used for training and the last 5000 for testing.

q	LVQ2.1 Error %	MProtSVM Error %
1	7.43	6.45
5	4.68	3.63
10	4.35	3.28
15	3.52	2.80

Table 1: Comparison of generalization performances between MProtSVM and LVQ with increasing number of prototypes/codewords (NIST dataset, $\tau = .05$, $\beta = 0.002 \times q$).

q	USPS Error (%)	q	LETTER Error (%)
1	8.12	1	21.36
3	6.13	3	9.64
5	5.83	5	6.42
10	5.48	10	4.84
15	5.23	15	3.16
20	5.00	20	2.94

Table 2: (a) Test error of MProtSVM on the USPS dataset ($\tau = .05$, $\beta = 0.00137 \times q$), with an increasing number of prototypes; (b) Test error of MProtSVM on the LETTER dataset ($\tau = .05$, $\beta = 0.00043 \times q$), with an increasing number of prototypes.

A first set of experiments have been performed to compare the generalization performance of our (linear) model versus LVQ (Kohonen et al., 1996), which seemed to us the most comparable model, into an OCR task. For this, we have reported the results obtained by the LVQ2.1 version of the algorithm in (Sona et al., 2000) on the NIST problem. Configurations with a higher number of codewords started to overfit the data. As it can be seen in Table 1, MProtSVM performs significantly better with the same number of parameters. This can be due to the more effective control of the margin for our model w.r.t. LVQ models. On the same dataset, the tangent-distance based TVQ algorithm (Aiolli and Sperduti, 2002b) has obtained the best result, a remarkable 2.1% test error, and polynomial SVM's have obtained a 2.82% test error. These last results should not surprise since their models are well suited for OCR tasks. Here and in the following experiments we report the value of the factor $\beta = (m \times q)/n$ defined as the number of prototypes produced as a fraction of the cardinality of the training set. This represent a sort of factor of compression in the model.

A second set of experiments have been performed to test the MProtSVM model against state-of-the-art methods on two well known datasets: UCI Irvine USPS and LETTER. The obtained results are reported in Table 2. As it is possible to see, by combining a reasonably high number of linear prototypes, we have been able to obtain performances almost comparable with the ones obtained using non-linear models. In fact, on the USPS dataset, we obtained a 4.63% error using an our own SProtSVM implementation with polynomial kernel of degree 3 and without further preprocessing of the data. Finally, a 5.63% test error performance has been obtained using 1-NN. Concerning the LETTER dataset, the results should be compared to versus the 1.95% obtained in (Crammer and Singer, 2001) by SProtSVM with exponential kernel and to the 4.34% obtained by 1-NN. Although obtained with a slightly different split of the LETTER dataset (16000 examples for training and 4000 for test), we would like to mention the results reported in (Michie et al., 1994) where LVQ yielded a 7.9%.

From these experiments it is clear that MProtSVM returns far more compact models with respect to state of the art non-linear kernel methods allowing a (one or two order) reduced response time in classification while preserving a good generalization performance. In fact, the above experiments have shown very low values for the compression factor β (e.g. 26×20 prototypes in the LETTER dataset gives $\beta = 0.013$ and 10×20 prototypes for USPS gives $\beta = 0.0274$). Notice that β can be directly compared with the fraction of support vectors in kernel machines. Thus, MProtSVMs also give us a way to decide (before training) the compression factor we want to obtain.

	— Vectors —		**Errors**					
Dataset	**SVM**	**RVM**	**SVM**	**RVM**	**MProtSVM**			
					q=1	q=3	q=5	q=10
Banana	135.2	11.4	10.9	**10.8**	46.0	12.8	[11.0]	11.0
Breast Cancer	116.7	6.3	**26.9**	29.9	28.2	**26.9**	27.5	[27.0]
German	411.2	12.5	22.6	**22.2**	[23.6]	23.8	23.5	23.7
Image	166.6	34.6	3.0	3.9	15.0	3.2	2.7	[**2.5**]
Titanic	93.7	65.3	**22.1**	23.0	[22.5]	22.2	22.2	22.2
Waveform	146.4	14.6	10.3	10.9	13.3	10.8	**10.0**	[10.2]

Table 3: Comparison of solution complexity and generalization error of MProtSVM with respect to SVM and Tipping's RVM on a set of UCI binary datasets. The results quoted for SVM and RVM are taken from (Tipping, 2001). Values in brackets are the ones obtained using the model suggested by model selection performed over the number of prototypes.

To further validate our claim, we made a comparison of our technique against others that explicitly try to obtain compact models. In Table 3 we reported the results obtained with six binary

Dataset	**# Features**	**Train Size**	**Test Size**
Banana	2	400	4900
Breast Cancer	9	200	77
German	20	700	300
Image	18	1300	1010
Titanic	3	150	2051
Waveform	21	400	4600

Table 4: General information about the UCI binary datasets used in the experiments.

problems (see Table 4 for general information about these datasets) from the benchmark of Rätsch available over the web[2], exactly the ones used by (Tipping, 2001) of which we are reporting the obtained results for RVM and SVM. They correspond to averages over the first 10 splits of the collection. For each dataset, it is reported the average number of support vectors generated by SVM and RVM, the generalization error obtained with these two methods and the results obtained using four very simple MProtSVM configurations, namely made of 1, 3, 5 and 10 prototypes per class (2, 6, 10 and 20 vectors in total, respectively[3]). It is possible to see how very compact and simple models performs as good as (sometimes better than) state-of-the-art methods. Values in brackets represent the error value obtained using the model suggested by the validation procedure we have performed over the number of models per class.

Finally, in Table 5 we report an example of the values obtained for the objective function of the primal problem in Eq. (8) along with their corresponding test errors obtained using different configurations and lowering the simulated annealing parameter τ on the USPS dataset. As expected, once fixed a raw in the table, better values for the primal can generally be obtained with lower values of τ. Moreover, as the number of prototypes per class increases, the choice of small τ tends to be more crucial. Anyway, higher values for τ, and thus not optimal values for the primal, can nevertheless lead to good generalization performances. Notice that from the fact that the primal value is just a way to approximate the theoretical SRM principle and from the non-optimality of the parameter C in these experiments, better values for the primal does not necessarily correspond to better values for the test error.

9. Conclusions

We have proposed an extension of multiclass SVM able to deal with several prototypes per class. This extension defines a non-convex problem. We suggested to solve this problem by using a novel efficient optimization procedure within an annealing framework where the energy function corresponds to the primal of the problem. Experimental results on some popular benchmarks demonstrated that it is possible to reach very competitive performances by using few linear models per class instead of a single model per class with kernel. This allows the user to get very compact models which are very fast in classifying new patterns. Thus, according to the computational constraints, the user may decide how to balance the trade-off between better accuracy and speed of

2. `http://ida.first.gmd.de/~raetsch`
3. When one prototype per class is used in a binary problem, as in this case, MProtSVM actually generates two vectors that are the same with sign inverted. Thus, they can be compacted into one vector only with no loss of information.

q	$\tau = 0.2$	$\tau = 0.1$	$\tau = 0.05$	$\tau = 0.03$
3	7.44626 (6.33%)	7.28049 (6.03%)	7.08138 (6.13%)	7.04274 (6.48%)
5	7.49136 (6.08%)	7.27318 (5.63%)	7.10498 (5.83%)	7.00946 (5.58%)
10	7.82233 (5.58%)	7.51780 (5.88%)	7.27596 (5.48%)	7.12517 (5.23%)
15	7.82222 (5.33%)	7.57009 (5.73%)	7.38722 (5.33%)	7.22250 (5.53%)
20	7.78410 (5.48%)	7.79388 (5.72%)	7.49125 (5.38%)	7.21303 (5.53%)

Table 5: Primal values and generalization error obtained with different configurations varying the parameter τ for the USPS dataset.

classification. Finally, it should be noted that the proposed approach compares favorably versus LVQ, a learning procedure that, similarly to the proposed approach, returns a set of linear models.

Preliminary experiments with kernels have shown negligible improvements that makes us to consider this extension not worthwhile of further investigations. An alternative more interesting extension would be to try to combine different types of kernels together in the same model.

Acknowledgments

This work has been developed when Fabio Aiolli was a PhD student and postdoc at the Department of Computer Science, Univ. of Pisa. The research was partially supported by Italian MIUR project 2003091149_005. We would like to thank Y. Singer, K. Crammer, and the anonymous reviewers for their valuable suggestions on how to improve the paper.

References

F. Aiolli and A. Sperduti. An efficient SMO-like algortihm for multiclass SVM. In *Proceedings of IEEE workshop on Neural Networks for Signal Processing*, pages 297–306, 2002a.

F. Aiolli and A. Sperduti. A re-weighting strategy for improving margins. *Artificial Intelligence Journal*, 137/1-2:197–216, 2002b.

F. Aiolli and A. Sperduti. Multi-prototype support vector machine. In *Proceedings of International Joint Conference of Artificial Intelligence (IJCAI)*, 2003.

E. Allwein, R. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 2000.

K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, pages 35–46, 2000.

K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based machines. *Journal of Machine Learning Research*, 2(Dec):265–292, 2001.

T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

T. Downs, K. E. Gates, and A. Masters. Exact simplification of support vector solutions. *Journal of Machine Learning Research*, 2:293–297, 2001.

G. M. Fung, O. L. Mangasarian, and A. J. Smola. Minimal kernel classifiers. *Journal of Machine Learning Research*, 3:303–321, 2002.

Y. Guermeur, A. Elisseeff, and H. Paugam-Moisy. A new multi-class SVM based on a uniform convergence result. In *Proceedings of the IJCNN*, 2000.

T. Joachims. Making large-scale SVM learning practical. In *Advances in Kernel Methods - Support Vector Learning*. B. Schlkopf and C. Burges and A. Smola (ed.), MIT Press, 1999.

M. J. Kearns and U. V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.

S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. Improvements to platt's smo algorithm for SVM classifier design. Technical Report CD-99-14, Control Division, Dept. of Mechanical and Production Engineering, National University of Singapore, 1999.

T. Kohonen, J. Hynninen, J. Kangas, J. Laaksonen, and K. Torkkola. Lvq_pak: The learning vector quantization program package. Technical Report A30, Helsinki University of Technology, Laboratory of Computer and Information Science, January 1996. http://www.cis.hut.fi/nnrc/nnrc-programs.html.

H. R. Lourenco, O. C. Martin, and T. Stutzle. Iterated local search. *Handbook of Metaheuristics*, Ed. F. Glover and G. Kochenberger, International Series in Operations Research & Management Science(57):321–353, 2002.

D. Michie, D. Speigelhalter, and C. Taylor. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, 1994.

T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

J. Platt, N. Cristianini, and J. Shawe Taylor. Large margin DAGs for multiclass classification. In S. A. Solla, T. K. Leen, and K. R. Muller, editors, *Advances in Neural Information Processing Systems*. MIT Press, 2000.

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

J. R. Quinlan. *C4.5: Programs for Empirical Learning*. Morgan Kaufmann, San Francisco, CA, 1993.

D. E. Rumelhart, G. E. Hinton, and R.J Williams. Learning internal representation by error propagation. In *Parallel Distributed Processing - Explorations in the Microstructure of cognition*, chapter 8, pages 318–362. MIT Press, 1986.

B. Schölkopf, S. Mika, C. J. C. Burges, P. Knirsch, K. R. Muller, G. Rätsch, and A. J. Smola. Input space versus feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, 5 (10):1000–1017, 1999.

B. Schölkopf and C. Burges and V. Vapnik. Extracting support data for a given task. In *First International Conference on Knowledge Discovery & Data Mining*, pages 252–257, 1995.

D. Sona, A. Sperduti, and A. Starita. Discriminant pattern recognition using transformation invariant neurons. *Neural Computation*, 12(6), 2000.

M. E. Tipping. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, 2001.

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

J. Weston and C. Watkins. Multiclass support vector machines. In M. Verleysen, editor, *Proceedings of ESANN99*. D. Facto Press, 1999.

Journal of Machine Learning Research 6 (2005) 851-881 Submitted 11/03; Revised 9/04; Published 5/05

Prioritization Methods for Accelerating MDP Solvers

David Wingate WINGATED@CS.BYU.EDU
Kevin D. Seppi KSEPPI@CS.BYU.EDU
Computer Science Department
Brigham Young University
Provo, UT 84602, USA

Editor: Sridhar Mahadevan

Abstract

The performance of value and policy iteration can be dramatically improved by eliminating redundant or useless backups, and by backing up states in the right order. We study several methods designed to accelerate these iterative solvers, including prioritization, partitioning, and variable reordering. We generate a family of algorithms by combining several of the methods discussed, and present extensive empirical evidence demonstrating that performance can improve by several orders of magnitude for many problems, while preserving accuracy and convergence guarantees.

Keywords: Markov Decision Processes, value iteration, policy iteration, prioritized sweeping, dynamic programming

1. Introduction

This paper systematically explores the idea of minimizing the computational effort needed to compute the optimal policy (with its value function) of a discrete, stationary Markov Decision Process using an iterative solver such as value or policy iteration. The theme of our exploration can be stated generally as "backing up states in the right order," and to accomplish that, we present and discuss several methods of differing complexity which structure value dependency and prioritize computation to follow those dependencies. We have named the resulting family of algorithms *General Prioritized Solvers*, or GPS.

Many problems in reinforcement learning are well modeled as MDPs. Optimal policies for such MDPs are often computed by iteratively improving an existing policy, which can be accomplished by computing (or approximating) the value function of the existing policy. Computing each value function is generally a non-trivial task, meaning that the ability to compute them quickly enables larger and more complicated problems to be solved. As Andre et al. (1998) point out, there is also a classic tradeoff in reinforcement learning between spending time acting in the environment and spending time planning what to do in the environment. GPS is designed to help navigate that tradeoff – and help other algorithms navigate that tradeoff – by allocating computational effort intelligently.

GPS can also improve the performance of algorithms which rely on accurate value function estimates to make *other* decisions, by reducing their computational overhead. For example, Munos and Moore (2002) use the value function to guide discretization decisions, and Kearns and Singh (2002) use it to decide between exploration and exploitation. Value iteration is also used as part of larger algorithms: RTDP (Barto et al., 1995) performs some value iteration off-line between executing controls, and Modified Policy Iteration (Puterman and Shin, 1978) performs some value iteration between policy improvement steps. In addition, GPS can enhance algorithms that propagate different forms of information (not just *value* information). For example, Munos and Moore (2002) propagate both "influence" and "variance" throughout a problem using a form of value iteration. In this more abstract sense, the principle of propagating knowledge throughout a space as quickly and efficiently as possible is applicable to almost all systems.

Two principal observations motivated this work. First, many backups performed by value iteration can be useless. Value iteration is almost a pessimal algorithm, in the sense that it never leverages any advantage a sparse transition matrix (and/or sparse reward function) may offer: it always iterates over and updates every state, even if such a backup does not (or cannot) change the value function. An intuitive improvement is this: if, on the previous sweep, only a handful of states changed value, why back up the value of *every* state on the next sweep? The only useful backups will be to those states which depend upon states that changed on the previous sweep. Similar observations about the efficient ordering of work can be made about policy iteration: it is best to wait to compute the policy of a state s until a good policy for the dependents of s has been determined.

Second, almost all backups are naively ordered. For example, ordering the states in an acyclic problem such that the rows in the transition matrix are triangular (corresponding to a topological sort) yields a $O(n)$ solution; but solving the same system in an arbitrary order yields an expected $O(n^2)$ solution time. Additionally, as information backpropagates through a value function estimate, the optimal ordering may change. Dynamically generating a good backup ordering in an efficient way is one of the central issues we examine.

The idea of efficient computation applied to value iteration and policy iteration is not new, but it has not received a dedicated treatment. This paper makes a fourfold contribution: first, it studies prioritization metrics systematically, comparing and contrasting them to each other. Most other papers have only presented a metric in isolation, as a heuristic performance enhancer. Prioritized Sweeping (Moore and Atkeson, 1993), for instance, uses Bellman error as a priority metric, but we demonstrate that another equally simple metric can perform better. Second, this paper points out how the complexity introduced with the priority metrics can be managed through the use of partitioning, which is an issue other researchers have not addressed. Partitioning also enables prioritized policy iteration, which has not been studied previously. Third, this paper introduces a new priority metric, *H2*, and an effective variable reordering algorithm designed to improve performance. Both are studied empirically, and some general guidelines for their use are established. Fourth, and somewhat in contrast to most asynchronous value iteration proofs of convergence (such as Bertsekas, 1982, 1983; Gullapalli and Barto, 1994), we point out that not every state needs to be backed up during each sweep in order to guarantee convergence. In fact, some states may never need to be backed up at all, which is valuable for optimizing performance.

The paper is organized as follows. Section 2 describes GPS, and discusses prioritization, partitioning, variable reordering, and convergence and stopping criteria. Section 3 presents our experimental setup and Section 4 presents the experimental results. Section 5 briefly points out related work, and Section 6 presents conclusions and ideas for future research. Additionally, an on-line appendix is available, which is described at the end of the paper.

2. The GPS Family of Algorithms

There are three principal enhancements we use to accelerate value and policy iteration: prioritization, partitioning and variable reordering. Sections 2.1, 2.3 and 2.5 discuss each in detail, along with issues that each raises. A general discussion of convergence, stopping, and complexity is deferred until Section 2.6. We shall study many combinations of these enhancements, but consider all variants to be members of a single family. We begin with a prototype MDP solver, which has elements common to all members:

Algorithm 1 Abstract GPS

```
Initialization
1: //  Partition the problem
2: //  Order variables within each partition
3: //  Compute initial partition priorities

Main Loop
1: repeat
2:    //  Select a partition $p$
3:    //  Compute the optimal policy and value function of states in $p$,
      //       keeping all other partitions constant
4:    //  Recompute priorities of partitions depending on $p$
5: until  convergence
```

To simplify the following discussions, all of the MDPs we consider are discounted, infinite horizon, stationary and positive bounded (all rewards are positive and finite).

2.1 Prioritization

The first method we use to improve efficiency is the prioritization of backups. Instead of naively sweeping over the entire problem, we wish to work our way backwards through the problem: we correct the value function estimate (and policy) for a state s by backing it up, and then correct the value function estimate (and policy) for all states which depend upon s. This has the effect of focusing computation in regions of the problem which are expected to be maximally productive, and simultaneously avoids useless backups.

To accomplish this, we begin with a standard value function definition:

$$V(s) = \max_{a\in A}\left\{R(s,a)+\gamma\sum_{s'\in S}Pr(s'|s,a)V(s')\right\}. \tag{1}$$

Here, $s\in S$ is a state, $a\in A$ is an action, $\gamma\in[0,1)$ is the discount factor, $R(s,a)$ is the reward function, and $Pr(s'|s,a)$ is the probability of transitioning to state s' if action a is taken in state s. Algorithm 2 shows the traditional value iteration algorithm, without prioritization.

We use *Bellman error* to characterize how useful any given backup is, and then construct different metrics based on the Bellman error as the priority in a priority queue:

$$B_t(s) = \max_{a\in A}\left\{R(s,a)+\gamma\sum_{s'\in S}Pr(s'|s,a)V_t(s')\right\} - V_t(s).$$

Algorithm 2 Standard Value Iteration

```
1: $V_0 \leftarrow 0$
2: repeat
3:    for all $s \in S$ do
4:       $V_t(s) \leftarrow \max_{a \in A} \{R(s,a) + \gamma \sum_{s' \in S} Pr(s'|s,a) V_{t-1}(s')\}$
5:    end for
6:    $t \leftarrow t+1$
7: until convergence
```

Note that this should not be considered a one-step temporal difference: Bellman error represents the amount of *potential* change to the value function, assuming that a certain state was backed up, as opposed to the *actual* difference between two value function estimates separated by one timestep. Peng and Williams (1993) called this value the *prediction difference*. We will let

$$M_t = \|B_t\|_\infty$$

be the largest potential update in the system, where $\|\cdot\|_\infty$ represents max-norm. This quantity is commonly called the *Bellman error magnitude* (Williams and Baird, 1993).

We build different prioritization metrics upon the Bellman error function. The first metric we will analyze, *H1*, is equal to the Bellman error itself:

$$H1_t(s) = B_t(s).$$

The second metric is:

$$H2_t(s) = \begin{cases} B_t(s) + V_t(s) & \text{if } B_t(s) > \varepsilon \\ 0 & \text{otherwise.} \end{cases}$$

When it is not important which prioritization metric is used, we will use $H_t(s)$ to refer to a generic one. The next section discusses the semantics of each metric.

Once a state s is backed up, the priority of any state depending on s must be recomputed. The *state dependents of a state* is the set of all states who have some probability of transitioning to s, and therefore whose value depend on the value of s. We define it as

$$SDS(s) = \left\{s' : \exists a Pr(s'|s,a) \neq 0\right\}.$$

Algorithm 3 shows a prioritized version of value iteration.

As noted, we only consider positive bounded MDPs. Creating a positive bounded MDP can be accomplished by adding a constant C to the reward function; since we will initialize the value function estimate to 0, this ensures that $V_t \leq V^*$ (where V^* is the value of the optimal policy π^*). This does not change the resulting policy, and, as Zhang et al. (1999) point out, "the value function of the original [MDP] equals that of the transformed [MDP] minus $C/(1-\gamma)$, where C is the constant added." This stipulation is required by the *H2* metric, and simplifies some of the bounds provided in Section 2.6.

Algorithm 3 Prioritized Value Iteration

```
1: repeat
2:     $s \leftarrow \arg\max_{\xi \in S} H(\xi)$
3:     $V(s) \leftarrow \max_{a \in A} \{R(s,a) + \gamma \sum_{s' \in S} Pr(s'|s,a) V(s')\}$
4:     for all $s' \in SDS(s)$ do
5:         // recompute $H(s')$
6:     end for
7: until convergence
```

2.2 Selecting Metrics

The *H1* prioritization metric is the most obvious metric, and has been studied before (although not in contrast to other metrics, and in somewhat different contexts than ours). Using it, GPS can be thought of as a greedy reduction in the error of the value function estimate. This has the tendency to propagate information quickly throughout the state space, but it also tends to leave large regions only partially converged, and therefore does not necessarily propagate *correct* information quickly.

The *H2* metric has a very different effect on computation order. The intuition is this: if there is a value that is more than ε away from its optimal value, the value will eventually need to be corrected. Since large values (generated from large rewards, or small loops) have greater influence on the value function than small values, *H2* converges large values before propagating their influence throughout the state space. This tends to ensure that regions are fully converged before anything depending on the region is processed. Experimental results illustrating these effects are shown in Figure 1.

The results in Section 4 demonstrate that neither *H1*, *H2*, nor standard value iteration induce an optimal backup ordering for all MDPs. However, each performs better than the others for some problems. The question of which metric should be used on a new problem naturally arises, but it is difficult to find topological features which accurately predict the performance of each metric. Often, the best metric seems to be a hybrid of all three.

Normal value iteration yields a very good backup order when a problem is close to being fully connected (and thus, whenever states are highly interdependent). The obvious corollary is that value iteration is also very good for any subgraph that is close to fully connected. Value iteration performs poorly when the problem exhibits highly sequential (and thus, asymmetric) dependencies, which can be due to a large number of strongly connected components, or a large graph diameter relative to the number of nodes.

The *H1* metric performs best in graphs which have highly sequential dependencies, which occur in acyclic graphs and in graphs with long loops. The *H1* metric excels at avoiding useless backups, but tends not to iron out feedback loops completely, meaning that states within such loops must often be processed multiple times.

The advantage of the *H2* metric is more difficult to quantify. *H2* tries to ensure that states have converged before moving on to those states' dependents. Conceptually, this is an appealing idea, but practically it is very difficult to make it work well without the addition of partitions (discussed in the next section). *H2* needs some cycles to generate a different order than *H1*, but does poorly with too many cycles. Figure 2 illustrates a problem for which *H2* is highly suboptimal, and Figure 3 shows performance visually: *H2* selects one state and "spirals" its value upwards, then selects another state and spirals, then a third, and back to the first, in a loop. However, value iteration works on all four

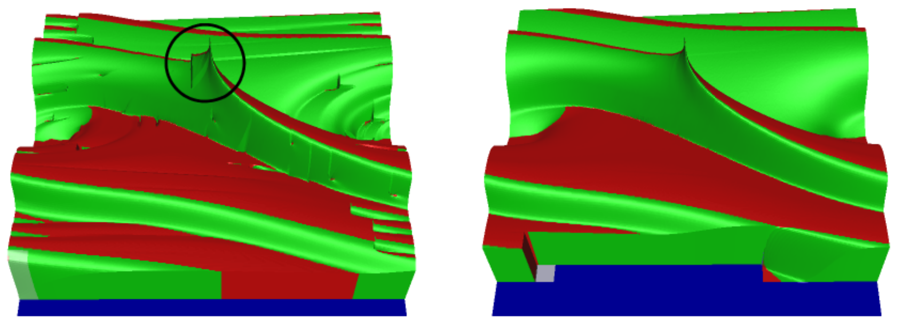


Figure 1: Images of partially converged value functions for the SAP problem (described in Section 3). The left function was generated with *H1*, and the right function was generated with *H2*. For both images, the *x* axis represents position, the *y* axis represents velocity, and the *z* axis represents the value of the state. Notice the “stair step” in the left image near the primary reward (the peak in the middle of the space). This will eventually need to be corrected, and the change will propagate throughout the entire problem, resulting in extra computation. In addition, notice the many imperfections; each of them will eventually need to be corrected. The right function is much cleaner, because *H2* tends to drive regions of the problem to convergence before moving on. Green (light gray) and red (medium gray) are different controls; a dark blue color (dark gray) indicates that a state has never been processed. Both images are frames from a video which is available in the on-line appendix.

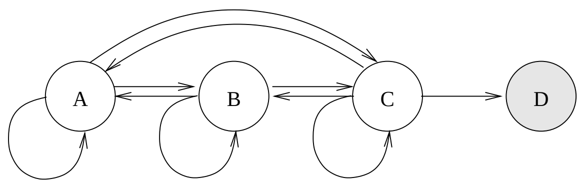


Figure 2: An example problem for which the *H2* priority metric yields a highly suboptimal backup order, but for which normal round-robin updating yields an almost optimal backup order. The *H1* metric, used with partitions, also generates a suboptimal backup order. State *D* is an absorbing reward state. Only one action is available at each state. Transitions to other states all have equal probability.

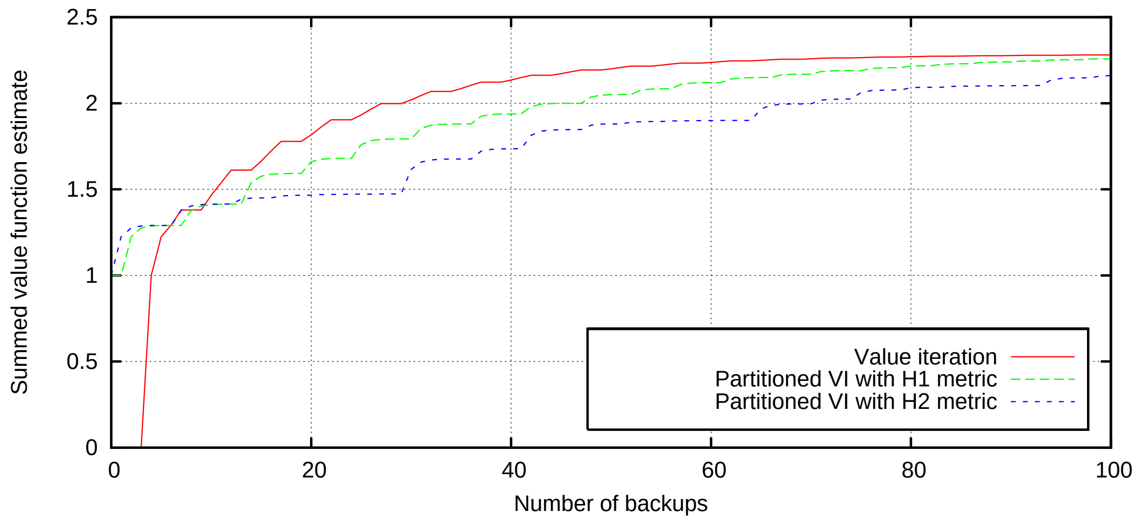


Figure 3: Performance of two GPS variants on the problem in Figure 2, using one state per partition and a value iteration subsolver (see Figure 6 for an explanation of the different algorithms). Shown is $\sum_{s \in S} V(s)$ versus the number of backups. Reaching a higher sum in fewer backups is better.

states in a round-robin fashion, which is nearly optimal (it is not fully optimal because it repeatedly backs up state D, even though it does not need to do so).

H2 performs best in a *hybrid* setting, which we illustrate using Figure 4. Each cloud represents a cluster of highly interdependent states (perhaps even strongly connected components); clusters are weakly connected to each other. The values of states within each cluster should be converged before moving on to process the next cluster, but within each cluster, standard value iteration should be employed. By themselves, each metric performs poorly: value iteration performs useless backups by working on clusters two and three before information has propagated back to them; *H1* has the tendency to prematurely move on to the second and third clusters before the first cluster has converged, and *H2* correctly prioritizes clusters, but functions poorly within each cluster.

A good algorithm should select a cluster and work on it until convergence, then move on to the next cluster. This is exactly the way that GPS functions, except that it also employs partitions (as discussed in the next section). Either *H1* or *H2* serves as a guide between partitions, but within each partition, round-robin updating occurs.

2.3 Partitioning

Although prioritization reduces the total number of backups performed, the overhead of managing the priority queue can be prohibitively high. Each state s (and each $s' \in SDS(s)$) must be extracted, reprioritized, and reinserted into the queue, resulting in several $O(\log n)$ operations per backup (where n is the number of states). Figure 5 illustrates this overhead empirically: on one problem, although one variant of GPS with one state per partition performs far fewer backups than normal value iteration, it takes far longer to solve the problem.

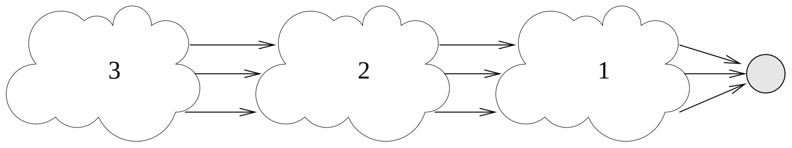


Figure 4: An example illustrating when hybrid metrics are close to optimal. Clouds represent clusters of highly interdependent states; arrows represent some of the arcs in the transition matrix. Variants of GPS perform very well on problems of this sort if partitions correspond to clusters.

Two observations direct our solution: first, we can accept some backups that do not occur in strict priority order. Second, any single state (typically) depends on multiple other states; it would be ideal to postpone the reprioritization of a state until multiple dependencies have been backed up. A good principle is to group states together into sets, and to work on the sets, instead of individual states. This accomplishes both goals: it efficiently approximates the backup order induced by the priority metric, and it tends to ensure that multiple dependencies are resolved before moving on. The specific partitioning used therefore navigates the trade-off between useless backups (there might be states in the partition that did not need to be processed) and priority queue overhead (it is faster to update them anyway, because it takes too long to determine which ones are useless). Additionally, with partitioning in place, a prioritized version of policy iteration may be created, as described in the next section.

Our partitioned, prioritized algorithm selects a high-priority partition p, solves the states in the partition, and then reprioritizes any partition which depends upon anything in p. Thus, running GPS with a single partition containing all states is equivalent to normal value/policy iteration, while running it with a single state per partition generates backups in strict priority order (this is actually not always the case, as explained in the next section).

We use the following definitions to describe a partitioned, prioritized algorithm. Let each $p \in P$ be a partition, which is a set of states. We define the *state dependents of a partition* to be the set of all states whose value depends on some state in the partition p:

$$SDP(p) = \bigcup_{s \in p} SDS(s).$$

Let P_s be a function mapping states to their partitions. We define the *partition dependents of a state* to be the set of partitions which contain a state whose value depends on s:

$$PDS(s) = \bigcup_{s' \in SDS(s)} P_s(s').$$

We define the *partition dependents of a partition* to be the set of all partitions that contain at least one state that depends on the value of at least one state in p:

$$PDP(p) = \bigcup_{s \in p} PDS(s).$$

We define the priority between two partitions as

$$HPP_t(p, p') = \max_{s \in p \cap SDP(p')} H_t(s).$$

Note that in general, $HPP_t(p,p') \neq HPP_t(p',p)$. We define the priority of a partition as

$$HP_t(p) = \max_{p'} HPP_t(p,p').$$

Algorithm 4 shows a general partitioned, prioritized solver.

As shown in Figure 5, adding more states to the partitions dramatically improves performance. Both variants of GPS perform fewer backups to the value function than normal value iteration, but because the variant with 200 states per partition eliminates priority queue overhead, the time needed for it to reach a solution drops by two orders of magnitude. Counter-intuitively, it even performed fewer backups than the variant which used only one state per partition. This indicates that the intra-partition backup order is better than the order imposed by the priority metric.

Figure 3 demonstrates one situation in which using priority metrics with partitioning can be suboptimal, and that it is not always desirable to solve partitions exactly. In this example, the best solver is normal value iteration, which can be thought of as an algorithm which solves each partition inexactly. That is, it performs exactly one backup within each partition, and then moves on to the next partition, in a round-robin fashion. Both of the other algorithms attempt to solve each partition to within ε of optimal; because they back up the states in the partition multiple times, the value function of the states slowly spirals upwards. Once they are within ε of their optimal value, the solvers select another partition. This example suggests that that partitioned solvers will be suboptimal whenever partitions are highly intra-dependent *and* highly inter-dependent. Of course, the example also illustrates the fact that the prioritization metrics (either at the state level, or the partition level) are only an approximation of the optimal backup ordering.

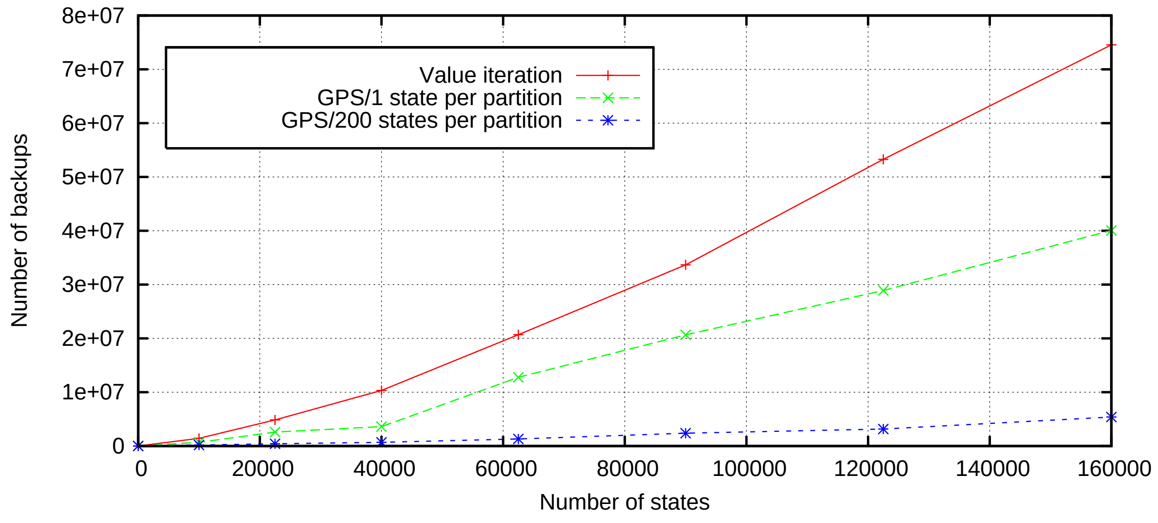
There are many possible ways to generate good partitions. If states have geometrical information associated with them, it can be used to generate partitions containing states that are near to each other. If not, more general *k*-way graph partitioning algorithms, such as multilevel coarsening or recursive spectral bisection, may be used (see Alpert, 1996, for an excellent dissertation on the subject). *k*-way graph partitioners generate partitions that minimize the cumulative weight of cross-partition edges, which is desirable because it tends to ensure that highly interdependent states are in the same partition. Automatic, variable resolution partitioners could be used, such as those described by Moore and Atkeson (1995) or Munos and Moore (2002). It may also be possible that techniques from state aggregation literature may help. Dean and Givan (1997) describe a “stable cluster” creation technique, for instance, with properties that are desirable for a partition.

Combining partitioning with prioritization is useful for other reasons, which are not explored in this work. Partitioning is a good domain decomposition, which enables an efficient, naturally parallelizable algorithm (Wingate and Seppi, 2004b). In addition, the *H2* metric (combined with partitioning) exhibits excellent disk-based-cache behavior, which is desirable when attempting to solve problems so large they cannot fit into available RAM (Wingate and Seppi, 2004a).

Our experiments tested both geometrically generated partitions as well as partitions generated by the METIS package (see, for example, Karypis and Kumar, 1998). Section 4 presents results exploring different edge cut criteria.

2.4 Solving a Partition

Once a partition p has been selected, we must compute the optimal policy and the corresponding value function of the states in p, while treating the values of the rest of the states in the problem as constants. Any MDP solver, such as value iteration, policy iteration, or linear programming, could



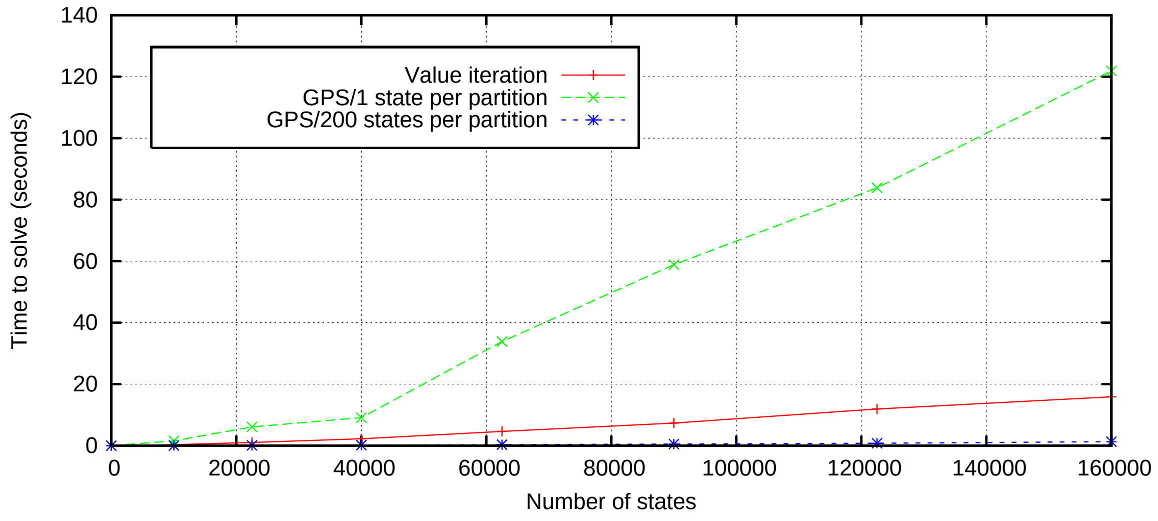


Figure 5: Performance on the MCAR problem (see Section 3) as a function of the number of states used to discretize the problem. Lower times are better. Using one state per partition, one variant of GPS performs half as many backups as normal value iteration, but it takes far longer to complete. Priority queue overhead accounts for most of this discrepancy. Using partitions greatly improves performance: in the bottom graph, another GPS variant (which has 200 states per partition) requires only 1.2 seconds to solve the problem, and is barely visible above the horizontal axis.

Algorithm 4 Prioritized, Partitioned Value and Policy Iteration

Initialization

```
 1: for all $s \in S$ do
 2:     $V_0(s) \leftarrow 0$
 3:     $H_0(s) \leftarrow \max_{a \in A} R(s,a)$
 4: end for
 5: for all $p \in P$ do
 6:     $HP_0(p) \leftarrow \max_{s \in p, a \in A} R(s,a)$
 7:     for all $p' \in P$ do
 8:         $HPP_0(p,p') \leftarrow 0$
 9:     end for
10: end for
11: $p \leftarrow \arg\max_{\xi \in P} HP_0(\xi)$
12: $t \leftarrow 1$
```

Main loop

```
 1: repeat
 2:     // compute the optimal policy and value function of states in $p$
 3:     solve($p$)
 4:
 5:     // update partition priority for all dependent partitions
 6:     for all $p' \in PDP(p)$ do
 7:         $HPP_t(p',p) \leftarrow 0$
 8:         $h_{max} \leftarrow 0$
 9:         for all $s' \in p' \cap SDP(p)$ do
10:             // recompute $H_t(s')$
11:             $h_{max} \leftarrow \max(h_{max}, H_t(s'))$
12:         end for
13:         $HPP_t(p',p) \leftarrow h_{max}$
14:         $HP_t(p') \leftarrow \max_{\xi} HPP_t(p',\xi)$
15:     end for
16:     $p \leftarrow \arg\max_{\xi \in P} HP_t(\xi)$
17:     $t \leftarrow t+1$
18: until convergence
```

be used. It is even possible to use a partitioned, prioritized solver (indeed, such an idea could be extended to more than two levels), although hierarchical partitioning is not explored in this work. We require that the value of the states in the partition be computed accurately (that is, to within ε of exact) because the value function may be needed in the context of a larger algorithm, but more importantly because both priority metrics depend upon accurate value function estimates. In other words, if we use policy iteration as a solver, and use an iterative policy evaluation method, we cannot stop when just the *policy* has converged; rather, we must wait until the value function has converged as well.

It is clear how to use value iteration to solve p, but it is less clear how to use policy iteration. Policy improvement is easy, but how do we evaluate the value of the states in p? Recall that policy evaluation is the process of computing the value function for a single policy π. This eliminates the max operator in Eq. 1, simplifying it to $V(s) = R(s,\pi(s)) + \gamma \sum_{s' \in S} Pr(s'|s,a)V(s')$. This is a linear system of $|S|$ equations in $|S|$ unknowns; in matrix-vector notation, it is equivalently expressed as $v = r_\pi + \gamma P_\pi v$, where P is the transition probability matrix of policy π and r_π is the reward under policy π. Note that this has the same form as the more general problem $x = Ax + b$, which is equivalent to $(I - A)x = b$.

The key observation is the fact that if the value and policy of states outside the partition are held constant, their values may be temporarily "folded" into the right-hand side vector, and a new sub-problem created. As an example, consider the general problem $Ax = b$, where A is a 2x2 matrix. Suppose that we wish to solve only for variable 0 while holding variable 1 constant. We may expand this system (indexing with array notation) as

$$\begin{aligned} A[0,0]x[0] + A[0,1]x[1] &= b[0] \\ A[1,0]x[0] + A[1,1]x[1] &= b[1] \end{aligned}$$

and, since $x[1]$ is constant, rewrite it as

$$\begin{aligned} A[0,0]x[0] &= b[0] - A[0,1]x[1] &&= b'[0] \\ A[1,0]x[0] &= b[1] - A[1,1]x[1] &&= b'[1], \end{aligned}$$

where b' is the new vector that results from the folding operation. This yields a set of two equations with one unknown. Either may be used to solve for variable 0.

More generally, assume that we are given an $n \times n$ matrix A and a partition p which contains a set of variable indices that we wish to solve for. The equation for variable i is

$$\begin{aligned} \sum_{j=0}^{n} A[i,j]x[j] &= b[i] \\ \sum_{j \in p} A[i,j]x[j] + \sum_{j \notin p} A[i,j]x[j] &= b[i] \\ \sum_{j \in p} A[i,j]x[j] &= b[i] - \sum_{j \notin p} A[i,j]x[j] \\ \sum_{j \in p} A[i,j]x[j] &= b'[i]. \end{aligned}$$

This yields n equations in $|p|$ unknowns, which is an over-constrained system. We wish to select a subset of the equations with which to work; we adopt the convention that we will use the equations corresponding to the variable indices in the partition. Specifically, we may define a diagonal *selector matrix* as

$$K_p[i,i] = \begin{cases} 1 & i \in p \\ 0 & \text{otherwise.} \end{cases}$$

Then, to select a subset of equations from A, x and b, we will let $A' = K_p A K_p$, $x' = K_p x$, and $b' = K_p b$. Note that A' is still an $n \times n$ matrix, but with many empty rows and columns. If row i in A' is empty, column i will also be empty, and the corresponding entries in both both x' and b' will also be zero. The entire system may be compacted by eliminating such zero entries and mapping variable indices from the original system to new variable indices in a compacted system, which may then be passed to an arbitrary subsolver. Selection and compaction can be accomplished simultaneously through the use of prolongation and restriction operators (Saad, 1996), but we have adopted the current approach for simplicity of explanation. We note that GPS with this fold/extract method can be considered a prioritized Multiplicative Schwarz Procedure (Schwarz, 1890; Saad, 1996).

Once the problem $A'x' = b'$ is constructed, any number of direct or indirect linear system solvers can be used. Exact policy evaluation usually involves inverting a matrix, which is typically a $O(n^3)$ operation. Since we only need to solve $A'x' = b'$ to within ε, and because of our interest in performance, we we can opt instead for approximate policy evaluation. Fortunately, this does not compromise the accuracy of the final solution: Bertsekas and Tsitsiklis (1996) establish the fundamental soundness of approximate policy evaluation, and provides bounds on the optimality of the final policy based on the evaluation error.

In this work, we use Richardson iteration (which is equivalent to value iteration) and GMRES as policy evaluators. GMRES is considered by the numerical analysis community to be the standard iterative method for large, sparse, asymmetric matrices. Saad (1996) presents an excellent discussion of iterative methods and Barrett et al. (1994) present template-based implementations of common iterative methods.

2.5 Variable Reordering

As noted previously, it is possible to use standard value iteration to solve a partition. We would like to optimize this step in the algorithm, but we have already seen that we cannot use a priority queue – the overhead is excessive, which is why we used partitions in the first place. Instead of a good *dynamic* ordering, we therefore opt for a good *static* ordering. Specifically, we wish to reorder the states in the partition such that for each sweep, they are backed up in an approximately optimal order. This ordering of states is computed once, during initialization. Note that variable reordering is only effective when Gauss-Seidel iterations are used; it does not affect Jacobi-style iterations, and may not affect other methods of solving linear systems. In particular, variable reordering is only applicable to prioritized policy iteration if partitions are evaluated using a Gauss-Seidel iterative method.

We wish to back up a state s only when all of the states s depends on have been backed up. This suggests the use of a topological sort on the states, and indeed, this yields an optimal ordering of states if the graph is acyclic. Since a topological sort is not defined for cyclic graphs, a more general possibility is to reorder the states in the matrix to make the matrix “near triangular.” Specifically, we wish to permute the iteration matrix to minimize the maximum row sum in the upper triangle.

Algorithm 5 Variable reordering

```
1:  // Initialization: dc is an array representing the in-degree of each state
2:  dc ← 0
3:  for all $s \in p$ do
4:      for all $a \in A$ do
5:          for all $s' \in p$ do if $((Pr(s'|s,a) \neq 0)$ then increment(dc[$s'$])
6:      end for
7:  end for
8:
9:  // Main loop: finalorder is an array representing the final state ordering
10: for $i = 0..|p|-1$ do
11:     let $s$ be the index of the smallest non-negative value in dc
12:     dc[$s$] ← $-1$
13:     finalorder[$|p|-1-i$] ← $s$
14:     for all $a \in A$ do
15:         for all $s' \in p$ do if $((Pr(s'|s,a) \neq 0)$ then decrement(dc[$s'$])
16:     end for
17: end for
```

The reason for this follows. We begin with the generic problem $x = Ax + b$, where A is a matrix and x, b are column vectors. This suggests an iterative method of the form

$$x_{t+1} = Ax_t + b \tag{2}$$

which corresponds to a Jacobi-style iterative method. Now, let $A = (L + D + U)$, where L is the lower-triangular part of A, D is the diagonal part, and U is the upper-triangular part. Using Gauss-Seidel iterations, Eq. 2 can be expressed as $x_{t+1} = Lx_{t+1} + (U + D)x_t + b$ (recall that Gauss-Seidel iterations use state values as soon as they are available). This can be rearranged to yield

$$x_{t+1} = (I - L)^{-1}(U + D)x_t + (I + L)^{-1}b. \tag{3}$$

This is the most basic regular splitting of the matrix A; see Puterman (1994) for a more comprehensive treatment of regular splittings and of the specifics of Gauss-Seidel vs. Jacobi iterations.

It is well known that both asynchronous and synchronous relaxations of the form $x_{t+1} = f(x_t)$ converge when f satisfies the definitions of a contraction mapping (Bertsekas and Tsitsiklis, 1989). Proofs of contraction have been constructed for several important cases, including linear relaxations. Iterations in the form of Eq. 2 are guaranteed to converge if the spectral radius $\rho(A) < 1$.

The convergence of Eq. 3 is therefore governed by $\rho((I - L)^{-1}(U + D))$, which is a difficult expression to simplify. However, it is also well known that $\rho(AB) < \|A\|\|B\|$ for any matrix norm; since the infinity-norm of a matrix corresponds to the largest row-sum in the matrix, convergence will be governed by $\|(I - L)^{-1}\|_\infty$ and $\|(U + D)\|_\infty$. (Tangentially, we note that, because $(I - L)$ is lower-triangular, its inverse is also lower-triangular, and the eigenvalues therefore correspond to the diagonal entries. It is easy to show that the diagonals will still be ones after inverting, meaning that $\rho((I - L)^{-1})$ is simply 1). We therefore seek some permutation of A which will minimize either $\|(I - L)^{-1}\|_\infty$ (which is difficult to do because of the inverse), or $\|(U + D)\|_\infty$. Note that if such a minimizing permutation is obtained, and the graph defined by the matrix is acyclic, $(U+D)$ is empty, and the system converges in one pass.

As Knuth (1993) states, this problem is NP-complete, because it includes as a very special case the "Feedback Arc Set" problem, which "was shown to be NP-complete in Karp's classic paper (Miller and Thatcher, 1972)". However, heuristics have been proposed: Knuth (1993) experiments with a downhill method to find a permutation that is locally optimal (in the sense that moving any individual row increases the row-sum), but his method is computationally expensive. Modified topological sorts (in which any edge that completes a cycle is deleted) have also been proposed which are efficient, and which form the basis of the strategy we follow.

The foregoing analysis has been in terms of a generic matrix A. However, the transition matrix of a discrete MDP depends on the operative policy at any given time. What matrix should be used to compute a new variable ordering? We use an aggregate matrix which incorporates all of the transitions possible under any policy, all of which are weighted equally. This represents an implicit assumption that we are optimizing for an expected case where all policies are equally likely, but if additional information about the likelihood of different policies is available, it could be leveraged during this phase.

The final variable reordering algorithm is a modified topological sort, and is shown in Algorithm 5. The algorithm operates on a partition p, and generates the array `finalorder`, which lists the order in which states should be backed up.

2.6 Convergence, Stopping Criteria and Complexity

Convergence of GPS with a value-iteration subsolver is established by noting that it is an asynchronous variant of traditional value iteration. Convergence is guaranteed for such algorithms provided that every state is backed up infinitely often (Bertsekas, 1982, 1983; Gullapalli and Barto, 1994). Practically, this can be guaranteed as long as no state is starved of backups. In our algorithm, states will be backed up until they have converged, at which point a new set of states will be selected. If a state has some $B_t(s) > \varepsilon$, it will eventually be backed up; if no such state exists, the problem is solved. Convergence of approximate policy iteration and asynchronous policy iteration has been established by Bertsekas and Tsitsiklis (1996); once again, the stipulation is that each state must be visited infinitely often, and once again, it is clear that our algorithm does not starve any state that has $B_t(s) > \varepsilon$.

Here, we also note that if a state s never has $B_t(s) > \varepsilon$, it never needs to be backed up. Of course, there is no performance gain in detecting that a single state never needs to be backed up, because it takes just as long to compute $B_t(s)$ as it does to back the state up – but partitions change that. Because the states in a partition p are "blocked off" from the rest of the problem, detection of the fact that they may not need to be backed up can be highly efficient: only B_t of the cross-partition dependencies must be examined. If they never move above ε, nothing in p needs to be backed up (assuming the partition is internally solved). Thus, the "infinite updates" stipulation of most asynchronous convergence proofs represent sufficient conditions, but not necessary conditions.

Stopping criteria are easily established. The largest difference between a value function estimate and the optimal value function can be characterized in terms of the Bellman error magnitude. This has already been accomplished by Williams and Baird (1993); similar results can be easily derived by using equation 6.3.7 of Puterman's book (Puterman, 1994):

$$\|V_t - V^*\| \leq M_{t-1}/(1-\gamma). \tag{4}$$

Name	Description
PI-Rich	Policy iteration with Richardson policy evaluator
PI-GMRES	Policy iteration with GMRES policy evaluator
PPI-Rich-H1	Partitioned PI, prioritized w/*H1*, using Richardson evaluator
PPI-Rich-H2	Same, but with *H2* priority metric
PPI-GMRES-H1	Partitioned PI, prioritized w/*H1*, using GMRES evaluator
PPI-GMRES-H2	Same, but with *H2* priority metric
VI	Standard value iteration
VI-VRE	Value iteration with partitions, NO priority, and variable reordering
PVI-H1	Partitioned value iteration, prioritized with *H1*
PVI-H1-VRE	Same, plus variable reordering
PVI-H2	Partitioned value iteration, prioritized with *H2*
PVI-H2-VRE	Same, plus variable reordering

Figure 6: Algorithms tested.

The maximum difference provides a natural stopping criterion. The algorithm can stop when $M_t < \varepsilon(1-\gamma)$, and will be guaranteed to have an ε-optimal policy. A more common bound (for example, Puterman, 1994) is that if $\|V_{t+1} - V_t\| < \varepsilon(1-\gamma)/2\gamma$, then $\|V_{t+1} - V^*\| < \varepsilon/2$. The slight difference in the two equations can be accounted for by noting that we previously stipulated that all rewards be positive (which allows us to provide a tighter bound by avoiding absolute values), and because of a minor difference in time subscripting.

The space complexity of GPS is quite good. The largest overhead comes from the need to store a partial inverse model, but this is always a subset of the whole problem. Additional memory is needed for the priority queue ($O(|P|)$), for the state-to-partition mapping ($O(|S|)$), and the partition-to-state maps ($O(|P| + |S|)$).

3. Experimental Setup

We tested twelve different combinations of the basic enhancements we have discussed, which are shown in Figure 6. Most are self-explanatory, except the VI-VRE algorithm. VI-VRE was designed to isolate the impact of variable reordering, so to accomplish this, the problem was partitioned, and variables were reordered within each partition. Then, for each sweep, the algorithm iterated over all partitions, and backed up each state within each partition in the prescribed order.

All value iteration and Richardson iteration solvers used Gauss-Seidel backups. When GMRES was needed, we used the AZTEC package (Tuminaro et al., 1999), which is an implementation of several iterative methods developed by Sandia National Laboratories.

Two types of partitioning were tested. Most experiments used graph-based partitions generated by the METIS package (Karypis and Kumar, 1998), but some used geometrical information. This was done by laying down a regular grid of partitions over the state space. Various grids were tested; the best was a simple grid with square cells.

The algorithms were tested against several problems of differing complexity. Success was measured by the amount of time taken, by the number of backups performed, and by how accurate the resulting value function was. We tested two types of problems. The first were deterministic mini-

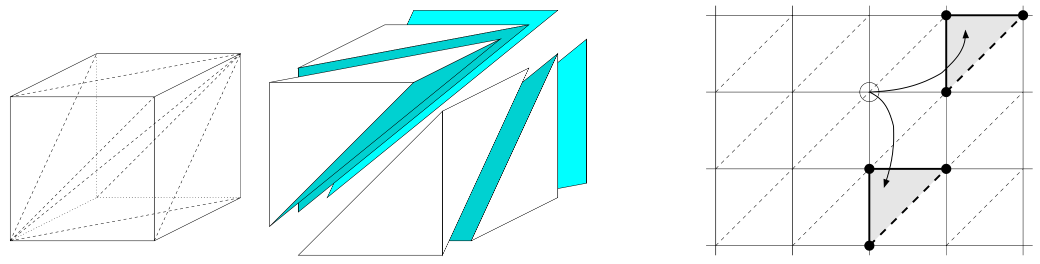


Figure 7: On the left, the Kuhn triangulation of a (3d) cube. A d-dimensional hypercube is tessellated (implicitly) into $d!$ simplices. On the right, control of each (s,a) pair is tracked until the resulting state s' enters a new hypercube. Barycentric coordinates relative to the enclosing simplex are computed, and are used to represent probabilistic transitions to vertices.

mum time optimal control problems, which are continuous time, and involve continuous action and state dimensions (these were discretized as described below). These control problems were selected because the number of states used in the discretization could be varied at will, while maintaining a constant expected degree, thus allowing us to generate families of highly related MDPs. The second set of problems were versions of the SysAdmin problem described by Guestrin et al. (2003). These are inherently discrete, stochastic problems with very dense transition matrices.

We will briefly describe the process used to discretize the control problems, but we note here that this process is tangential to the research focus of this paper. There are many other methods which could have been used to discretize the problems; naturally, this particular method introduces a bias with respect to the original problem, but since the solution engine simply expects a discrete MDP, the details of where it came from are somewhat irrelevant.

To discretize the space, we use the general numerical stochastic control techniques described by Kushner and Dupuis (2001); our specific implementation closely followed that of Munos and Moore (2002) (except that no *variable* discretization is used). Instead, the space is discretized once in the initialization phase. We refer the reader to their work for a complete description of the technique with comprehensive citations on component elements.

Figure 7 illustrates the discretization process. The state space is divided into hypercubes by regularly dividing each dimension, and a Kuhn triangulation is implemented (implicitly) inside of each hypercube. The use of Kuhn triangles is particularly appropriate because once discretized, each state depends upon exactly $d+1$ other states. In addition, the combination of hypercubes and Kuhn triangles has excellent space and time performance characteristics, which greatly accelerated the experimental cycle. The hypercubes completely tessellate the space, and the Kuhn triangles completely tessellate each hypercube. The vertices defining the hypercube grid are used as the states in the MDP. The transition matrix is computed by iterating over each vertex s. For each available action a, the system dynamics are integrated using Runge-Kutta and tracked until the resulting state s' enters a new hypercube. The barycentric coordinates of s' with respect to the enclosing simplex are then computed. The state s can then be said to transition non-deterministically to a vertex in the enclosing simplex with probability equal to the related barycentric coordinate (since barycentric coordinates always sum to one). As Munos and Moore (2002) state, "doing this interpolation is thus mathematically equivalent to probabilistically jumping to a vertex: we approximate a *deterministic* continuous process by a *stochastic* discrete one" (emphasis in original).

Approximation of the value function is performed by computing exact values at each of the vertices, and interpolating the value across the interior of each hypercube. Interpolation is linear within each simplex. Since these problems are continuous time, a slightly different form of the value function equation was used:

$$V_t(s,a) = \int_0^\tau \gamma^t R(s(t),a)dt + \gamma^\tau \sum_{s' \in S} Pr(s'|s(\tau),a) \max_{a' \in A} V_{t-1}(s',a')$$

where τ is the amount of time it took for s' to enter the new hypercube (or exit the state space), with the convention that $\tau = \infty$ if s' never exited the original hypercube.

All results were obtained on a dual Opteron 246 with 8G of RAM. For partitioned solvers, there were always about 200 states in each partition, unless otherwise indicated. Results are deterministic, so wallclock tests were only run enough times to ensure accuracy.

3.1 Problem Details

Mountain car (MCAR) is a two-dimensional minimum-time optimal control problem. A small car must rock back and forth until it gains enough momentum to carry itself up to the top of the hill (see Figure 8). In order to receive any reward, the car must exit the state space on the right-hand side (positive position), with a velocity close to zero. In order to make results more comparable to the other problems studied, the reward function was modified from the traditional gradient reward to be a single-point reward: the agent received a reward only upon exiting the state space with a velocity (within a threshold) of zero. This did not substantially change the shape of the resulting value function. The state space is defined by position ($x \in [-1,1]$) and velocity ($\dot{x} \in [-4,4]$).

The single-arm pendulum (SAP) is also a two dimensional minimum time optimal control problem. The agent has two actions available (bang-bang control is sufficient for this problem) representing positive and negative torques applied to rotating pendulum, which the agent must learn to swing up and balance. Similar to MCAR, the agent cannot move the pendulum from the bottom to the top directly, but must learn to rock it back and forth. Rewards are zero everywhere but in the balanced region. The state space is defined by the angle of the link (θ_1) and the angular velocity of the link ($\dot{\theta}_1 \in [-15,15]$). The two actions are ± 10 Newton.

The double-arm pendulum (DAP) is a four dimensional minimum time optimal control problem (see Figure 8). It is similar to SAP, except that there are two links. It is the second link which the agent must balance vertically, but it is a free-swinging link. This variant of DAP is different from the easier Acrobot problem, where force is applied at the junction between the two links (Sutton, 1996), and from a horizontal double-arm pendulum (where the main link rotates in the horizontal plane, and the secondary link rotates vertically with respect to the main link). This version of DAP is the complete swing-up-and-balance problem; other variants only treat the balancing aspect. Rewards are zero everywhere but in the balanced region. The state space is defined by the two link angles (θ_1,θ_2) and their angular velocities ($\dot{\theta}_1 \in [-10,10]$ radians/s, $\dot{\theta}_2 \in [-15,15]$ radians/s). Values outside these ranges are almost impossible to achieve, and are not necessary for the optimal policy. The two actions are ± 10 Newton.

SysAdmin is an inherently discrete, highly non-deterministic problem with a fairly large number of actions. A systems administrator must maintain a network of n connected computers. At

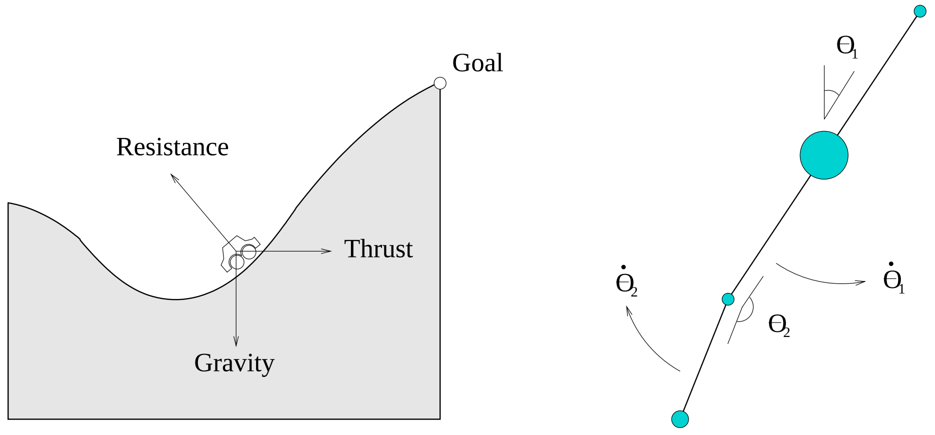


Figure 8: The left figure shows the MCAR problem (figure adapted from Munos and Moore, 2002). The car must rock itself back and forth to generate enough momentum to exit the state space. The state space is described by the position and velocity of the car. The right figure shows the DAP problem. The agent must swing the secondary link into the vertical position and keep it there. The state space is described by four variables: θ_1, θ_2, $\dot{\theta}_1$ and $\dot{\theta}_2$. The same dynamics are used for the SAP problem.

each timestep, the administrator is paid proportionally to the number of machines that are on-line; the goal of the problem is to maximize the amount of money made. The state space is a binary vector, with each bit representing whether or not a machine is working. Machines fail with a fixed probability, with a failed machine increasing the probability of failure of any machine connected to it. The administrator has $n+1$ actions available: action i corresponds to rebooting machine i, and action $n+1$ means that nothing is rebooted. A rebooted machine is guaranteed to be working on the next timestep. We used a network of 10 machines, connected in a ring topology; other network sizes were tested, with substantially similar results. See the paper by Guestrin et al. (2003) for specifics on generating the transition probabilities.

4. Results

Our experiments generated many positive results. First, the enhancements we have discussed accelerated the solution to many problems by as much as two orders of magnitude, while maintaining a high degree of accuracy. The *H2* metric usually outperformed the *H1* metric, and variable reordering almost always helped. On the MCAR problem, many of the algorithms avoided many useless backups, to the point of never processing several states. In addition, GPS exhibited fairly good space complexity. However, there were some negative results. First, all of the enhancements (with the exception of variable reordering) exhibited very poor behavior on the SysAdmin problem. Second, we observed that it was difficult to tune the parameters of the algorithm. Finally, we observed that none of our enhancements were optimal.

4.1 Positive Results

Figures 9, 10 and 11 show that the enhancements constituting GPS clearly accelerate normal value and policy iteration for the SAP problem; substantially similar results were obtained for the MCAR problem. The gains were even better for the DAP problem, as shown in Figures 12 and 13, although the enhancements that worked the best on SAP and MCAR were different than the enhancements that worked the best on DAP. To solve a 160,000 state version of SAP, for example, VI required about 23 seconds, but PVI-H2-VRE required about 1.24 seconds. For a 6.25 million state version of DAP, VI required 110.64 seconds, but PVI-H1 required only 8.28 seconds. Similar performance gains were observed by enhancing policy iteration: PI-Rich and PI-GMRES solved the 6.25 million state DAP in about 250 seconds each; the partitioned, prioritized variants ranged in times between 12.53 and 22.77 seconds. It is also interesting to note that GPS solved both MCAR and SAP in about the same amount of time for any given discretization, even though they represent very different problems (in the sense that value information flows through them very differently).

The *H2* metric usually exhibited better behavior than the *H1* metric; the only exception was the DAP problem (this is discussed in the next section). The results also demonstrate that, while solving the control problems, the relative advantages of the enhancements were consistent as the problem size changed. This could be due to the fact that although the discretization level is changing, other fundamental properties of the MDP are not: the expected degree of any vertex is constant, the number of actions is very small, and there is only one reward in the system.

Variable reordering was almost always very effective on these control problems. In most experiments, reordering reduced time and backups by at least a factor of 2, and even in the cases when it did not help, we never saw a situation where it hurt in any statistically significant way. VRE thus appears to be a good enhancement, considering the low overhead and ease of implementation.

An additional advantage of prioritization is that some of the prioritized algorithms never processed certain states in the MCAR problem. Figure 16 demonstrates this graphically: large regions of the state space (indicated by a dark blue color) were never processed, because the agent can never reach the goal state from them. This is a significant result from a practical standpoint: no additional code or information about the problem was necessary, but a full 19% of the state space was never processed. This behavior manifested itself in the algorithm by a priority of zero that never changed.

For all policy iteration variants, the Richardson iteration subsolver often outperformed the GMRES subsolver. This is surprising, since Krylov methods are generally considered superior to their more naive counterparts. It is possible that the use of preconditioning may show GMRES in a more positive light, but such experiments are left to future research. We consider this a positive result because it indicates that using "naive" algorithms such as Modified Policy Iteration (Puterman and Shin, 1978) (which interleaves rounds of policy improvement with rounds of value iteration) may not be such a bad idea after all; the use of more sophisticated general matrix solvers may not be very fruitful because of the extremely specific type of matrix that is used in RL problems.

To validate the resulting policies from GPS, a 75,000,000 state version of DAP was run (an empirically determined minimum resolution needed for an effective control policy). This was done in the continuous domain by overlaying the discretization structure on the original space. Policies are exact at the vertices, so distance-weighted vertex voting was used to generate policies inside each simplex. A very good result is that GPS solved it (to $\varepsilon = 0.0001$) in only about four hours. The resulting control policy performed perfectly, and represented the first time we solved DAP using any algorithm. We should note that this problem was run on an SGI Origin 3000; the SGI was about one third as fast as the Opteron used in all of the other experiments, and is a shared machine which is always heavily used.

4.2 Negative Results

The most significant negative result is the poor performance of GPS on the SysAdmin problem. Figure 14 shows some representative results: the best performers are standard PI and VI, and the worst are any GPS variant (except, of course, the variants which used only variable reordering; these showed slightly improved performance). As it turns out, SysAdmin is an example of the theoretically worst-case performance for GPS when compared to standard methods, and therefore represents the other extreme of the performance spectrum.

The problem lies in state interdependence. The transition matrix of this problem is extremely dense: under any policy, there is a non-zero probability of transitioning from any state to 50% of all of the other states. This is problematic for two reasons: first, the reprioritization calculations are very expensive, and second, the problem is not amenable to a divide-and-conquer strategy. Instead, this problem is like the one shown in Figure 2, where the best solution is a round-robin backup ordering. Thus, all of the overhead associated with maintaining priorities is *pure* overhead, in the sense that solution times would have been just as good without the effort.

The negative results are therefore not all that surprising, but deserve more quantification. A theoretical worst-case scenario for GPS can be estimated by considering a fully connected MDP ($Pr(s'|s,a) \neq 0 \;\; \forall s,a,s'$). The two critical factors in the reprioritization overhead are 1) the fact that backing up a state is costly, which is true for any algorithm but which additionally implies that recomputing H_t is costly, and 2) the fact that every state is dependent upon every other state, implying that $|SDP(p)|$ will be large for all p, and that H_t will need to be recomputed for a large number of states. The very worst case occurs when just one state is included in each partition. Assume that just one backup per state s is needed to solve a partition. Then, to reprioritize dependent partitions, we must recompute the priority for every dependent state ($O(|S|)$), by recomputing $H_t(s)$, which costs $O(|S||A|)$. This incurs $O(|S|^2|A|)$ overhead per backup. Using fewer partitions helps, but even then performance is worse than that of normal VI, because the same $O(|S|^2|A|)$ reprioritization overhead is incurred whenever a partition is solved. The extra overhead is not justified when a round-robin method works just as well.

Additional experiments were conducted to discover which features of SysAdmin lead to the poor performance. These involved reducing the number of available actions, pruning low-probability transitions, testing various partition sizes, etc. GPS still performed poorly on these modified problems. The current hypothesis is that all versions of SysAdmin considered still exhibit the same basic interdependence, which means that it is difficult to partition the problem cleanly, which implies that solving one partition before moving on to the next is not the best strategy. This leads to the following general conclusion: highly interdependent states must be solved concurrently; spending too much time solving in isolation any one part of a problem which is highly interdependent on another part is wasted effort. This suggests that a better strategy for future versions of GPS may be to solve each partition inexactly, instead of to within ε, but such an algorithm is left to future research.

The MCAR and SAP problems demonstrate that reordering often helps, that Richardson is better as a policy evaluator than GMRES, and that the using the *H2* priority metric often yields better performance than using the *H1* metric. However, DAP represents an exception to all three: Figure 12 shows that both variable reordering and the *H2* metric worsened performance, and Figure 13 shows

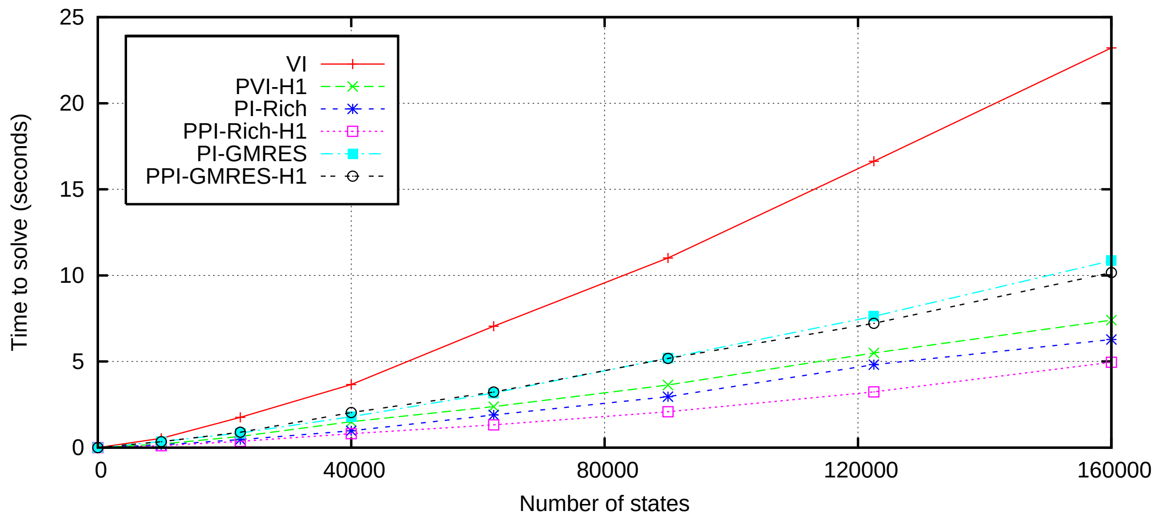


Figure 9: Impact of using the *H1* metric on the SAP problem. The *y* axis is time to solve in seconds (lower is better), and the *x* axis is the number of states used to discretize the problem. The reader should compare VI to PVI-H1, PI-Rich to PPI-Rich-H1, etc. These results show that using partitioning and prioritization with *H1* always improves performance on this problem (although it does not improve by much between PI-GMRES and PPI-GMRES-H1). The improvement is especially large for VI. Partitions were generated using METIS.

GMRES outperforming Richardson, as well as *H1* outperforming *H2*. In both sets of experiments, the largest performance gain seems to be due to the general idea of prioritization and partitioning; the specific variant of GPS used does not affect performance proportionately as much. Figure 12 also shows that, for a small number of states, VI slightly outperformed any GPS variant. Since each state always depends on $d+1$ other states, the graph diameter of these problems may be smaller.

Neither our priority metrics, our variable reordering algorithm, nor our partitioning methods are optimal. For example, Figure 15 clearly shows that graph-based partitioning is not as effective as partitions based on geometrical information. Although the minimum-cut partitioning problem is NP-complete, it is unlikely that the suboptimality in our partitioning is what causes the poor performance; a more likely explanation is that we are partitioning based on the wrong criteria. Figure 5 shows that PVI-H2-VRE/200 performs fewer backups than PVI-H2-VRE/1. With PVI-H2-VRE/1, backups are executed in strict priority order, but with PVI-H2-VRE/200, backups are executed in an order that is partly due to the priority metric, and partly due to variable reordering. This fact implies that *H2* generates a suboptimal ordering.

In order to obtain the best results, partition sizes had to be selected manually (however, it is also possible to present this as a positive result: the fact that the system was fast enough to allow us to tune this parameter is significant). Figures 14 and 15 demonstrate that partitioning is largely problem-dependent: on the SysAdmin problem, adding any partitions always hurt performance. For MCAR, adding partitions initially improved performance, but adding too many worsened it again. We do not know how to predict a good number of partitions, except to observe that using somewhere between 100 and 400 states per partition tended to yield very good results on the control problems.

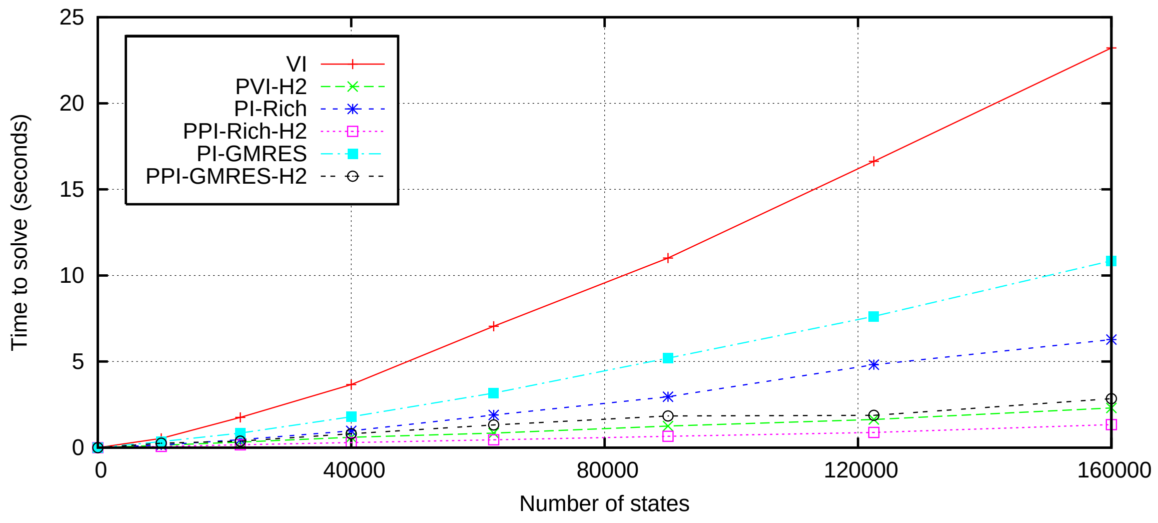


Figure 10: Impact of using the *H2* metric on the SAP problem. The results show that using partitioning and prioritization with *H2* always improves performance. In contrast to the previous figure, it substantially improves PI-GMRES. Partitions were generated using METIS.

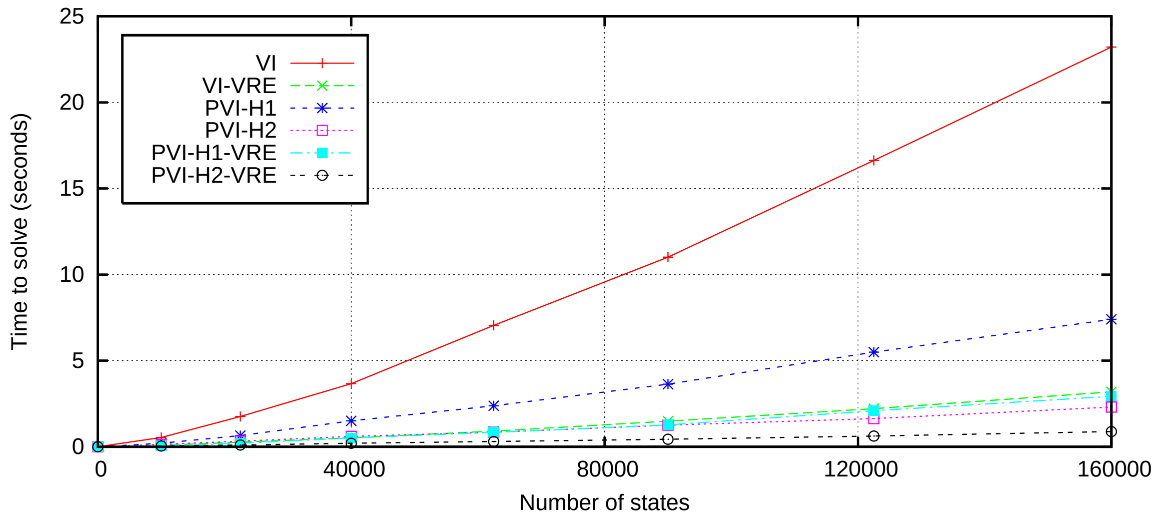


Figure 11: Impact of using VRE on SAP. VRE always improves performance on this problem, but the impact is especially dramatic on normal VI (improving performance by an order of magnitude). The lowest line shows the very best times obtained for solving SAP using any algorithm – at a size of 160,000, it takes about 0.85 seconds. Partitions were generated using METIS.

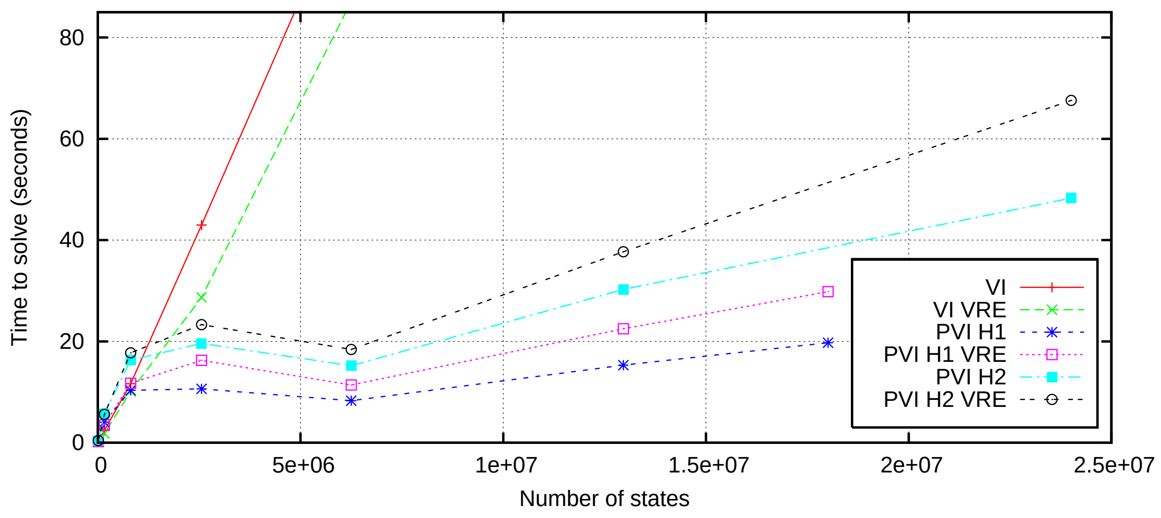


Figure 12: Results of the value iteration algorithms on the DAP problem. The results contrast somewhat with the results for MCAR and SAP: *H1* outperformed *H2*, and VRE negatively affected performance. However, any GPS variant greatly outperformed standard VI, with or without VRE. Partitions were generated using geometrical information.

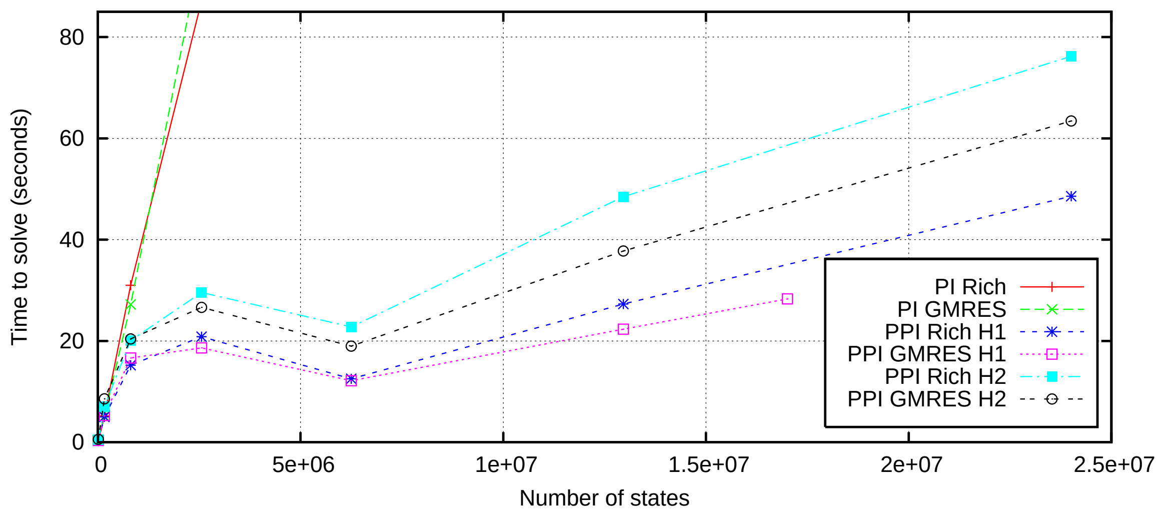


Figure 13: Results of the policy iteration algorithms on the DAP problem. Once again, the results contrast with MCAR and SAP: *H1* outperformed *H2*, and GMRES outperformed Richardson iteration. However, like the value iteration algorithms, GPS variants greatly outperformed their naive counterpart. Partitions were generated using geometrical information.

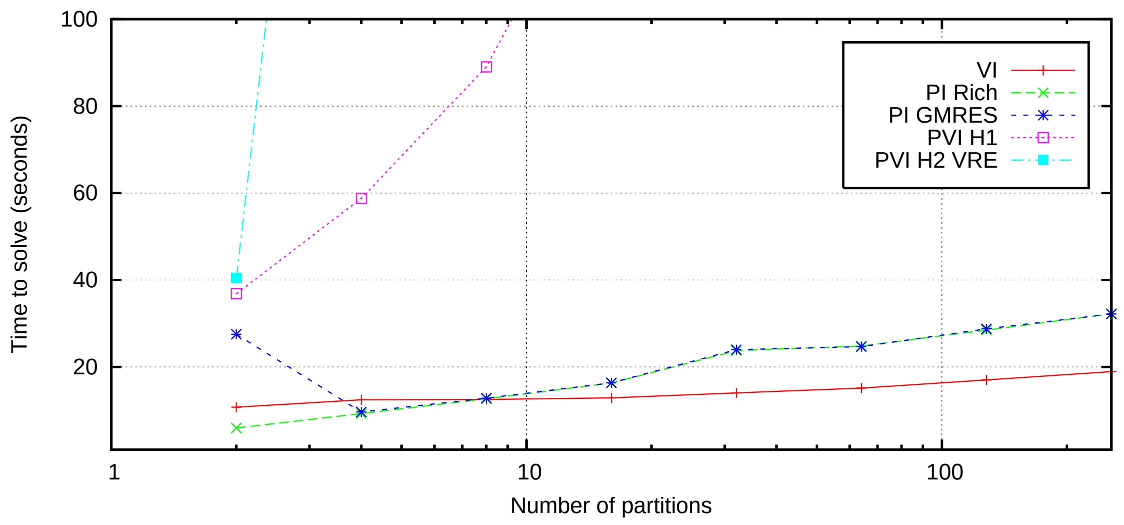


Figure 14: Performance on the SysAdmin-10 problem, as a function of the number of partitions used. Only a representative sample of all algorithms is shown. The results contrast sharply with previous results: VI and PI dramatically outperform GPS variants, and adding partitions never helps any algorithm. VRE barely affects performance. Partitions were generated using METIS.

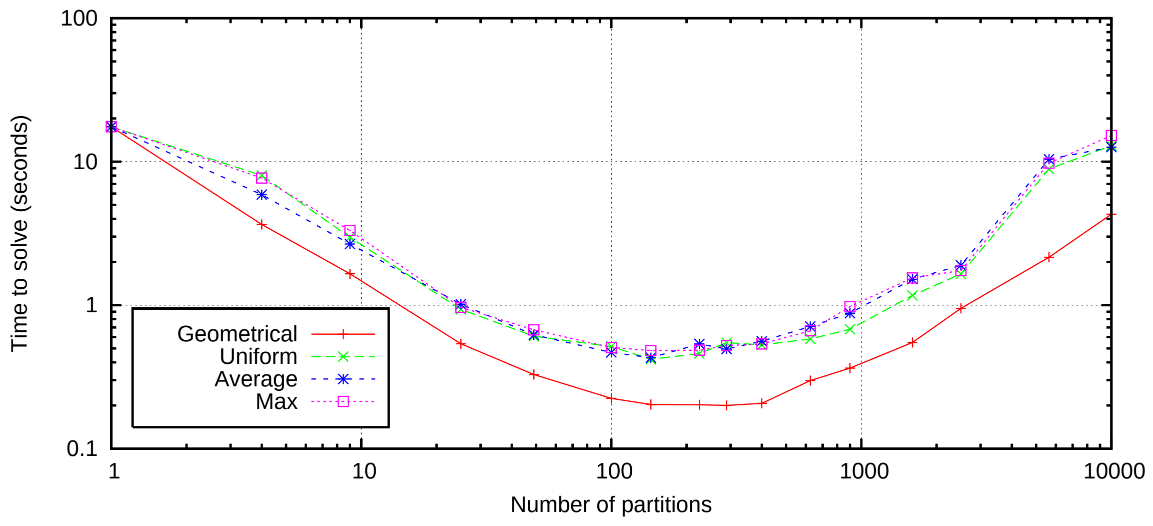


Figure 15: Results contrasting partitioning methods (note the double log scale). Shown are solution times using METIS generated partitions (with three different edge-weight criteria), compared to a geometrically generated partitioning. The edge-weight criteria are 1) maximum probability under any policy, 2) average over all policies, and 3) uniform cost. The geometric partitions perform best. Results come from the MCAR problem using a 300x300 state discretization and the PVI-H2-VRE solver.

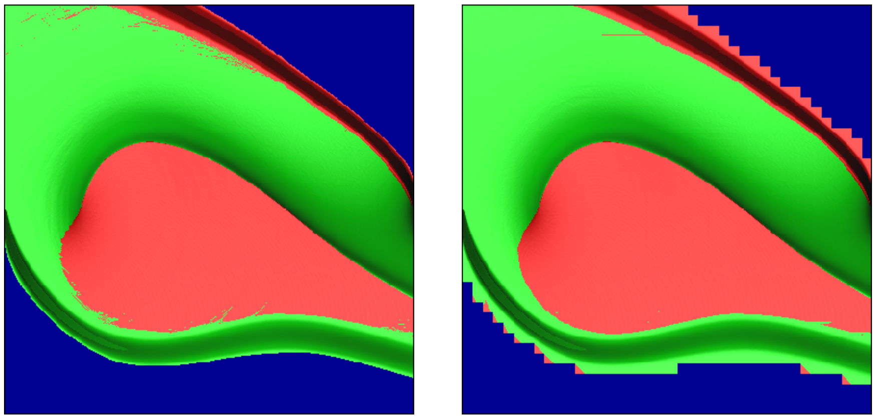


Figure 16: The MCAR control policy. Green (light gray) is positive thrust, red (medium gray) is negative thrust, and dark blue (dark gray) indicates that the partition was never processed. On the left: using one state per partition, the resolution of the unvisited states is very high, and corresponds exactly to the discontinuities in the value function. On the right: same, but using 100 states per partition. A partition can be skipped only if all of the states inside can be skipped.

5. Related Work

This work is about the efficient backpropagation of correct value function estimates. Other researchers who have investigated similar issues of efficiency have produced results that are tangible and compelling, but their algorithms are not directly comparable to ours because they have been developed in the context of on-line, model-free learning. Our algorithms, in contrast, explicitly assume the availability of a complete model.

The difference in these domains is significant and shifts the emphasis of the work. Model-free algorithms do not have the luxury of executing backups to states they have not visited; model-based algorithms, in contrast, can execute backups to any state, in any order. This fact frees us to examine different types of questions. For example, most model-free algorithms must content themselves with backpropagating information along experience traces, but there is no reason to suppose that an experience trace (played in any order) represents an *optimal* sequence of backups. It is a surrogate for what is truly desired: the ability to digest the consequences of corrected value function estimates as quickly and thoroughly as possible, throughout the entire problem. Thus, instead of examining questions related to maximizing the utility of experience traces, this work examines questions related to finding globally optimal backup sequences.

There are three primary classes of methods that researchers have used to accelerate the backpropagation of correct value information. Algorithmically, these methods form a poor basis for comparison, but conceptually, they illustrate several important points.

First is the class of *trace propagation* methods, such as TD(λ) (Sutton, 1988), Q(λ) (Peng and Williams, 1994), SARSA(λ) (Rummery and Niranjan, 1994), Fast Q(λ) (Reynolds, 2002) and Experience Stack Replay (Reynolds, 2002). These methods store a record of past experiences. As value function estimates are corrected, the changes are propagated backwards along the experience trace. Relative to value iteration, these methods derive enhanced performance partly from backing up states in a principled order (that is, backwards) and by only backing up a subset of all states. These ideas of principled ordering and partial sweeps are central in this work.

Second, there are *forced generalization* methods, such as Eligibility Traces (Singh and Sutton, 1996), PQ-learning (Zhu and Levinson, 2002) and Propagation-TD (Preux, 2002). These methods attempt to compute the value for a state based on information that was not directly associated with an experience trace. States selected for backup may have been part of a previous experience trace, or may have a geometrical or geodesic relationship to states along the actual trace (this happens implicitly with function approximators, but is forced to happen explicitly in these tabular methods).

Third, there are *prioritized computation* methods, such as Prioritized Sweeping (Moore and Atkeson, 1993) and Queue-DYNA (Peng and Williams, 1993). These methods order the backups in a principled way by constructing priority queues based on Bellman error. The idea of prioritizing backups is also central to our paper, but these methods raise many questions that merit further study. It is from these questions that our work springs.

Other researchers have considered extensions to the three basic classes previously enumerated, but the extensions do not match our domain of interest. For example, Andre et al. (1998) propose a continuous extension to Prioritized Sweeping, and Zhang and Zhang (2001) discuss a method for accelerating the convergence of value iteration in POMDPs. It is also well known that the dual of any MDP can be solved by linear programming. However, Littman et al. (1995) point out that "existing algorithms for solving LPs with provable polynomial-time performance are impractical for most MDPs. Practical algorithms for solving LPs based on the simplex method appear prone to the same sort of worst-case behavior as policy iteration and value iteration." Guestrin et al. (2003) present efficient algorithms for solving factored MDPs, but the efficiency they describe relies on closed-form expressions for state spaces that are never explicitly enumerated. Gordon (1999) provides a thorough survey of other MDP solution techniques, such as state aggregation, interpolated value iteration, approximate policy iteration, policies without values, etc.

6. Conclusions and Future Research

Based on our observations, there are several important conclusions which clarify directions for future research. In the quest for an optimal sequence of backups, the gains to be had from prioritized computation are real and compelling, but there is a lack of understanding as to what constitutes optimality and how it can be achieved. A better understanding of why GPS works is needed: more principled approaches to selecting priority metrics, reordering methods, and partitioning schemes are essential. Ideally, such principled methods would all be combined in a unified architecture.

Partitioning with a priority metric seems to be the most important improvement. Even though we observed that our partitioning criteria was suboptimal, and that our reordering algorithm was suboptimal, and that both *H1* and *H2* are suboptimal, the fact that they were not perfect seemed to make less of a difference than the fact that we used them at all. This was shown clearly by DAP: the addition of VRE and the specific priority metric used did not affect things proportionately as much as the initial use of partitions.

The variability in these results make it clear that more theory is needed to guide the development and selection of such enhancements. The most useful would be problem features and optimality definitions that would indicate which metric, reordering method and partitioning scheme are maximally effective, and which would guide the development of new enhancements. These may include distributional properties of the reward functions, distributional properties of transition matrices, strongly/weakly connected component analyses, etc. When do our enhancements work well? The rule of thumb seems to be "on problems with large diameter;" there is also evidence that they work well on problems with sparse rewards and a sparse transition matrix. We conclude that perhaps the control problems selected show GPS in its very best light – however, it was only GPS which made it possible for us to solve a 75,000,000, explicitly enumerated discrete state MDP. Without question, the algorithms are effective on certain types of problems.

More generally, the results indicate that dramatically improved performance is possible for algorithms that exploit problem-specific structure in an intelligent way. This motivates research into new types of representations (and companion algorithms) that are designed from the beginning with efficiency in mind. There are also many improvements that could be made using current ideas. For example, variable reordering can be considered a surrogate for an intra-partition priority metric. In the same way that partitioning a problem alleviates suboptimal backups, partitioning a partition might improve efficiency. The choice of a single-level partitioning scheme was arbitrary; perhaps a better solution is to generate a continuum of partitions with priority metrics at each level.

Overall, the results of this work have improved our ability to experiment with RL problems and have opened the doors of several fascinating avenues of research. At the very least, this work has enabled certain very large MDPs to be solved in tractable amounts of time and space, which would not have been possible otherwise; hopefully, this advance will allow researchers to design and tackle ever larger and more relevant problems. But the results also indicate that there are strong possibilities for even more efficient solution methods in the future, some of which may be radically different than anything considered to date, and which may enable even more intelligent systems to be created.

On-Line Appendix

The reader is encouraged to refer to

http://aml.cs.byu.edu/papers/prioritization_methods/

for additional multi-media materials. Several videos are available which graphically demonstrate the different backup orders imposed by the different priority metrics. The GPS source code is also available for download.

Acknowledgments

The authors wish to thank Todd Peterson, Michael Goodrich, Dan Ventura and Martha Wingate for their patient revisions and cogent suggestions. They would also like to thank the anonymous reviewers for their attention to detail and excellent feedback.

David Wingate is supported under a National Science Foundation Graduate Research Fellowship.

References

Charles J. Alpert. *Multi-way graph and hypergraph partitioning*. PhD thesis, University of California Los Angeles, Los Angeles, CA, 1996.

David Andre, Nir Friedman, and Ronald Parr. Generalized prioritized sweeping. *Advances in Neural Information Processing Systems*, 10:1001–1007, 1998.

Richard Barrett, Michael Berry, Tony F. Chan, James Demmel, June Donato, Jack Dongarra, Victor Eijkhout, Roldan Pozo, Charles Romine, and Henk Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. SIAM, Philadelphia, PA, 1994.

Andrew G. Barto, S. J. Bradtke, and Satinder P. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72(1):81–138, 1995.

Dimitri P. Bertsekas. Distributed dynamic programming. *IEEE Transactions on Automatic Control*, 27:610–616, 1982.

Dimitri P. Bertsekas. Distributed asynchronous computation of fixed points. *Mathematics Programming*, 27:107–120, 1983.

Dimitri P. Bertsekas and John Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.

Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, Englewood Cliffs, NJ, 1989.

Thomas Dean and Robert Givan. Model minimization in Markov Decision Processes. In *Proceedings of The Fourteenth National Conference on Artificial Intelligence*, pages 106–111, 1997.

Geoffrey J. Gordon. *Approximate Solutions to Markov Decision Processes*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 1999.

Carlos Guestrin, Daphne Koller, Ronald Parr, and Shobha Venkataraman. Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research*, 19:399–468, 2003.

Vijaykumar Gullapalli and Andrew G. Barto. Convergence of indirect adaptive asynchronous value iteration algorithms. *Advances in Neural Information Processing Systems*, 6:695–702, 1994.

George Karypis and Vipin Kumar. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing*, 48:96–129, 1998.

Michael J. Kearns and Satinder P. Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49:209–232, 2002.

Donald E. Knuth. *The Stanford GraphBase: A Platform for Combinatorial Computing*. ACM Press, New York, NY, 1993.

Harold J. Kushner and Paul Dupuis. *Numerical methods for stochastic control problems in continuous time, Second Edition*. Springer-Verlag, New York, NY, 2001.

Michael L. Littman, Thomas L. Dean, and Leslie P. Kaelbling. On the complexity of solving Markov Decision Problems. In *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence*, pages 394–402, 1995.

Raymond E. Miller and James W. Thatcher. *Complexity of computer computations*. Plenum Press, New York, NY, 1972.

Andrew W. Moore and Christopher G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, 13:103–130, 1993.

Andrew W. Moore and Christopher G. Atkeson. The parti-game algorithm for variable resolution reinforcement learning in multidimensional state space. *Machine Learning*, 21:199–233, 1995.

Remi Munos and Andrew W. Moore. Variable resolution discretization in optimal control. *Machine Learning*, 49:291–323, 2002.

Jing Peng and John Williams. Efficient learning and planning within the dyna framework. In *Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, pages 437–454, 1993.

Jing Peng and Ronald J. Williams. Incremental multi-step Q-learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 226–232, 1994.

Philippe Preux. Propagation of Q-values in tabular TD(lambda). In *Proceedings of the Thirteenth European Conference on Machine Learning*, pages 369–380, 2002.

Martin L. Puterman. *Markov Decision Processes–Discrete Stochastic Dynamic Programming*. John Wiley and Sons, Inc., New York, NY, 1994.

Martin L. Puterman and Moon C. Shin. Modified policy iteration algorithms for discounted Markov Decision Problems. *Management Science*, 24:1127–1137, 1978.

Stuart I. Reynolds. *Reinforcement Learning with Exploration*. PhD thesis, University of Birmingham, Birmingham, United Kingdom, 2002.

Gavin A. Rummery and Mahesan Niranjan. On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166, Cambridge University, Cambridge, United Kingdom, 1994.

Yousef Saad. *Iterative Methods for Sparse Linear Systems*. PWS Publishing, Boston, 1996.

Hermann A. Schwarz. *Gesammelte Mathematische Abhandlungen*, volume 2. Springer-Verlag, 1890.

Satinder P. Singh and Richard S. Sutton. Reinforcement learning with replacing eligibility traces. *Machine Learning*, 22:123–158, 1996.

Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

Richard S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. *Advances in Neural Information Processing Systems*, 8:1038–1044, 1996.

Ray S. Tuminaro, Mike Heroux, S. A. Hutchinson, and John N. Shadid. *Official Aztec User's Guide: Version 2.1*. Sandia National Laboratory, Albuquerque, NM, 1999.

Ronald J. Williams and Leemon C. Baird. Tight performance bounds on greedy policies based on imperfect value functions. Technical Report NU-CCS-93-14, Northeastern University, Boston, MA, 1993.

David Wingate and Kevin D. Seppi. Cache efficiency of priority metrics for MDP solvers. In *AAAI Workshop on Learning and Planning in Markov Processes*, pages 103–106, 2004a.

David Wingate and Kevin D. Seppi. P3VI: A partitioned, prioritized, parallel value iterator. In *Proceedings of the Twenty-First International Conference on Machine Learning*, pages 863–870, 2004b.

Nevin L. Zhang, Stephen S. Lee, and Weihong Zhang. A method for speeding up value iteration in partially observable Markov Decision Processes. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 696–703, 1999.

Nevin L. Zhang and Weihong Zhang. Speeding up the convergence of value iteration in partially observable Markov Decision Processes. *Journal of Artificial Intelligence Research*, 14:29–51, 2001.

Weiyu Zhu and Stephen Levinson. PQ-learning: an efficient robot learning method for intelligent behavior acquisition. In *Proceedings of the Seventh International Conference on Intelligent Autonomous Systems*, 2002.

Journal of Machine Learning Research 6 (2005) 883–904 Submitted 12/04; Revised 3/05; Published 5/05

Learning from Examples as an Inverse Problem

Ernesto De Vito DEVITO@UNIMO.IT
Dipartimento di Matematica
Università di Modena e Reggio Emilia
Modena, Italy
and INFN, Sezione di Genova,
Genova, Italy

Lorenzo Rosasco ROSASCO@DISI.UNIGE.IT
Andrea Caponnetto CAPONNETTO@DISI.UNIGE.IT
Umberto De Giovannini UMBERTO.DEGIOVANNINI@FASTWEBNET.IT
Francesca Odone ODONE@DISI.UNIGE.IT
DISI
Università di Genova,
Genova, Italy

Editor: Peter Bartlett

Abstract

Many works related learning from examples to regularization techniques for inverse problems, emphasizing the strong algorithmic and conceptual analogy of certain learning algorithms with regularization algorithms. In particular it is well known that regularization schemes such as Tikhonov regularization can be effectively used in the context of learning and are closely related to algorithms such as support vector machines. Nevertheless the connection with inverse problem was considered only for the discrete (finite sample) problem and the probabilistic aspects of learning from examples were not taken into account. In this paper we provide a natural extension of such analysis to the continuous (population) case and study the interplay between the discrete and continuous problems. From a theoretical point of view, this allows to draw a clear connection between the consistency approach in learning theory and the stability convergence property in ill-posed inverse problems. The main mathematical result of the paper is a new probabilistic bound for the regularized least-squares algorithm. By means of standard results on the approximation term, the consistency of the algorithm easily follows.

Keywords: statistical learning, inverse problems, regularization theory, consistency

1. Introduction

The main goal of learning from examples is to infer an estimator from a finite set of examples. The crucial aspect in the problem is that the examples are drawn according to a fixed but unknown probabilistic input-output relation and the desired property of the selected function is to be descriptive also of new data, i.e. it should *generalize*. The fundamental work of Vapnik and further developments (see Vapnik (1998); Alon et al. (1997) and Bartlett and Mendelson (2002) for recent results) show that the key to obtain a meaningful solution is to control the complexity of the hypothesis space. Interestingly, as pointed out in a number of papers (see Poggio and Girosi (1992); Evgeniou et al. (2000) and references therein), this is in essence the idea underlying regularization techniques for ill-posed problems (Tikhonov and Arsenin, 1977; Engl et al., 1996). Not surprisingly the form of the algorithms proposed in both theories is strikingly similar (Mukherjee et al., 2002) and the point of view of regularization is indeed not new to learning (Poggio and Girosi, 1992; Evgeniou et al., 2000; Vapnik, 1998; Arbib, 1995; Fine, 1999; Kecman, 2001; Schölkopf and Smola, 2002). In particular it allowed to cast a large class of algorithms in a common framework, namely regularization networks or regularized kernel methods (Evgeniou et al., 2000; Schölkopf and Smola, 2002).

Anyway a careful analysis shows that a rigorous mathematical connection between learning theory and the theory of ill-posed inverse problems is not straightforward since the settings underlying the two theories are different. In fact learning theory is intrinsically probabilistic whereas the theory of inverse problem is mostly deterministic. Statistical methods were recently applied in the context of inverse problems (Kaipio and Somersalo, 2005). Anyway a Bayesian point of view is considered which differs from the usual learning theory approach. Recently the connection between learning and inverse problems was considered in the restricted setting in which the elements of the input space are fixed and not probabilistically drawn (Mukherjee et al., 2004; Kurkova, 2004). This corresponds to what is usually called nonparametric regression with fixed design (Györfi et al., 1996) and when the noise level is fixed and known, the problem is well studied in the context of inverse problems (Bertero et al., 1988). In the case of fixed design on a finite grid the problem is mostly that we are dealing with an ill-conditioned problem, that is *unstable* w.r.t. the data. Though such setting is indeed close to the algorithmic setting from a theoretical perspective it is not general enough to allow a consistency analysis of a given algorithm since it does not take care of the random sampling providing the data. In this paper we extend the analysis to the setting of nonparametric regression with random design (Györfi et al., 1996).

Our analysis and contribution develop in two steps. First, we study the mathematical connections between learning theory and inverse problems theory. We consider the specific case of quadratic loss and analyse the population case (i.e. when the probability distribution is known) to show that the discrete inverse problem which is solved in practice can be seen as the stochastic discretization of an infinite dimensional inverse problem. This ideal problem is, in general,*ill-posed* (Tikhonov and Arsenin, 1977) and its solution corresponds to the target function which is the final goal in learning theory. This clarifies in particular the following important fact. Regularized solutions in learning problems should not only provide stable approximate solutions to the discrete problem but especially give continuous estimates of the solution to the ill-posed infinite dimensional problem. Second, we exploit the established connection to study the regularized least-squares algorithm. This passes through the definition of a natural notion of discretization noise providing a straightforward relation between the number of available data and the noise affecting the problem. Classical regularization theory results can be easily adapted to the needs of learning. In particular our definition of noise together with well-known results concerning Tikhonov regularization for inverse problems with modelling error can be applied to derive a new probabilistic bound for the estimation error of regularized least squares improving recently proposed results (Cucker and Smale, 2002a; De Vito et al., 2004). The approximation term can be studied through classical spectral theory arguments. The consistency of the algorithm easily follows. As the major aim of the paper was to investigate the relation between learning from examples and inverse problem we just prove convergence without dealing with rates. Anyway the approach proposed in Cucker and Smale (2002a); De Vito et al. (2004) to study the approximation term can be straightforwardly applied to derive explicit rates under suitable a priori conditions.

Several theoretical results are available on regularized kernel methods for large class of loss functions. The stability approach proposed in Bousquet and Elisseeff (2002) allows to find data-dependent generalization bounds. In Steinwart (2004) it is proved that such results as well as other probabilistic bounds can be used to derive consistency results without convergence rates. For the specific case of regularized least-squares algorithm a functional analytical approach to derive consistency results for regularized least squares was proposed in Cucker and Smale (2002a) and eventually refined in De Vito et al. (2004) and Smale and Zhou (2004b). In the latter the connection between learning and sampling theory is investigated. Some weaker results in the same spirit of those presented in this paper can be found in Rudin (2004). Anyway none of the mentioned papers exploit the connection with inverse problems. The arguments used to derive our results are close to those used in the study of stochastic inverse problems discussed in Vapnik (1998). From the algorithmic point of view Ong and Canu (2004) apply other techniques than Tikhonov regularization in the context of learning. In particular several iterative algorithms are considered and convergence with respect to the regularization parameter (semiconvergence) is proved.

The paper is organized as follows. After recalling the main concepts and notation of statistical learning (Section 2) and of inverse problems (Section 3), in Section 4 we develop a formal connection between the two theories. In Section 5 the main results are stated, discussed and proved. In the Appendix we collect some technical results we need in our proofs. Finally in Section 6 we conclude with some remarks and open problems.

2. Learning from Examples

We briefly recall some basic concepts of statistical learning theory (for details see Vapnik (1998); Evgeniou et al. (2000); Schölkopf and Smola (2002); Cucker and Smale (2002b) and references therein).

In the framework of learning from examples, there are two sets of variables: the input space X, which we assume to be a compact subset of $\mathbb{R}^n$, and the output space Y, which is a subset of $\mathbb{R}$ contained in $[-M,M]$ for some $M \geq 0$. The relation between the input $x \in X$ and the output $y \in Y$ is described by a probability distribution $\rho(x,y) = \nu(x)\rho(y|x)$ on $X \times Y$. The distribution ρ is known only through a sample $\mathbf{z} = (\mathbf{x},\mathbf{y}) = ((x_1,y_1),\ldots,(x_\ell,y_\ell))$, called *training set*, drawn independently and identically distributed (i.i.d.) according to ρ. Given the sample $\mathbf{z}$, the aim of learning theory is to find a function $f_{\mathbf{z}} : X \to \mathbb{R}$ such that $f_{\mathbf{z}}(x)$ is a good estimate of the output y when a new input x is given. The function $f_{\mathbf{z}}$ is called *estimator* and the map providing $f_{\mathbf{z}}$, for any training set $\mathbf{z}$, is called *learning algorithm*.

Given a measurable function $f : X \to \mathbb{R}$, the ability of f to describe the distribution ρ is measured by its *expected risk* defined as

$$I[f] = \int_{X \times Y} V(f(x),y)\, d\rho(x,y),$$

where $V(f(x),y)$ is the *loss function*, which measures the cost paid by replacing the true label y with the estimate $f(x)$. In this paper we consider the square loss

$$V(f(x),y) = (f(x)-y)^2.$$

With this choice, it is well known that the regression function

$$g(x) = \int_Y y\, d\rho(y|x)$$

is well defined (since Y is bounded) and is the minimizer of the expected risk over the space of all the measurable real functions on X. In this sense g can be seen as the ideal estimator of the distribution probability ρ. However, the regression function cannot be reconstructed exactly since only a finite, possibly small, set of examples $\mathbf{z}$ is given.

To overcome this problem, in the framework of the regularized least squares algorithm (Wahba, 1990; Poggio and Girosi, 1992; Cucker and Smale, 2002b; Zhang, 2003), an hypothesis space $\mathcal{H}$ of functions is fixed and the estimator $f_{\mathbf{z}}{}^{\lambda}$ is defined as the solution of the regularized least squares problem,

$$\min_{f \in \mathcal{H}} \{ \frac{1}{\ell} \sum_{i=1}^{\ell} (f(x_i) - y_i)^2 + \lambda \Omega(f) \}, \tag{1}$$

where Ω is a penalty term and λ is a positive parameter to be chosen in order to ensure that the discrepancy.

$$I[f_{\mathbf{z}}{}^{\lambda}] - \inf_{f \in \mathcal{H}} I[f]$$

is small with high probability. Since ρ is unknown, the above difference is studied by means of a probabilistic bound $\mathcal{B}(\lambda,\ell,\eta)$, which is a function depending on the regularization parameter λ, the number ℓ of examples and the confidence level $1-\eta$, such that

$$\mathbf{P}\left[I[f_{\mathbf{z}}^{\lambda}] - \inf_{f\in\mathcal{H}} I[f] \leq \mathcal{B}(\lambda,\ell,\eta)\right] \geq 1-\eta.$$

We notice that, in general, $\inf_{f\in\mathcal{H}} I[f]$ is larger than $I[g]$ and represents a sort of irreducible error (Hastie et al., 2001) associated with the choice of the space $\mathcal{H}$. We do not require the infimum $\inf_{f\in\mathcal{H}} I[f]$ to be achieved. If the minimum on $\mathcal{H}$ exists, we denote the minimizer by $f_{\mathcal{H}}$.

In particular, the learning algorithm is *consistent* if it is possible to choose the regularization parameter, as a function of the available data $\lambda = \lambda(\ell,\mathbf{z})$, in such a way that

$$\lim_{\ell\to+\infty} \mathbf{P}\left[I[f_{\mathbf{z}}^{\lambda(\ell,\mathbf{z})}] - \inf_{f\in\mathcal{H}} I[f] \geq \varepsilon\right] = 0, \tag{2}$$

for every $\varepsilon > 0$. The above convergence in probability is usually called *(weak) consistency* of the algorithm (see Devroye et al. (1996) for a discussion on the different kind of consistencies).

In this paper we assume that the hypothesis space $\mathcal{H}$ is a reproducing kernel Hilbert space (RKHS) on X with a continuous kernel K. We recall the following facts (Aronszajn, 1950; Schwartz, 1964). The kernel $K : X\times X \to \mathbb{R}$ is a continuous symmetric positive definite function, where *positive definite* means that

$$\sum_{i,j} a_i a_j K(x_i,x_j) \geq 0.$$

for any $x_1,\ldots x_n \in X$ and $a_1,\ldots a_n \in \mathbb{R}$.

The space $\mathcal{H}$ is a real separable Hilbert space whose elements are real continuous functions defined on X. In particular, the functions $K_x = K(\cdot,x)$ belong to $\mathcal{H}$ for all $x\in X$, and

$$\begin{aligned} \mathcal{H} &= \overline{\operatorname{span}\{K_x \,|\, x\in X\}} \\ \langle K_x, K_t\rangle_{\mathcal{H}} &= K(x,t) \quad \forall x,t\in X, \end{aligned}$$

where $\langle\cdot,\cdot\rangle_{\mathcal{H}}$ is the scalar product in $\mathcal{H}$. Moreover, since the kernel is continuous and X is compact

$$\kappa = \sup_{x\in X} \sqrt{K(x,x)} = \sup_{x\in X} \|K_x\|_{\mathcal{H}} < +\infty, \tag{3}$$

where $\|\cdot\|_{\mathcal{H}}$ is the norm in $\mathcal{H}$. Finally, given $x\in X$, the following *reproducing* property holds

$$f(x) = \langle f, K_x\rangle_{\mathcal{H}} \quad \forall f\in\mathcal{H}. \tag{4}$$

In particular, in the learning algorithm (1) we choose the penalty term

$$\Omega(f) = \|f\|_{\mathcal{H}}^{2},$$

so that, by a standard convex analysis argument, the minimizer $f_{\mathbf{z}}^{\lambda}$ exists, is unique and can be computed by solving a linear finite dimensional problem, (Wahba, 1990).

With the above choices, we will show that the consistency of the regularized least squares algorithm can be deduced using the theory of linear inverse problems we review in the next section.

3. Ill-Posed Inverse Problems and Regularization

In this section we give a very brief account of the main concepts of linear inverse problems and regularization theory (see Tikhonov and Arsenin (1977); Groetsch (1984); Bertero et al. (1985, 1988); Engl et al. (1996); Tikhonov et al. (1995) and references therein).

Let $\mathcal{H}$ and $\mathcal{K}$ be two Hilbert spaces and $A : \mathcal{H} \to \mathcal{K}$ a linear bounded operator. Consider the equation

$$Af = g \tag{5}$$

where $g \in \mathcal{K}$ is the *exact* datum. Finding the function f satisfying the above equation, given A and g, is the linear inverse problem associated to (5). In general the above problem is ill-posed, that is, the solution either not exists, is not unique or does not depend continuously on the datum g. Existence and uniqueness can be restored introducing the Moore-Penrose generalized solution $f^\dagger$ defined as the minimal norm solution of the least squares problem

$$\min_{f \in \mathcal{H}} \|Af - g\|^2_{\mathcal{K}}. \tag{6}$$

It can be shown (Tikhonov et al., 1995) that the generalized solution $f^\dagger$ exists if and only if $Pg \in \text{Range}(A)$, where P is the projection on the closure of the range of A. However, the generalized solution $f^\dagger$ does not depend continuously on the datum g, so that finding $f^\dagger$ is again an ill-posed problem. This is a problem since the exact datum g is not known, but only a *noisy* datum $g_\delta \in \mathcal{K}$ is given, where $\|g - g_\delta\|_{\mathcal{K}} \leq \delta$. According to Tikhonov regularization (Tikhonov and Arsenin, 1977) a possible way to find a solution depending continuously on the data is to replace Problem (6) with the following convex problem

$$\min_{f \in \mathcal{H}} \{\|Af - g_\delta\|^2_{\mathcal{K}} + \lambda \|f\|^2_{\mathcal{H}}\}, \tag{7}$$

and, for $\lambda > 0$, the unique minimizer is given by

$$f^\lambda_\delta = (A^*A + \lambda I)^{-1} A^* g_\delta, \tag{8}$$

where A^* the adjoint operator of A. A crucial issue is the choice of the regularization parameter λ as a function of the noise. A basic requirement is that the *reconstruction error*

$$\left\| f^\lambda_\delta - f^\dagger \right\|_{\mathcal{H}}$$

is small. In particular, λ must be selected, as a function of the noise level δ and the data g_δ, in such a way that the regularized solution $f^{\lambda(\delta, g_\delta)}_\delta$ converges to the generalized solution, that is,

$$\lim_{\delta \to 0} \left\| f^{\lambda(\delta, g_\delta)}_\delta - f^\dagger \right\|_{\mathcal{H}} = 0, \tag{9}$$

for any g such that $f^\dagger$ exists.

Remark 1 *We briefly comment on the well known difference between ill-posed and ill-conditioned problems (Bertero et al., 1988). Finite dimensional problems are often well-posed. In particular it can be shown that if a solution exists unique then continuity of A^{-1} is always ensured. Nonetheless regularization is needed since the problems are usually ill conditioned and lead to unstable solutions.*

Sometimes, another measure of the error, namely the *residual*, is considered according to the following definition

$$\left\|Af_\delta^\lambda - Pg\right\|_{\mathcal{K}} = \left\|Af_\delta^\lambda - Af^\dagger\right\|_{\mathcal{K}}, \tag{10}$$

which will be important in our analysis of learning. Comparing (9) and (10), it is clear that while studying the convergence of the residual we do not have to assume that the generalized solution exists.

We conclude this section noting that the above formalism can be easily extended to the case of a noisy operator $A_\delta : \mathcal{H} \to \mathcal{K}$ where

$$\|A - A_\delta\| \leq \delta,$$

and $\|\cdot\|$ is the operator norm (Tikhonov et al., 1995).

4. Learning as an Inverse Problem

The similarity between regularized least squares and Tikhonov regularization is apparent comparing Problems (1) and (7). However while trying to formalize this analogy several difficulties emerge.

- To treat the problem of learning in the setting of ill-posed inverse problems we have to define a direct problem by means of a suitable operator A between two Hilbert spaces $\mathcal{H}$ and $\mathcal{K}$.
- The nature of the noise δ in the context of statistical learning is not clear .
- We have to clarify the relation between consistency, expressed by (2), and the convergence considered in (9).

In the following we present a possible way to tackle these problems and show the problem of learning can be indeed rephrased in a framework close to the one presented in the previous section.

We let $L^2(X,\nu)$ be the Hilbert space of square integrable functions on X with respect to the marginal measure ν and we define the operator $A : \mathcal{H} \to L^2(X,\nu)$ as

$$(Af)(x) = \langle f, K_x \rangle_{\mathcal{H}},$$

where K is the reproducing kernel of $\mathcal{H}$. The fact that K is bounded, see (3), ensures that A is a bounded linear operator. Two comments are in order. First, from (4) we see that the action of A on an element f is simply

$$(Af)(x) = f(x) \quad \forall x \in x,\ f \in \mathcal{H},$$

that is, A is the canonical inclusion of $\mathcal{H}$ into $L^2(X,\nu)$. However it is important to note that A changes the norm since $\|f\|_{\mathcal{H}}$ is different to $\|f\|_{L^2(X,\nu)}$. Second, to avoid pathologies connected with subsets of zero measure, we assume that ν is not degenerate.[1] This condition and the fact that K is continuous ensure that A is injective (see the Appendix for the proof).

It is known that, considering the quadratic loss function, the expected risk can be written as

$$\begin{aligned} I[f] &= \int_X (f(x)-g(x))^2\, d\nu(x) + \int_{X\times Y} (y-g(x))^2\, d\rho(x,y) \\ &= \|f-g\|^2_{L^2(X,\nu)} + I[g], \end{aligned}$$

where g is the regression function (Cucker and Smale, 2002b) and f is any function in $L^2(X,\nu)$. If f belongs to the hypothesis space $\mathcal{H}$, the definition of the operator A allows to write

$$I[f] = \|Af-g\|^2_{L^2(X,\nu)} + I[g]. \tag{11}$$

Moreover, if P is the projection on the closure of the range of A, that is, the closure of $\mathcal{H}$ into $L^2(X,\nu)$, then the definition of projection gives

$$\inf_{f\in\mathcal{H}} \|Af-g\|^2_{L^2(X,\nu)} = \|g-Pg\|^2_{L^2(X,\nu)}\,. \tag{12}$$

Given $f \in \mathcal{H}$, clearly $PAf = Af$, so that

$$I[f] - \inf_{f\in\mathcal{H}} I[f] = \|Af-g\|^2_{L^2(X,\nu)} - \|g-Pg\|^2_{L^2(X,\nu)} = \|Af-Pg\|^2_{L^2(X,\nu)}\,, \tag{13}$$

which is the square of the residual of f.

Now, comparing (11) and (6), it is clear that the expected risk admits a minimizer $f_{\mathcal{H}}$ on the hypothesis space $\mathcal{H}$ if and only if $f_{\mathcal{H}}$ is precisely the generalized solution $f^\dagger$ of the linear inverse problem

$$Af = g. \tag{14}$$

The fact that $f_{\mathcal{H}}$ is the minimal norm solution of the least squares problem is ensured by the fact that A is injective.

Let now $\mathbf{z} = (\mathbf{x},\mathbf{y}) = ((x_1,y_1),\ldots,(x_\ell,y_\ell))$ be the training set. The above arguments can be repeated replacing the set X with the finite set $\{x_1,\ldots,x_\ell\}$. We now get a discretized version of A by defining the *sampling operator* (Smale and Zhou, 2004a)

$$A_{\mathbf{x}} : \mathcal{H} \to \mathbf{E}^\ell \quad (A_{\mathbf{x}} f)_i = \langle f, K_{x_i}\rangle_{\mathcal{H}} = f(x_i),$$

1. This means that all the open non-void subsets of X have strictly positive measure.

where $\mathbf{E}^\ell = \mathbb{R}^\ell$ is the finite dimensional euclidean space endowed with the scalar product

$$\langle \mathbf{w}, \mathbf{w}' \rangle_{\mathbf{E}^\ell} = \frac{1}{\ell} \sum_{i=1}^{\ell} w_i w_i'.$$

It is straightforward to check that

$$\frac{1}{\ell} \sum_{i=1}^{\ell} (f(x_i) - y_i)^2 = \| A_{\mathbf{x}} f - \mathbf{y} \|_{\mathbf{E}^\ell}^2 ,$$

so that the estimator $f_{\mathbf{z}}{}^{\lambda}$ given by the regularized least squares algorithm, see Problem (1), is the Tikhonov regularized solution of the discrete problem

$$A_{\mathbf{x}} f = \mathbf{y}. \tag{15}$$

At this point it is useful to remark the following three facts. First, in learning from examples rather than finding a stable approximation to the solution of the noisy (discrete) Problem (15), we want to find a meaningful approximation to the solution of the exact (continuous) Problem (14) (compare with Kurkova (2004)). Second, in statistical learning theory, the key quantity is the residual of the solution, which is a weaker measure than the reconstruction error, usually studied in the inverse problem setting. In particular, consistency requires a weaker kind of convergence than the one usually studied in the context of inverse problems . Third, we observe that in the context of learning the existence of the minimizer $f_{\mathcal{H}}$, that is, of the generalized solution, is no longer needed to define good asymptotic behavior. In fact when the projection of the regression function is not in the range of A the ideal solution $f_{\mathcal{H}}$ does not exist but this is not a problem since Eq. (12) still holds.

After this preliminary considerations in the next section we further develop our analysis stating the main mathematical results of this paper.

5. Regularization, Stochastic Noise and Consistency

Table 1 compares the classical framework of inverse problems (see Section 3) with the formulation of learning proposed above. We note some differences. First, the noisy data space $\mathbf{E}^\ell$ is different from the exact data space $L^2(X, \nu)$ so that A and $A_{\mathbf{x}}$ belong to different spaces, as well as g and $\mathbf{y}$. A measure of the difference between $A_{\mathbf{x}}$ and A, and between g and $\mathbf{y}$ is then required. Second, both $A_{\mathbf{x}}$ and $\mathbf{y}$ are random variables and we need to relate the noise δ to the number ℓ of examples in the training set $\mathbf{z}$. Given the above premise our derivation of consistency results is developed in two steps: we first study the residual of the solution by means of a measure of the noise due to discretization, then we show a possible way to give a probabilistic evaluation of the noise previously introduced.

Inverse problem	**Learning theory**
input space $\mathcal{H}$	hypothesis space RKHS $\mathcal{H}$
data space $\mathcal{K}$	target space $L^2(X,\nu)$
norm in $\mathcal{K}$ $\|f\|_{\mathcal{K}}$	norm in $L^2(X,\nu)$ $\|f\|_{L^2(X,\nu)}$
exact operator A	inclusion of $\mathcal{H}$ into $L^2(X,\nu)$
exact datum g	regression function $g(x) = \int_Y y\, d\rho(y\|x)$
generalized solution $f^\dagger$	ideal solution $f_{\mathcal{H}}$
reconstruction error $\left\|f - f^\dagger\right\|_{\mathcal{H}}$	residual $\|Af - Af_{\mathcal{H}}\|^2_{L^2(X,\nu)} = I[f] - I[f_{\mathcal{H}}]$
noisy data space $\mathcal{K}$	$\mathbf{E}^\ell$
noisy data $g_\delta \in \mathcal{K}$	$\mathbf{y} \in \mathbf{E}^\ell$
noisy operator $A_\delta : \mathcal{H} \to \mathcal{K}$	sampling operator $A_{\mathbf{x}} : \mathcal{H} \to \mathbf{E}^\ell$
Tikhonov regularization	Regularized least squares algorithm

Table 1: The above table summarizes the relation between the theory of inverse problem and the theory of learning from examples. When the projection of the regression function is not in the range of the operator A the ideal solution $f_{\mathcal{H}}$ does not exist. Nonetheless, in learning theory, if the ideal solution does not exist the asymptotic behavior can still be studied since we are looking for the residual.

5.1 Bounding the Residual of Tikhonov Solution

In this section we study the dependence of the minimizer of Tikhonov functional on the operator A and the data g. We indicate with $\mathcal{L}(\mathcal{H})$ and $\mathcal{L}(\mathcal{H},\mathcal{K})$ the Banach space of bounded linear operators from $\mathcal{H}$ into $\mathcal{H}$ and from $\mathcal{H}$ into $\mathcal{K}$ respectively. We denote with $\|\cdot\|_{\mathcal{L}(\mathcal{H})}$ the uniform norm in $\mathcal{L}(\mathcal{H})$ and, if $A \in \mathcal{L}(\mathcal{H},\mathcal{K})$, we recall that A^* is the adjoint operator. The Tikhonov solutions of Problems (14) and (15) can be written as

$$\begin{aligned} f^\lambda &= (A^*A + \lambda I)^{-1} A^* g, \\ f_{\mathbf{z}}^\lambda &= (A_{\mathbf{x}}^* A_{\mathbf{x}} + \lambda I)^{-1} A_{\mathbf{x}}^* \mathbf{y} \end{aligned}$$

(see for example Engl et al., 1996, Chapter 5, page 117). The above equations show that $f_{\mathbf{z}}^\lambda$ and f^λ depend only on $A_{\mathbf{x}}^* A_{\mathbf{x}}$ and A^*A, which are operators from $\mathcal{H}$ into $\mathcal{H}$, and on $A_{\mathbf{x}}^*\mathbf{y}$ and A^*g, which are elements of $\mathcal{H}$. This observation suggests that noise levels could be evaluated controlling $\|A_{\mathbf{x}}^*A_{\mathbf{x}} - A^*A\|_{\mathcal{L}(\mathcal{H})}$ and $\|A_{\mathbf{x}}^*\mathbf{y} - A^*g\|_{\mathcal{H}}$.

For this purpose, for every $\delta = (\delta_1, \delta_2) \in \mathbb{R}^2_+$, we define the collection of training sets

$$\mathcal{U}_\delta := \{\mathbf{z} \in (X \times Y)^\ell \mid \|A_{\mathbf{x}}^*\mathbf{y} - A^*g\|_{\mathcal{H}} \le \delta_1,\ \|A_{\mathbf{x}}^*A_{\mathbf{x}} - A^*A\|_{\mathcal{L}(\mathcal{H})} \le \delta_2\}.$$

Recalling that P is the projection on the closure of the range of A and $Y \subset [-M,M]$, we are ready to state the following theorem.

Theorem 2 *Given $\lambda > 0$, the following inequality holds*

$$\left| \left\| A f_{\mathbf{z}}^{\lambda} - Pg \right\|_{L^2(X,\nu)} - \left\| A f^{\lambda} - Pg \right\|_{L^2(X,\nu)} \right| \leq \frac{\delta_1}{2\sqrt{\lambda}} + \frac{M\delta_2}{4\lambda}$$

for any training set $\mathbf{z} \in \mathcal{U}_\delta$.

We postpone the proof to Section 5.4 and briefly comment on the above result. The first term in the l.h.s. of the inequality is exactly the residual of the regularized solution whereas the second term represents the approximation error, which does not depend on the sample. Our bound quantifies the difference between the residual of the regularized solutions of the exact and noisy problems in terms of the noise level $\delta = (\delta_1, \delta_2)$. As mentioned before this is exactly the kind of result needed to derive consistency. Our result bounds the residual both from above and below and is obtained introducing the collection $\mathcal{U}_\delta$ of training sets compatible with a certain noise level δ. It is left to quantify the noise level corresponding to a training set of cardinality ℓ. This will be achieved in a probabilistic setting in the next section, where we also discuss a standard result on the approximation error.

5.2 Stochastic Evaluation of the Noise and Approximation Term

In this section we give a probabilistic evaluation of the noise levels δ_1 and δ_2 and we analyze the behavior of the term $\left\| A f^{\lambda} - Pg \right\|_{L^2(X,\nu)}$. In the context of inverse problems a noise estimate is a part of the available data whereas in learning problems we need a probabilistic analysis.

Theorem 3 *Let $0 < \eta < 1$. Then*

$$\mathbf{P}\left[\|A^* g - A_{\mathbf{x}}{}^* \mathbf{y}\|_{\mathcal{H}} \leq \delta_1(\ell,\eta),\ \|A^*A - A_{\mathbf{x}}{}^* A_{\mathbf{x}}\|_{\mathcal{L}(\mathcal{H})} \leq \delta_2(\ell,\eta) \right] \geq 1 - \eta$$

where $\kappa = \sup_{x \in X} \sqrt{K(x,x)}$,

$$\delta_1(\ell,\eta) = \frac{M\kappa}{2} \psi\left(\frac{8}{\ell} \log \frac{4}{\eta} \right) \qquad\qquad \delta_2(\ell,\eta) = \frac{\kappa^2}{2} \psi\left(\frac{8}{\ell} \log \frac{4}{\eta} \right)$$

with $\psi(t) = \frac{1}{2}(t + \sqrt{t^2 + 4t}) = \sqrt{t} + o(\sqrt{t})$.

We refer again to Section 5.4 for the complete proof and add a few comments. The one proposed is just one of the possible probabilistic tools that can be used to study the above random variables. For example union bounds and Hoeffding's inequality can be used introducing a suitable notion of covering numbers on $X \times Y$.

An interesting aspect in our approach is that the collection of training sets compatible with a certain noise level δ does not depend on the regularization parameter λ. This last fact allows us to consider indifferently data independent parameter choices $\lambda = \lambda(\ell)$ as well as data dependent choices $\lambda = \lambda(\ell, \mathbf{z})$. Since through data dependent parameter choices the regularization parameter becomes a function of the given sample $\lambda(\ell, \mathbf{z})$, in general some further analysis is needed to ensure that the bounds hold uniformly w.r.t. λ.

We now consider the term $\left\|Af^{\lambda} - Pg\right\|_{L^2(X,\nu)}$ which does not depend on the training set $\mathbf{z}$ and plays the role of an approximation error (Smale and Zhou, 2003; Niyogi and Girosi, 1999). The following is a trivial modification of a classical result in the context of inverse problems (see for example Engl et al. (1996) Chapter 4, Theorem 4.1, p. 72).

Proposition 4 *Let f^{λ} the Tikhonov regularized solution of the problem $Af = g$, then the following convergence holds*

$$\lim_{\lambda \to 0^+} \left\|Af^{\lambda} - Pg\right\|_{L^2(X,\nu)} = 0.$$

We report the proof in the Appendix for completeness. The above proposition ensures that, independently of the probability measure ρ, the approximation term goes to zero as $\lambda \to 0$. Unfortunately it is well known, both in learning theory (see for example Devroye et al. (1996); Vapnik (1998); Smale and Zhou (2003); Steinwart (2004)) and inverse problems theory (Groetsch, 1984), that such a convergence can be arbitrarily slow and convergence rates can be obtained only under some assumptions either on the regression function g or on the probability measure ρ (Smale and Zhou, 2003). In the context of RKHS the issue was considered in Cucker and Smale (2002a); De Vito et al. (2004) and we can strightforwardly apply those results to obtain explicit convergence rates.

We are now in the position to derive the consistency result that we present in the following section.

5.3 Consistency and Regularization Parameter Choice

Combining Theorems 2 and 3 with Proposition 4, we easily derive the following result (see Section 5.4 for the proof).

Theorem 5 *Given $0 < \eta < 1$, $\lambda > 0$ and $\ell \in \mathbb{N}$, the following inequality holds with probability greater that $1 - \eta$*

$$\begin{aligned} I[f_{\mathbf{z}}^{\lambda}] - \inf_{f \in \mathcal{H}} I[f] &\leq \left[\left(\frac{M\kappa}{2\sqrt{\lambda}} + \frac{M\kappa^2}{4\lambda}\right)\psi\left(\frac{8}{\ell}\log\frac{4}{\eta}\right) + \left\|Af^{\lambda} - Pg\right\|_{L^2(X,\nu)}\right]^2 \qquad (16) \\ &= \left[M\kappa^2\sqrt{\frac{\log\frac{4}{\eta}}{2\lambda^2\ell}} + \left\|Af^{\lambda} - Pg\right\|_{L^2(X,\nu)} + o\left(\sqrt{\frac{1}{\lambda^2\ell}\log\frac{4}{\eta}}\right)\right]^2 \end{aligned}$$

where $\psi(\cdot)$ is defined as in Theorem 3. Moreover, if $\lambda = O(l^{-b})$ with $0 < b < \frac{1}{2}$, then

$$\lim_{\ell \to +\infty} \mathbf{P}\left[I[f_{\mathbf{z}}^{\lambda(\ell,\mathbf{z})}] - \inf_{f \in \mathcal{H}} I[f] \geq \varepsilon\right] = 0.$$

for every $\varepsilon > 0$.

As mentioned before, the second term in the right hand side of the above inequality is an approximation error and vanishes as λ goes to zero. The first term in the right hand side of Inequality (16) plays the role of sample error. It is interesting to note that since $\delta = \delta(\ell)$ we have an equivalence between the limit $\ell \to \infty$, usually studied in learning theory, and the limit $\delta \to 0$, usually considered for inverse problems. Our result presents the formal connection between the consistency approach considered in learning theory, and the regularization-stability convergence property used in ill-posed inverse problems. Although it is known that connections already exist, as far as we know, this is the first full connection between the two areas, for the specific case of square loss.

We now briefly compare our result with previous work on the consistency of the regularized least squares algorithm. Recently, several works studied the consistency property and the related convergence rate of learning algorithms inspired by Tikhonov regularization. For the classification setting, a general discussion considering a large class of loss functions can be found in Steinwart (2004), whereas some refined results for specific loss functions can be found in Chen et al. (2004) and Scovel and Steinwart (2003). For regression problems in Bousquet and Elisseeff (2002) a large class of loss functions is considered and a bound of the form

$$I[f_{\mathbf{z}}{}^{\lambda}] - I_{\mathbf{z}}[f_{\mathbf{z}}{}^{\lambda}] \leq O\left(\frac{1}{\sqrt{\ell}\lambda}\right)$$

is proved, where $I_{\mathbf{z}}[f_{\mathbf{z}}{}^{\lambda}]$ is the empirical error.[2] Such a bound allows to prove consistency using the error decomposition in Steinwart (2004). The square loss was considered in Zhang (2003) where, using leave-one out techniques, the following bound in expectation was proved

$$E_{\mathbf{z}}(I[f_{\mathbf{z}}{}^{\lambda}]) \leq O\left(\frac{1}{\ell\lambda}\right).$$

Techniques similar to those used in this paper are used in De Vito et al. (2004) to derive a bound of the form

$$I[f_{\mathbf{z}}{}^{\lambda}] - \inf_{f\in\mathcal{H}} I[f] \leq \left(S(\lambda,\ell) + \left\|Af^{\lambda} - Pg\right\|_{L^2(X,\nu)}\right)^2$$

where $S(\lambda,\ell)$ is a data-independent bound on $\left\|f_{\mathbf{z}}{}^{\lambda} - f^{\lambda}\right\|_{L^2(X,\nu)}$. In that case $S(\lambda,\ell) \leq O\left(\frac{1}{\sqrt{\ell}\lambda^{\frac{3}{2}}}\right)$ and we see that Theorem 4 gives $S(\lambda,\ell) \leq O\left(\frac{1}{\sqrt{\ell}\lambda}\right)$. Moreover in Cucker and Smale (2002a), Theorem 2 gives $O\left(\frac{\log\ell}{\sqrt{\ell}\lambda^2}\right)$ as it can be seen from Equation (3) at p. 12. Finally our results were recently improved in Smale and Zhou (2004b), where, using again techniques similar to those presented here, a bound of the form $S(\lambda,\ell) \leq O\left(\frac{1}{\sqrt{\ell}\lambda}\right) + O\left(\frac{1}{\ell\lambda^{\frac{3}{2}}}\right)$ is obtained. It is worth noting that in general working on the square root of the error leads to better overall results.

2. We recall that the empirical error is defined as $I_{\mathbf{z}}[f] = \frac{1}{\ell}\sum_{i=1}^{\ell} V(f(x_i), y_i)$.

5.4 Proofs

In this section we collect the proofs of the theorems that we stated in the previous sections. e first now prove the bound on the residual for the Tikhonov regularization.

Proof [of Theorem 2] The idea of the proof is to note that, by triangular inequality, we can write

$$\left| \left\| A f_{\mathbf{z}}^{\lambda} - Pg \right\|_{L^2(X,\nu)} - \left\| A f^{\lambda} - Pg \right\|_{L^2(X,\nu)} \right| \leq \left\| A f_{\mathbf{z}}^{\lambda} - A f^{\lambda} \right\|_{L^2(X,\nu)} \tag{17}$$

so that we can focus on the difference between the discrete and continuous solutions. By a simple algebraic computation we have that

$$\begin{aligned} f_{\mathbf{z}}^{\lambda} - f^{\lambda} &= (A_{\mathbf{x}}^* A_{\mathbf{x}} + \lambda I)^{-1} A_{\mathbf{x}}^* \mathbf{y} - (A^* A + \lambda I)^{-1} A^* g \\ &= [(A_{\mathbf{x}}^* A_{\mathbf{x}} + \lambda I)^{-1} - (A^* A + \lambda I)^{-1}] A_{\mathbf{x}}^* \mathbf{y} + (A^* A + \lambda I)^{-1} (A_{\mathbf{x}}^* \mathbf{y} - A^* g) \\ &= (A^* A + \lambda I)^{-1} (A^* A - A_{\mathbf{x}}^* A_{\mathbf{x}}) (A_{\mathbf{x}}^* A_{\mathbf{x}} + \lambda I)^{-1} A_{\mathbf{x}}^* \mathbf{y} + (A^* A + \lambda I)^{-1} (A_{\mathbf{x}}^* \mathbf{y} - A^* g). \end{aligned} \tag{18}$$

and we see that the relevant quantities for the definition of the noise appear.

We claim that

$$\left\| A(A^* A + \lambda I)^{-1} \right\|_{\mathcal{L}(\mathcal{H})} = \frac{1}{2\sqrt{\lambda}} \tag{19}$$

$$\left\| (A_{\mathbf{x}}^* A_{\mathbf{x}} + \lambda I)^{-1} A_{\mathbf{x}}^* \right\|_{\mathcal{L}(\mathcal{H})} = \frac{1}{2\sqrt{\lambda}}. \tag{20}$$

Indeed, let $A = U|A|$ be the polar decomposition of A. The spectral theorem implies that

$$\begin{aligned} \|A(A^* A + \lambda I)^{-1}\|_{\mathcal{L}(\mathcal{H})} &= \|U|A|(|A|^2 + \lambda I)^{-1}\|_{\mathcal{L}(\mathcal{H})} = \||A|(|A|^2 + \lambda I)^{-1}\|_{\mathcal{L}(\mathcal{H})} \\ &= \sup_{t \in [0, \||A|\|} \frac{t}{t^2 + \lambda}. \end{aligned}$$

A direct computation of the derivative shows that the maximum of $\frac{t}{t^2+\lambda}$ is $\frac{1}{2\sqrt{\lambda}}$ and (19) is proved. Formula (20) follows replacing A with $A_{\mathbf{x}}$.

Last step is to plug Equation (18) into (17) and use Cauchy-Schwartz inequality. Since $\|\mathbf{y}\|_{\mathbf{E}^\ell} \leq M$, (19) and (20) give

$$\left| \|A f_{\mathbf{z}}^{\lambda} - Pg\|_{L^2} - \|A f^{\lambda} - Pg\|_{L^2} \right| \leq \frac{M}{4\lambda} \|A^* A - A_{\mathbf{x}}^* A_{\mathbf{x}}\|_{\mathcal{L}(\mathcal{H})} + \frac{1}{2\sqrt{\lambda}} \|A_{\mathbf{x}}^* \mathbf{y} - A^* g\|_{\mathcal{H}}$$

so that the theorem is proved. ■

The proof of Theorem 2 is a straightforward application of Lemma (8) (see Appendix) .

Proof [Theorem 2] The proof is a simple consequence of estimate (26) applied to the random variables

$$\begin{aligned} \xi_1(x,y) &= y K_x \\ \xi_2(x,y) &= \langle \cdot, K_x \rangle_{\mathcal{H}} K_x = K_x \otimes K_x \end{aligned}$$

where

1. ξ_1 takes value in $\mathcal{H}$, $L_1 = \kappa M$ and $v_1^* = A^* g$, see (21), (23);

2. ξ_2 takes vales in the Hilbert space of Hilbert-Schmidt operators, which can be identified with $\mathcal{H} \otimes \mathcal{H}$, $L_2 = \kappa^2$ and $v_2^* = T$, see (22), (24).

Replacing η with $\eta/2$, (26) gives

$$\begin{aligned} \|A^*g - A_{\mathbf{x}}{}^*\mathbf{y}\|_{\mathcal{H}} \leq \delta_1(\ell,\eta) &= \frac{M\kappa}{2}\psi\left(\frac{8}{\ell}\log\frac{4}{\eta}\right) \\ \|A^*A - A_{\mathbf{x}}{}^*A_{\mathbf{x}}\|_{\mathcal{L}(\mathcal{H})} \leq \delta_2(\ell,\eta) &= \frac{\kappa^2}{2}\psi\left(\frac{8}{\ell}\log\frac{4}{\eta}\right), \end{aligned}$$

respectively, so that the thesis follows. ■

Finally we combine the above results to prove the consistency of the regularized least squares algorithm.

Proof [Theorem 4] Theorem 1 gives

$$\|Af_{\mathbf{z}}^{\lambda} - Pg\|_{L^2(X,\nu)} \leq \left(\frac{1}{2\sqrt{\lambda}}\delta_1 + \frac{M}{4\lambda}\delta_2\right) + \|Af^{\lambda} - Pg\|_{L^2(X,\nu)}.$$

Equation (13) and the estimates for the noise levels δ_1 and δ_2 given by Theorem 2 ensure that

$$\sqrt{I[f_{\mathbf{z}}{}^{\lambda}] - \inf_{f\in\mathcal{H}} I[f]} \leq \left(\frac{M\kappa}{2\sqrt{\lambda}} + \frac{M\kappa^2}{4\lambda}\right)\psi\left(\frac{8}{\ell}\log\frac{4}{\eta}\right) + \left\|Af^{\lambda} - Pg\right\|_{L^2(X,\nu)}$$

and (16) simply follows taking the square of the above inequality. Let now $\lambda = 0(\ell^{-b})$ with $0 < b < \frac{1}{2}$, the consistency of the regularised least squares algorithm is proved by inverting the relation between ε and η and using the result of Proposition (4) (see Appendix). ■

6. Conclusions

In this paper we analyse the connection between the theory of statistical learning and the theory of ill-posed problems. More precisely we show that, considering the quadratic loss function, the problem of finding the best solution $f_{\mathcal{H}}$ for a given hypothesis space $\mathcal{H}$ is a linear inverse problem and that the regularized least squares algorithm is the Tikhonov regularization of the discretized version of the above inverse problem. As a consequence, the consistency of the algorithm is traced back to the well known convergence property of the Tikhonov regularization. A probabilistic estimate of the noise is given based on a elegant concentration inequality in Hilbert spaces.

An open problem is extending the above results to arbitrary loss functions. For other choices of loss functions the problem of finding the best solution gives rise to a non linear ill-posed problem and the theory for this kind of problems is much less developed than the corresponding theory for linear problems. Moreover, since, in general, the expected risk $I[f]$ for arbitrary loss function does not define a metric, the relation between the expected risk and the residual is not clear. Further problems are the choice of the regularization parameter, for example by means of the generalized Morozov principle (Engl et al., 1996) and the extension of our analysis to a wider class of regularization algorithms.

Acknowledgments

We would like to thank M.Bertero, C. De Mol, M. Piana, T. Poggio, S. Smale, G. Talenti and A. Verri for useful discussions and suggestions. This research has been partially funded by the INFM Project MAIA, the FIRB Project ASTAA and the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778.

Appendix A. Technical Results

First, we collect some useful properties of the operators A and $A_{\mathbf{x}}$.

Proposition 6 *The operator A is a Hilbert-Schmidt operator from $\mathcal{H}$ into $L^2(X,\nu)$ and*

$$A^*\phi = \int_X \phi(x) K_x \, d\nu(x), \tag{21}$$

$$A^*A = \int_X \langle \cdot, K_x \rangle_{\mathcal{H}} K_x \, d\nu(x), \tag{22}$$

where $\phi \in L^2(X,\nu)$, the first integral converges in norm and the second one in trace norm.

Proof The proof is standard and we report it for completeness.
Since the elements $f \in \mathcal{H}$ are continuous functions defined on a compact set and ν is a probability measure, then $f \in L^2(X,\nu)$, so that A is a linear operator from $\mathcal{H}$ to $L^2(X,\nu)$. Moreover the Cauchy-Schwartz inequality gives

$$|(Af)(x)| = |\langle f, K_x \rangle_{\mathcal{H}}| \leq \kappa \, \|f\|_{\mathcal{H}},$$

so that $\|Af\|_{L^2(X,\nu)} \leq \kappa \|f\|_{\mathcal{H}}$ and A is bounded.

We now show that A is injective. Let $f \in \mathcal{H}$ and $W = \{x \in X \mid f(x) \neq 0\}$. Assume $Af = 0$, then W is a open set, since f is continuous, and W has null measure, since $(Af)(x) = f(x) = 0$ for ν-almost all $x \in X$. The assumption that ν is not degenerate ensures W be the empty set and, hence, $f(x) = 0$ for all $x \in X$, that is, $f = 0$.

We now prove (21). We first recall the map

$$X \ni x \mapsto K_x \in \mathcal{H}$$

is continuous since $\|K_t - K_x\|_{\mathcal{H}}{}^2 = K(t,t) + K(x,x) - 2K(x,t)$ for all $x, t \in X$, and K is a continuous function. Hence, given $\phi \in L^2(X, \nu)$, the map $x \mapsto \phi K_x$ is measurable from X to $\mathcal{H}$. Moreover, for all $x \in X$,

$$\|\phi(x)K_x\|_{\mathcal{H}} = |\phi(x)|\sqrt{K(x,x)} \leq |\phi(x)|\kappa.$$

Since ν is finite, ϕ is in $L^1(X, \nu)$ and, hence, ϕK_x is integrable, as a vector valued map. Finally, for all $f \in \mathcal{H}$,

$$\int_X \phi(x) \langle K_x, f\rangle_{\mathcal{H}} \, d\nu(x) = \langle \phi, Af\rangle_{L^2(X,\nu)} = \langle A^*\phi, f\rangle_{\mathcal{H}},$$

so, by uniqueness of the integral, Equation (21) holds.

Equations (22) is a consequence of Equation (21) and the fact that the integral commutes with the scalar product.

We now prove that A is a Hilbert-Schmidt operator. Let $(e_n)_{n\in\mathbb{N}}$ be a Hilbert basis of $\mathcal{H}$. Since A^*A is a positive operator and $|\langle K_x, e_n\rangle_{\mathcal{H}}|^2$ is a positive function, by monotone convergence theorem, we have that

$$\begin{aligned}
\operatorname{Tr}(A^*A) &= \sum_n \int_X |\langle e_n, K_x\rangle_{\mathcal{H}}|^2 \, d\nu(x) \\
&= \int_X \sum_n |\langle e_n, K_x\rangle_{\mathcal{H}}|^2 \, d\nu(x) \\
&= \int_X \langle K_x, K_x\rangle_{\mathcal{H}} \, d\nu(x) \\
&= \int_X K(x,x) \, d\nu(x) < \kappa^2
\end{aligned}$$

and the thesis follows. ■

Corollary 7 *The sampling operator $A_{\mathbf{x}} : \mathcal{H} \to \mathbf{E}^\ell$ is a Hilbert-Schmidt operator and*

$$A_{\mathbf{x}}{}^*\mathbf{y} = \frac{1}{\ell}\sum_{i=1}^{\ell} y_i K_{x_i} \tag{23}$$

$$A_{\mathbf{x}}{}^*A_{\mathbf{x}} = \frac{1}{\ell}\sum_{i=1}^{\ell} \langle \cdot, K_{x_i}\rangle_{\mathcal{H}} K_{x_i}. \tag{24}$$

Proof The content of the proposition is a restatement of Proposition 6 and the fact that the integrals reduce to sums. ■

For sake of completeness we report a standard proof on the convergence of the approximation error.

Proof [of Proposition 4] Consider the polar decomposition $A = U|A|$ of A (see, for example, Lang (1993)), where $|A|^2 = A^*A$ is a positive operator on $\mathcal{H}$ and U is a partial isometry such that the projector P on the range of A is $P = UU^*$. Let $dE(t)$ be the spectral measure of $|A|$. Recalling that

$$f^\lambda = (A^*A+\lambda)^{-1}A^*g = (|A|^2+\lambda)^{-1}|A|U^*g$$

the spectral theorem gives

$$\begin{aligned}\left\|Af^\lambda - Pg\right\|^2_{\mathcal{K}} &= \||U|A|(|A|^2+\lambda)^{-1}|A|U^*g - UU^*g\|^2_{\mathcal{K}} = \\ &= \left\|\left(|A|^2\left(|A|^2+\lambda\right)^{-1}-1\right)U^*g\right\|^2_{\mathcal{H}} = \\ &= \int_0^{\||A|\|}\left(\frac{t^2}{t^2+\lambda}-1\right)^2 d\langle E(t)U^*g, U^*g\rangle_{\mathcal{H}}.\end{aligned}$$

Let $r_\lambda(t) = \frac{t^2}{t^2+\lambda} - 1 = -\frac{\lambda}{t^2+\lambda}$, then

$$|r_\lambda(t)| \le 1 \quad \text{and} \quad \lim_{\lambda\to 0^+} r_\lambda(t) = 0 \quad \forall t > 0,$$

so that the dominated convergence theorem gives that

$$\lim_{\lambda\to 0^+}\left\|Af^\lambda - Pg\right\|^2_{\mathcal{K}} = 0.$$

■

Finally, to prove our estimate of the noise we need the following probabilistic inequality due to Pinelis and Sakhanenko (1985). (See Yurinsky, 1995, for the version presented int he following.)

Lemma 8 *Let Z be a probability space and ξ be a random variable on X taking value in a real separable Hilbert space $\mathcal{H}$. Assume that the expectation value $v^* = \mathbb{E}[\xi]$ exists and there are two positive constants H and σ such that*

$$\begin{aligned}\|\xi(z) - v^*\|_{\mathcal{H}} &\le H \quad \text{a.}s \\ \mathbb{E}[\|\xi - v^*\|^2_{\mathcal{H}}] &\le \sigma^2.\end{aligned}$$

If z_i are drawn i.i.d. from Z, then, with probability greater than $1-\eta$,

$$\left\|\frac{1}{\ell}\sum_{i=1}^{\ell}\xi(z_i) - v^*\right\| \le \frac{\sigma^2}{H}g\left(\frac{2H^2}{\ell\sigma^2}\log\frac{2}{\eta}\right) = \delta(\ell,\eta) \tag{25}$$

where $g(t) = \frac{1}{2}(t+\sqrt{t^2+4t})$. In particular

$$\delta(\ell,\eta) = \sigma\sqrt{\frac{2}{\ell}\log\frac{2}{\eta}} + o\left(\sqrt{\frac{1}{\ell}\log\frac{2}{\eta}}\right)$$

Proof It is just a testament to Th. 3.3.4 of Yurinsky (1995), see also Steinwart (2003). Consider the set of independent random variables with zero mean $\xi_i = \xi(z_i) - v^*$ defined on the probability space Z^ℓ. Since, ξ_i are identically distributed, for all $m \geq 2$ it holds

$$\sum_{i=1}^{\ell} \mathbb{E}[\|\xi_i\|_{\mathcal{H}}^m] \leq \frac{1}{2} m! B^2 H^{m-2},$$

with the choice $B^2 = \ell\sigma^2$. So Th. 3.3.4 of Yurinsky (1995) can be applied and it ensures

$$\mathbf{P}\left[\frac{1}{\ell}\left\|\sum_{i=1}^{\ell}(\xi(z_i) - v^*)\right\| \geq \frac{xB}{\ell}\right] \leq 2\exp\left(-\frac{x^2}{2(1 + xHB^{-1})}\right)$$

for all $x \geq 0$. Letting $\delta = \frac{xB}{\ell}$, we get the equation

$$\frac{1}{2}(\frac{\ell\delta}{B})^2 \frac{1}{1 + \ell\delta HB^{-2}} = \frac{\ell\delta^2\sigma^{-2}}{2(1 + \delta H\sigma^{-2})} = \log\frac{2}{\eta},$$

since $B^2 = \ell\sigma^2$. Defining $t = \delta H\sigma^{-2}$

$$\frac{\ell\sigma^2}{2H^2}\frac{t^2}{1+t} = \log\frac{2}{\eta}.$$

The thesis follows, observing that g is the inverse of $\frac{t^2}{1+t}$ and that $g(t) = \sqrt{t} + o(\sqrt{t})$. ∎

We notice that, if ξ is bounded by L almost surely, then v^* exists and we can choose $H = 2L$ and $\sigma = L$ so that

$$\delta(\ell, \eta) = \frac{L}{2} g\left(\frac{8}{\ell}\log\frac{2}{\eta}\right). \tag{26}$$

In Smale and Y. (2004) a better estimate is given, replacing the function $\frac{t^2}{1+t}$ with $t\log(1+t)$, anyway the asymptotic rate is the same.

References

N. Alon, S. Ben-David, N. Cesa-Bianchi, and D. Haussler. Scale sensitive dimensions, uniform convergence, and learnability. *Journal of the ACM*, 44:615–631, 1997.

A. Arbib, M. *The Handbook of Brain Theory and Neural Networks*. The MIT Press, Cambridge, MA, 1995.

N. Aronszajn. Theory of reproducing kernels. *Trans. Amer. Math. Soc.*, 68:337–404, 1950.

P. L. Bartlett and S. Mendelson. Rademacher and Gaussian complexities. *Journal of Machine Learning Research*, 3:463–482, 2002.

M. Bertero, C. De Mol, and E. R. Pike. Linear inverse problems with discrete data. I. General formulation and singular system analysis. *Inverse Problems*, 1(4):301–330, 1985.

M. Bertero, C. De Mol, and E. R. Pike. Linear inverse problems with discrete data. II. Stability and regularisation. *Inverse Problems*, 4(3):573–594, 1988.

O. Bousquet and A. Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2:499–526, 2002.

D. Chen, Q. Wu, Y. Ying, and D. Zhou. Support vector machine soft margin classifiers: Error analysis. *Journal of Machine Learning research*, 5:1143–1175, 2004.

F. Cucker and S. Smale. Best choices for regularization parameters in learning theory: on the bias-variance problem. *Foundations of Computationals Mathematics*, 2:413–428, 2002a.

F. Cucker and S. Smale. On the mathematical foundations of learning. *Bull. Amer. Math. Soc. (N.S.)*, 39(1):1–49 (electronic), 2002b.

E. De Vito, A. Caponnetto, and L. Rosasco. Model selection for regularized least-squares algorithm in learning theory. *to be published in Foundations of Computational Mathematics*, 2004.

L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Number 31 in Applications of mathematics. Springer, New York, 1996.

H. W. Engl, M. Hanke, and A. Neubauer. *Regularization of inverse problems*, volume 375 of *Mathematics and its Applications*. Kluwer Academic Publishers Group, Dordrecht, 1996.

T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. *Adv. Comp. Math.*, 13:1–50, 2000.

L. Fine, T. *Feedforward Neural Network Methodology*. Springer-Verlag, 1999.

C. W. Groetsch. *The theory of Tikhonov regularization for Fredholm equations of the first kind*, volume 105 of *Research Notes in Mathematics*. Pitman (Advanced Publishing Program), Boston, MA, 1984.

M. Györfi, L.and Kohler, A. Krzyzak, and H. Walk. *A Distribution-free Theory of Non-parametric Regression*. Springer Series in Statistics, New York, 1996, 1996.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, New York, 2001.

J. Kaipio and E. Somersalo. *Statistical and Computational Inverse Problems*. Springer, 2005.

V. Kecman. *Learning and Soft Computing*. The MIT Press, Cambridge, MA, 2001.

V. Kurkova. Learning from data as an inverse problem. In J. Antoch, editor, *COMPSTAT2004*, pages 1377–1384. Springer-Verlag, 2004.

S. Lang. *Real and Functional Analysis*. Springer, New York, 1993.

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. Technical Report CBCL Paper 223, Massachusetts Institute of Technology, january revision 2004.

S. Mukherjee, R. Rifkin, and T. Poggio. Regression and classification with regularization. *Lectures Notes in Statistics: Nonlinear Estimation and Classification, Proceedings from MSRI Workshop*, 171:107–124, 2002.

P. Niyogi and F. Girosi. Generalization bounds for function approximation from scattered noisy data. *Adv. Comput. Math.*, 10:51–80, 1999.

C.S. Ong and S. Canu. Regularization by early stopping. Technical report, Computer Sciences Laboratory, RSISE, ANU, 2004.

I. F. Pinelis and A. I. Sakhanenko. Remarks on inequalities for probabilities of large deviations. *Theory Probab. Appl.*, 30(1):143–148, 1985. ISSN 0040-361X.

T. Poggio and F. Girosi. A theory of networks for approximation and learning. In C. Lau, editor, *Foundation of Neural Networks*, pages 91–106. IEEE Press, Piscataway, N.J., 1992.

C. Rudin. A different type of convergence for statistical learning algorithms. Technical report, Program in Applied and Computational Mathematics Princeton University, 2004.

B. Schölkopf and A.J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002. URL `http://www.learning-with-kernels.org`.

L. Schwartz. Sous-espaces hilbertiens d'espaces vectoriels topologiques et noyaux associés (noyaux reproduisants). *J. Analyse Math.*, 13:115–256, 1964.

C. Scovel and I. Steinwart. Fast rates support vector machines. *submitted to Annals of Statistics*, 2003.

S. Smale and Yao Y. Online learning algorithms. Technical report, Toyota Technological Institute, Chicago, 2004.

S. Smale and D. Zhou. Estimating the approximation error in learning theory. *Analysis and Applications*, 1(1):1–25, 2003.

S. Smale and D. Zhou. Shannon sampling and function reconstruction from point values. *Bull. Amer. Math. Soc. (N.S.)*, 41(3):279–305 (electronic), 2004a.

S. Smale and D. Zhou. Shannon sampling II : Connections to learning theory. *preprint*, 2004b.

I. Steinwart. Sparseness of support vector machines. *Journal of Machine Learning Research*, 4: 1071–1105, 2003.

I. Steinwart. Consistency of support vector machines and other regularized kernel machines. *accepted on IEEE Transaction on Information Theory*, 2004.

A. N. Tikhonov, A. V. Goncharsky, V. V. Stepanov, and A. G. Yagola. *Numerical methods for the solution of ill-posed problems*, volume 328 of *Mathematics and its Applications*. Kluwer Academic Publishers Group, Dordrecht, 1995. Translated from the 1990 Russian original by R. A. M. Hoksbergen and revised by the authors.

A.N. Tikhonov and V.Y. Arsenin. *Solutions of Ill Posed Problems*. W. H. Winston, Washington, D.C., 1977.

V. N. Vapnik. *Statistical learning theory*. Adaptive and Learning Systems for Signal Processing, Communications, and Control. John Wiley & Sons Inc., New York, 1998. A Wiley-Interscience Publication.

G. Wahba. *Spline models for observational data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1990.

V. Yurinsky. *Sums and Gaussian vectors*, volume 1617 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1995.

T. Zhang. Leave-one-out bounds for kernel methods. *Neural Computation*, 13:1397–1437, 2003.

Journal of Machine Learning Research 6 (2005) 905–936 Submitted 11/04; Revised 4/05; Published 5/05

Loopy Belief Propagation: Convergence and Effects of Message Errors

Alexander T. Ihler IHLER@ALUM.MIT.EDU
Donald Bren School of Information and Computer Science
University of California, Irvine
Irvine, CA 92697 USA

John W. Fisher III FISHER@CSAIL.MIT.EDU

Alan S. Willsky WILLSKY@MIT.EDU
Massachusetts Institute of Technology
Cambridge, MA 02139, USA

Editor: David Maxwell Chickering

Abstract

Belief propagation (BP) is an increasingly popular method of performing approximate inference on arbitrary graphical models. At times, even further approximations are required, whether due to quantization of the messages or model parameters, from other simplified message or model representations, or from stochastic approximation methods. The introduction of such errors into the BP message computations has the potential to affect the solution obtained adversely. We analyze the effect resulting from message approximation under two particular measures of error, and show bounds on the accumulation of errors in the system. This analysis leads to convergence conditions for traditional BP message passing, and both strict bounds and estimates of the resulting error in systems of approximate BP message passing.

Keywords: belief propagation, sum-product, convergence, approximate inference, quantization

1. Introduction

Graphical models and message-passing algorithms defined on graphs comprise a growing field of research. In particular, the *belief propagation* (or sum-product) algorithm has become a popular means of solving inference problems exactly or approximately. One part of its appeal lies in its optimality for tree-structured graphical models (models which contain no loops). However, its is also widely applied to graphical models with cycles. In these cases it may not converge, and if it does its solution is approximate; however in practice these approximations are often good. Recently, some additional justifications for loopy belief propagation have been developed, including a handful of convergence results for graphs with cycles (Weiss, 2000; Tatikonda and Jordan, 2002; Heskes, 2004).

The approximate nature of loopy belief propagation is often a more than acceptable price for performing efficient inference; in fact, it is sometimes desirable to make *additional* approximations. There may be a number of reasons for this—for example, when exact message representation is computationally intractable, the messages may be approximated stochastically (Koller et al., 1999) or deterministically by discarding low-likelihood states (Coughlan and Ferreira, 2002). For belief propagation involving continuous, non-Gaussian potentials, some form of approximation is required to obtain a finite parameterization for the messages (Sudderth et al., 2003; Isard, 2003; Minka, 2001). Additionally, simplification of complex graphical models through edge removal, quantization of the potential functions, or other forms of distributional approximation may be considered in this framework. Finally, one may wish to approximate the messages and reduce their representation size for another reason—to decrease the communications required for distributed inference applications. In distributed message passing, one may approximate the transmitted message to reduce its representational cost (Ihler et al., 2004a), or discard it entirely if it is deemed "sufficiently similar" to the previously sent version (Chen et al., 2004). Through such means one may significantly reduce the amount of communications required.

Given that message approximation may be desirable, we would like to know what effect the errors introduced have on our overall solution. In order to characterize the approximation effects in graphs with cycles, we analyze the deviation from the solution given by "exact" loopy belief propagation (*not*, as is typically considered, the deviation of loopy BP from the true marginal distributions). As a byproduct of this analysis, we also obtain some results on the convergence of loopy belief propagation.

We begin in Section 2 by briefly reviewing the relevant details of graphical models and belief propagation. Section 4 then examines the consequences of measuring a message error by its dynamic range. In particular, we explain the utility of this measure and its behavior with respect to the operations of belief propagation. This allows us to derive conditions for the convergence of traditional loopy belief propagation, and bounds on the distance between any pair of BP fixed points (Sections 5.1–5.2), and these results are easily extended to many approximate forms of BP (Section 5.3). If the errors introduced are independent, as is a typical assumption in, for example, quantization analysis (Gersho and Gray, 1991; Willsky, 1978), tighter estimates of the resulting error can be obtained (Section 5.5).

It is also instructive to examine other measures of message error, in particular ones which emphasize more average-case (as opposed to pointwise or worst-case) differences. To this end, we consider a KL-divergence based measure in Section 6. While the analysis of the KL-divergence measure is considerably more difficult and does not lead to strict guarantees, it serves to give some intuition into the behavior of perturbed BP under an average-case difference measure.

2. Graphical Models

Graphical models (Lauritzen, 1996; Kschischang et al., 2001) provide a convenient means of representing conditional independence relations among large numbers of random variables. Specifically, each node s in an undirected graph is associated with a random variable x_s, while the set of edges $\mathcal{E}$ is used to describe the conditional dependency structure of the variables through *graph separation*. If every path between two sets A and C passes through another set B [see Figure 1(a)], the sets of variables $\mathbf{x}_A = \{x_s : s \in A\}$ and $\mathbf{x}_C = \{x_s : s \in C\}$ must be independent given the values of $\mathbf{x}_B = \{x_s : s \in B\}$. Thus, the distribution $p(\mathbf{x}_A, \mathbf{x}_B, \mathbf{x}_C)$ can be written in the form $p(\mathbf{x}_B)p(\mathbf{x}_A|\mathbf{x}_B)p(\mathbf{x}_C|\mathbf{x}_B)$.

It can be shown that a distribution $p(\mathbf{x})$ is consistent with (i.e., satisfies the conditional independence relations specified by) an undirected graph if it factors into a product of potential functions ψ defined on the cliques (fully-connected subsets) of the graph, and that the converse is also true if $p(\mathbf{x})$ is strictly positive (Clifford, 1990). For convenience, we confine our attention to graphical

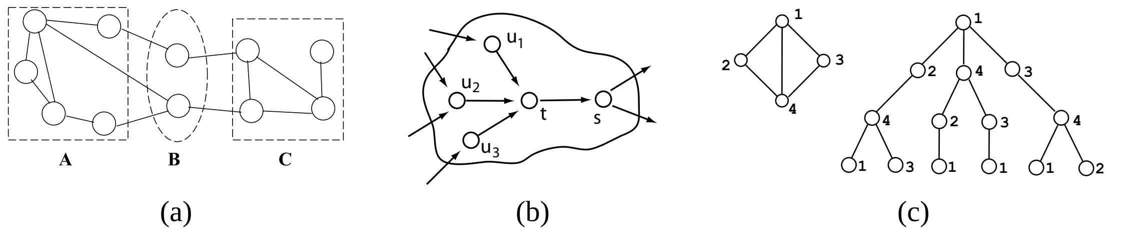


Figure 1: (a) Graphical models describe statistical dependency; here, the sets A and C are independent given B. (b) BP propagates information from t and its neighbors u_i is to s by a simple message-passing procedure; this procedure is exact on a tree, but approximate in graphs with cycles. (c) For a graph with cycles, one may show an equivalence between n iterations of loopy BP and the depth-n computation tree [shown here for $n = 3$ and rooted at node 1; example from Tatikonda and Jordan (2002)].

models with at most pairwise potential functions, so that the distribution factors according to

$$p(\mathbf{x}) = \prod_{(s,t)\in\mathcal{E}} \psi_{st}(x_s, x_t) \prod_s \psi_s(x_s).$$

This is a typical assumption for belief propagation, and can be taken without incurring any real loss of generality since a graphical model with higher-order potential functions may always be converted to a graphical model with only pairwise potential functions through a process of variable augmentation, though this may also increase the nodes' state dimension undesirably; see, for example, Weiss (2000).

2.1 Belief Propagation

The goal of belief propagation (BP) (Pearl, 1988), also called the sum-product algorithm, is to compute the marginal distribution $p(x_t)$ at each node t. BP takes the form of a message-passing algorithm between nodes, expressed in terms of an update to the outgoing message at iteration i from each node t to each neighbor s in terms of the previous iteration's incoming messages from t's neighbors Γ_t [see Figure 1(b)],

$$m^i_{ts}(x_s) \propto \int \psi_{ts}(x_t, x_s)\psi_t(x_t) \prod_{u\in\Gamma_t\setminus s} m^{i-1}_{ut}(x_t)dx_t. \tag{1}$$

Typically each message is normalized so as to integrate to unity (and we assume that such normalization is possible). For discrete-valued random variables, of course, the integral is replaced by a summation. At any iteration, one may calculate the *belief* at node t by

$$M^i_t(x_t) \propto \psi_t(x_t) \prod_{u\in\Gamma_t} m^i_{ut}(x_t). \tag{2}$$

For tree-structured graphical models, belief propagation can be used to efficiently perform exact marginalization. Specifically, the iteration (1) converges in a finite number of iterations (at most the length of the longest path in the graph), after which the belief (2) equals the correct marginal $p(x_t)$. However, as observed by Pearl (1988), one may also apply belief propagation to arbitrary graphical models by following the same *local* message passing rules at each node and ignoring the presence of cycles in the graph; this procedure is typically referred to as "loopy" BP.

For loopy BP, the sequence of messages defined by (1) is not guaranteed to converge to a fixed point after any number of iterations. Under relatively mild conditions, one may guarantee the existence of fixed points (Yedidia et al., 2004). However, they may not be unique, nor are the results exact [the belief M_t^i does not converge to the true marginal $p(x_t)$]. In practice however the procedure often arrives at a reasonable set of approximations to the correct marginal distributions.

2.2 Computation Trees

It is sometimes convenient to think of loopy BP in terms of its *computation tree*. Tatikonda and Jordan (2002) showed that the effect of n iterations of loopy BP at any particular node s is equivalent to exact inference on a tree-structured 'unrolling" of the graph from s. A small graph, and its associated 4-level computation tree rooted at node 1, are shown in Figure 1(c).

The computation tree with depth n consists of all length-n paths emanating from s in the original graph which do not immediately backtrack (though they may eventually repeat nodes).[1] We draw the computation tree as consisting of a number of *levels*, corresponding to each node in the tree's distance from the root, with the root node at level 0 and the leaf nodes at level n. Each level may contain multiple replicas of each node, and thus there are potentially many replicas of each node in the graph. The root node s has replicas of all neighbors Γ_s in the original graph as children, while all other nodes have replicas of all neighbors except their parent node as children.

Each edge in the computation tree corresponds to both an edge in the original graph *and* an iteration in the BP message-passing algorithm. Specifically, assume an equivalent initialization of both the loopy graph and computation tree—i.e., the initial messages m_{ut}^0 in the loopy graph are taken as inputs to the leaf nodes. Then, the upward messages from level n to level $n-1$ match the messages m_{ut}^1 in the first iteration of loopy BP, and more generally, a upward message m_{ut}^i on the computation tree which originates from a node u on level $n-i+1$ to its parent node t on level $n-i$ is identical to the message from node u to node t in the i^{th} iteration of loopy BP (out of n total iterations) on the original graph. Thus, the incoming messages to the root node (level 0) correspond to the messages in the n^{th} iteration of loopy BP.

2.3 Message Approximations

Let us now consider the concept of *approximate* BP messages. We begin by assuming that the "true" messages $m_{ts}(x_s)$ are some fixed point of BP, so that $m_{ts}^i = m_{ts}^{i+1}$. We may ask what happens when these messages are perturbed by some (perhaps small) error function $e_{ts}(x_s)$. Although there are certainly other possibilities, the fact that BP messages are combined by taking their product makes it natural to consider multiplicative message deviations (or additive in the log-domain):

$$\hat{m}_{ts}^i(x_s) = m_{ts}(x_s)e_{ts}^i(x_s).$$

To facilitate our analysis, we split the message update operation (1) into two parts. In the first, we focus on the message *products*

$$\hat{M}_{ts}^i(x_t) \propto \psi_t(x_t) \prod_{u\in\Gamma_t\setminus s} \hat{m}_{ut}^i(x_t) \qquad\qquad \hat{M}_t^i(x_t) \propto \psi_t(x_t) \prod_{u\in\Gamma_t} \hat{m}_{ut}^i(x_t) \tag{3}$$

1. Thus in Figure 1(c), the computation tree includes the sequence $1-2-4-1$, but not the sequence $1-2-4-2$.

where the proportionality constant is chosen to normalize $\hat{M}$. The second operation, then, is the message *convolution*

$$\hat{m}_{ts}^{i+1}(x_s) \propto \int \psi_{ts}(x_s, x_t) \hat{M}_{ts}^{i}(x_t) dx_t \tag{4}$$

where again $\hat{M}$ is a normalized message or product of messages.

In this paper, we use the convention that lowercase quantities ($m_{ts}, e_{ts}, \ldots$) refer to messages and message errors, while uppercase ones ($M_{ts}, E_{ts}, M_t, \ldots$) refer to their products—at node t, the product of all incoming messages and the local potential is denoted $M_t(x_t)$, its approximation $\hat{M}_t(x_t) = M_t(x_t)E_t(x_t)$, with similar definitions for M_{ts}, $\hat{M}_{ts}$, and E_{ts}.

3. Overview of Results

To orient the reader, we lay out the order and general results which are obtained in this paper. We begin in Section 4 by examining a *dynamic range* measure $d\left(e\right)$ of the variability of a message error $e(x)$ (or more generally of any function) and show how this measure behaves with respect to the BP equations (1) and (2). Specifically, we show in Section 4.2 that the measure $\log d\left(e\right)$ is sub-additive with respect to the product operation (3), and contractive with respect to the convolution operation (4).

Applying these results to traditional belief propagation results in a new sufficient condition for BP convergence (Section 5.1), specifically

$$\max_{s,t} \sum_{u \in \Gamma_t \setminus s} \frac{d\left(\psi_{ut}\right)^2 - 1}{d\left(\psi_{ut}\right)^2 + 1} < 1; \tag{5}$$

and this condition may be further improved in many cases. The condition (5) can be shown to be slightly stronger than the sufficient condition given in Tatikonda and Jordan (2002), and empirically appears to be stronger than that of Heskes (2004). In experiments, the condition appears to be tight (exactly predicting uniqueness or non-uniqueness of fixed points) for at least some problems, such as binary–valued random variables with attractive potentials. More importantly, however, the *method* in which it is derived allows us to generalize to many other situations:

1. Using the same methodology, we may demonstrate that any two BP fixed points must be within a ball of a calculable diameter; the condition (5) is equivalent to this diameter being zero (Section 5.2).

2. Both the diameter of the bounding ball and the convergence criterion (5) are easily improved for graphical models with irregular geometry or potential strengths, leading to better conditions on graphs which are more “tree-like” (Section 5.3).

3. The same analysis may also be applied to the case of quantized or otherwise approximated messages and models (potential functions), yielding bounds on the resulting error (Section 5.4).

4. If we regard the message errors as a stochastic process, a similar analysis with a few additional, intuitive assumptions gives alternate, tighter estimates (though not necessarily bounds) of performance (Section 5.5).

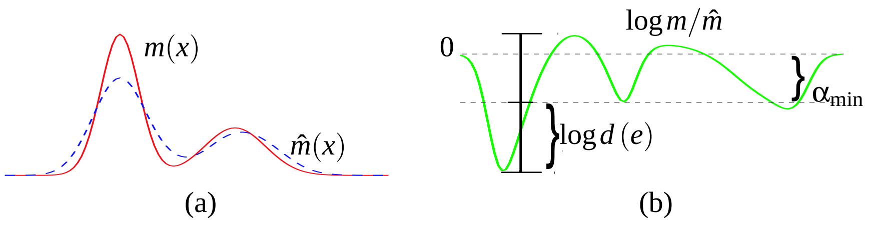


Figure 2: (a) A message $m(x)$ and an example approximation $\hat{m}(x)$; (b) their log-ratio $\log m(x)/\hat{m}(x)$, and the error measure $\log d\,(e)$.

Finally, in Section 6 we perform the same analysis for a less strict measure of message error [i.e., disagreement between a message $m(x)$ and its approximation $\hat{m}(x)$], namely the Kullback-Leibler divergence. This analysis shows that, while failing to provide strict bounds in several key ways, one is still able to obtain some intuition into the behavior of approximate message passing under an average-case difference measure.

In the next few sections, we first describe the dynamic range measure and discuss some of its salient properties (Section 4). We then apply these properties to analyze the behavior of loopy belief propagation (Section 5). Almost all proofs are given in an in-line fashion, as they frequently serve to give intuition into the method and meaning of each result.

4. Dynamic Range Measure

In order to discuss the effects and propagation of errors, we first require a measure of the difference between two messages. In this section, we examine the following measure on $e_{ts}(x_s)$: let $d\,(e_{ts})$ denote the function's *dynamic range*,[2] specifically

$$d\,(e_{ts}) = \sup_{a,b} \sqrt{e_{ts}(a)/e_{ts}(b)}. \tag{6}$$

Then, we have that $m_{ts} \equiv \hat{m}_{ts}$ (i.e., the pointwise equality condition $m_{ts}(x) = \hat{m}_{ts}(x) \forall x$) if and only if $\log d\,(e_{ts}) = 0$. Figure 2 shows an example of $m(x)$ and $\hat{m}(x)$ along with their associated error $e(x)$.

4.1 Motivation

We begin with a brief motivation for this choice of error measure. It has a number of desirable features; for example, it is directly related to the pointwise log error between the two distributions.

Lemma 1. *The dynamic range measure* (6) *may be equivalently defined by*

$$\log d\,(e_{ts}) = \inf_{\alpha} \sup_{x} |\log \alpha m_{ts}(x) - \log \hat{m}_{ts}(x)| = \inf_{\alpha} \sup_{x} |\log \alpha - \log e_{ts}(x)|.$$

Proof. The minimum is given by $\log\alpha = \frac{1}{2}(\sup_a \log e_{ts}(a) + \inf_b \log e_{ts}(b))$, and thus the right-hand side is equal to $\frac{1}{2}(\sup_a \log e_{ts}(a) - \inf_b \log e_{ts}(b))$, or $\frac{1}{2}(\sup_{a,b} \log e_{ts}(a)/e_{ts}(b))$, which by definition is $\log d\,(e_{ts})$. $\square$

2. This measure has also been independently investigated to provide a stability analysis for the max-product algorithm in Bayes' nets (acyclic, directed graphical models) (Chan and Darwiche, 2005). While similar in some ways, the analysis for acyclic graphs is considerably simpler; loopy graphs require demonstrating a rate of contraction, which we show is possible for the sum-product algorithm (Theorem 8).

The scalar α serves the purpose of "zero-centering" the function $\log e_{ts}(x)$ and making the measure invariant to simple rescaling. This invariance reflects the fact that the scale factor for BP messages is essentially arbitrary, defining a class of equivalent messages. Although the scale factor cannot be completely ignored, it takes on the role of a nuisance parameter. The inclusion of α in the definition of Lemma 1 acts to select particular elements of the equivalence classes (with respect to rescaling) from which to measure distance—specifically, choosing the closest such messages in a log-error sense. The log-error, dynamic range, and the minimizing α are depicted in Figure 2.

Lemma 1 allows the dynamic range measure to be related directly to an approximation error in the log-domain when both messages are normalized to integrate to unity, using the following theorem:

Theorem 2. *The dynamic range measure can be used to bound the log-approximation error:*

$$|\log m_{ts}(x) - \log \hat{m}_{ts}(x)| \le 2 \log d\,(e_{ts}) \qquad \forall x.$$

Proof. We first consider the magnitude of $\log \alpha$:

$$\begin{aligned}
\forall x, \qquad & \left| \log \frac{\alpha m_{ts}(x)}{\hat{m}_{ts}(x)} \right| \le \log d\,(e_{ts}) \\
\Rightarrow \qquad & \frac{1}{d\,(e_{ts})} \le \frac{\alpha m_{ts}(x)}{\hat{m}_{ts}(x)} \le d\,(e_{ts}) \\
\Rightarrow \qquad & \int \hat{m}_{ts}(x) dx \frac{1}{d\,(e_{ts})} \le \alpha \int m_{ts}(x) dx \le \int \hat{m}_{ts}(x) dx d\,(e_{ts})
\end{aligned}$$

and since the messages are normalized, $|\log \alpha| \le \log d\,(e_{ts})$. Then by the triangle inequality,

$$|\log m_{ts}(x) - \log \hat{m}_{ts}(x)| \le |\log \alpha m_{ts}(x) - \log \hat{m}_{ts}(x)| + |\log \alpha| \le 2 \log d\,(e_{ts}). \qquad \square$$

In this light, our analysis of message approximation (Section 5.4) may be equivalently regarded as a statement about the required quantization level for an accurate implementation of loopy belief propagation. Interestingly, it may also be related to a floating-point precision on $m_{ts}(x)$.

Lemma 3. *Let $\hat{m}_{ts}(x)$ be an F-bit mantissa floating-point approximation to $m_{ts}(x)$. Then, $\log d\,(e_{ts}) \le 2^{-F} + O(2^{-2F})$.*

Proof. For an F-bit mantissa, we have $|m_{ts}(x) - \hat{m}_{ts}(x)| < 2^{-F} \cdot 2^{\lfloor \log_2 m_{ts}(x) \rfloor} \le 2^{-F} \cdot m_{ts}(x)$. Then, using the Taylor expansion of $\log\left[1 + (\frac{\hat{m}}{m} - 1)\right] \approx (\frac{\hat{m}}{m} - 1)$ we have that

$$\begin{aligned}
\log d\,(e_{ts}) &\le \sup_x \left| \log \frac{\hat{m}(x)}{m(x)} \right| \\
&\le \sup_x \frac{\hat{m}(x) - m(x)}{m(x)} + O\left(\left(\sup_x \frac{\hat{m}(x) - m(x)}{m(x)} \right)^2 \right) \\
&\le 2^{-F} + O\left(2^{-2F}\right). \qquad \square
\end{aligned}$$

Thus our measure of error is, to first order, similar to the typical measure of precision in floating-point implementations of belief propagation on microprocessors. We may also relate $d\,(e)$ to other measures of interest, such as the Kullback-Leibler (KL) divergence.

Lemma 4. *The KL-divergence satisfies the inequality* $D(m_{ts} \| \hat{m}_{ts}) \leq 2 \log d\left(e_{ts}\right)$

Proof. By Theorem 2, we have

$$D(m_{ts} \| \hat{m}_{ts}) = \int m_{ts}(x) \log \frac{m_{ts}(x)}{\hat{m}_{ts}(x)} dx \leq \int m_{ts}(x) \left(2 \log d\left(e_{ts}\right)\right) dx = 2 \log d\left(e_{ts}\right). \qquad \square$$

Finally, a bound on the dynamic range or the absolute log-error can also be used to develop confidence intervals for the maximum and median of the distribution.

Lemma 5. *Let* $\hat{m}(x)$ *be an approximation of* $m(x)$ *with* $\log d\left(\hat{m}/m\right) \leq \varepsilon$*, so that*

$$\hat{m}^{+}(x) = \exp(2\varepsilon)\hat{m}(x) \qquad\qquad \hat{m}^{-}(x) = \exp(-2\varepsilon)\hat{m}(x)$$

are upper and lower pointwise bounds on $m(x)$*, respectively. Then we have a confidence region on the maximum of* $m(x)$ *given by*

$$\arg\max_x m(x) \in \{x : \hat{m}^{+}(x) \geq \max_y \hat{m}^{-}(y)\}$$

and an upper bound μ *on the median of* $m(x)$*, i.e.,*

$$\int_{-\infty}^{\mu} m(x) \geq \int_{\mu}^{\infty} m(x) \qquad \text{where} \qquad \int_{-\infty}^{\mu} \hat{m}^{-}(x) = \int_{\mu}^{\infty} \hat{m}^{+}(x)$$

with a similar lower bound.

Proof. The definitions of $\hat{m}^{+}$ and $\hat{m}^{-}$ follow from Theorem 2. Given these bounds, the maximum value of $m(x)$ must be larger than the maximum value of $\hat{m}^{-}(x)$, and this is only possible at locations x for which $\hat{m}^{+}(x)$ is also greater than the maximum of $\hat{m}^{-}$. Similarly, the left integral of $m(x)$ ($-\infty$ to μ) must be larger than the integral of $\hat{m}^{-}(x)$, while the right integral (μ to ∞) must be smaller than for $\hat{m}^{+}(x)$. Thus the median of $m(x)$ must be less than μ. $\square$

These bounds and confidence intervals are illustrated in Figure 3: given the approximate message $\hat{m}$ (solid black), a bound on the error yields $\hat{m}^{+}(x)$ and $\hat{m}^{-}(x)$ (dotted lines), which yield confidence regions on the maximum and median values of $m(x)$.

4.2 Additivity and Error Contraction

We now turn to the properties of our dynamic range measure with respect to the operations of belief propagation. First, we consider the error resulting from taking the product (3) of a number of incoming approximate messages.

Theorem 6. *The log of the dynamic range measure is sub-additive:*

$$\log d\left(E_{ts}^{i}\right) \leq \sum_{u \in \Gamma_t \setminus s} \log d\left(e_{ut}^{i}\right) \qquad\qquad \log d\left(E_{t}^{i}\right) \leq \sum_{u \in \Gamma_t} \log d\left(e_{ut}^{i}\right).$$

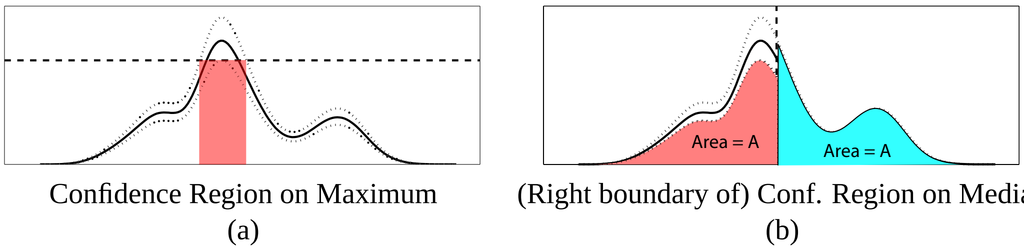


Figure 3: Using the error measure (6) to find confidence regions on maximum and median locations of a distribution. The distribution estimate $\hat{m}(x)$ is shown in solid black, with $|\log m(x)/\hat{m}(x)| \leq \frac{1}{4}$ bounds shown as dotted lines. Then, the maximum value of $m(x)$ must lie above the shaded region, and the median value is less than the dashed vertical line; a similar computation gives a lower bound.

Proof. We show the left-hand sub-additivity statement; the right follows from a similar argument. By definition, we have

$$\log d\left(E^i_{ts}\right) = \log d\left(\hat{M}^i_{ts}/M^i_{ts}\right) = \frac{1}{2}\log \sup_{a,b} \prod e^i_{ut}(a)/\prod e^i_{ut}(b).$$

Increasing the number of degrees of freedom gives

$$\leq \frac{1}{2}\log \prod \sup_{a_u,b_u} e^i_{ut}(a_u)/e^i_{ut}(b_u) = \sum \log d\left(e^i_{ut}(x)\right). \qquad \square$$

Theorem 6 allows us to bound the error resulting from a combination of the incoming approximations from two different neighbors of the node t. It is also important that $\log d(e)$ satisfy the triangle inequality, so that the application of two successive approximations results in an error which is bounded by the sum of their respective errors.

Theorem 7. *The log of the dynamic range measure satisfies the triangle inequality:*

$$\log d(e_1 e_2) \leq \log d(e_1) + \log d(e_2).$$

Proof. This follows from the same argument as Theorem 6. $\square$

We may also derive a minimum rate of contraction occurring with the convolution operation (4). We characterize the strength of the potential ψ_{ts} by extending the definition of the dynamic range measure:

$$d(\psi_{ts})^2 = \sup_{a,b,c,d} \frac{\psi_{ts}(a,b)}{\psi_{ts}(c,d)}. \tag{7}$$

When this quantity is finite, it represents a minimum rate of *mixing* for the potential, and thus causes a contraction on the error. This fact is exhibited in the following theorem.

Theorem 8. *When $d(\psi_{ts})$ is finite, the dynamic range measure satisfies a rate of contraction:*

$$d\left(e^{i+1}_{ts}\right) \leq \frac{d(\psi_{ts})^2 d\left(E^i_{ts}\right) + 1}{d(\psi_{ts})^2 + d\left(E^i_{ts}\right)}. \tag{8}$$

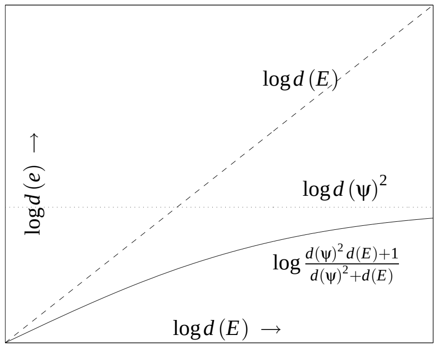


Figure 4: Three bounds on the error output $d(e)$ as a function of the error on the product of incoming messages $d(E)$.

Proof. See Appendix A. □

Two limits are of interest. First, if we examine the limit as the potential strength $d(\psi)$ grows, we see that the error cannot increase due to convolution with the pairwise potential ψ. Similarly, if the potential strength is finite, the outgoing error cannot be arbitrarily large (independent of the size of the incoming error).

Corollary 9. *The outgoing message error $d(e_{ts})$ is bounded by*

$$d\left(e_{ts}^{i+1}\right) \le d\left(E_{ts}^{i}\right) \qquad\qquad d\left(e_{ts}^{i+1}\right) \le d(\psi_{ts})^2 .$$

Proof. Let $d(\psi_{ts})$ or $d\left(E_{ts}^{i}\right)$ tend to infinity in Theorem 8. □

The contractive bound (8) is shown in Figure 4, along with the two simpler bounds of Corollary 9, shown as straight lines. Moreover, we may evaluate the asymptotic behavior by considering the derivative

$$\frac{\partial}{\partial d(E)} \left. \frac{d(\psi)^2 d(E)+1}{d(E)+d(\psi)^2} \right|_{d(E)\to 1} = \frac{d(\psi)^2-1}{d(\psi)^2+1} = \tanh(\log d(\psi)).$$

The limits of this bound are quite intuitive: for $\log d(\psi) = 0$ (independence of x_t and x_s), this derivative is zero; increasing the error in incoming messages m_{ut}^{i} has no effect on the error in m_{ts}^{i+1}. For $d(\psi) \to \infty$, the derivative approaches unity, indicating that for very large $d(\psi)$ (strong potentials) the propagated error can be nearly unchanged.

We may apply these bounds to investigate the behavior of BP in graphs with cycles. We begin by examining loopy belief propagation with exact messages, using the previous results to derive a new sufficient condition for BP convergence to a unique fixed point. When this condition is not satisfied, we instead obtain a bound on the relative distances between any two fixed points of the loopy BP equations. This allows us to consider the effect of introducing additional errors into the messages passed at each iteration, showing sufficient conditions for this operation to converge, and a bound on the resulting error from exact loopy BP.

5. Applying Dynamic Range to Graphs with Cycles

In this section, we apply the framework developed in Section 4, along with the computation tree formalism of Tatikonda and Jordan (2002), to derive results on the behavior of traditional belief propagation (in which messages and potentials are represented exactly). We then use the same methodology to analyze the behavior of loopy BP for quantized or otherwise approximated messages and potential functions.

5.1 Convergence of Loopy Belief Propagation

The work of Tatikonda and Jordan (2002) showed that the convergence and fixed points of loopy BP may be considered in terms of a Gibbs measure on the graph's computation tree. In particular, this led to the result that loopy BP is guaranteed to converge if the graph satisfies Dobrushin's condition (Georgii, 1988). Dobrushin's condition is a global measure, and difficult to verify; given in Tatikonda and Jordan (2002) is the easier to check sufficient condition (often called Simon's condition),

Theorem 10 (Simon's condition). *Loopy belief propagation is guaranteed to converge if*

$$\max_t \sum_{u\in\Gamma_t} \log d\left(\psi_{ut}\right) < 1. \tag{9}$$

where $d\left(\psi\right)$ is defined as in (7).

Proof. See Tatikonda and Jordan (2002). □

Using the previous section's analysis, we obtain the following, stronger condition, and (after the proof) show analytically how the two are related.

Theorem 11 (BP convergence). *Loopy belief propagation is guaranteed to converge if*

$$\max_{(s,t)\in\mathcal{E}} \sum_{u\in\Gamma_t\setminus s} \frac{d\left(\psi_{ut}\right)^2-1}{d\left(\psi_{ut}\right)^2+1} < 1 \tag{10}$$

Proof. By induction. Let the "true" messages m_{ts} be any fixed point of BP, and consider the incoming error observed by a node t at level $n-1$ of the computation tree (corresponding to the first iteration of BP), and having parent node s. Suppose that the total incoming error $\log d\left(E_{ts}^1\right)$ is bounded above by some constant $\log\varepsilon^1$ for all $(t,s)\in\mathcal{E}$. Note that this is trivially true (for any n) for the constant $\log\varepsilon^1 = \max_t \sum_{u\in\Gamma_t} \log d\left(\psi_{ut}\right)^2$, since the error on any message m_{ut} is bounded above by $d\left(\psi_{ut}\right)^2$.

Now, assume that $\log d\left(E_{ut}^i\right) \le \log\varepsilon^i$ for all $(u,t)\in\mathcal{E}$. Theorem 8 bounds the maximum log-error $\log d\left(E_{ts}^{i+1}\right)$ at any replica of node t with parent s, where s is on level $n-i$ of the tree (which corresponds to the i^{th} iteration of loopy BP) by

$$\log d\left(E_{ts}^{i+1}\right) \le g_{ts}(\log\varepsilon^i) = G_{ts}(\varepsilon^i) = \sum_{u\in\Gamma_t\setminus s} \log \frac{d\left(\psi_{ut}\right)^2\varepsilon^i+1}{d\left(\psi_{ut}\right)^2+\varepsilon^i}. \tag{11}$$

We observe a contraction of the error between iterations i and $i+1$ if the bound $g_{ts}(\log\varepsilon^i)$ is smaller than $\log\varepsilon^i$ for every $(t,s)\in\mathcal{E}$, and asymptotically achieve $\log\varepsilon^i\to 0$ if this is the case for any value of $\varepsilon^i > 1$.

Defining $z = \log \varepsilon$, we may equivalently show $g_{ts}(z) < z$ for all $z > 0$. This can be guaranteed by the conditions $g_{ts}(0) = 0$, $g'_{ts}(0) < 1$, and $g''_{ts}(z) \leq 0$ for each t,s. The first is easy to verify, as is the last (term by term) using the identity $g''_{ts}(z) = \varepsilon^2 G''_{ts}(\varepsilon) + \varepsilon G'_{ts}(\varepsilon)$; the second ($g'_{ts}(0) < 1$) can be rewritten to give the convergence condition (10). $\square$

We may relate Theorem 11 to Simon's condition by expanding the set $\Gamma_t \setminus s$ to the larger set Γ_t, and observing that $\log x \geq \frac{x^2-1}{x^2+1}$ for all $x \geq 1$ with equality as $x \to 1$. Doing so, we see that Simon's condition is sufficient to guarantee Theorem 11, but that Theorem 11 may be true (implying convergence) when Simon's condition is not satisfied. The improvement over Simon's condition becomes negligible for highly-connected systems with weak potentials, but can be significant for graphs with low connectivity. For example, if the graph consists of a single loop then each node t has at most two neighbors. In this case, the contraction (11) tells us that the outgoing message in either direction is *always* as close or closer to the BP fixed point than the incoming message. Thus we easily obtain the result of Weiss (2000), that (for finite-strength potentials) BP always converges to a unique fixed point on graphs containing a single loop. Simon's condition, on the other hand, is too loose to demonstrate this fact. The form of the condition in Theorem 11 is also similar to a result shown for binary spin models; see Georgii (1988) for details.

However, both Theorem 10 and Theorem 11 depend only on the pairwise potentials $\psi_{st}(x_s, x_t)$, and not on the single-node potentials $\psi_s(x_s)$, $\psi_t(x_t)$. As noted by Heskes (Heskes, 2004), this leaves a degree of freedom to which the single-node potentials may be chosen so as to minimize the (apparent) strength of the pairwise potentials. Thus, (9) can be improved slightly by writing

$$\max_t \sum_{u \in \Gamma_t} \min_{\psi_u, \psi_t} \log d\left(\frac{\psi_{ut}}{\psi_u \psi_t}\right) < 1 \tag{12}$$

and similarly for (10) by writing

$$\max_{(s,t) \in \mathcal{E}} \sum_{u \in \Gamma_t \setminus s} \min_{\psi_u, \psi_t} \frac{d\left(\frac{\psi_{ut}}{\psi_u \psi_t}\right)^2 - 1}{d\left(\frac{\psi_{ut}}{\psi_u \psi_t}\right)^2 + 1} < 1. \tag{13}$$

To evaluate this quantity, one may also observe that

$$\min_{\psi_u, \psi_t} d\left(\frac{\psi_{ut}}{\psi_u \psi_t}\right)^4 = \sup_{a,b,c,d} \frac{\psi_{ts}(a,b)}{\psi_{ts}(a,d)} \frac{\psi_{ts}(c,d)}{\psi_{ts}(c,b)}.$$

In general we shall ignore this subtlety and simply write our results in terms of $d(\psi)$, as given in (9) and (10). For binary random variables, it is easy to see that the minimum–strength ψ_{ut} has the form

$$\psi_{ut} = \begin{bmatrix} \eta & 1-\eta \\ 1-\eta & \eta \end{bmatrix},$$

and that when the potentials are of this form (such as in the examples of this section) the two conditions are completely equivalent.

We provide a more empirical comparison between our condition, Simon's condition, and the recent work of Heskes (2004) shortly. Similarly to Heskes (2004), we shall see that it is possible to use the graph geometry to improve our bound (Section 5.3); but perhaps more importantly (and in contrast to both other methods), when the condition is *not* satisfied, we still obtain useful information about the relationship between any pair of fixed points (Section 5.2), allowing its extension to quantized or otherwise distorted versions of belief propagation (Section 5.4).

5.2 Distance of Multiple Fixed Points

Theorem 11 may be extended to provide not only a sufficient condition for a unique BP fixed point, but an upper bound on distance between the beliefs generated by successive BP updates and any BP fixed point. Specifically, the proof of Theorem 11 relied on demonstrating a bound $\log \varepsilon^i$ on the distance from some arbitrarily chosen fixed point $\{M_t\}$ at iteration i. When this bound decreases to zero, we may conclude that only one fixed point exists. However, even should it decrease only to some positive constant, it still provides information about the distance between any iteration's belief and the fixed point. Moreover, applying this bound to another, different fixed point $\{\tilde{M}_t\}$ tells us that all fixed points of loopy BP must lie within a sphere of a given diameter [as measured by $\log d\left(M_t/\tilde{M}_t\right)$]. These statements are made precise in the following two theorems:

Theorem 12 (BP distance bound). *Let $\{M_t\}$ be any fixed point of loopy BP. Then, after $n > 1$ iterations of loopy BP resulting in beliefs $\{\hat{M}_t^n\}$, for any node t and for all x*

$$\log d\left(M_t/\hat{M}_t^n\right) \leq \sum_{u\in\Gamma_t} \log \frac{d\left(\psi_{ut}\right)^2 \varepsilon^{n-1}+1}{d\left(\psi_{ut}\right)^2+\varepsilon^{n-1}}$$

where ε^i is given by $\varepsilon^1 = \max_{s,t} d\left(\psi_{st}\right)^2$ and

$$\log \varepsilon^{i+1} = \max_{(s,t)\in\mathcal{E}} \sum_{u\in\Gamma_t\setminus s} \log \frac{d\left(\psi_{ut}\right)^2 \varepsilon^{i}+1}{d\left(\psi_{ut}\right)^2+\varepsilon^{i}}.$$

Proof. The result follows directly from the proof of Theorem 11. ☐

We may thus infer a distance bound between any two BP fixed points:

Theorem 13 (Fixed-point distance bound). *Let $\{M_t\}$, $\{\tilde{M}_t\}$ be the beliefs of any two fixed points of loopy BP. Then, for any node t and for all x*

$$|\log M_t(x)/\tilde{M}_t(x)| \leq 2\log d\left(M_t/\tilde{M}_t\right) \leq 2\sum_{u\in\Gamma_t} \log \frac{d\left(\psi_{ut}\right)^2 \varepsilon+1}{d\left(\psi_{ut}\right)^2+\varepsilon} \tag{14}$$

where ε is the largest value satisfying

$$\log \varepsilon = \max_{(s,t)\in\mathcal{E}} G_{ts}(\varepsilon) = \max_{(s,t)\in\mathcal{E}} \sum_{u\in\Gamma_t\setminus s} \log \frac{d\left(\psi_{ut}\right)^2 \varepsilon+1}{d\left(\psi_{ut}\right)^2+\varepsilon}. \tag{15}$$

Proof. The inequality $|\log M_t(x)/\tilde{M}_t(x)| \leq 2\log d\left(M_t/\tilde{M}_t\right)$ follows from Theorem 2. The rest follows from Theorem 12—taking the "approximate" messages to be any other fixed point of loopy BP, we see that the error cannot decrease over any number of iterations. However, by the same argument given in Theorem 11, $g_{ts}''(z) < 0$, and for z sufficiently large, $g_{ts}(z) < z$. Thus (15) has at most one solution greater than unity, and $\varepsilon^{i+1} < \varepsilon^i$ for all i with $\varepsilon^i \to \varepsilon$ as $i \to \infty$. Letting the number of iterations $i \to \infty$, we see that the message "errors" $\log d\left(M_{ts}/\tilde{M}_{ts}\right)$ must be at most ε, and thus the difference in M_t (the belief of the root node of the computation tree) must satisfy (14). ☐

Thus, if the value of $\log \varepsilon$ is small (the sufficient condition of Theorem 11 is nearly satisfied) then although we cannot guarantee convergence to a unique fixed point, we can still make a strong statement: that the set of fixed points are all mutually close (in a log-error sense), and reside within a ball of diameter described by (14). Moreover, even though it is possible that loopy BP does not converge, and thus even after infinite time the messages may not correspond to *any* fixed point of the BP equations, we are guaranteed by Theorem 12 that the resulting belief estimates *will* asymptotically approach the same bounding ball [achieving distance at most (14) from *all* fixed points].

5.3 Path-Counting

If we are willing to put a bit more effort into our bound-computation, we may be able to improve it further, since the bounds derived using computation trees are very much "worst-case" bounds. In particular, the proof of Theorem 11 assumes that, as a message error propagates through the graph, repeated convolution with *only* the strongest set of potentials is possible. But often even if the worst potentials are quite strong, every cycle which contains them may also contain several weaker potentials. Using an iterative algorithm much like belief propagation itself, we may obtain a more globally aware estimate of how errors can propagate through the graph.

Theorem 14 (Non-uniform distance bound). *Let $\{M_t\}$ be any fixed point belief of loopy BP. Then, after $n \geq 1$ iterations of loopy BP resulting in beliefs $\{\hat{M}_t^n\}$, for any node t and for all x*

$$|\log M_t(x)/\hat{M}_t(x)| \leq 2 \log d\left(M_t/\hat{M}_t^n\right) \leq 2 \sum_{u \in \Gamma_t} \log \upsilon_{ut}^n$$

where υ_{ut}^i is defined by the iteration

$$\log \upsilon_{ts}^{i+1} = \log \frac{d(\psi_{ts})^2 \varepsilon_{ts}^i + 1}{d(\psi_{ts})^2 + \varepsilon_{ts}^i} \qquad \log \varepsilon_{ts}^i = \sum_{u \in \Gamma_t \setminus s} \log \upsilon_{ut}^i \tag{16}$$

with initial condition $\upsilon_{ut}^1 = d(\psi_{ut})^2$.

Proof. Again we consider the error $\log d\left(E_{ts}^i\right)$ incoming to node t with parent s, where t is at level $n - i + 1$ of the computation tree. Using the same arguments as Theorem 11 it is easy to show by induction that the error products $\log d\left(E_{ts}^i\right)$ are bounded above by ε_{ts}^i, and the individual message errors $\log d\left(e_{ts}^i\right)$ are bounded above by υ_{ts}^i, and . Then, by additivity we obtain the stated bound on $d(E_t^n)$ at the root node. □

The iteration defined in Theorem 14 can also be interpreted as a (scalar) message-passing procedure, or may be performed offline. As before, if this procedure results in $\log \varepsilon_{ts} \to 0$ for all $(t,s) \in \mathcal{E}$ we are guaranteed that there is a unique fixed point for loopy BP; if not, we again obtain a bound on the distance between any two fixed-point beliefs. When the graph is perfectly symmetric (every node has identical neighbors and potential strengths), this yields the same bound as Theorem 12; however, if the potential strengths are inhomogeneous Theorem 14 provides a strictly better bound on loopy BP convergence and errors.

This situation is illustrated in Figure 5—we specify two different graphical models defined on a 5×5 grid in terms of their potential strengths $\log d(\psi)^2$, and compute bounds on the dynamic range $d\left(M_t/\tilde{M}_t\right)$ of any two fixed point beliefs M_t, $\tilde{M}_t$ for each model. (Note that, while potential strength

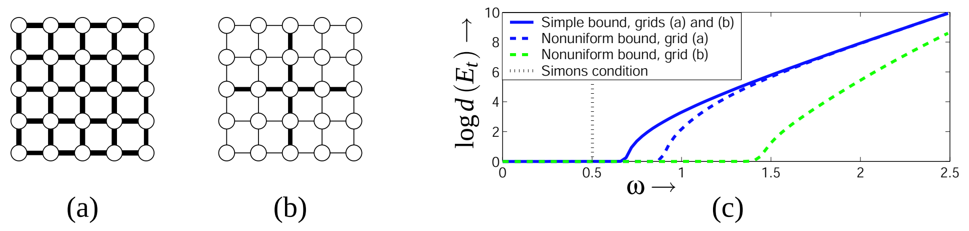


Figure 5: (a-b) Two small (5×5) grids. In (a), the potentials ψ are all of equal strength ($\log d\,(\psi)^2 = \omega$), while in (b) several potentials (thin lines) are weaker ($\log d\,(\psi)^2 = .5\omega$). The methods described may be used to compute bounds (c) on the distance $d\,(E_t)$ between any two fixed point beliefs as a function of potential strength ω.

does not completely specify the graphical model, it is sufficient for all the bounds considered here.) One grid (a) has equal-strength potentials $\log d\,(\psi)^2 = \omega$, while the other has many weaker potentials ($\omega/2$). The worst-case bounds are the same (since both have a node with four strong neighbors), shown as the solid curve in (c). However, the dashed curves show the estimate of (16), which improves only slightly for the strongly coupled graph (a) but considerably for the weaker graph (b). All three bounds give considerably more information than Simon's condition (dotted vertical line).

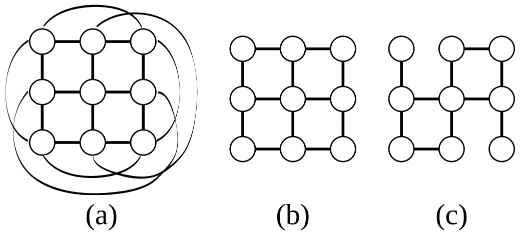
Having shown how our bound may be improved for irregular graph geometry, we may now compare our bounds to two other known uniqueness conditions (Tatikonda and Jordan, 2002; Heskes, 2004). Simon's condition can be related analytically, as described in Section 5.1. On the other hand, the recent work of Heskes (2004) takes a very different approach to uniqueness based on analysis of the minima of the Bethe free energy, which directly correspond to stable fixed points of BP (Yedidia et al., 2004). This leads to an alternate sufficient condition for uniqueness. As observed in Heskes (2004) it is unclear whether a unique fixed point necessarily implies convergence of loopy BP. In contrast, our approach gives a sufficient condition for the convergence of BP to a unique solution, which implies uniqueness of the fixed point.

Showing an analytic relation between all three approaches does not appear straightforward; to give some intuition, we show the three example binary graphs compared in Heskes (2004), whose structures are shown in Figure 6(a-c) and whose potentials are parameterized by a scalar $\eta > .5$, namely

$$\psi = \begin{bmatrix} \eta & 1-\eta \\ 1-\eta & \eta \end{bmatrix} \tag{17}$$

(so that $d\,(\psi)^2 = \frac{\eta}{1-\eta}$). The trivial solution $M_t = [.5; .5]$ is always a fixed point, but may not be stable; the precise η_{crit} at which this fixed point becomes unstable (implying the existence of other, stable fixed points) can be found empirically for each case (Heskes, 2004); the same values may also be found algebraically by imposing symmetry requirements on the messages (Yedidia et al., 2004). This value may then be compared to the uniqueness bounds of Tatikonda and Jordan (2002), the bound of Heskes (2004), and this work; these are shown in Figure 6.

Notice that our bound is always better than Simon's condition, though for the perfectly symmetric graph the margin is not large (and decreases further with increased connectivity, for example a cubic lattice). Additionally, in all three examples our method appears to outperform that of Heskes



Method	(a)	(b)	(c)
Simon's condition	.62	.62	.62
Heskes' condition	.55	.58	.65
This work	.67	.79	.88
Empirical	.67	.79	.88

η_{crit}

Figure 6: Comparison of various uniqueness bounds: for binary potentials parameterized by η, we find the predicted η_{crit} at which loopy BP can no longer be guaranteed to be unique. For these simple problems, the η_{crit} at which the trivial (correct) solution becomes unstable may be found empirically. Examples and empirical values of η_{crit} from Heskes (2004).

(2004), though without analytic comparison it is unclear whether this is always the case. In fact, for these simple binary examples, our bound appears to be tight.

However, our method also allows us to make statements about the results of loopy BP after finite numbers of iterations, up to some finite degree of numerical precision in the final results. For example, we may also find the value of η below which BP will attain a particular precision, say $\log d\left(M_t/\hat{M}_t^n\right) < 10^{-3}$ in at least $n = 100$ iterations [obtaining the values $\{.66, .77, .85\}$ for the grids in Figure 6(a), (b), and (c), respectively].

5.4 Introducing Intentional Message Errors and Censoring

As discussed in the introduction, we may wish to introduce or allow *additional* errors in our messages at each stage, in order to improve the computational or communication efficiency of the algorithm. This may be the result of an actual distortion imposed on the message (perhaps to decrease its complexity, for example quantization), or the result of censoring the message update (reusing the message from the previous iteration) when the two are sufficiently similar. Errors may also arise from quantization or other approximation of the potential functions. Such additional errors may be easily incorporated into our framework.

Theorem 15. *If at every iteration of loopy BP, each message is further approximated in such a way as to guarantee that the additional distortion has maximum dynamic range at most δ, then for any fixed point beliefs $\{M_t\}$, after $n \geq 1$ iterations of loopy BP resulting in beliefs $\{\hat{M}_t^n\}$ we have*

$$\log d\left(M_t/\hat{M}_t^n\right) \leq \sum_{u\in\Gamma_t} \log \upsilon_{ut}^n$$

where υ_{ut}^i is defined by the iteration

$$\log \upsilon_{ts}^{i+1} = \log \frac{d\left(\psi_{ts}\right)^2 \varepsilon_{ts}^i + 1}{d\left(\psi_{ts}\right)^2 + \varepsilon_{ts}^i} + \log \delta \qquad\qquad \log \varepsilon_{ts}^i = \sum_{u\in\Gamma_t\setminus s} \log \upsilon_{ut}^i$$

with initial condition $\upsilon_{ut}^1 = \delta\, d\left(\psi_{ut}\right)^2$.

Proof. Using the same logic as Theorems 12 and 14, apply additivity of the log dynamic range measure to the additional distortion $\log \delta$ introduced to each message. □

As with Theorem 14, a simpler bound can also be derived (similar to Theorem 12). Either gives a bound on the maximum total distortion from any true fixed point which will be incurred by quantized or censored belief propagation. Note that (except on tree-structured graphs) this does *not* bound the error from the true marginal distributions, only from the loopy BP fixed points.

It is also possible to interpret the additional error as arising from an approximation to the correct single-node and pairwise potentials ψ_t, ψ_{ts}.

Theorem 16. *Suppose that $\{M_t\}$ are a fixed point of loopy BP on a graph defined by potentials ψ_{ts} and ψ_t, and let $\{\hat{M}_t^n\}$ be the beliefs of n iterations of loopy BP performed on a graph with potentials $\hat{\psi}_{ts}$ and $\hat{\psi}_t$, where $d\left(\hat{\psi}_{ts}/\psi_{ts}\right) \le \delta_1$ and $d\left(\hat{\psi}_t/\psi_t\right) \le \delta_2$. Then,*

$$\log d\left(M_t/\hat{M}_t^n\right) \le \sum_{u\in\Gamma_t} \log \upsilon_{ut}^n + \log \delta_2$$

where υ_{ut}^i is defined by the iteration

$$\log \upsilon_{ts}^{i+1} = \log \frac{d\left(\psi_{ts}\right)^2 \varepsilon_{ts}^i + 1}{d\left(\psi_{ts}\right)^2 + \varepsilon_{ts}^i} + \log \delta_1 \qquad\qquad \log \varepsilon_{ts}^i = \log \delta_2 + \sum_{u\in\Gamma_t\setminus s} \log \upsilon_{ut}^i$$

with initial condition $\upsilon_{ut}^1 = \delta_1\, d\left(\psi_{ut}\right)^2$.

Proof. We first extend the contraction result given in Appendix A by applying the inequality

$$\frac{\int \psi(x_t,a)\frac{\hat{\psi}(x_t,a)}{\psi(x_t,a)} M(x_t)E(x_t)dx_t}{\int \psi(x_t,b)\frac{\hat{\psi}(x_t,b)}{\psi(x_t,b)} M(x_t)E(x_t)dx_t} \le \frac{\int \psi(x_t,a)M(x_t)E(x_t)dx_t}{\int \psi(x_t,b)M(x_t)E(x_t)dx_t} \cdot d\left(\hat{\psi}/\psi\right)^2 .$$

Then, proceeding similarly to Theorem 15 yields the definition of υ_{ts}^i, and including the additional errors $\log \delta_2$ in each message product (resulting from the product with $\hat{\psi}_t$ rather than ψ_t) gives the definition of ε_{ts}^i. $\square$

Incorrect models $\hat{\psi}$ may arise when the exact graph potentials have been estimated or quantized; Theorem 16 gives us the means to interpret the (worst-case) overall effects of using an approximate model. As an example, let us again consider the model depicted in Figure 6(b). Suppose that we are given *quantized* versions of the pairwise potentials, $\hat{\psi}$, specified by the value (rounded to two decimal places) $\eta = .65$. Then, the true potential ψ has $\eta \in .65 \pm .005$, and thus is within $\delta_1 \approx 1.022 = \frac{(.35)(.655)}{(.345)(.65)}$ of the known approximation $\hat{\psi}$. Applying the recursion of Theorem 16 allows us to conclude that the solution obtained using the approximate model $\hat{\psi}$ and true model ψ are within $\log d\left(e\right) \le .36$, or alternatively that the beliefs found using the approximate model are correct to within a multiplicative factor of about 1.43. The same $\hat{\psi}$, with η assumed correct to three decimal places, gives a bound $\log d\left(e\right) \le .04$, or multiplicative factor of 1.04.

5.5 Stochastic Analysis

Unfortunately, the bounds given by Theorem 16 are often pessimistic compared to actual performance. We may use a similar analysis, coupled with the assumption of uncorrelated message errors, to obtain a more realistic estimate (though no longer a strict bound) on the resulting error.

Proposition 17. *Suppose that the errors* $\log e_{ts}$ *are random and uncorrelated, so that at each iteration i, for* $s \neq u$ *and any x,* $E\left[\log e^i_{st}(x) \cdot \log e^i_{ut}(x)\right] = 0$*, and that at each iteration of loopy BP, the additional error (in the log domain) imposed on each message is uncorrelated with variance at most* $(\log \delta)^2$*. Then,*

$$E\left[\left(\log d\left(E^i_t\right)\right)^2\right] \leq \sum_{u \in \Gamma_t} \left(\sigma^i_{ut}\right)^2 \tag{18}$$

where $\sigma^1_{ts} = \log d\left(\psi_{ts}\right)^2$ *and*

$$\left(\sigma^{i+1}_{ts}\right)^2 = \left(\log \frac{d\left(\psi_{ts}\right)^2 \lambda^i_{ts} + 1}{d\left(\psi_{ts}\right)^2 + \lambda^i_{ts}}\right)^2 + (\log \delta)^2 \qquad\qquad \left(\log \lambda^i_{ts}\right)^2 = \sum_{u \in \Gamma_t \setminus s} \left(\sigma^i_{ut}\right)^2.$$

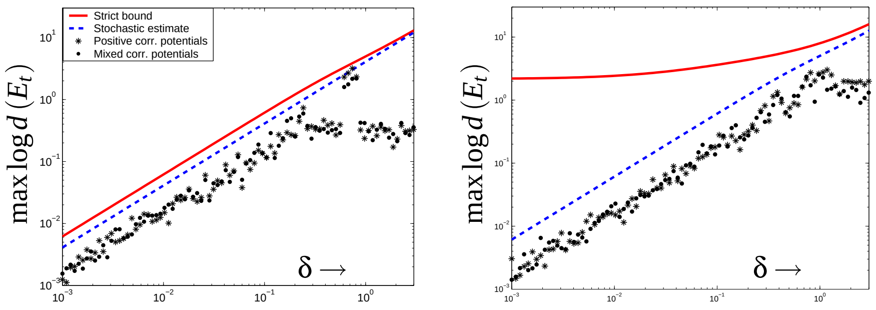
Proof. Let us define the (nuisance) scale factor $\alpha^i_{ts} = \arg\min_\alpha \sup_x |\log \alpha e^i_{ts}(x)|$ for each error e^i_{ts}, and let $\zeta^i_{ts}(x) = \log \alpha^i_{ts} e^i_{ts}(x)$. Now, we model the error function $\zeta^i_{ts}(x)$ (for each x) as a random variable with mean zero, and bound the standard deviation of $\zeta^i_{ts}(x)$ by σ^i_{ts} at each iteration i; under the assumption that the errors in any two incoming messages are uncorrelated, we may assert additivity of their variances. Thus the variance of $\sum_{\Gamma_t \setminus s} \zeta^i_{ut}(x)$ is bounded by $(\log \lambda^i_{ts})^2$. The contraction of Theorem 8 is a non-linear relationship; we estimate its effect on the error variance using a simple sigma-point quadrature ("unscented") approximation (Julier and Uhlmann, 1996), in which the standard deviation σ^{i+1}_{ts} is estimated by applying Theorem 8's nonlinear contraction to the standard deviation of the error on the incoming product $(\log \lambda^i_{ts})$. □

The assumption of uncorrelated errors is clearly questionable, since propagation around loops may couple the incoming message errors. However, similar assumptions have yielded useful analysis of quantization effects in assessing the behavior and stability of digital filters (Willsky, 1978). It is often the case that empirically, such systems behave similarly to the predictions made by assuming uncorrelated errors. Indeed, we shall see that in our simulations, the assumption of uncorrelated errors provides a good estimate of performance.

Given the bound (18) on the variance of $\log d(E)$, we may apply a Chebyshev-like argument to provide probabilistic guarantees on the magnitude of errors $\log d(E)$ observed in practice. In our experiments (Section 5.6), the 2σ distance was almost always larger than the observed error. The probabilistic bound derived using (18) is typically much smaller than the bound of Theorem 15 due to the strictly sub-additive relationship between the standard deviations. However, the underlying assumption of uncorrelated errors makes the estimate obtained using (18) unsuitable for deriving strict convergence guarantees.

5.6 Experiments

We demonstrate the dynamic range error bounds for quantized messages with a set of Monte Carlo trials. In particular, for each trial we construct a binary–valued 5×5 grid with uniform potential strengths, which are either (1) all positively correlated, or (2) randomly chosen to be positively or negatively correlated (equally likely); we also assign random single-node potentials to each variable x_s. We then run a quantized version of BP for $n = 100$ iterations from the same initial conditions, rounding each log-message to discrete values separated by $2 \log \delta$ (ensuring that the newly introduced error satisfies $d(e) \leq \delta$). Figure 7 shows the maximum belief error in each of 100 trials of this procedure for various values of δ.



(a) $\log d\,(\psi)^2 = .25$ (b) $\log d\,(\psi)^2 = 1$

Figure 7: Maximum belief errors incurred as a function of the quantization error. The scatterplot indicates the maximum error measured in the graph for each of 200 Monte Carlo runs; this is strictly bounded above by Theorem 15, solid, and bounded with high probability (assuming uncorrelated errors) by Proposition 17, dashed.

.

Also shown are two performance estimators—the *bound* on belief error developed in Section 5.4, and the 2σ estimate computed assuming uncorrelated message errors as in Section 5.5. As can be seen, the stochastic estimate is a much tighter, more accurate assessment of error, but it does not possess the same strong theoretical guarantees. Since [as observed for digital filtering applications (Willsky, 1978)] the errors introduced by quantization are typically close to independent, the assumptions underlying the stochastic estimate are reasonable, and empirically we observe that the estimate and actual errors behave similarly.

6. KL-Divergence Measures

Although the dynamic range measure introduced in Section 4 leads to a number of strong guarantees, its performance criterion may be unnecessarily (and undesirably) strict. Specifically, it provides a *pointwise* guarantee, that m and $\hat{m}$ are close for every possible state x. For continuous-valued states, this is an extremely difficult criterion to meet—for instance, it requires that the messages' tails match almost exactly. In contrast, typical measures of the difference between two distributions operate by an average (mean squared error or mean absolute error) or weighted average (Kullback-Leibler divergence) evaluation. To address this, let us consider applying a measure such as the Kullback-Leibler (KL) divergence,

$$D(p\|\hat{p}) = \int p(x) \log \frac{p(x)}{\hat{p}(x)} dx.$$

The pointwise guarantees of Section 4 are necessary to bound performance even in the case of "unlikely" events. More specifically, the tails of a message approximation can become important if two parts of the graph strongly disagree, in which case the tails of each message are the only overlap of significant likelihood. One way to discount this possibility is to consider the graph potentials themselves (in particular, the single node potentials ψ_t) as a realization of random variables which "typically" agree, then apply a probabilistic measure to estimate the typical performance. From this viewpoint, since a strong disagreement between parts of the graph is unlikely we will be able to relax our error measure in the message tails.

Unfortunately, many of the properties which we relied on for analysis of the dynamic range measure do not strictly hold for a KL-divergence measure of error, resulting in an *approximation*, rather than a bound, on performance. In Appendix B, we give a detailed analysis of each property, showing the ways in which each aspect can break down and discussing the reasonability of simple approximations. In this section, we apply these approximations to develop a KL-divergence based estimate of error.

6.1 Local Observations and Parameterization

To make this notion concrete, let us consider a graphical model in which the single-node potential functions are specified in terms of a set of observation variables $\mathbf{y} = \{y_t\}$; in this section we will examine the average (expected) behavior of BP over multiple realizations of the observation variables $\mathbf{y}$. We further assume that both the prior $p(\mathbf{x})$ and likelihood $p(\mathbf{y}|\mathbf{x})$ exhibit conditional independence structure, expressed as a graphical model. Specifically, we assume throughout this section that the observation likelihood factors as

$$p(\mathbf{y}|\mathbf{x}) = \prod_t p(y_t|x_t), \tag{19}$$

in other words, that each observation variable y_t is *local* to (conditionally independent given) one of the x_t. As for the prior model $p(\mathbf{x})$, for the moment we confine our attention to tree-structured distributions, for which one may write (Wainwright et al., 2003)

$$p(\mathbf{x}) = \prod_{(s,t)\in\mathcal{E}} \frac{p(x_s,x_t)}{p(x_s)p(x_t)} \prod_s p(x_s). \tag{20}$$

The expressions (19)-(20) give rise to a convenient parameterization of the joint distribution, expressed as

$$p(\mathbf{x},\mathbf{y}) \propto \prod_{(s,t)\in\mathcal{E}} \psi_{st}(x_s,x_t) \prod_s \psi_s^x(x_s)\psi_s^y(x_s) \tag{21}$$

where

$$\psi_{st}(x_s,x_t) = \frac{p(x_s,x_t)}{p(x_s)p(x_t)} \qquad \text{and} \qquad \psi_s^x(x_s) = p(x_s) \qquad , \qquad \psi_s^y(x_s) = p(y_s|x_s). \tag{22}$$

Our goal is to compute the posterior marginal distributions $p(x_s|\mathbf{y})$ at each node s; for the tree-structured distribution (21) this can be performed exactly and efficiently by BP. As discussed in the previous section, we treat the $\{y_t\}$ as random variables; thus almost all quantities in this graph are themselves random variables (as they are dependent on the y_t), so that the single node observation potentials $\psi_s^y(x_s)$, messages $m_{st}(x_t)$, *etc.* are random functions of their argument x_s. The potentials due to the prior (ψ_{st} and ψ_s^x), however, are not random variables as they do not depend on any of the observations y_t.

For models of the form (21)-(22), the (unique) BP message fixed point consists of normalized versions of the likelihood functions $m_{ts}(x_s) \propto p(\mathbf{y}_{ts}|x_s)$, where $\mathbf{y}_{ts}$ denotes the set of all observations $\{y_u\}$ such that t separates u from s. In this section it is also convenient to perform a *prior-weighted* normalization of the messages m_{ts}, so that $\int p(x_s) m_{ts}(x_s) = 1$ (as opposed to $\int m_{ts}(x_s) = 1$ as assumed previously); we again assume this prior-weighted normalization is always possible (this is trivially the case for discrete-valued states $\mathbf{x}$). Then, for a tree-structured graph, the prior-weight normalized fixed-point message from t to s is precisely

$$m_{ts}(x_s) = p(\mathbf{y}_{ts}|x_s)/p(\mathbf{y}_{ts}) \tag{23}$$

and the products of incoming messages to t, as defined in Section 2.3, are equal to

$$M_{ts}(x_t) = p(x_t|\mathbf{y}_{ts}) \qquad\qquad M_t(x_t) = p(x_t|\mathbf{y}).$$

We may now apply a *posterior-weighted log-error* measure, defined by

$$\mathcal{D}(m_{ut}\|\hat{m}_{ut}) = \int p(x_t|\mathbf{y}) \log \frac{m_{ut}(x_t)}{\hat{m}_{ut}(x_t)} dx_t; \tag{24}$$

and may relate (24) to the Kullback-Leibler divergence.

Lemma 18. *On a tree-structured graph, the error measure $\mathcal{D}(M_t, \hat{M}_t)$ is equivalent to the KL-divergence of the true and estimated posterior distributions at node t:*

$$\mathcal{D}(M_t\|\hat{M}_t) = D(p(x_t|\mathbf{y})\|\hat{p}(x_t|\mathbf{y})).$$

Proof. This follows directly from the definitions of $\mathcal{D}$, and the fact that on a tree, the unique fixed point has beliefs $M_t(x_t) = p(x_t|\mathbf{y})$. □

Again, the error $\mathcal{D}(m_{ut}\|\hat{m}_{ut})$ is a function of the observations $\mathbf{y}$, both explicitly through the term $p(x_t|\mathbf{y})$ and implicitly through the message $m_{ut}(x_t)$, and is thus also a random variable. Although the definition of $\mathcal{D}(m_{ut}\|\hat{m}_{ut})$ involves the *global* observation $\mathbf{y}$ and thus cannot be calculated at node u without additional (non-local) information, we will primarily be interested in the expected value of these errors over many realizations $\mathbf{y}$, which is a function only of the distribution. Specifically, we can see that in expectation over the data $\mathbf{y}$, it is simply

$$E\left[\mathcal{D}(m_{ut}\|\hat{m}_{ut})\right] = E\left[\int p(x_t) m_{ut}(x_t) \log \frac{m_{ut}(x_t)}{\hat{m}_{ut}(x_t)} dx_t\right]. \tag{25}$$

One nice consequence of the choice of potential functions (22) is the locality of prior information. Specifically, if *no* observations $\mathbf{y}$ are available, and only prior information is present, the BP messages are trivially constant [$m_{ut}(x) = 1\ \forall x$]. This ensures that any message approximations affect only the data likelihood, and not the prior $p(x_t)$; this is similar to the motivation of Paskin and Guestrin (2004), in which an additional message-passing procedure is used to create this parameterization.

Finally, two special cases are of note. First, if x_s is discrete-valued and the prior distribution $p(x_s)$ constant (uniform), the expected message distortion with prior-normalized messages, $E[\mathcal{D}(m\|\hat{m})]$, and the KL-divergence of traditionally normalized messages behave equivalently, i.e.,

$$E\left[\mathcal{D}(m_{ts}\|\hat{m}_{ts})\right] = E\left[D\left(\frac{m_{ts}}{\int m_{ts}} \| \frac{\hat{m}_{ts}}{\int \hat{m}_{ts}}\right)\right]$$

where we have abused the notation of KL-divergence slightly to apply it to the normalized likelihood $m_{ts}/\int m_{ts}$. This interpretation leads to the same message-censoring criterion used in Chen et al. (2004).

Secondly, when the state x_s is a discrete-valued random variable taking on one of M possible values, a straightforward uniform quantization of the value of $p(x_s)m(x_s)$ results in a bound on the divergence (25). Specifically, we have the following lemma:

Lemma 19. *For an M-ary discrete variable x, the quantization*

$$p(x)m(x) \to \{\varepsilon, 3\varepsilon, \ldots, 1-\varepsilon\}$$

results in an expected divergence bounded by

$$E\left[\mathcal{D}(m(x)\|\hat{m}(x))\right] \leq (2\log 2 + M)M\varepsilon + O(M^3\varepsilon^2).$$

Proof. Define $\mu(x) = p(x)m(x)$, and $\bar{\mu}(x) \in \{\varepsilon, 3\varepsilon, \ldots, 1-\varepsilon\}$ (for each x) to be its quantized value. Then, the prior-normalized approximation $\hat{m}(x)$ satisfies

$$p(x)\hat{m}(x) = \bar{\mu}(x) \;/\; \sum_x \bar{\mu}(x) = \bar{\mu}(x)/C$$

where $C \in [1-M\varepsilon, 1+M\varepsilon]$. The expected divergence

$$\begin{aligned} E\left[\mathcal{D}(m(x)\|\hat{m}(x))\right] &= \sum_x p(x)m(x)\log\frac{m(x)}{\hat{m}(x)} \\ &\leq \sum_x \mu(x)\log\frac{\mu(x)}{\bar{\mu}(x)} + \sum_x |\log C|. \end{aligned}$$

The first sum is at its maximum for $\mu(x) = 2\varepsilon$ and $\bar{\mu}(x) = \varepsilon$, which results in the value $\sum_x (2\log 2)\varepsilon$. Applying the Taylor expansion of the log, the second sum $\sum|\log C|$ is bounded above by $M^2\varepsilon + O(M^3\varepsilon^2)$. □

Thus, for example, for uniform quantization of a message with binary–valued state x, fidelity up to two significant digits ($\varepsilon = .005$) results in an error $\mathcal{D}$ which, on average, is less than .034.

We now state the approximations which will take the place of the fundamental properties used in the preceding sections, specifically versions of the triangle inequality, sub-additivity, and contraction. Although these properties do *not* hold in general, in practice useful estimates are obtained by making approximations corresponding to each property and following the same development used in the preceding sections. (In fact, experimentally these estimates still appear quite conservative.) A more detailed analysis of each property, along with justification for the approximation applied, is given in Appendix B.

6.2 Approximations

Three properties of the dynamic range described in Section 4 are important in the error analysis of Section 5—a form of the triangle inequality, enabling the accumulation of errors in successive approximations to be bounded by the sum of the individual errors, a form of sub-additivity, enabling the accumulation of errors in the message product operation to be bounded by the sum of incoming errors, and a rate of contraction due to convolution with each pairwise potential. We assume the following three properties for the expected error; see Appendix B for a more detailed discussion.

Approximation 20 (Triangle Inequality). *For a true BP fixed-point message m_{ut} and two approximations $\hat{m}_{ut}$, $\tilde{m}_{ut}$, we assume*

$$\mathcal{D}(m_{ut}\|\tilde{m}_{ut}) \le \mathcal{D}(m_{ut}\|\hat{m}_{ut}) + \mathcal{D}(\hat{m}_{ut}\|\tilde{m}_{ut}). \tag{26}$$

Comment. This is not strictly true for arbitrary $\hat{m}$, $\tilde{m}$, since the KL-divergence (and thus $\mathcal{D}$) does not satisfy the triangle inequality.

Approximation 21 (Sub-additivity). *For true BP fixed-point messages $\{m_{ut}\}$ and approximations $\{\hat{m}_{ut}\}$, we assume*

$$\mathcal{D}(M_{ts}\|\hat{M}_{ts}) \le \sum_{u\in\Gamma_t\setminus s} \mathcal{D}(m_{ut}\|\hat{m}_{ut}). \tag{27}$$

Approximation 22 (Contraction). *For a true BP fixed-point message product M_{ts} and approximation $\hat{M}_{ts}$, we assume*

$$\mathcal{D}(m_{ts}\|\hat{m}_{ts}) \le (1-\gamma_{ts})\mathcal{D}(M_{ts}\|\hat{M}_{ts}) \tag{28}$$

where

$$\gamma_{ts} = \min_{a,b}\int \min\left[\rho(x_s,x_t=a)\,,\,\rho(x_s,x_t=b)\right] dx_s \qquad \rho(x_s,x_t) = \frac{\psi_{ts}(x_s,x_t)\psi^x_s(x_s)}{\int \psi_{ts}(x_s,x_t)\psi^x_s(x_s)dx_s}.$$

Comment. For tree-structured graphical models with the parametrization described by (21)-(22), $\rho(x_s,x_t) = p(x_s|x_t)$, and γ_{ts} corresponds to the rate of contraction described by Boyen and Koller (1998).

6.3 Steady-State Errors

Applying these approximations to graphs with cycles, and following the same development used for constructing the strict bounds of Section 5, we find the following estimates of steady-state error. Note that, other than those outlined in the previous section (and described in Appendix B), this development involves no additional approximations.

Approximation 23. *After $n \ge 1$ iterations of loopy BP subject to additional errors at each iteration of magnitude (measured by $\mathcal{D}$) bounded above by some constant δ, with initial messages $\{m^0_{tu}\}$ satisfying $\mathcal{D}(m_{tu}\|m^0_{tu})$ less than some constant C, results in an expected KL-divergence between a true BP fixed point $\{M_t\}$ and the approximation $\{\hat{M}^n_t\}$ bounded by*

$$E_{\mathbf{y}}\left[D(M_t\|\hat{M}^n_t)\right] = E_{\mathbf{y}}\left[\mathcal{D}(\mathcal{M}_t\|\hat{\mathcal{M}}^n_t)\right] \le \sum_{u\in\Gamma_t}((1-\gamma_{ut})\varepsilon^{n-1}_{ut}+\delta)$$

where $\varepsilon^0_{ts} = C$ and

$$\varepsilon^i_{ts} = \sum_{u\in\Gamma_t\setminus s}((1-\gamma_{ut})\varepsilon^{i-1}_{ut}+\delta).$$

Comment. The argument proceeds similarly to that of Theorem 15. Let ε^i_{ts} bound the quantity $\mathcal{D}(\mathcal{M}_{ts}\|\hat{\mathcal{M}}^i_{ts})$ at each iteration i, and apply Approximations 20-22.

We refer to the estimate described in Approximation 23 as a “bound-approximation”, in order to differentiate it from the stochastic error estimate presented next.

Just as a stochastic analysis of message error gave a tighter estimate for the pointwise difference measure, we may obtain an alternate Chebyshev-like “bound” by assuming that the message perturbations are uncorrelated (already an assumption of the KL additivity analysis) and that we require only an estimate which exceeds the expected error with high probability.

Approximation 24. *Under the same assumptions as Approximation 23, but describing the error in terms of its variance and assuming that these errors are uncorrelated gives the estimate*

$$E\left[\mathcal{D}(\mathcal{M}_t \| \hat{\mathcal{M}}_t^n)^2\right] \leq \sum_{u \in \Gamma_t} (\sigma_{ut}^{n-1})^2$$

where $(\sigma_{ts}^0)^2 = C$ *and*

$$(\sigma_{ts}^i)^2 = \sum_{u \in \Gamma_t \setminus s} ((1-\gamma_{ut})\sigma_{ut}^{i-1})^2 + \delta^2.$$

Comment. The argument proceeds similarly to Proposition 17, by induction on the claim that $(\sigma_{ut}^i)^2$ bounds the variance at each iteration *i*. This again applies Theorem 29 ignoring any effects due to loops, as well as the assumption that the message errors are uncorrelated (implying additivity of the variances of each incoming message). As in Section 5.5, we take the 2σ value as our performance estimate.

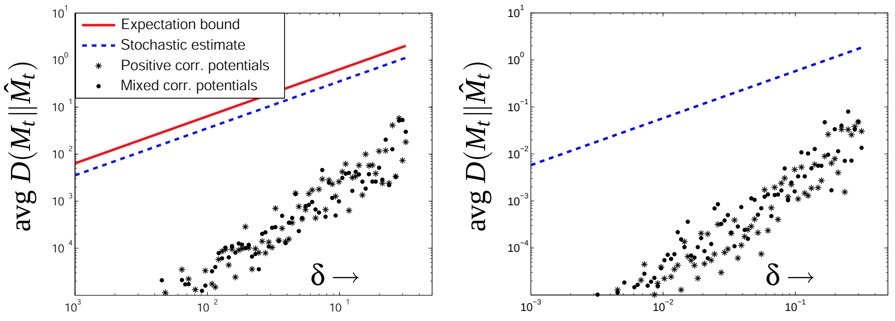
6.4 Experiments

Once again, we demonstrate the utility of these two estimates on the same uniform grids used in Section 5.6. Specifically, we generate 200 example realizations of a 5×5 binary grid and its observation potentials (100 with strictly attractive potentials and 100 with mixed potentials), and compare a quantized version of loopy BP with the solution obtained by exact loopy BP, as a function of KL-divergence bound δ incurred by the quantization level ε (see Lemma 18).

Figure 8(a) shows the maximum KL-divergence from the correct fixed point resulting in each Monte Carlo trial for a grid with relatively weak potentials (in which loopy BP is analytically guaranteed to converge). As can be seen, both the bound (solid) and stochastic estimate (dashed) still provide conservative estimates of the expected error. In Figure 8(b) we repeat the same analysis but with stronger pairwise potentials (for which convergence to a unique solution is not guaranteed but typically occurs in practice). In this case, the bound-based estimate of KL-divergence is trivially infinite—its linear rate of contraction is insufficient to overcome the accumulation rate. However, the greater sub-additivity in the stochastic estimate leads to the non-trivial curve shown (dashed), which still provides a reasonable (and still conservative) estimate of the performance in practice.

7. Conclusions and Future Directions

We have described a framework for the analysis of belief propagation stemming from the view that the message at each iteration is some noisy or erroneous version of some true BP fixed point. By measuring and bounding the error at each iteration, we may analyze the behavior of various forms of BP and test for convergence to the ideal fixed-point messages, or bound the total error from any such fixed point.



(a) $\log d(\psi)^2 = .25$ (b) $\log d(\psi)^2 = 1$

Figure 8: KL-divergence of the beliefs as a function of the added message error δ. The scatterplots indicates the average error measured in the graph for each of 200 Monte Carlo runs, along with the expected divergence bound (solid) and 2σ stochastic estimate (dashed). For stronger potentials, the upper bound may be trivially infinite; in this example the stochastic estimate still gives a reasonable gauge of performance.

.

In order to do so, we introduced a measure of the pointwise dynamic range, which represents a strong condition on the agreement between two messages; after showing its utility for common inference tasks such as MAP estimation and its transference to other common measures of error, we showed that under this measure the influence of message errors is both sub-additive and measurably contractive. These facts led to conditions under which traditional belief propagation may be shown to converge to a unique fixed point, and more generally a bound on the distance between any two fixed points. Furthermore, it enabled analysis of quantized, stochastic, or other approximate forms of belief propagation, yielding conditions under which they may be guaranteed to converge to some unique region, as well as bounds on the ensuing error over exact BP. If we further assume that the message perturbations are uncorrelated, we obtain an alternate, tighter estimate of the resulting error.

The second measure considered an average case error similar to the Kullback-Liebler divergence, in expectation over the possible realizations of observations within the graph. While this gives no guarantees about any particular realization, the difference measure itself is able to be much less strict (allowing poor approximations in the distribution tails, for example). Analysis of this case is substantially more difficult and leads to approximations rather than guarantees, but explains some of the observed similarities in behavior among the two forms of perturbed BP. Simulations indicate that these estimates remain sufficiently accurate to be useful in practice.

Further analysis of the propagation of message errors has the potential to give an improved understanding of when and why BP converges (or fails to converge), and potentially also the role of the message schedule in determining the performance. Additionally, there are many other possible measures of the deviation between two messages, any of which may be able to provide an alternative set of bounds and estimates on performance of BP using either exact or approximate messages.

Acknowledgments

The authors would like to thank Tom Heskes, Martin Wainwright, Erik Sudderth, and Lei Chen for many helpful discussions. Thanks also to the anonymous reviewers of JMLR for their insightful comments, and for suggesting the improvement of equation (13). This research was supported in part by AFOSR grant F49620-00-0362 and by ODDR&E MURI through ARO grant DAAD19-00-0466. Portions of this work have appeared as a conference paper (Ihler et al., 2004b).

Appendix A. Proof of Theorem 8

Because all quantities in this section refer to the pair (t,s), we suppress the subscripts. The error measure $d(e)$ is given by

$$d(e)^2 = d(\hat{m}/m)^2 = \max_{a,b} \frac{\int \psi(x_t,a)M(x_t)E(x_t)dx_t}{\int \psi(x_t,a)M(x_t)dx_t} \cdot \frac{\int \psi(x_t,b)M(x_t)dx_t}{\int \psi(x_t,b)M(x_t)E(x_t)dx_t} \tag{29}$$

subject to a few constraints: positivity of the messages and potential functions, normalization of the message product M, and the definitions of $d(E)$ and $d(\psi)$. In order to analyze the maximum possible value of $d(e)$ for any functions ψ, M, and E, we make repeated use of the following property:

Lemma 25. *For f_1, f_2, g_1, g_2 all positive,*

$$\frac{f_1+f_2}{g_1+g_2} \le \max\left[\frac{f_1}{g_1}, \frac{f_2}{g_2}\right].$$

Proof. Assume without loss of generality that $f_1/g_1 \ge f_2/g_2$. Then we have $f_1/g_1 \ge f_2/g_2 \Rightarrow f_1g_2 \ge f_2g_1 \Rightarrow f_1g_1 + f_1g_2 \ge f_1g_1 + f_2g_1 \Rightarrow \frac{f_1}{g_1} \ge \frac{f_1+f_2}{g_1+g_2}$. □

This fact, extended to more general sums, may be applied directly to (29) to prove Corollary 9. However, a more careful application leads to the result of Theorem 8. The following lemma will assist us:

Lemma 26. *The maximum of $d(e)$ with respect to $\psi(x_t,a)$, $\psi(x_t,b)$, and $E(x_t)$ is attained at some extremum of their feasible function space. Specifically,*

$$\psi(x,a) = 1 + (d(\psi)^2 - 1)\chi_A(x) \qquad E(x) = 1 + (d(E)^2 - 1)\chi_E(x)$$
$$\psi(x,b) = 1 + (d(\psi)^2 - 1)\chi_B(x)$$

where χ_A, χ_B, and χ_E are indicator functions taking on only values 0 and 1.

Proof. We simply show the result for $\psi(x,a)$; the proofs for $\psi(x,b)$ and $E(x)$ are similar. First, observe that without loss of generality we may scale $\psi(x,a)$ so that its minimum value is 1. Now consider a convex combination of any two possible functions: let $\psi(x_t,a) = \alpha_1\psi_1(x_t,a) + \alpha_2\psi_2(x_t,a)$ with $\alpha_1 \ge 0$, $\alpha_2 \ge 0$, and $\alpha_1 + \alpha_2 = 1$. Then, applying Lemma 25 to the left-hand term of (29) we have

$$\frac{\alpha_1 \int \psi_1(x_t,a)M(x_t)E(x_t)dx_t + \alpha_2 \int \psi_2(x_t,a)M(x_t)E(x_t)dx_t}{\alpha_1 \int \psi_1(x_t,a)M(x_t)dx_t + \alpha_2 \int \psi_2(x_t,a)M(x_t)dx_t}$$
$$\le \max\left[\frac{\int \psi_1(x_t,a)M(x_t)E(x_t)dx_t}{\int \psi_1(x_t,a)M(x_t)dx_t}, \frac{\int \psi_2(x_t,a)M(x_t)E(x_t)dx_t}{\int \psi_2(x_t,a)M(x_t)dx_t}\right]. \tag{30}$$

Thus, $d(e)$ is maximized by taking whichever of ψ_1, ψ_2 results in the largest value—an extremum. It remains only to describe the form of such a function extremum. Any potential $\psi(x,a)$ may be considered to be the convex combination of functions of the form $\left(d(\psi)^2 - 1\right)\chi(x) + 1$, where χ takes on values $\{0,1\}$. This can be seen by the construction

$$\psi(x,a) = \int_0^1 \left(d(\psi)^2 - 1\right)\chi_m^y(x,a) + 1 \; dy$$

$$\text{where} \qquad \chi_m^y(x,a) = \begin{cases} 1 & \psi(x,a) \ge 1 + (d(\psi)^2 - 1)y \\ 0 & \text{otherwise.} \end{cases}$$

Thus, the maximum value of $d(e)$ will be attained by a potential equal to one of these functions. $\square$

Applying Lemma 26, we define the shorthand

$$\begin{aligned} M_A &= \int M(x)\chi_A(x) & M_B &= \int M(x)\chi_B(x) & M_E &= \int M(x)\chi_E(x) \\ M_{AE} &= \int M(x)\chi_A(x)\chi_E(x) & M_{BE} &= \int M(x)\chi_B(x)\chi_E(x) & & \\ \alpha &= d(\psi)^2 - 1 & \beta &= d(E)^2 - 1, & & \end{aligned}$$

giving

$$d(e)^2 \le \max_M \frac{1 + \alpha M_A + \beta M_E + \alpha\beta M_{AE}}{1 + \alpha M_B + \beta M_E + \alpha\beta M_{BE}} \cdot \frac{1 + \alpha M_B}{1 + \alpha M_A}.$$

Using the same argument outlined by Equation 30, one may argue that the scalars M_{AE}, M_{BE}, M_A, and M_B must also be extremum of their constraint sets. Noticing that M_{AE} should be large and M_{BE} small, we may summarize the constraints by

$$0 \le M_A,\, M_B,\, M_E \le 1 \qquad M_{AE} \le \min[M_A,\, M_E] \qquad M_{BE} \ge \max[0,\, M_E - (1 - M_B)]$$

(where the last constraint arises from the fact that $M_E + M_B - M_{BE} \le 1$). We then consider each possible case: $M_A \le M_E$, $M_A \ge M_E$, ... In each case, we find that the maximum is found at the extrema $M_{AE} = M_A = M_E$ and $M_E = 1 - M_B$. This gives

$$d(e)^2 \le \max_M \frac{1 + (\alpha + \beta + \alpha\beta)M_E}{1 + \alpha + (\beta - \alpha)M_E} \cdot \frac{1 + \alpha - \alpha M_E}{1 + \alpha M_E}.$$

The maximum with respect to M_E (whose optimum is not an extreme point) is given by taking the derivative and setting it to zero. This procedure gives a quadratic equation; solving and selecting the positive solution gives $M_E = \frac{1}{\beta}(\sqrt{\beta + 1} - 1)$. Finally, plugging in, simplifying, and taking the square root yields

$$d(e) \le \frac{d(\psi)^2 d(E) + 1}{d(\psi)^2 + d(E)}.$$

$\square$

Appendix B. Properties of the Expected Divergence

We begin by examining the properties of the expected divergence (25) on tree-structured graphical models parameterized by (21)-(22); we discuss the application of these results to graphs with cycles in Appendix B.4. Recall that, for tree-structured models described by (21)-(22), the prior-weight normalized messages of the (unique) fixed point are equivalent to

$$m_{ut}(x_t) = p(\mathbf{y}_{ut}|x_t)/p(\mathbf{y}_{ut}),$$

and that the message products are given by

$$M_{ts}(x_t) = p(x_t|\mathbf{y}_{ts})M_t(x_t) = p(x_t|\mathbf{y}).$$

Furthermore, let us define the *approximate* messages $\hat{m}_{ut}(x)$ in terms of some approximate likelihood function, i.e., $\hat{m}_{ut}(x) = \hat{p}(\mathbf{y}_{ut}|x_t)/\hat{p}(\mathbf{y}_{ut})$ where $\hat{p}(\mathbf{y}_{ut}) = \int \hat{p}(\mathbf{y}_{ut}|x_t)p(x_t)dx_t$. We may then examine each of the three properties in turn: the triangle inequality, additivity, and contraction.

B.1 Triangle Inequality

Kullback-Leibler divergence is not a true distance, and in general, it does not satisfy the triangle inequality. However, the following generalization does hold.

Theorem 27. *For a tree-structured graphical model parameterized as in* (21)-(22), *and given the true BP message $m_{ut}(x_t)$ and two approximations $\hat{m}_{ut}(x_t)$, $\tilde{m}_{ut}(x_t)$, suppose that $m_{ut}(x_t) \le c_{ut}\hat{m}_{ut}(x_t)\ \forall x_t$. Then,*

$$\mathcal{D}(m_{ut}\|\tilde{m}_{ut}) \le \mathcal{D}(m_{ut}\|\hat{m}_{ut}) + c_{ut}\mathcal{D}(\hat{m}_{ut}\|\tilde{m}_{ut})$$

and furthermore, if $\hat{m}_{ut}(x_t) \le c^*_{ut}\tilde{m}_{ut}(x_t)\ \forall x_t$, then $m_{ut}(x_t) \le c_{ut}c^*_{ut}\tilde{m}_{ut}(x_t)\ \forall x_t$.

Comment. Since $m, \hat{m}$ are prior-weight normalized ($\int p(x)m(x) = \int p(x)\hat{m}(x) = 1$), for a strictly positive prior $p(x)$ we see that $c_{ut} \ge 1$, with equality if and only if $m_{ut}(x) = \hat{m}_{ut}(x)\ \forall x$. However, this is often quite conservative and Approximation 20 ($c_{ut} = 1$) is sufficient to estimate the resulting error. Moreover, we shall see that the constants $\{c_{ut}\}$ are also affected by the product operation, described next.

B.2 Near-Additivity

For BP fixed-point messages $\{m_{ut}(x_t)\}$, approximated by the messages $\{\hat{m}_{ut}(x_t)\}$, the resulting error is not quite guaranteed to be sub-additive, but is almost so.

Theorem 28. *The expected error $E[\mathcal{D}(M_t\|\hat{M}_t)]$ between the true and approximate beliefs is nearly sub-additive; specifically,*

$$E\left[\mathcal{D}(M_t\|\hat{M}_t)\right] \le \sum_{u\in\Gamma_t} E\left[\mathcal{D}(m_{ut}\|\hat{m}_{ut})\right] + \left(\hat{I} - I\right) \tag{31}$$

$$\textit{where} \qquad I = E\left[\log\ p(\mathbf{y})/\prod_{u\in\Gamma_t} p(\mathbf{y}_{ut})\right] \qquad \textit{and} \qquad \hat{I} = E\left[\log\ \hat{p}(\mathbf{y})/\prod_{u\in\Gamma_t} \hat{p}(\mathbf{y}_{ut})\right].$$

Moreover, if $m_{ut}(x_t) \le c_{ut}\hat{m}_{ut}(x_t)$ for all x_t and for each $u \in \Gamma_t$, then

$$M_t(x_t) \le \prod_{u\in\Gamma_t} c_{ut} C_t^* \hat{M}_t(x_t) \qquad\qquad C_t^* = \frac{\hat{p}(\mathbf{y})}{\prod_{u\in\Gamma_t}\hat{p}(\mathbf{y}_{ut})}\frac{\prod_{u\in\Gamma_t} p(\mathbf{y}_{ut})}{p(\mathbf{y})} \tag{32}$$

Proof. By definition we have

$$E[\mathcal{D}(M_t\|\hat{M}_t)] = E\left[\int p(x_t,\mathbf{y})\log\frac{M_t(x_t)}{\hat{M}_t(x_t)}dx_t\right] = E\left[\int p(x_t|\mathbf{y})\log\frac{p(x_t)}{p(x_t)}\frac{p(\mathbf{y}|x_t)}{\hat{p}(\mathbf{y}|x_t)}\frac{\hat{p}(\mathbf{y})}{p(\mathbf{y})}dx_t\right].$$

Using the Markov property of (21) to factor $p(\mathbf{y}|x_t)$, we have

$$= E\left[\int p(x_t|\mathbf{y})\sum_{u\in\Gamma_t}\log\frac{p(\mathbf{y}_{ut}|x_t)}{\hat{p}(\mathbf{y}_{ut}|x_t)} + p(x_t|\mathbf{y})\log\frac{\hat{p}(\mathbf{y})}{p(\mathbf{y})}dx_t\right]$$

and, applying the identity $m_{ut}(x_t) = p(\mathbf{y}_{ut}|x_t)/p(\mathbf{y}_{ut})$ gives

$$\begin{aligned}
&= \sum_{u\in\Gamma_t} E\left[\int p(x_t|\mathbf{y})\log\frac{m_{ut}(x_t)}{\hat{m}_{ut}(x_t)}\right] + E\left[\log\frac{\hat{p}(\mathbf{y})}{\prod_u \hat{p}(\mathbf{y}_{ut})}\frac{\prod_u p(\mathbf{y}_{ut})}{p(\mathbf{y})}\right]dx_t \\
&= \sum_{u\in\Gamma_t} E\left[\mathcal{D}(m_{ut}\|\hat{m}_{ut})\right] + (\hat{I} - I)
\end{aligned}$$

where $\hat{I}$, I are as defined. Here, I is the mutual information (the divergence from independence) of the variables $\{\mathbf{y}_{ut}\}_{u\in\Gamma_t}$. Equation (32) follows from a similar argument. □

Unfortunately, it is *not* the case that the quantity $\hat{I} - I$ must necessarily be less than or equal to zero. To see how it may be positive, consider the following example. Let $x = [x_a, x_b]$ be a two-dimensional binary random variable, and let y_a and y_b be observations of the specified dimension of x. Then, if y_a and y_b are independent ($I = 0$), the true messages $m_a(x)$ and $m_b(x)$ have a regular structure; in particular, m_a and m_b have the forms $[p_1 p_2 p_1 p_2]$ and $[p_3 p_3 p_4 p_4]$ for some $p_1, \ldots, p_4$. However, we have placed no such requirements on the message *errors* $\hat{m}/m$; they have the potentially arbitrary forms $e_a = [e_1 e_2 e_3 e_4]$, *etc.*. If either message error e_a, e_b does *not* have the same structure as m_a, m_b respectively (even if they are random and independent), then $\hat{I}$ will in general not be zero. This creates the *appearance* of information between y_a and y_b, and the KL-divergence will not be strictly sub-additive.

However, this is not a typical situation. One may argue that in most problems of interest, the information I between observations is non-zero, and the types of message perturbations [particularly random errors, such as appear in stochastic versions of BP (Sudderth et al., 2003; Isard, 2003; Koller et al., 1999)] tend to degrade this information on average. Thus, is is reasonable to assume that $\hat{I} \le I$.

A similar quantity defines the multiplicative constant C_t^* in (32). When $C_t^* \le 1$, it acts to reduce the constant which bounds M_t by $\hat{M}_t$; if this occurs "typically", it lends additional support for Approximation (20). Moreover, if $E[C_t^*] \le 1$, then by Jensen's inequality, we have $\hat{I} - I \le 0$, ensuring sub-additivity as well.

B.3 Contraction

Analysis of the contraction of expected KL-divergence is also non-trivial; however, the work of Boyen and Koller (1998) has already considered this problem in some depth for the specific case of directed Markov chains (in which additivity issues do not arise) and projection-based approximations (for which KL-divergence does satisfy a form of the triangle inequality). We may directly apply their findings to construct Approximation 22.

Theorem 29. *On a tree-structured graphical model parameterized as in* (21)-(22)*, the error measure* $\mathcal{D}(M,\hat{M})$ *satisfies the inequality*

$$E\left[\mathcal{D}(m_{ts}\|\hat{m}_{ts})\right] \le (1-\gamma_{ts})E\left[\mathcal{D}(M_{ts}\|\hat{M}_{ts})\right]$$

$$\textit{where} \qquad \gamma_{ts} = \min_{a,b}\int \min\left[p(x_s|x_t=a)\,,\,p(x_s|x_t=b)\right]dx_s.$$

Proof. For a detailed development, see Boyen and Koller (1998); we merely sketch the proof here. First, note that

$$\begin{aligned}
E\left[\mathcal{D}(m_{ts}\|\hat{m}_{ts})\right] &= E\left[\int p(x_s|\mathbf{y})\log\frac{p(\mathbf{y}_{ts}|x_s)}{p(\mathbf{y}_{ts})}\frac{\hat{p}(\mathbf{y}_{ts})}{\hat{p}(\mathbf{y}_{ts}|x_s)}\right] \\
&= E\left[\int p(x_s|\mathbf{y}_{ts})\log\frac{p(x_s|\mathbf{y}_{ts})}{\hat{p}(x_s|\mathbf{y}_{ts})}\right] \\
&= E\left[D(p(x_s|\mathbf{y}_{ts})\|\hat{p}(x_s|\mathbf{y}_{ts}))\right]
\end{aligned}$$

(which is the quantity considered by Boyen and Koller, 1998) and further that

$$p(x_s|\mathbf{y}_{ts}) = \int p(x_s|x_t)p(x_t|\mathbf{y}_{ts})dx_t.$$

By constructing two valid conditional distributions $f_1(x_s|x_t)$ and $f_2(x_s|x_t)$ such that f_1 has the form $f_1(x_s|x_t) = f_1(x_s)$ (independence of x_s, x_t), and

$$p(x_s|x_t) = \gamma_{ts}f_1(x_s|x_t) + (1-\gamma_{ts}f_2(x_s|x_t)$$

one may use the convexity of KL-divergence to show

$$\begin{aligned}
D(p(x_s|\mathbf{y}_{ts})\|\hat{p}(x_s|\mathbf{y}_{ts})) \le \gamma_{ts}D(f_1 * p(x_t|\mathbf{y}_{ts})\|f_1 * \hat{p}(x_t|\mathbf{y}_{ts}))+ \\
(1-\gamma_{ts})D(f_2 * p(x_t|\mathbf{y}_{ts})\|f_2 * \hat{p}(x_t|\mathbf{y}_{ts}))
\end{aligned}$$

where "$*$" denotes convolution, i.e., $f_1 * p(x_t|\mathbf{y}_{ts}) = \int f_1(x_s|x_t)p(x_t|\mathbf{y}_{ts})dx_t$. Since the conditional f_1 induces independence between x_s and x_t, the first divergence term is zero, and since f_2 is a valid conditional distribution, the second divergence term is less than $D(p(x_t|\mathbf{y}_{ts})\|\hat{p}(x_t|\mathbf{y}_{ts}))$ (see Cover and Thomas, 1991). Thus we have a minimum rate of contraction of $(1-\gamma_{ts})$. $\square$

It is worth noting that Theorem 29 gives a *linear* contraction rate. While this makes for simpler recurrence relations than the nonlinear contraction found in Section 4.2, it has the disadvantage that, if the rate of error addition exceeds the rate of contraction it may result in a trivial (infinite) bound. Theorem 29 is the best contraction rate currently known for arbitrary conditional distributions, although certain special cases (such as binary–valued random variables) appear to admit stronger contractions.

B.4 Graphs with Cycles

The analysis and discussion of each property (Appendices B.1–B.3) also relied on assuming a tree-structured graphical model, and using the direct relationship between messages and likelihood functions for the parameterization (21)-(22). However, for BP on general graphs, this parameterization is not valid.

One way to generalize this choice is given by the re-parameterization around some fixed point of loopy BP on the graphical model of the prior. If the original potentials $\tilde{\psi}_{st}, \tilde{\psi}_s^x$ specify the prior distribution [cf. (22)],

$$p(\mathbf{x}) \propto \prod_{(s,t)\in\mathcal{E}} \tilde{\psi}_{st}(x_s,x_t) \tilde{\prod}_s \psi_s^x(x_s) \tag{33}$$

then given a BP fixed point $\{\tilde{M}_{st}, \tilde{M}_s\}$ of (33), we may choose a new parameterization of the same prior ψ_{st}, ψ_s^x given by

$$\psi_{st}(x_s,x_t) = \frac{\tilde{M}_{st}(x_s)\tilde{M}_{ts}(x_t)\tilde{\psi}_{st}(x_s,x_t)}{\tilde{M}_s(x_s)\tilde{M}_t(x_t)} \qquad \text{and} \qquad \psi_s^x(x_s) = \tilde{M}_s(x_s). \tag{34}$$

This parameterization ensures that uninformative messages $[m_{ut}(x_t) = 1 \; \forall x_t]$ comprise a fixed point for the graphical model of $p(\mathbf{x})$ as described by the new potentials $\{\psi_{st}, \psi_s\}$. For a tree-structured graphical model, this recovers the parameterization given by (22).

However, the messages of loopy BP are no longer precisely equal to the likelihood functions $m(x) = p(\mathbf{y}|x)/p(\mathbf{y})$, and thus the expectation applied in Theorem 28 is no longer consistent with the messages themselves. Additionally, the additivity and contraction statements were developed under the assumption that the observed data $\mathbf{y}$ along different branches of the tree are conditionally *independent*; in graphs with cycles, this is not the case. In the computation tree formalism, instead of being conditionally independent, the observations $\mathbf{y}$ actually *repeat* throughout the tree.

However, the assumption of independence is precisely the same assumption applied by loopy belief propagation itself to perform tractable approximate inference. Thus, for problems in which loopy BP is well-behaved and results in answers similar to the true posterior distributions, we may expect our estimates of belief error to be similarly incorrect but near to the true divergence.

In short, all three properties required for a strict analysis of the propagation of errors in BP fail, in one sense or another, for graphs with cycles. However, for many situations of practical interest, they are quite close to the real average-case behavior. Thus we may expect that our approximations give rise to reasonable estimates of the total error incurred by approximate loopy BP, an intuition which appears to be borne out in our simulations (Section 6.4).

References

X. Boyen and D. Koller. Tractable inference for complex stochastic processes. In *Uncertainty in Artificial Intelligence*, pages 33–42, 1998.

H. Chan and A. Darwiche. A distance measure for bounding probabilistic belief change. *International Journal of Approximate Reasoning*, 38(2):149–174, Feb 2005.

L. Chen, M. Wainwright, M. Cetin, and A. Willsky. Data association based on optimization in graphical models with application to sensor networks. Submitted to *Mathematical and Computer Modeling*, 2004.

P. Clifford. Markov random fields in statistics. In G. R. Grimmett and D. J. A. Welsh, editors, *Disorder in Physical Systems*, pages 19–32. Oxford University Press, Oxford, 1990.

J. M. Coughlan and S. J. Ferreira. Finding deformable shapes using loopy belief propagation. In *European Conference on Computer Vision 7*, May 2002.

H. Georgii. *Gibbs measures and phase transitions*. Studies in Mathematics. de Gruyter, Berlin / New York, 1988.

A. Gersho and R. M. Gray. *Vector quantization and signal compression*. Kluwer, Boston, 1991.

T. Heskes. On the uniqueness of loopy belief propagation fixed points. *Neural Computation*, 16(11):2379–2413, 2004.

A. T. Ihler, J. W. Fisher III, and A. S. Willsky. Communication-constrained inference. Technical Report 2601, MIT, Laboratory for Information and Decision Systems, 2004a.

A. T. Ihler, J. W. Fisher III, and A. S. Willsky. Message errors in belief propagation. In *Neural Information Processing Systems*, 2004b.

M. Isard. PAMPAS: Real–valued graphical models for computer vision. In *IEEE Computer Vision and Pattern Recognition*, 2003.

S. Julier and J. Uhlmann. A general method for approximating nonlinear transformations of probability distributions. Technical report, RRG, Dept. of Eng. Science, Univ. of Oxford, 1996.

D. Koller, U. Lerner, and D. Angelov. A general algorithm for approximate inference and its application to hybrid Bayes nets. In *Uncertainty in Artificial Intelligence 15*, pages 324–333, 1999.

F. Kschischang, B. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, February 2001.

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

T. Minka. Expecatation propagation for approximate bayesian inference. In *Uncertainty in Artificial Intelligence*, 2001.

M. A. Paskin and C. E. Guestrin. Robust probabilistic inference in distributed systems. In *Uncertainty in Artificial Intelligence 20*, 2004.

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

E. B. Sudderth, A. T. Ihler, W. T. Freeman, and A. S. Willsky. Nonparametric belief propagation. In *IEEE Computer Vision and Pattern Recognition*, 2003.

S. Tatikonda and M. Jordan. Loopy belief propagation and gibbs measures. In *Uncertainty in Artificial Intelligence*, 2002.

M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. Tree–based reparameterization analysis of sum–product and its generalizations. *IEEE Transactions on Information Theory*, 49(5), May 2003.

Y. Weiss. Correctness of local probability propagation in graphical models with loops. *Neural Computation*, 12(1), 2000.

A. Willsky. Relationships between digital signal processing and control and estimation theory. *Proceedings of the IEEE*, 66(9):996–1017, September 1978.

J. S. Yedidia, W. T. Freeman, and Y. Weiss. Constructing free energy approximations and generalized belief propagation algorithms. Technical Report 2004-040, MERL, May 2004.

Journal of Machine Learning Research 6 (2005) 937–965 Submitted 7/04; Revised 4/05; Published 6/05

Learning a Mahalanobis Metric from Equivalence Constraints

Aharon Bar-Hillel AHARONBH@CS.HUJI.AC.IL
Tomer Hertz TOMBOY@CS.HUJI.AC.IL
Noam Shental FENOAM@CS.HUJI.AC.IL
Daphna Weinshall DAPHNA@CS.HUJI.AC.IL
School of Computer Science & Engineering and Center for Neural Computation
The Hebrew University of Jerusalem
Jerusalem, Israel 91904

Editor: Greg Ridgeway

Abstract

Many learning algorithms use a metric defined over the input space as a principal tool, and their performance critically depends on the quality of this metric. We address the problem of learning metrics using side-information in the form of equivalence constraints. Unlike labels, we demonstrate that this type of side-information can sometimes be automatically obtained without the need of human intervention. We show how such side-information can be used to modify the representation of the data, leading to improved clustering and classification.

Specifically, we present the Relevant Component Analysis (RCA) algorithm, which is a simple and efficient algorithm for learning a Mahalanobis metric. We show that RCA is the solution of an interesting optimization problem, founded on an information theoretic basis. If dimensionality reduction is allowed within RCA, we show that it is optimally accomplished by a version of Fisher's linear discriminant that uses constraints. Moreover, under certain Gaussian assumptions, RCA can be viewed as a Maximum Likelihood estimation of the within class covariance matrix. We conclude with extensive empirical evaluations of RCA, showing its advantage over alternative methods.

Keywords: clustering, metric learning, dimensionality reduction, equivalence constraints, side information.

1. Introduction

A number of learning problems, such as clustering and nearest neighbor classification, rely on some a priori defined distance function over the input space. It is often the case that selecting a "good" metric critically affects the algorithms' performance. In this paper, motivated by the wish to boost the performance of these algorithms, we study ways to learn a "good" metric using side information.

One difficulty in finding a "good" metric is that its quality may be context dependent. For example, consider an image-retrieval application which includes many facial images. Given a query image, the application retrieves the most similar faces in the database according to some pre-determined metric. However, when presenting the query image we may be interested in retrieving other images of the same person, or we may want to retrieve other faces with the same facial expression. It seems difficult for a pre-determined metric to be suitable for two such different tasks.

In order to learn a context dependent metric, the data set must be augmented by some additional information, or side-information, relevant to the task at hand. For example we may have access to the labels of *part* of the data set. In this paper we focus on another type of side-information, in which *equivalence constraints* between a few of the data points are provided. More specifically we assume knowledge about small groups of data points that are known to originate from the same class, although their label is unknown. We term these small groups of points *"chunklets"*.

A key observation is that in contrast to explicit labels that are usually provided by a human instructor, in many unsupervised learning tasks equivalence constraints may be extracted with minimal effort or even automatically. One example is when the data is inherently sequential and can be modelled by a Markovian process. Consider for example movie segmentation, where the objective is to find all the frames in which the same actor appears. Due to the continuous nature of most movies, faces extracted from successive frames in roughly the same location can be assumed to come from the same person. This is true as long as there is no scene change, which can be robustly detected (Boreczky and Rowe, 1996). Another analogous example is speaker segmentation and recognition, in which the conversation between several speakers needs to be segmented and clustered according to speaker identity. Here, it may be possible to automatically identify small segments of speech which are likely to contain data points from a single yet *unknown* speaker.

A different scenario, in which equivalence constraints are the natural source of training data, occurs when we wish to learn from several teachers who do not know each other and who are not able to coordinate among themselves the use of common labels. We call this scenario 'distributed learning'.[1] For example, assume that you are given a large database of facial images of many people, which cannot be labelled by a small number of teachers due to its vast size. The database is therefore divided (arbitrarily) into P parts (where P is very large), which are then given to P teachers to annotate. The labels provided by the different teachers may be inconsistent: as images of the same person appear in more than one part of the database, they are likely to be given different names. Coordinating the labels of the different teachers is almost as daunting as labelling the original data set. However, equivalence constraints can be easily extracted, since points which were given the same tag by a certain teacher are known to originate from the same class.

In this paper we study how to use equivalence constraints in order to learn an optimal Mahalanobis metric between data points. Equivalently, the problem can also be posed as learning a good representation function, transforming the data representation by the square root of the Mahalanobis weight matrix. Therefore we shall discuss the two problems interchangeably.

In Section 2 we describe the proposed method–the Relevant Component Analysis (RCA) algorithm. Although some of the interesting results can only be proven using explicit Gaussian assumptions, the optimality of RCA can be shown with some relatively weak assumptions, restricting the discussion to linear transformations and the Euclidean norm. Specifically, in Section 3 we describe a novel information theoretic criterion and show that RCA is its optimal solution. If Gaussian assumptions are added the result can be extended to the case where dimensionality reduction is permitted, and the optimal solution now includes Fisher's linear discriminant (Fukunaga, 1990) as an intermediate step. In Section 4 we show that RCA is also the optimal solution to another optimization problem, seeking to minimize within class distances. Viewed this way, RCA is directly compared to another recent algorithm for learning Mahalanobis distance from equivalence constraints, proposed by Xing et al. (2002). In Section 5 we show that under Gaussian assumptions RCA can be interpreted as the maximum-likelihood (ML) estimator of the within class covariance matrix. We also provide a bound over the variance of this estimator, showing that it is at most twice the variance of the ML estimator obtained using the fully labelled data.

1. A related scenario (which we call 'generalized relevance feedback'), where users of a retrieval engine are asked to annotate the retrieved set of data points, has similar properties.

The successful application of RCA in high dimensional spaces requires dimensionality reduction, whose details are discussed in Section 6. An online version of the RCA algorithm is presented in Section 7. In Section 8 we describe extensive empirical evaluations of the RCA algorithm. We focus on two tasks–data retrieval and clustering, and use three types of data: (a) A data set of frontal faces (Belhumeur et al., 1997); this example shows that RCA with partial equivalence constraints typically yields comparable results to supervised algorithms which use fully labelled training data. (b) A large data set of images collected by a real-time surveillance application, where the equivalence constraints are gathered automatically. (c) Several data sets from the UCI repository, which are used to compare between RCA and other competing methods that use equivalence constraints.

1.1 Related Work

There has been much work on learning representations and distance functions in the supervised learning settings, and we can only briefly mention a few examples. Hastie and Tibshirani (1996) and Jaakkola and Haussler (1998) use labelled data to learn good metrics for classification. Thrun (1996) learns a distance function (or a representation function) for classification using a "leaning-to-learn" paradigm. In this setting several related classification tasks are learned using several labelled data sets, and algorithms are proposed which learn representations and distance functions in a way that allows for the transfer of knowledge between the tasks. In the work of Tishby et al. (1999) the joint distribution of two random variables X and Z is assumed to be known, and one seeks a compact representation of X which bears high relevance to Z. This work, which is further developed in Chechik and Tishby (2003), can be viewed as supervised representation learning.

As mentioned, RCA can be justified using information theoretic criteria on the one hand, and as an ML estimator under Gaussian assumptions on the other. Information theoretic criteria for unsupervised learning in neural networks were studied by Linsker (1989), and have been used since in several tasks in the neural network literature. Important examples are self organizing neural networks (Becker and Hinton, 1992) and Independent Component Analysis (Bell and Sejnowski, 1995)). Viewed as a Gaussian technique, RCA is related to a large family of feature extraction techniques that rely on second order statistics. This family includes, among others, the techniques of Partial Least-Squares (PLS) (Geladi and Kowalski, 1986), Canonical Correlation Analysis (CCA) (Thompson, 1984) and Fisher's Linear Discriminant (FLD) (Fukunaga, 1990). All these techniques extract linear projections of a random variable X, which are relevant to the prediction of another variable Z in various settings. However, PLS and CCA are designed for regression tasks, in which Z is a continuous variable, while FLD is used for classification tasks in which Z is discrete. Thus, RCA is more closely related to FLD, as theoretically established in Section 3.3. An empirical investigation is offered in Section 8.1.3, in which we show that RCA can be used to enhance the performance of FLD in the fully supervised scenario.

In recent years some work has been done on using equivalence constraints as side information. Both positive ('a is similar to b') and negative ('a is dissimilar from b') equivalence constraints were considered. Several authors considered the problem of semi-supervised clustering using equivalence constraints. More specifically, positive and negative constraints were introduced into the complete linkage algorithm (Klein et al., 2002), the K-means algorithm (Wagstaff et al., 2001) and the EM of a Gaussian mixture model (Shental et al., 2004). A second line of research, to which this work belongs, focuses on learning a 'good' metric using equivalence constraints. Learning a Mahalanobis metric from both positive and negative constraints was addressed in the work of Xing et al. (2002), presenting an algorithm which uses gradient ascent and iterative projections to solve a convex non linear optimization problem. We compare this optimization problem to the one solved by RCA in Section 4, and empirically compare the performance of the two algorithms in Section 8. The initial description of RCA was given in the context of image retrieval (Shental et al., 2002), followed by the work of Bar-Hillel et al. (2003). Recently Bilenko et al. (2004) suggested a K-means based clustering algorithm that also combines metric learning. The algorithm uses both positive and negative constraints and learns a single or multiple Mahalanobis metrics.

2. Relevant Component Analysis: The Algorithm

Relevant Component Analysis (RCA) is a method that seeks to identify and down-scale global unwanted variability within the data. The method changes the feature space used for data representation, by a global linear transformation which assigns large weights to "relevant dimensions" and low weights to "irrelevant dimensions" (see Tenenbaum and Freeman, 2000). These "relevant dimensions" are estimated using *chunklets*, that is, small subsets of points that are known to belong to the same although *unknown* class. The algorithm is presented below as Algorithm 1 (Matlab code can be downloaded from the authors' sites).

Algorithm 1 The RCA algorithm

Given a data set $X = \{x_i\}_{i=1}^N$ and n chunklets $C_j = \{x_{ji}\}_{i=1}^{n_j}$ $j = 1 \dots n$, do

1. Compute the within chunklet covariance matrix (Figure 1d)

$$\hat{C} = \frac{1}{N} \sum_{j=1}^{n} \sum_{i=1}^{n_j} (x_{ji} - m_j)(x_{ji} - m_j)^t, \quad (1)$$

   where m_j denotes the mean of the j'th chunklet.

2. If needed, apply dimensionality reduction to the data using $\hat{C}$ as described in Algorithm 2 (see Section 6).

3. Compute the whitening transformation associated with $\hat{C}$: $W = \hat{C}^{-\frac{1}{2}}$ (Figure 1e), and apply it to the data points: $X_{new} = WX$ (Figure 1f), where X refers to the data points after dimensionality reduction when applicable. Alternatively, use the inverse of $\hat{C}$ in the Mahalanobis distance: $d(x_1, x_2) = (x_1 - x_2)^t \hat{C}^{-1} (x_1 - x_2)$.

More specifically, points x_1 and x_2 are said to be related by a positive constraint if it is known that both points share the same (unknown) label. If points x_1 and x_2 are related by a positive constraint, and x_2 and x_3 are also related by a positive constraint, then a chunklet $\{x_1, x_2, x_3\}$ is formed. Generally, chunklets are formed by applying transitive closure over the whole set of positive equivalence constraints.

The RCA transformation is intended to reduce clutter, so that in the new feature space, the inherent structure of the data can be more easily unravelled (see illustrations in Figure 1a-f). To this end, the algorithm estimates the within class covariance of the data $cov(X|Z)$ where X and Z describe the data points and their labels respectively. The estimation is based on positive equivalence constraints only, and does not use any explicit label information. In high dimensional data, the estimated matrix can be used for semi-supervised dimensionality reduction. Afterwards, the data set is whitened with respect to the estimated within class covariance matrix. The whitening transformation W (in Step 3 of Algorithm 1) assigns lower weights to directions of large variability, since this variability is mainly due to within class changes and is therefore “irrelevant” for the task of classification.

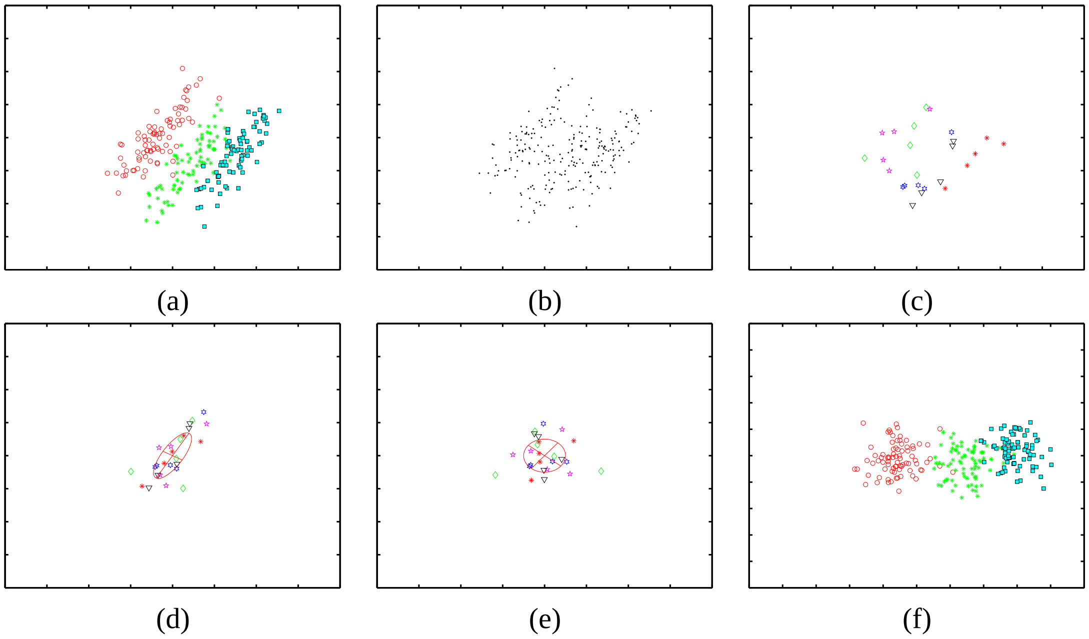


Figure 1: An illustrative example of the RCA algorithm applied to synthetic Gaussian data. (a) The fully labelled data set with 3 classes. (b) Same data unlabelled; clearly the classes' structure is less evident. (c) The set of chunklets that are provided to the RCA algorithm (points that share the same color and marker type form a chunklet). (d) The centered chunklets, and their empirical covariance. (e) The whitening transformation applied to the chunklets. (f) The original data after applying the RCA transformation.

The theoretical justifications for the RCA algorithm are given in Sections 3-5. In the following discussion, the term ‘RCA’ refers to the algorithm either with or without dimensionality reduction (optional Step 2). Usually the exact meaning can be readily understood in context. When we specifically discuss issues regarding the use of dimensionality reduction, we may use the explicit terms ‘RCA with (or without) dimensionality reduction’.

RCA does not use negative equivalence constraints. While negative constraints clearly contain useful information, they are less informative than positive constraints (see counting argument below). They are also much harder to use computationally, due partly to the fact that unlike positive constraints, negative constraints are not transitive. In our case, the naïve incorporation of negative constraints leads to a matrix solution which is the difference of two positive definite matrices, and as a results does not necessarily produce a legitimate Mahalanobis metric. An alternative approach, which modifies the optimization function to incorporate negative constraints, as used for example by Xing et al. (2002), leads to a non-linear optimization problem with the usual associated drawbacks of increased computational load and some uncertainty about the optimality of the final solution.[2] In contrast, RCA is the closed form solution of several interesting optimization problem, whose computation is no more complex than a single matrix inversion. Thus, in the tradeoff between runtime efficiency and asymptotic performance, RCA chooses the former and ignores the information given by negative equivalence constraints.

There is some evidence supporting the view that positive constraints are more informative than negative constraints. Firstly, a simple counting argument shows that positive constraints exclude more labelling possibilities than negative constraints. If for example there are M classes in the data, two data points have M^2 possible label combinations. A positive constraint between the points reduces this number to M combinations, while a negative constraint gives a much more moderate reduction to $M(M-1)$ combinations. (This argument can be made formal in information theoretic terms.) Secondly, empirical evidence from clustering algorithms which use both types of constraints shows that in most cases positive constraints give a much higher performance gain (Shental et al., 2004; Wagstaff et al., 2001). Finally, in most cases in which equivalence constraints are gathered automatically, only positive constraints can be gathered.

Step 2 of the RCA algorithm applies dimensionality reduction to the data if needed. In high dimensional spaces dimensionality reduction is almost always essential for the success of the algorithm, because the whitening transformation essentially re-scales the variability in all directions so as to equalize them. Consequently, dimensions with small total variability cause instability and, in the zero limit, singularity.

As discussed in Section 6, the optimal dimensionality reduction often starts with Principal Component Analysis (PCA). PCA may appear contradictory to RCA, since it eliminates principal dimensions with small variability, while RCA emphasizes principal dimensions with small variability. One should note, however, that the principal dimensions are computed in different spaces. The dimensions eliminated by PCA have small variability in the original data space (corresponding to $Cov(X)$), while the dimensions emphasized by RCA have low variability in a space where each point is translated according to the centroid of its own chunklet (corresponding to $Cov(X|Z)$). As a result, the method ideally emphasizes those dimensions with large total variance, but small within class variance.

3. Information Maximization with Chunklet Constraints

How can we use chunklets to find a transformation of the data which improves its representation? In Section 3.1 we state the problem for general families of transformations and distances, presenting an information theoretic formulation. In Section 3.2 we restrict the family of transformation to non-singular linear maps, and use the Euclidean metric to measure distances. The optimal solution is then given by RCA. In Section 3.3 we widen the family of permitted transformations to include non-invertible linear transformations. We show that for normally distributed data RCA is the optimal transformation when its dimensionality reduction is obtained with a constraints based Fisher's Linear Discriminant (FLD).

2. Despite the problem's convexity, the proposed gradient based algorithm needs tuning of several parameters, and is not guaranteed to find the optimum without such tuning. See Section 8.1.5 for relevant empirical results.

3.1 An Information Theoretic Perspective

Following Linsker (1989), an information theoretic criterion states that an optimal transformation of the input X into its new representation Y, should seek to maximize the mutual information $I(X,Y)$ between X and Y under suitable constraints. In the general case a set $X = \{x_i\}$ of data points in $\mathcal{R}^D$ is transformed into the set $Y = \{f(x_i)\}$ of points in $\mathcal{R}^K$. We seek a deterministic function $f \in F$ that maximizes $I(X,Y)$, where F is the family of permitted transformation functions (a "hypotheses family").

First, note that since f is deterministic, maximizing $I(X,Y)$ is achieved by maximizing the entropy $H(Y)$ alone. To see this, recall that by definition

$$I(X,Y) = H(Y) - H(Y|X)$$

where $H(Y)$ and $H(Y|X)$ are differential entropies, as X and Y are continuous random variables. Since f is deterministic, the uncertainty concerning Y when X is known is minimal, thus $H(Y|X)$ achieves its lowest possible value at $-\infty$.[3] However, as noted by Bell and Sejnowski (1995), $H(Y|X)$ does not depend on f and is constant for every finite quantization scale. Hence maximizing $I(X,Y)$ with respect to f can be done by considering only the first term $H(Y)$.

Second, note also that $H(Y)$ can be increased by simply 'stretching' the data space. For example, if $Y = f(X)$ for an invertible continuous function, we can increase $H(Y)$ simply by choosing $Y = \lambda f(X)$ for any $\lambda > 1$. In order to avoid the trivial solution $\lambda \to \infty$, we can limit the distances between points contained in a single chunklet . This can be done by constraining the average distance between a point in a chunklet and the chunklet's mean. Hence the optimization problem is

$$\max_{f \in F} H(Y_f) \qquad s.t. \;\; \frac{1}{N} \sum_{j=1}^{n} \sum_{i=1}^{n_j} ||y_{ji} - m_j^y|| \le \kappa \tag{2}$$

where $\{y_{ji}\}_{j=1,i=1}^{n\;,\;n_j}$ denote the set of points in n chunklets after the transformation, m_j^y denotes the mean of chunklet j after the transformation, and κ is a constant.

3.2 RCA: The Optimal Linear Transformation for the Euclidean Norm

Consider the general problem (2) for the family F of invertible linear transformations, and using the squared Euclidean norm to measure distances. Since f is invertible, the connection between the densities of $Y = f(X)$ and X is expressed by $p_y(y) = \frac{p_x(x)}{|J(x)|}$, where $|J(x)|$ is the Jacobian of the transformation. From $p_y(y)dy = p_x(x)dx$, it follows that $H(Y)$ and $H(X)$ are related as follows:

$$H(Y) = -\int_y p(y) \log p(y) dy = -\int_x p(x) \log \frac{p(x)}{|J(x)|} dx = H(X) + \langle \log |J(x)| \rangle_x .$$

For the linear map $Y = AX$ the Jacobian is constant and equals $|A|$, and it is the only term in $H(Y)$ that depends on the transformation A. Hence Problem (2) is reduced to

$$\max_A log|A| \qquad s.t. \;\; \frac{1}{N} \sum_{j=1}^{n} \sum_{i=1}^{n_j} ||y_{ji} - m_j^y||_2^2 \le \kappa .$$

3. This non-intuitive divergence is a result of the generalization of information theory to continuous variables, that is, the result of ignoring the discretization constant in the definition of differential entropy.

Multiplying a solution matrix A by $\lambda > 1$ increases both the $log|A|$ argument and the constrained sum of within chunklet distances. Hence the maximum is achieved at the boundary of the feasible region, and the constraint becomes an equality. The constant κ only determines the scale of the solution matrix, and is not important in most clustering and classification tasks, which essentially rely on relative distances. Hence we can set $\kappa = 1$ and solve

$$\max_A log|A| \quad s.t.\ \frac{1}{N}\sum_{j=1}^{n}\sum_{i=1}^{n_j}||y_{ji} - m_j^y||_2^2 = 1. \tag{3}$$

Let $B = A^tA$; since B is positive definite and $\log|A| = \frac{1}{2}\log|B|$, Problem (3) can be rewritten as

$$\max_{B\succ 0} log|B| \quad s.t.\ \frac{1}{N}\sum_{j=1}^{n}\sum_{i=1}^{n_j}||x_{ji} - m_j||_B^2 = 1, \tag{4}$$

where $||.||_B$ denotes the Mahalanobis distance with weight matrix B. The equivalence between the problems is valid since for any $B \succ 0$ there is an A such that $B = A^tA$, and so a solution to (4) gives us a solution to (3) (and vice versa).

The optimization problem (4) can be solved easily, since the constraint is linear in B. The solution is $B = \frac{1}{D}\hat{C}^{-1}$, where $\hat{C}$ is the average chunklet covariance matrix (1) and D is the dimensionality of the data space. This solution is identical to the Mahalanobis matrix compute by RCA up to a global scale factor, or in other words, RCA is a scaled solution of (4).

3.3 Dimensionality Reduction

We now solve the optimization problem (4) for the family of general linear transformations, that is, $Y = AX$ where $A \in \mathcal{M}_{K\times D}$ and $K \leq D$. In order to obtain workable analytic expressions, we assume that the distribution of X is a multivariate Gaussian, from which it follows that Y is also Gaussian with the entropy

$$H(Y) \quad = \quad \frac{D}{2}\log 2\pi e + \frac{1}{2}\log|\Sigma_y| = \frac{D}{2}\log 2\pi e + \frac{1}{2}\log|A\Sigma_x A^t|.$$

Following the same reasoning as in Section 3.2 we replace the inequality with equality and let $\kappa = 1$. Hence the optimization problem becomes

$$\max_A \log|A\Sigma_x A^t| \quad s.t.\ \frac{1}{N}\sum_{j=1}^{n}\sum_{i=1}^{n_j}||x_{ji} - m_j||_{A^tA}^2 = 1. \tag{5}$$

For a given target dimensionality K, the solution of the problem is Fisher linear discriminant (FLD),[4] followed by the whitening of the within chunklet covariance in the reduced space. A sketch of the proof is given in Appendix A. The optimal RCA procedure therefore includes dimensionality reduction. Since the FLD transformation is computed based on the estimated within chunklet covariance matrix, it is essentially a semi-supervised technique, as described in Section 6. Note that after the FLD step, the within class covariance matrix in the reduced space is always diagonal, and Step 3 of RCA amounts to the scaling of each dimension separately.

4. Fisher Linear Discriminant is a linear projection A from $\mathcal{R}^D$ to $\mathcal{R}^K$ with $K < D$, which maximizes the determinant ratio $\max_{A\in\mathcal{M}_{K\times D}} \frac{AS_tA^t}{AS_wA^t}$, where S_t and S_w denote the total covariance and the within class covariance respectively.

4. RCA and the Minimization of Within Class Distances

In order to gain some intuition about the solution provided by the information maximization criterion (2), let us look at the optimization problem obtained by reversing the roles of the maximization term and the constraint term in problem (4):

$$\min_B \frac{1}{N} \sum_{j=1}^{n} \sum_{i=1}^{n_j} ||x_{ji} - m_j||_B^2 \quad s.t. \;\; |B| \geq 1. \tag{6}$$

We interpret problem (6) as follows: a Mahalanobis distance B is sought, which minimizes the sum of all within chunklet squared distances, while $|B| \geq 1$ prevents the solution from being achieved by "shrinking" the entire space. Using the Kuhn-Tucker theorem, we can reduce (6) to

$$\min_B \sum_{j=1}^{n} \sum_{i=1}^{n_j} ||x_{ji} - m_j||_B^2 - \lambda \log |B| \quad s.t. \;\; \lambda \geq 0, \;\; \lambda \log |B| = 0. \tag{7}$$

Differentiating this Lagrangian shows that the minimum is given by $B = |\hat{C}|^{\frac{1}{D}} \hat{C}^{-1}$, where $\hat{C}$ is the average chunklet covariance matrix. Once again, the solution is identical to the Mahalanobis matrix in RCA up to a scale factor.

It is interesting, in this respect, to compare RCA with the method proposed recently by Xing et al. (2002). They consider the related problem of learning a Mahalanobis distance using side information in the form of pairwise constraints (Chunklets of size > 2 are not considered). It is assumed that in addition to the set of positive constraints Q_P, one is also given access to a set of negative constraints Q_N–a set of pairs of points known to be dissimilar. Given these sets, they pose the following optimization problem:

$$\min_B \sum_{(x_1,x_2)\in Q_P} ||x_1 - x_2||_B^2 \qquad s.t. \sum_{(x_1,x_2)\in Q_N} ||x_1 - x_2||_B \geq 1, \quad B \succeq 0. \tag{8}$$

This problem is then solved using gradient ascent and iterative projection methods.

In order to allow a clear comparison of RCA with (8), we reformulate the argument of (6) using only within chunklet pairwise distances. For each point x_{ji} in chunklet j we have

$$x_{ji} - m_j = x_{ji} - \frac{1}{n_j} \sum_{k=1}^{n_j} x_{jk} = \frac{1}{n_j} \sum_{k=1}^{n_j} (x_{ji} - x_{jk}).$$

Problem (6) can now be rewritten as

$$\min_B \frac{1}{N} \sum_{j=1}^{n} \frac{1}{n_j^2} \sum_{i=1}^{n_j} ||\sum (x_{ji} - x_{jk})||_B^2 \quad s.t. \;\; |B| \geq 1. \tag{9}$$

When only chunklets of size 2 are given, as in the case studied by Xing et al. (2002), (9) reduces to

$$\min_B \frac{1}{2N} \sum_{j=1}^{n} ||x_{j1} - x_{j2}||_B^2 \quad s.t. \;\; |B| \geq 1. \tag{10}$$

Clearly the minimization terms in problems (10) and (8) are identical up to a constant ($\frac{1}{2N}$). The difference between the two problems lies in the constraint term: the constraint proposed by Xing et al. (2002) uses pairs of dissimilar points, whereas the constraint in the RCA formulation affects global scaling so that the 'volume' of the Mahalanobis neighborhood is not allowed to shrink indefinitely. As a result Xing et al. (2002) are faced with a much harder optimization problem, resulting in a slower and less stable algorithm.

5. RCA and Maximum Likelihood: The Effect of Chunklet Size

We now consider the case where the data consists of several normally distributed classes sharing the same covariance matrix. Under the assumption that the chunklets are sampled i.i.d. and that points within each chunklet are also sampled i.i.d., the likelihood of the chunklets' distribution can be written as

$$\prod_{j=1}^{n}\prod_{i=1}^{n_j}\frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma|^{\frac{1}{2}}}\exp^{(-\frac{1}{2}(x_{ji}-m_j)^t\Sigma^{-1}(x_{ji}-m_j))}.$$

Writing the log-likelihood while neglecting constant terms and denoting $B=\Sigma^{-1}$, we obtain

$$\sum_{j=1}^{n}\sum_{i=1}^{n_j}||x_{ji}-m_j||_B^2-N\log|B|, \tag{11}$$

where N is the total number of points in chunklets. Maximizing the log-likelihood is equivalent to minimizing (11), whose minimum is obtained when B equals the RCA Mahalanobis matrix (1). Note, moreover, that (11) is rather similar to the Lagrangian in (7), where the Lagrange multiplier is replaced by the constant N. Hence, under Gaussian assumptions, the solution of Problem (7) is probabilistically justified by a maximum likelihood formulation.

Under Gaussian assumptions, we can further define an *unbiased* version of the RCA estimator. Assume for simplicity that there are N constrained data points divided into n chunklets of size k each. The *unbiased* RCA estimator can be written as

$$\hat{C}(n,k)=\frac{1}{n}\sum_{j=1}^{n}\frac{1}{k-1}\sum_{i=1}^{k}(x_{ji}-m_i)(x_{ji}-m_i)^t,$$

where $\hat{C}(n,k)$ denotes the empirical mean of the covariance estimators produced by each chunklet. It is shown in Appendix B that the variance of the elements $\hat{C}_{ij}$ of the estimating matrix is bounded by

$$Var(\hat{C}_{ij}(n,k))\leq(1+\frac{1}{k-1})Var(\hat{C}_{ij}(1,nk)), \tag{12}$$

where $\hat{C}_{ij}(1,nk)$ is the estimator when all the $N=nk$ points are known to belong to the same class, thus forming the best estimate possible from N points. This bound shows that the variance of the RCA estimator rapidly converges to the variance of the best estimator, even for chunklets of small size. For the smallest possible chunklets, of size 2, the variance is only twice as high as the best possible.

6. Dimensionality Reduction

As noted in Section 2, RCA may include dimensionality reduction. We now turn to address this issue in detail. Step 3 of the RCA algorithm decreases the weight of principal directions along which the within class covariance matrix is relatively high, and increases the weight of directions along which it is low. This intuition can be made precise in the following sense:

Denote by $\{\lambda^i\}_{i=1}^D$ the eigenvalues of the within class covariance matrix, and consider the squared distance between two points from the same class $||x_1-x_2||^2$. We can diagonalize the within class covariance matrix using an orthonormal transformation which does not change the distance. Therefore, let us assume without loss of generality that the covariance matrix is diagonal.

Before whitening, the average squared distance is $E[||x_1 - x_2||^2] = 2\sum_{j=1}^{D} \lambda^j$ and the average squared distance in direction i is $E[(x_1^i - x_2^i)^2] = 2\lambda^i$. After whitening these values become $2D$ and 2, respectively. Let us define the weight of dimension i, $W(i) \in [0,1]$, as

$$W(i) = \frac{E[(x_1^i - x_2^i)^2]}{E[||x_1 - x_2||^2]}$$

Now the ratio between the weight of each dimension before and after whitening is given by

$$\frac{W_{before}(i)}{W_{after}(i)} = \frac{\lambda^i}{\frac{1}{D}\sum_{j=1}^{D} \lambda^j}. \tag{13}$$

In Equation (13) we observe that the weight of each principal dimension increases if its initial within class variance was lower than the average, and vice versa. When there is high irrelevant noise along several dimensions, the algorithm will indeed scale down noise dimensions. However, when the irrelevant noise is scattered among many dimensions with low amplitude in each of them, whitening will amplify these noisy dimensions, which is potentially harmful. Therefore, when the data is initially embedded in a high dimensional space, the optional dimensionality reduction in RCA (Step 2) becomes mandatory.

We have seen in Section 3.3 that FLD is the dimensionality reduction technique which maximizes the mutual information under Gaussian assumptions. Traditionally FLD is computed from fully labelled training data, and the method therefore falls within supervised learning. We now extend FLD, using the same information theoretic criterion, to the case of partial supervision in the form of equivalence constraints. Specifically, denote by S_t and S_w the estimators of the total covariance and the within class covariance respectively. FLD maximizes the determinant ratio

$$\max_{A \in \mathcal{M}_{K \times D}} \frac{AS_tA^t}{AS_wA^t} \tag{14}$$

by solving a generalized eigenvector problem. The row vectors of the optimal matrix A are the first K eigenvectors of $S_w^{-1}S_t$. In our case the optimization problem is of the same form as in (14), with the within chunklet covariance matrix from (1) playing the role of S_w. We compute the projection matrix using SVD in the usual way, and term this FLD variant cFLD (constraints based FLD).

To understand the intuition behind cFLD, note that both PCA and cFLD remove dimensions with small total variance, and hence reduce the risk of RCA amplifying irrelevant dimensions with small variance. However, unsupervised PCA may remove dimensions that are important for the discrimination between classes, if their total variability is low. Intuitively, better dimensionality reduction can be obtained by comparing the total covariance matrix (used by PCA) to the within class covariance matrix (used by RCA), and this is exactly what the partially supervised cFLD is trying to accomplish in (14).

The cFLD dimensionality reduction can only be used if the rank of the within chunklet covariance matrix is higher than the dimensionality of the initial data space. If this condition does not hold, we use PCA to reduce the original data dimensionality as needed. The procedure is summarized below in Algorithm 2.

Algorithm 2 Dimensionality reduction: Step 2 of RCA

Denote by D the original data dimensionality. Given a set of chunklets $\{C_j\}_{j=1}^n$ do

1. Compute the rank of the estimated within chunklet covariance matrix $R = \sum_{j=1}^n (|C_j| - 1)$, where $|C_j|$ denotes the size of the j'th chunklet.
2. If $(D > R)$, apply PCA to reduce the data dimensionality to αR, where $0 < \alpha < 1$ (to ensure that cFLD provides stable results).
3. Compute the total covariance matrix estimate S_t, and estimate the within class covariance matrix using $S_w = \hat{C}$ from (1). Solve (14), and use the resulting A to achieve the target data dimensionality.

7. Online Implementation of RCA

The standard RCA algorithm presented in Section 2 is a batch algorithm which assumes that all the equivalence constraints are available at once, and that all the data is sampled from a stationary source. Such conditions are usually not met in the case of biological learning systems, or artificial sensor systems that interact with a gradually changing environment. Consider for example a system that tries to cluster images of different people collected by a surveillance camera in gradually changing illumination conditions, such as those caused by night and day changes. In this case different distance functions should be used during night and day times, and we would like the distance used by the system to gradually adapt to the current illumination conditions. An online algorithm for distance function learning is required to achieve such a gradual adaptation.

Here we briefly present an online implementation of RCA, suitable for a neural-network-like architecture. In this implementation a weight matrix $W \in \mathcal{M}_{D\times D}$, initiated randomly, is gradually developed to become the RCA transformation matrix. In Algorithm 3 we present the procedure for the simple case of chunklets of size 2. The extension of this algorithm to general chunklets is briefly described in Appendix C.

Algorithm 3 Online RCA for point pairs

Input: a stream of pairs of points (x_1^T, x_2^T), where x_1^T x_2^T are known to belong to the same class.
Initialize W to a symmetric random matrix with $||W|| << 1$.
At time step T do:

- receive pair x_1^T, x_2^T;
- let $h = x_1^T - x_2^T$;
- apply W to h, to get $y = Wh$;
- update $W = W + \eta(W - yy^t W)$.

where $\eta > 0$ determines the step size.

Assuming local stationarity, the steady state of this stochastic process can be found by equating the mean update to 0, where the expectation is taken over the next example pair (x_1^{T+1}, x_2^{T+1}). Using the notations of Algorithm 3, the resulting equation is

$$E[\eta(W - yy^tW)] = 0 \quad \Rightarrow \quad E[I - yy^t] = I - WE[hh^t]W^t = 0 \quad \Rightarrow \quad W = PE[hh^t]^{-\frac{1}{2}},$$

where P is an orthonormal matrix $PP^t = I$. The steady state W is the whitening transformation of the correlation matrix of h. Since $h = 2(x_1 - \frac{(x_1+x_2)}{2})$, it is equivalent (up to the constant 2) to the distance of a point from the center of its chunklet. The correlation matrix of h is therefore equivalent to the within chunklet covariance matrix. Thus W converges to the RCA transformation of the input population up to an orthonormal transformation. The resulting transformation is geometrically equivalent to RCA, since the orthonormal transformation P preserves vector norms and angles.

In order to evaluate the stability of the online algorithm we conducted simulations which confirmed that the algorithm converges to the RCA estimator (up to the transformation P), if the gradient steps decrease with time ($\eta = \eta_0/T$). However, the adaptation of the RCA estimator for such a step size policy can be very slow. Keeping η constant avoids this problem, at the cost of producing a noisy RCA estimator, where the noise is proportional to η. Hence η can be used to balance this tradeoff between adaptation, speed and accuracy.

8. Experimental Results

The success of the RCA algorithm can be measured directly by measuring neighborhood statistics, or indirectly by measuring whether it improves clustering results. In the following we tested RCA on three different applications using both direct and indirect evaluations.

The RCA algorithm uses only partial information about the data labels. In this respect it is interesting to compare its performance to unsupervised and supervised methods for data representation. Section 8.1 compares RCA to the unsupervised PCA and the fully supervised FLD on a facial recognition task, using the YaleB data set (Belhumeur et al., 1997). In this application of face recognition, RCA appears very efficient in eliminating irrelevant variability caused by varying illumination. We also used this data set to test the effect of dimensionality reduction using cFLD, and the sensitivity of RCA to average chunklet size and the total amount of points in chunklets.

Section 8.2 presents a more realistic surveillance application in which equivalence constraints are gathered automatically from a Markovian process. In Section 8.3 we conclude our experimental validation by comparing RCA with other methods which make use of equivalence constraints in a clustering task, using a few benchmark data sets from the UCI repository (Blake and Merz, 1998). The evaluation of different metrics below is presented using *cumulative neighbor purity* graphs, which display the average (over all data points) percentage of correct neighbors among the first k neighbors, as a function of k.

8.1 Applying RCA to Facial Recognition

The task here is to classify facial images with respect to the person photographed. In these experiments we consider a retrieval paradigm reminiscent of nearest neighbor classification, in which a query image leads to the retrieval of its nearest neighbor or its K-nearest neighbors in the data set. Using a facial image database, we begin by evaluating nearest neighbor classification with the RCA distance, and compare its performance to supervised and unsupervised learning methods. We then move on to address more specific issues: In 8.1.4 we look more closely at the two steps of RCA, Step 2 (cFLD dimensionality reduction) and Step 3 (whitening w.r.t. $\hat{C}$), and study their contribution to performance in isolation. In 8.1.5 the retrieval performance of RCA is compared with the

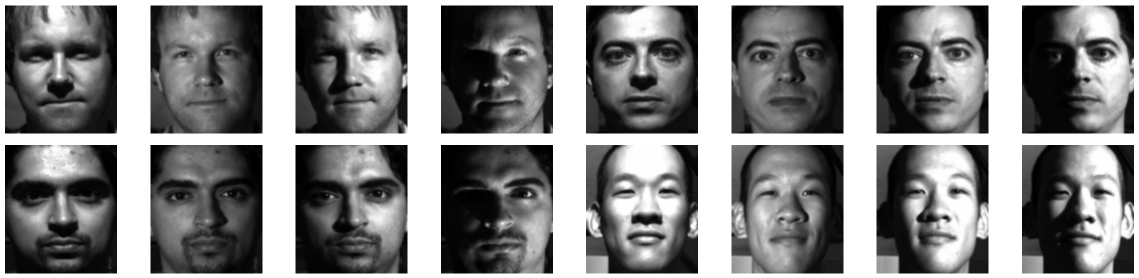


Figure 2: A subset of the YaleB database which contains 1920 frontal face images of 30 individuals taken under different lighting conditions.

algorithm presented by Xing et al. (2002). Finally in 8.1.6 we evaluate the effect of chunklets sizes on retrieval performance, and compare it to the predicted effect of chunklet size on the variance of the RCA estimator.

8.1.1 THE DATA SET

We used a subset of the yaleB data set (Belhumeur et al., 1997), which contains facial images of 30 subjects under varying lighting conditions. The data set contains a total of 1920 images, including 64 frontal pose images of each subject. The variability between images of the same person is mainly due to different lighting conditions. These factors caused the variability among images belonging to the same subject to be greater than the variability among images of different subjects (Adini et al., 1997). As preprocessing, we first automatically centered all the images using optical flow. Images were then converted to vectors, and each image was represented using its first 60 PCA coefficients. Figure 2 shows a few images of four subjects.

8.1.2 OBTAINING EQUIVALENCE CONSTRAINTS

We simulated the *'distributed learning'* scenario presented in Section 1 in order to obtain equivalence constraints. In this scenario, we obtain equivalence constraints using the help of T teachers. Each teacher is given a random selection of L data points from the data set, and is asked to give his own labels to all the points, effectively partitioning the data set into equivalence classes. Each teacher therefore provides both positive and negative constraints. Note however that RCA only uses the positive constraints thus gathered. The total number of points in chunklets grows linearly with TL, the number of data points seen by all teachers. We control this amount, which provides a loose bound on the number of points in chunklets, by varying the number of teachers T and keeping L constant. We tested a range of values of T for which TL is 10%, 30%, or 75% of the points in the data set.[5]

The parameter L controls the distribution of chunklet sizes. More specifically, we show in Appendix D that this distribution is controlled by the ratio $r = \frac{L}{M}$ where M is the number of classes in the data. In all our experiments we have used $r = 2$. For this value the expected chunklet size is

5. In this scenario one usually obtains mostly 'negative' equivalence constraints, which are pairs of points that are known to originate from different classes. RCA does *not* use these 'negative' equivalence constraints.

roughly 2.9 and we typically obtain many small chunklets. Figure 3 shows a histogram of typical chunklet sizes, as obtained in our experiments.[6]

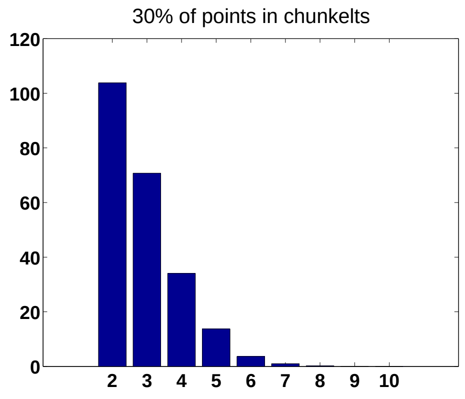


Figure 3: Sample chunklet size distribution obtained using the distributed learning scenario on a subset of the yaleB data set with 1920 images from $M = 30$ classes. L is chosen such that $r = \frac{L}{M} = 2$. The histogram is plotted for distributed learning with 30% of the data points in chunklets.

8.1.3 RCA ON THE CONTINUUM BETWEEN SUPERVISED AND UNSUPERVISED LEARNING

The goal of our main experiment in this section was to assess the relative performance of RCA as a semi-supervised method in a face recognition task. To this extent we compared the following methods:

- Eigenfaces (Turk and Pentland, 1991): this unsupervised method reduces the dimensionality of the data using PCA, and compares the images using the Euclidean metric in the reduced space. Images were normalized to have zero mean and unit variance.
- Fisherfaces (Belhumeur et al., 1997): this supervised method starts by applying PCA dimensionality reduction as in the Eigenfaces method. It then uses all the data labels to compute the FLD transformation (Fukunaga, 1990), and transforms the data accordingly.
- RCA: the RCA algorithm with dimensionality reduction as described in Section 6, that is, PCA followed by cFLD. We varied the amount of data in constraints provided to RCA, using the *distributed learning* paradigm described above.

The left panel in Figure 4 shows the results of the different methods. The graph presents the performance of RCA for low, moderate and high amounts of constrained points. As can be seen, even with low amounts of equivalence constraints the performance of RCA is much closer to the performance of the supervised FLD than to the performance of the unsupervised PCA. With Moderate and high amounts of equivalence constraints RCA achieves neighbor purity rates which are

6. We used a different sampling scheme in the experiments which address the effect of chunklet size, see Section 8.1.6.

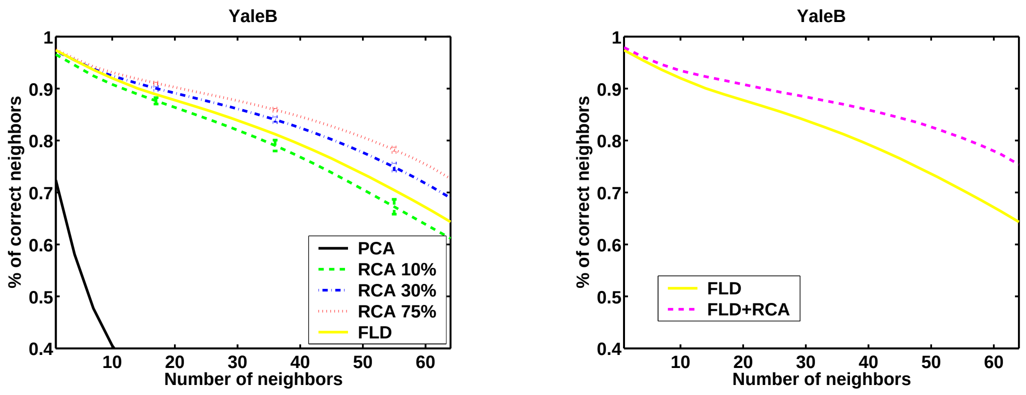


Figure 4: Left: Cumulative purity graphs for the following algorithms and experimental conditions: Eigenface (PCA), RCA 10%, RCA 30%, RCA 75%, and Fisherface (FLD). The percentages stated for RCA are the fractions of data points presented to the 'distributed learning' oracle, as discussed in Section 8.1.2. The data was reduced to dimension 60 using PCA for all the methods. It was then further reduced to dimension 30 using cFLD in the three RCA variants, and using FLD for the Fisherface method. Results were averaged over 50 constraints realizations. The error bars give the Standard Errors of the Mean (SEMs). Right: Cumulative purity graphs for the fully supervised FLD, with and without fully labelled RCA. Here RCA dramatically enhances the performance of FLD.

higher than those achieved by the fully supervised Fisherfaces method, while relying only on fragmentary chunklets with unknown class labels. This somewhat surprising result stems from the fact that the fully supervised FLD in these experiments was not followed by whitening.

In order to clarify this last point, note that RCA can also be used when given a fully labelled training set. In this case, chunklets correspond uniquely and fully to classes, and the cFLD algorithm for dimensionality reduction is equivalent to the standard FLD. In this setting RCA can be viewed as an augmentation of the standard, fully supervised FLD, which whitens the output of FLD w.r.t the within class covariance. The right panel in Figure 4 shows comparative results of FLD with and without whitening in the fully labelled case.

In order to visualize the effect of RCA in this task we also created some "RCAfaces", following Belhumeur et al. (1997): We ran RCA on the images after applying PCA, and then reconstructed the images. Figure 5 shows a few images and their reconstruction. Clearly RCA dramatically reduces the effect of varying lighting conditions, and the reconstructed images of the same individual look very similar to each other. The Eigenfaces (Turk and Pentland, 1991) method did not produce similar results.

8.1.4 SEPARATING THE CONTRIBUTION OF THE DIMENSIONALITY REDUCTION AND WHITENING STEPS IN RCA

Figure 4 presents the results of RCA including the semi-supervised dimensionality reduction of cFLD. While this procedure yields the best results, it mixes the separate contributions of the two main steps of the RCA algorithm, that is, dimensionality reduction via cFLD (Step 2) and whitening

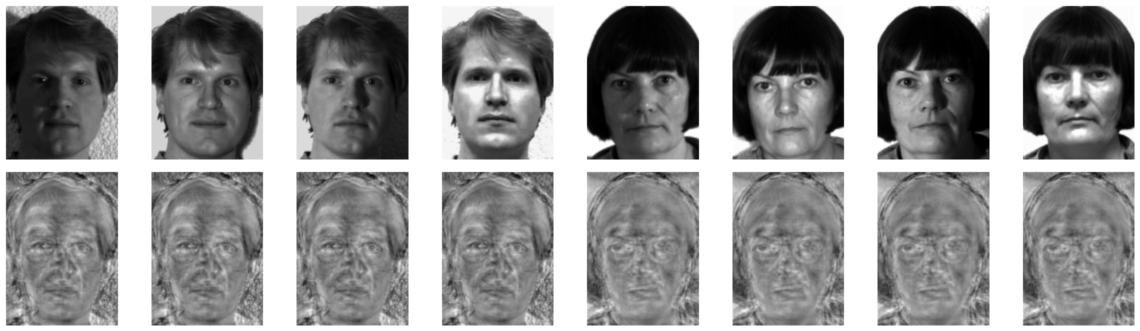


Figure 5: Top: Several facial images of two subjects under different lighting conditions. Bottom: the same images from the top row after applying PCA and RCA and then reconstructing the images. Clearly RCA dramatically reduces the effect of different lighting conditions, and the reconstructed images of each person look very similar to each other.

of the inner chunklet covariance matrix (Step 3). In the left panel of Figure 6 these contributions are isolated.

It can be seen that when cFLD and whitening are used separately, they both provide considerable improvement in performance. These improvements are only partially dependent, since the performance gain when combining both procedures is larger than either one alone. In the right panel of Figure 6 we present learning curves which show the performance of RCA with and without dimensionality reduction, as a function of the amount of supervision provided to the algorithm. For small amounts of constraints, both curves are almost identical. However, as the number of constraints increases, the performance of RCA dramatically improves when using cFLD.

8.1.5 COMPARISON WITH THE METHOD OF XING ET AL.

In another experiment we compared the algorithm of Xing et al. (2002) to RCA on the YaleB data set using code obtained from the author's web site. The experimental setup was the one described in Section 8.1.2, with 30% of the data points presented to the distributed learning oracle. While RCA uses only the positive constraints obtained, the algorithm of Xing et al. (2002) was given both the positive and negative constraints, as it can make use of both. Results are shown in Figure 7, showing that this algorithm failed to converge when given high dimensional data, and was outperformed by RCA in lower dimensions.

8.1.6 THE EFFECT OF DIFFERENT CHUNKLET SIZES

In Section 5 we showed that RCA typically provides an estimator for the within class covariance matrix, which is not very sensitive to the size of the chunklets. This was done by providing a bound on the variance of the elements in the RCA estimator matrix $\hat{C}(n,k)$. We can expect that lower variance of the estimator will go hand in hand with higher purity performance. In order to empirically test the effect of chunklets' size, we fixed the number of equivalence constraints, and

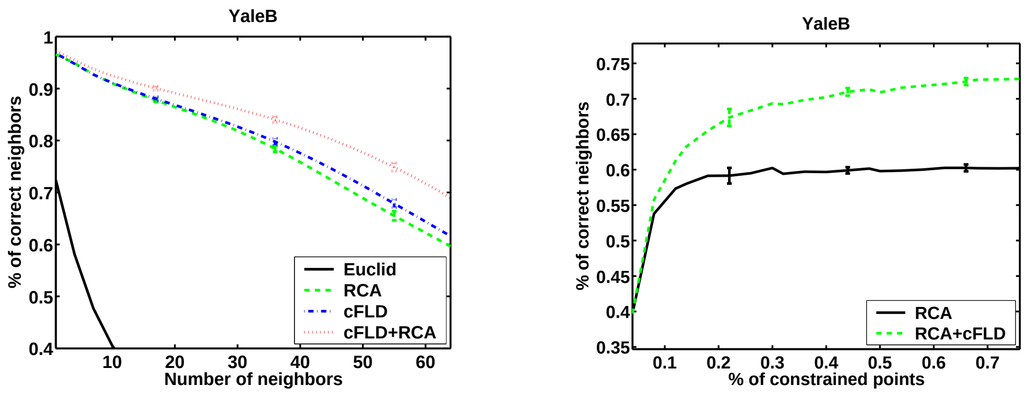


Figure 6: Left: Cumulative purity graphs for 4 experimental conditions: original space, RCA without cFLD, cFLD only, and RCA with cFLD (using the Euclidean norm in all cases). The data was reduced to 60 dimensions using unsupervised PCA. The semi supervised techniques used constraints obtained by distributed learning with 30% of the data points. RCA without cFLD was performed in the space of 60 PCA coefficients, while in the last 2 conditions dimensionality was further reduced to 30 using the constraints. Results were averaged over 50 constraints realizations. Right: Learning curves–neighbor purity performance for 64 neighbors as a function of the amount of constraints. The performance is measured by averaging (over all data points) the percentage of correct neighbors among the first 64 neighbors. The amount of constraints is measured using the percentage of points given to the distributed learning oracle. Results are averaged over 15 constraints realizations. Error bars in both graphs give the standard errors of the mean.

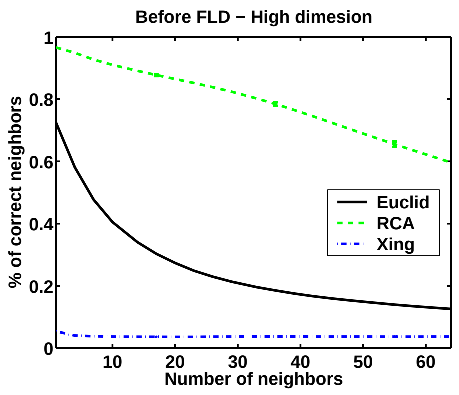
varied the size of the chunklets S in the range $\{2-10\}$. The chunklets were obtained by randomly selecting 30% of the data (total of $P = 1920$ points) and dividing it into chunklets of size S.[7]

The results can be seen in Figure 8. As expected the performance of RCA improves as the size of the chunklets increases. Qualitatively, this improvement agrees with the predicted improvement in the RCA estimator's variance, as most of the gain in performance is already obtained with chunklets of size $S = 3$. Although the bound presented is not tight, other reasons may account for the difference between the graphs, including the weakness of the Gaussian assumption used to derive the bound (see Section 9), and the lack of linear connection between the estimator's variance and purity performance.

8.2 Using RCA in a Surveillance Application

In this application, a stationary indoor surveillance camera provided short video clips whose beginning and end were automatically detected based on the appearance and disappearance of moving targets. The database therefore included many clips, each displaying only one person of unknown identity. Effectively each clip provided a chunklet. The task in this case was to cluster together all clips in which a certain person appeared.

7. When necessary, the remaining $mod(0.3P, S)$ points were gathered into an additional smaller chunklet.



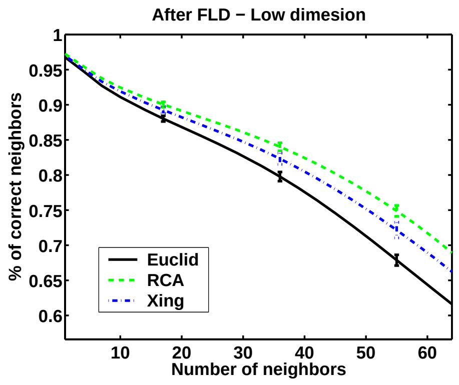
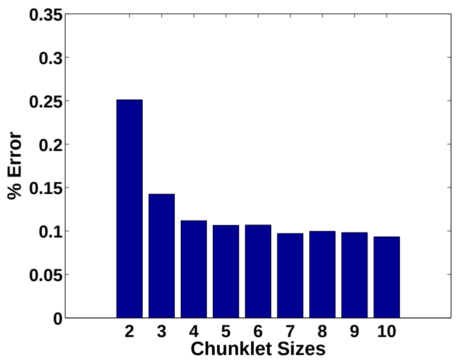


Figure 7: The method of Xing et al. (2002) and RCA on the YaleB facial image data set. Left: Neighbor purity results obtained using 60 PCA coefficients. The algorithm of Xing et al. (2002) failed to converge and returned a metric with chance level performance. Right: Results obtained using a 30 dimensional representation, obtained by applying cFLD to the 60 PCA coefficients. Results are averaged over 50 constraints realizations. The error bars give the standard errors of the mean.



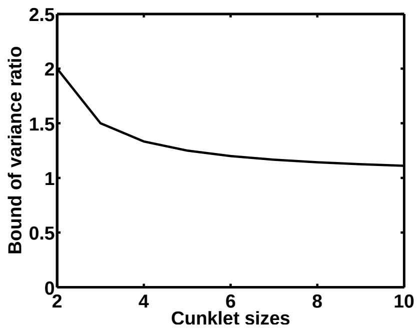


Figure 8: Left: Mean error rate on all 64 neighbors on the yaleB data set when using 30% of the data in chunklets. In this experiment we varied the chunklet sizes while fixing the total amount of points in chunklets. Right: the theoretical bound over the ratio between the variance of the RCA matrix elements and the variance of the best possible estimator using the same number of points (see inequality 12). The qualitative behavior of the graphs is similar, seemingly because a lower estimator variance tends to imply better purity performance.

The task and our approach: The video clips were highly complex and diversified, for several reasons. First, they were entirely unconstrained: a person could walk everywhere in the scene, coming closer to the camera or walking away from it. Therefore the size and resolution of each image varied dramatically. In addition, since the environment was not constrained, images included

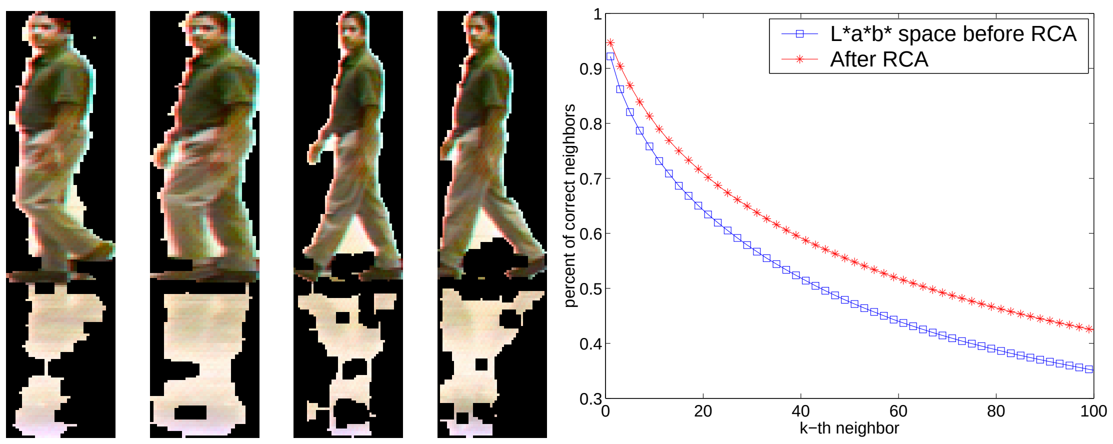


Figure 9: Left: several images from a video clip of one subject. Right: cumulative neighbor purity results before and after RCA.

varying occlusions, reflections and (most importantly from our perspective) highly variable illumination. In fact, the illumination changed dramatically across the scene both in intensity (from brighter to darker regions), and in spectrum (from neon light to natural lighting). Figure 9 shows several images from one input clip.

We sought to devise a representation that would enable the effective clustering of clips, focusing on color as the only low-level attribute that could be reliably used in this application. Therefore our task was to accomplish some sort of color constancy, that is, to overcome the general problem of irrelevant variability due to the varying illumination. This is accomplished by the RCA algorithm.

Image representation and RCA Each image in a clip was represented by its color histogram in $L^*a^*b^*$ space (we used 5 bins for each dimension). We used the clips as chunklets in order to compute the RCA transformation. We then computed the distance between pairs of images using two methods: *L1* and RCA (Mahalanobis). We used over 6000 images from 130 clips (chunklets) of 20 different people. Figure 9 shows the cumulative neighbor purity over all 6000 images. One can see that RCA makes a significant contribution by bringing 'correct' neighbors closer to each other (relative to other images). However, the effect of RCA on retrieval performance here is lower than the effect gained with the YaleB data base. While there may be several reasons for this, an important factor is the difference between the way chunklets were obtained in the two data sets. The automatic gathering of chunklets from a Markovian process tends to provide chunklets with dependent data points, which supply less information regarding the within class covariance matrix.

8.3 RCA and Clustering

In this section we evaluate RCA's contribution to clustering, and compare it to alternative algorithms that use equivalence constraints. We used six data sets from the UCI repository. For each data set we randomly selected a set Q_P of pairwise positive equivalence constraints (or chunklets of size 2). We compared the following clustering algorithms:

*a***.** K-means using the default Euclidean metric and no side-information (Fukunaga, 1990).

*b***.** Constrained K-means + Euclidean metric: the K-means version suggested by Wagstaff et al. (2001), in which a pair of points $(x_i, x_j) \in Q_P$ is always assigned to the same cluster.

*c***.** Constrained K-means + the metric proposed by Xing et al. (2002): The metric is learnt from constraints in Q_P. For fairness we replicated the experimental design employed by Xing et al. (2002), and allowed the algorithm to treat all unconstrained pairs of points as negative constraints (the set Q_N).

*d***.** Constrained K-means + RCA: Constrained K-means using the RCA Mahalanobis metric learned from Q_P.

*e***.** EM: Expectation Maximization of a Gaussian Mixture model (using no side-information).

*f***.** Constrained EM: EM using side-information in the form of equivalence constraints (Shental et al., 2004), when using the RCA distance metric as the initial metric.

Clustering algorithms *a* and *e* are unsupervised and provide respective lower bounds for comparison with our algorithms *d* and *f*. Clustering algorithms *b* and *c* compete fairly with our algorithm *d*, using the same kind of side information.

Experimental setup To ensure fair comparison with Xing et al. (2002), we used exactly the same experimental setup as it affects the gathering of equivalence constraints and the evaluation score used. We tested all methods using two conditions, with: (i) "little" side-information Q_P, and (ii) "much" side-information. The set of pairwise similarity constraints Q_P was generated by choosing a random subset of all pairs of points sharing the same class identity c_i. Initially, there are N 'connected components' of unconstrained points, where N is the number of data points. Randomly choosing a pairwise constraint decreases the number of connected components by 1 at most. In the case of "little" ("much") side-information, pairwise constraints are randomly added until the number of different connected components K_c is roughly $0.9N$ $(0.7N)$. As in the work of Xing et al. (2002), no negative constraints were sampled.

Following Xing et al. (2002) we used a normalized accuracy score, the "Rand index" (Rand, 1971), to evaluate the partitions obtained by the different clustering algorithms. More formally, with binary labels (or two clusters), the accuracy measure can be written as

$$\sum_{i>j} \frac{1\{1\{c_i = c_j\} = 1\{\hat{c}_i = \hat{c}_j\}\}}{0.5m(m-1)},$$

where $1\{\dot{\}}$ denotes the indicator function $(1\{True\} = 1), 1\{False\} = 0)$, $\{\hat{c}_i\}_{i=1}^m$ denotes the cluster to which point x_i is assigned by the clustering algorithm, and c_i denotes the "correct" (or desirable) assignment. The score above is the probability that the algorithm's decision regarding the label equivalence of two points agrees with the decision of the "true" assignment c.[8]

Figure 10 shows comparative results using six different UCI data sets. Clearly the RCA metric significantly improved the results over the original K-means algorithms (both the constrained

8. As noted by Xing et al. (2002), this score should be normalized when the number of clusters is larger than 2. Normalization is achieved by sampling the pairs (x_i, x_j) such that x_i and x_j are from the same cluster with probability 0.5 and from different clusters with probability 0.5, so that "matches" and "mismatches" are given the same weight.

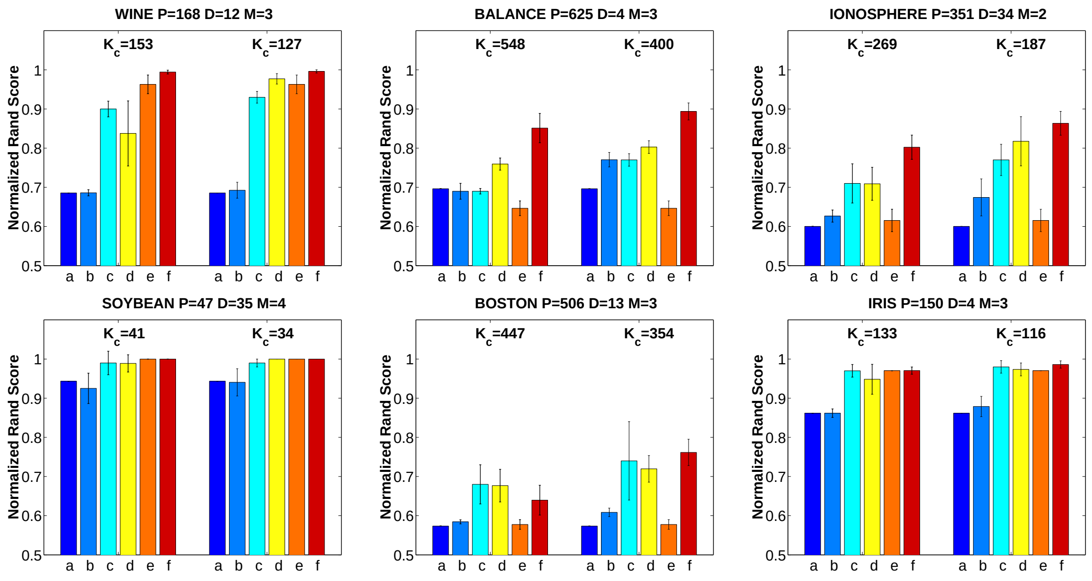


Figure 10: Clustering accuracy on 6 UCI data sets. In each panel, the six bars on the left correspond to an experiment with "little" side-information, and the six bars on the right correspond to "much" side-information. From left to right the six bars correspond respectively to the algorithms described in the text, as follows: (a) K-means over the original feature space (without using any side-information). (b) Constrained K-means over the original feature space. (c) Constrained K-means over the feature space suggested by Xing et al. (2002). (d) Constrained K-means over the feature space created by RCA. (e) EM over the original feature space (without using any side-information). (f) Constrained EM (Shental et al., 2004) over the feature space created by RCA. Also shown are P–the number of points, M–the number of classes, D–the dimensionality of the feature space, and K_c–the mean number of connected components. The results were averaged over 20 realizations of side-information. The error bars give the standard deviations. In all experiments we used K-means with multiple restarts as in done by Xing et al. (2002).

and unconstrained versions). Generally in the context of K-means, we observe that using equivalence constraints to find a better metric improves results much more than using this information to constrain the algorithm. RCA achieves comparable results to those reported by Xing et al. (2002), despite the big difference in computational cost between the two algorithms (see Section 9.1).

The last two algorithms in our comparisons use the EM algorithm to compute a generative Gaussian Mixture Model, and are therefore much more computationally intensive. We have added these comparisons because EM implicitly changes the distance function over the input space in a locally linear way (that is, like a Mahalanobis distance). It may therefore appear that EM can do everything that RCA does and more, without any modification. The histogram bins marked by (e) in Figure 10 clearly show that this is not the case. Only when we add constraints to the EM, and preprocess the data with RCA, do we get improved results as shown by the histogram bins marked by (f) in Figure 10.

9. Discussion

We briefly discuss running times in Section 9.1. The applicability of RCA in general conditions is then discussed in 9.2.

9.1 Runtime Performance

Computationally RCA relies on a few relatively simple matrix operations (inversion and square root) applied to a positive-definite square matrix, whose size is the reduced dimensionality of the data. This can be done fast and efficiently and is a clear advantage of the algorithm over its competitors.

9.2 Using RCA when the Assumptions Underlying the Method are Violated

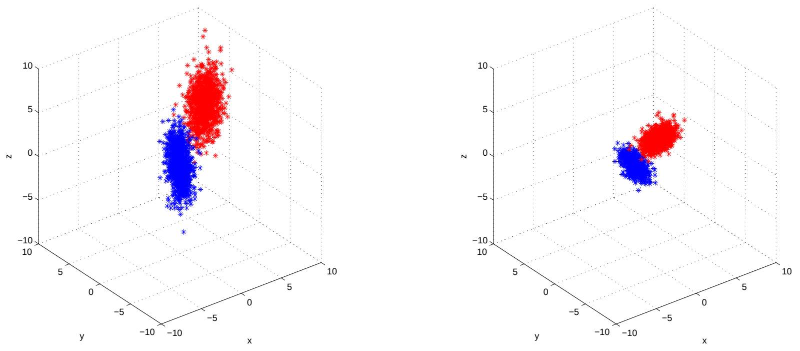


Figure 11: Extracting the shared component of the covariance matrix using RCA: In this example the data originates from 2 Gaussian sources with the following diagonal covariance matrices: $diag(C_1) = (\varepsilon, 1, 2)$ and $diag(C_2) = (1, \varepsilon, 2)$. (a) The original data points (b) The transformed data points when using RCA. In this example we used all of the points from each class as a single chunklet and therefore the chunklet covariance matrix is the average within-class covariance matrix. As can be seen RCA clearly downscales the irrelevant variability in the Z axis, which is the shared component of the 2 classes covariance matrices. Specifically, the eigenvalues of the covariance matrices for the two classes are as follows (for $\varepsilon = 0.1$): class 1–$(3.947, 1.045, 0.009)$ before RCA, and $(1.979, 1.001, 0.017)$ after RCA; class 2–$(3.953, 1.045, 0.010)$ before RCA, and $(1.984, 1.001, 0.022)$ after RCA. In this example, the condition numbers increased by a factor of 3.78 and 4.24 respectively for both classes.

In order to obtain a strict probabilistic justification for RCA, we listed in Section 5 the following assumptions:

1. The classes have multi-variate normal distributions.

2. All the classes share the same covariance matrix.

3. The points in each chunklet are an i.i.d. sample from the class.

What happens when these assumptions do not hold?

The first assumption gives RCA its probabilistic justification. Without it, in a distribution-free model, RCA is the best linear transformation optimizing the criteria presented in Sections 3-4: maximal mutual information, and minimal within-chunklet distance. These criteria are reasonable as long as the classes are approximately convex (as assumed by the use of the distance between chunklet's points and chunklet's means). In order to investigate this point empirically, we used Mardia's statistical tests for multi-variate normality (Mardia, 1970). These tests (which are based on skewness and kurtosis) showed that all of the data sets used in our experiments are significantly non-Gaussian (except for the Iris UCI data set). Our experimental results therefore clearly demonstrate that RCA performs well when the distribution of the classes in the data is not multi-variate normal.

The second assumption justifies RCA's main computational step, which uses the empirical average of all the chunklets covariance matrices in order to estimate the global within class covariance matrix. When this assumption fails, RCA effectively extracts the shared component of all the classes covariance matrices, if such component exists. Figure 11 presents an illustrative example of the use of RCA on data from two classes with different covariance matrices. A quantitative measure of RCA's partial success in such cases can be obtained from the change in the *condition number* (the ratio between the largest and smallest eigenvalues) of the within-class covariance matrices of each of the classes, before and after applying RCA. Since RCA attempts to whiten the within-class covariance, we expect the condition number of the within-class covariance matrices to decrease. This is indeed the case for the various classes in all of the data sets used in our experimental results.

The third assumption may break down in many practical applications, when chunklets are automatically collected and the points within a chunklet are no longer independent of one another. As a result chunklets may be composed of points which are rather close to each other, and whose distribution does not reflect all the typical variance of the true distribution. In this case RCA's performance is not guaranteed to be optimal (see Section 8.2).

10. Conclusion

We have presented an algorithm which uses side-information in the form of equivalence constraints, in order to learn a Mahalanobis metric. We have shown that our method is optimal under several criteria. Our empirical results show that RCA reduces irrelevant variability in the data and thus leads to considerable improvements in clustering and distance based retrieval.

Appendix A. Information Maximization with Non-Invertible Linear Transformations

Here we sketch the proof of the claim made in Section 3.3. As before, we denote by $\hat{C}$ the average covariance matrix of the chunklets. We can rewrite the constrained expression from Equation 5 as

$$\frac{1}{N}\sum_{j=1}^{n}\sum_{i=1}^{n_j}(x_{ji}-m_j)^t A^t A(x_{ji}-m_j) = tr(A^t A\hat{C}) = tr(A^t\hat{C}A).$$

Hence the Lagrangian can be written as

$$\log|A\Sigma_x A^t| - \lambda(tr(A\hat{C}A^t)-1).$$

Differentiating the Lagrangian with respect to A gives

$$\Sigma_x A^t (A\Sigma_x A^t)^{-1} = \lambda \hat{C} A^t.$$

Multiplying by A and rearranging terms, we get $\frac{I}{\lambda} = A\hat{C}A^t$. Hence as in RCA, A must whiten the data with respect to the chunklet covariance $\hat{C}$ in a yet to be determined subspace. We can now use the equality in (5) to find λ:

$$\begin{aligned} tr(A\hat{C}A^t) = tr(\frac{I}{\lambda}) = \frac{K}{\lambda} = 1 \Longrightarrow \ \lambda = K \\ \Longrightarrow \ A\hat{C}A^t = \frac{1}{K}I, \end{aligned}$$

where K is the dimension of the projection subspace.

Next, since in our solution space $A\hat{C}A^t = \frac{1}{K}I$, it follows that $\log|A\hat{C}A^t| = K\log\frac{1}{K}$ holds for all points. Hence we can modify the maximization argument as follows:

$$\log|A\Sigma_x A^t| \quad = \quad \log\frac{|A\Sigma_x A^t|}{|A\hat{C}A^t|} + K\log\frac{1}{K}$$

Now the optimization argument has a familiar form. It is known (Fukunaga, 1990) that maximizing the determinant ratio can be done by projecting the space on the span of the first K eigenvectors of $\hat{C}^{-1}\Sigma_x$. Denote by G the solution matrix for this unconstrained problem. This matrix orthogonally diagonalizes both $\hat{C}$ and Σ_x, so $G\hat{C}G^t = \Lambda_1$ and $G\Sigma_x G^t = \Lambda_2$ for Λ_1, Λ_2 diagonal matrices. In order to enforce the constraints we define the matrix $A = \sqrt{\frac{1}{K}}\Lambda_1^{-0.5}G$ and claim that A is the solution of the constrained problem. Notice that the value of the maximization argument does not change when we switch from A to G since A is a product of G and another full ranked matrix. It can also be shown that A satisfies the constraints and is thus the solution of the Problem (5).

Appendix B. Variance Bound on the RCA Covariance Estimator

In this appendix we prove Inequality 12 from Section 5. Assume we have $N = nk$ data points $X = \{x_{ji}\}_{i=1,j=1}^{n,k}$ in n chunklets of size k each. We assume that all chunklets are drawn independently from Gaussian sources with the same covariance matrix. Denoting by m_i the mean of chunklet i, the unbiased RCA estimator of this covariance matrix is

$$\hat{C}(n,k) = \frac{1}{n}\sum_{j=1}^{n}\frac{1}{k-1}\sum_{i=1}^{k}(x_{ji} - m_i)(x_{ji} - m_i)^T.$$

It is more convenient to estimate the convergence of the covariance estimate for data with a diagonal covariance matrix. We hence consider a diagonalized version of the covariance, and return to the original covariance matrix toward the end of the proof. Let U denote the diagonalization transformation of the covariance matrix C of the Gaussian sources, that is, $UCU^t = \Lambda$ where Λ is a diagonal matrix with $\{\lambda_i\}_{i=1}^{D}$ on the diagonal. Let $Z = UX = \{z_{ji}\}_{i=1,j=1}^{n,k}$ denote the transformed data. Denote the transformed within class covariance matrix estimation by $\hat{C}^u(n,k) = U\hat{C}(n,k)U^t$, and denote the chunklet means by $m_i^u = Um_i$. We can analyze the variance of $\hat{C}^u$ as follows:

$$var(\hat{C}^u(n,k)) = var[\frac{1}{n}\sum_{i=1}^{n}\frac{1}{k-1}\sum_{j=1}^{k}(z_{ji} - m_i^u)(z_{ji} - m_i^u)^T]$$

$$= \frac{1}{n} var[\frac{1}{k-1} \sum_{j=1}^{k} (z_{ji} - m_1^u)(z_{ji} - m_1^u)^T]. \tag{15}$$

The last equality holds since the summands of the external sum are sample covariance matrices of independent chunklets drawn from sources with the same covariance matrix.

The variance of the sample covariance, assessed from k points, for diagonalized Gaussian data is known to be (Fukunaga, 1990)

$$var(\hat{C}_{ii}) = \frac{2\lambda_i^2}{k-1}; \;\; var(\hat{C}_{ij}) = \frac{\lambda_i \lambda_j}{k}; \;\; cov(\hat{C}_{ij}, \hat{C}_{kl}) = 0.$$

Hence (15) is simply

$$var(\hat{C}_{ii}^u) = \frac{2\lambda_i^2}{n(k-1)}; \;\; var(\hat{C}_{ij}^u) = \frac{\lambda_i \lambda_j}{nk}; \;\; cov(\hat{C}_{ij}^u, \hat{C}_{kl}^u) = 0.$$

Replacing $N = nk$, we can write

$$var(\hat{C}_{ii}^u) = \frac{2\lambda_i^2}{N(1-\frac{1}{k})}; \;\; var(\hat{C}_{ij}^u) = \frac{\lambda_i \lambda_j}{N}; \;\; cov(\hat{C}_{ij}^u, \hat{C}_{kl}^u) = 0,$$

and for the diagonal terms $\hat{C}_{ii}^u$

$$var(\hat{C}^u(\frac{N}{k}, k)_{ii}) = \frac{2\lambda_i^2}{N(1-\frac{1}{k})} = \frac{k}{k-1}\frac{2\lambda_i^2}{N} \leq \frac{k}{k-1}\frac{2\lambda_i^2}{N-1} = \frac{k}{k-1} var(\hat{C}^u(1,N)_{ii}).$$

This inequality trivially holds for the off-diagonal covariance elements.

Getting back to the original data covariance, we note that in matrix elements notation $\hat{C}_{ij} = \sum_{q,r=1}^{D} \hat{C}_{qr}^u U_{iq} U_{jr}$ where D is the data dimension. Therefore

$$\frac{var[\hat{C}_{ij}(n,k)]}{var[\hat{C}_{ij}(1,nk)]} = \frac{\sum_{q,r=1}^{D} var[\hat{C}^u(n,k)_{qr} U_{iq} U_{jr}]}{\sum_{q,r=1}^{D} var[\hat{C}^u(1,nk)_{qr} U_{iq} U_{jr}]} \leq \frac{\sum_{q,r=1}^{D} \frac{k}{k-1} var[\hat{C}^u(1,nk)_{qr} U_{iq} U_{jr}]}{\sum_{q,r=1}^{D} var[\hat{C}^u(1,nk)_{qr} U_{iq} U_{jr}]} = \frac{k}{k-1},$$

where the first equality holds because $cov(\hat{C}_{ij}^u, \hat{C}_{kl}^u) = 0$.

Appendix C. Online RCA with Chunklets of General Size

The online RCA algorithm can be extended to handle a stream of chunklets of varying size. The procedure is presented in Algorithm 4.

The steady state of the weight matrix W can be analyzed in a way similar to the analysis in Section 3. The result is $W = PE[\frac{1}{n}\sum_{i=1}^{n}(x_i^T - m^T)(x_i^T - m^T)^t]^{-\frac{1}{2}}$ where P is an orthonormal matrix, and so W is equivalent to the RCA transformation of the current distribution.

Algorithm 4 Online RCA for chunklets of variable size

```
Input: a stream of chunklets where the points in a chunklet are known to belong to the same class.
Initialize $W$ to a symmetric random matrix with $||W|| << 1$.
At time step T do:
    • receive a chunklet $\{x_1^T, \ldots, x_n^T\}$ and compute its mean $m^T = \frac{1}{n}\sum_{i=1}^n x_i^T$;
    • compute $n$ difference vectors $h_i^T = x_i^T - m^T$;
    • transform $h_i^T$ using $W$, to get $y_i^T = W h_i^T$;
    • update $W = W + \eta \sum_{i=1}^n (W - y_i^T (y_i^T)^t W)$.
where $\eta > 0$ determines the step size.
```

Appendix D. The Expected Chunklet Size in the Distributed Learning Paradigm

We estimate the expected chunklet size obtained when using the distributed learning paradigm introduced in Section 8. In this scenario, we use the help of T teachers, each of which is provided with a random selection of L data points. Let us assume that the data contains M equiprobable classes, and that the size of the data set is large relative to L. Define the random variables x_i^j as the number of points from class i observed by teacher j. Due to the symmetry among classes and among teachers, the distribution of x_i^j is independent of i and j, thus defined as x. It can be well approximated by a Bernoulli distribution $B(L, \frac{1}{M})$, while considering only $x \geq 2$ (since $x = 0, 1$ do not form chunklets). Specifically,

$$p(x = i | x \neq 0, 1) = \frac{1}{1 - p(x=0) - p(x=1)} \begin{pmatrix} L \\ i \end{pmatrix} (\frac{1}{M})^i (1 - \frac{1}{M})^{L-i} \quad i = 2, 3, \ldots.$$

We can approximate $p(x=0)$ and $p(x=1)$ as

$$p(x=0) = (1 - \frac{1}{M})^L \approx e^{-\frac{L}{M}} \quad , \quad p(x=1) = \frac{L}{M}(1 - \frac{1}{M})^{L-1} \approx \frac{L}{M} e^{-\frac{L}{M}}.$$

Using these approximations, we can derive an approximation for the expected chunklet size as a function of the ratio $r = \frac{L}{M}$:

$$E(x | x \neq 0, x \neq 1) = \frac{\frac{L}{M} - p(x=1)}{1 - p(x=0) - p(x=1)} \simeq \frac{r(1 - e^{-r})}{1 - (r+1)e^{-r}}.$$

References

Y. Adini, Y. Moses, and S. Ullman. Face recognition: The problem of compensating for changes in illumination direction. In *proc. of IEEE PAMI*, volume 19(7), pages 721–732, 1997.

A. Bar-Hillel, T. Hertz, N. Shental, and D. weinshall. Learning distance functions using equivalence relations. In T. Fawcett and Nina Mishra, editors, *20th International Conference on Machine Learning*, Wahington DC, 2003. AAAI press.

S. Becker and G. E. Hinton. A self-organising neural network that discovers surfaces in random-dot stereograms. *Nature*, 355:161–163, 1992.

P. N. Belhumeur, J. Hespanha, and D. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE PAMI 8*, 19(7):711–720, 1997.

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

M. Bilenko, S. Basu, and R.J Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proc. 21st International Conf. on Machine Learning*, Banff Canada, 2004. AAAI press. URL `citeseer.ist.psu.edu/705723.html`.

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

J. S. Boreczky and L. A. Rowe. Comparison of video shot boundary detection techniques. *SPIE Storage and Retrieval for Still Images and Video Databases IV*, 2664:170–179, 1996.

G. Chechik and N. Tishby. Extracting relevant structures with side information. In S Becker, S. Thrune, and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15. The MIT Press, 2003.

K. Fukunaga. *Statistical Pattern Recognition*. Academic Press, San Diego, 2nd edition, 1990.

P. Geladi and B. Kowalski. Partial least squares regression: A tutorial. *Analytica Chimica Acta*, 185:1–17, 1986.

T. Hastie and R. Tibshirani. Discriminant adaptive nearest neighbor classification and regression. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 409–415. The MIT Press, 1996.

T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers, 1998.

D. Klein, S. Kamvar, and C. Manning. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In *Proc. 19th International Conf. on Machine Learning*, 2002. URL `citeseer.nj.nec.com/klein02from.html`.

R. Linsker. An application of the principle of maximum information preservation to linear systems. In David S. Touretzky, editor, *Advances in Neural Information Processing Systems*, pages 186–194. Morgan Kaufmann, 1989.

K. V. Mardia. Measures of multivariate skewness and kurtosis with applications. *Biometrika*, 36: 519–530, 1970.

W. M. Rand. Objective criteria for the evaluation of clustering method. *Journal of the American Statistical Association*, 66(366):846–850, 1971.

N. Shental, A. Bar-Hillel, T. Hertz, and D. Weinshall. Computing gaussian mixture models with em using equivalence constraints. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

N. Shental, T. Hertz, D. Weinshall, and M. Pavel. Adjustment learning and relevant component analysis. In A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, editors, *7th European Conference on Computer Vision*, volume 4, 2002.

J. B. Tenenbaum and W. T. Freeman. Separating style and content with bilinear models. *Neural Computation*, 12(6):1247–1283, 2000.

B. Thompson. *Canonical correlation analysis: Uses and interpretation*. Newbury Park CA: SAGE, 1984.

S. Thrun. Is learning the n-th thing any easier than learning the first? In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 640–646. The MIT Press, 1996.

N. Tishby, F. C. Pereira, and W. Bialek. The information bottleneck method. In *Proc. of the 37-th Annual Allerton Conference on Communication, Control and Computing*, pages 368–377, 1999.

M. A. Turk and A.P Pentland. Face recognition using eigenfaces. In *Proc. of IEEE CVPR*, pages 586–591, 1991.

K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. Constrained K-means clustering with background knowledge. In *Proc. 18th International Conf. on Machine Learning*, pages 577–584. Morgan Kaufmann, San Francisco, CA, 2001.

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*, volume 15. The MIT Press, 2002.

Journal of Machine Learning Research 6 (2005) 967–994 Submitted 2/04; Revised 8/04; Published 6/05

Algorithmic Stability and Meta-Learning

Andreas Maurer ANDREASMAURER@COMPUSERVE.COM
Adalbertstrasse 55
D-80799 München, Germany

Editor: Tommi Jaakkola

Abstract

A mechnism of transfer learning is analysed, where samples drawn from different learning tasks of an environment are used to improve the learners performance on a new task. We give a general method to prove generalisation error bounds for such meta-algorithms. The method can be applied to the bias learning model of J. Baxter and to derive novel generalisation bounds for meta-algorithms searching spaces of uniformly stable algorithms. We also present an application to regularized least squares regression.

Keywords: algorithmic stability, meta-learning, learning to learn

1. Introduction

We formally study the phenomenon of *transfer*, where novel tasks and concepts are learned more quickly and reliably through the application of past experience. Transfer is fundamental to human learning (see Robins, 1998, for an overview of the psychological literature) and offers a way to partially escape the implications of the *No Free Lunch Theorem* (NFLT).

The NFLT states that no algorithm is superior to another when averaged uniformly across all learning tasks. In a real environment, however, not all learning tasks occur equally likely. They are distributed according to some environmental distribution $\mathcal{E}$, which is far from uniform. By gathering information on this distribution of tasks, a learner can possibly find an algorithm to outperform other algorithms, but, of course, only on average over the distribution $\mathcal{E}$.

This mechanism of *meta-learning* has been analysed by Jonathan Baxter (1998, 2000) and there have been several successful experiments in practical machine-learning contexts (see Caruana, 1998; Thrun, 1996, 1998) and Section 6). In this paper we extend the results in Baxter (2000) and offer a general method to control the generalization error of meta-learning. We begin by reviewing some notions of learning theory.

Generalization error bounds. Statistical learning theory deals with *data* and *hypotheses*. A data point z may be an input-output pair $z = (x, y)$ and a hypothesis c may be some function $x \mapsto c(x)$, but for many theoretical results data and hypotheses can be arbitrary objects z and c, related only through a nonnegative *loss function* $l(c,z)$ which measures how poorly the hypothesis c applies to the data point z. The familiar square loss $l(c,(x,y)) = (c(x) - y)^2$ is an example where $z = (x,y)$ with $y \in \mathbb{R}$ and $c : x \mapsto c(x) \in \mathbb{R}$.

A *learning task* is modelled by a probability distribution D on the set of data points, $D(z)$ being interpreted as the probability that the data point z will be encountered under the conditions of the task D. For a given hypothesis c the *risk*

$$R(c,D) = E_{z\sim D}[l(c,z)] \tag{1}$$

measures how poorly the hypothesis c is expected to perform on D.

A *learning algorithm* A takes a *sample* $S = (z_1,...,z_m)$ of data, drawn iid from the distribution D defining the learning task, and computes a hypothesis $A(S)$. The returned hypothesis should work well on the same learning task D, so we want the risk $R(A(S),D)$ to be small. The quantity

$$E_{S\sim D^m}[R(A(S),D)] \tag{2}$$

would be a natural measure for the performance of a given algorithm A with respect to a given learning task D.

Unfortunately the distribution D itself is generally unknown, so that we cannot compute or bound (2) directly. We do, however, know the sample S which was drawn from D, and we may give a performance guarantee for A conditioned on S, but for arbitrary D. Such a *generalization error bound* is typically given by specifying a two argument function $B(\delta,S)$, where $\delta > 0$ is a confidence parameter, and the requirement that

$$\forall D, D^m\{S : R(A(S),D) \leq B(\delta,S)\} \geq 1-\delta. \tag{3}$$

The bound above states that with high probability $(1-\delta)$ in S the learning-result $A(S)$ will have risk bounded by B. Section 3 will give examples of generalization error bounds.

Meta-Learning. This paper describes a mechanism by which a sequence $\mathbf{S}=(S_1,...,S_n)$ of samples, drawn from different learning tasks $D_1,...,D_n$, can be used to improve and predict the performance of a learner on an *unknown future task*. We will give bounds analogous to (3) and also present a practical algorithm.

The crucial idea, due to J. Baxter (1998, 2000), is that the learning tasks D_i originate from an *environment* of tasks, which is a probability distribution $\mathcal{E}$ on the set of learning tasks. The encounter with a new learning task is thus modelled as a random event, a draw $D \sim \mathcal{E}$ of a task D. Subsequent to the draw of D a sample $S = (z_1,...,z_m)$ may be generated by a sequence of m *independent* draws from D. Let $\mathbf{D}_{\mathcal{E}}(S)$ be the overall probability for an m-sample S to arise in this way,

$$\mathbf{D}_{\mathcal{E}}(S) = E_{D\sim\mathcal{E}}[D^m(S)].$$

The accumulation of experience is then modelled by n *independent* draws of samples $S_i \sim \mathbf{D}_{\mathcal{E}}$, resulting in the sample-sequence or *meta-sample* $\mathbf{S}=(S_1,...,S_n)$ (also called 'support sets' by S. Thrun, 1998, or (n,m)-samples by J. Baxter, 2000). The probability for $\mathbf{S}$ to arise in this manner is $(\mathbf{D}_{\mathcal{E}})^n(\mathbf{S})$ and depends completely on the environment $\mathcal{E}$. We generally use m to denote the size of the ordinary samples and n for the size of the meta samples. We also use bold letters $\mathbf{D}$, $\mathbf{S}$, $\mathbf{l}$, etc to distinguish objects of meta-learning from the corresponding objects of ordinary learning D, S, l, etc.

A learners behaviour is formally described by a learning algorithm A. To say that the meta-sample S is used to determine the behaviour of the learner on future learning tasks can therefore be expressed in the equation

$$A = \mathbf{A}(\mathbf{S})$$

where $\mathbf{A}$ is a function which returns a learning algorithm for every meta-sample $\mathbf{S}$. The object $\mathbf{A}$ will be called a *meta-algorithm*. Since $\mathbf{A}(\mathbf{S})$ is an algorithm we can train it with a sample S to obtain a hypothesis $A(\mathbf{S})(S)$.

An example of a meta-algorithm is feature-learning where $\mathbf{A}$ selects a feature map to preprocess the input of a fixed algorithm. Another example is given in Section 6. In general, any method that adjusts the parameters of an algorithm on the basis of the experience made with other learning tasks can be regarded as a meta-algorithm.

To state generalization error bounds for meta-algorithms, we need to define a statistical measure of the performance of an algorithm A with respect to an environment $\mathcal{E}$, analogous to the risk $R(c,D)$ of a hypothesis c with respect to a task D. The risk (1) measures the expected loss of a hypothesis for future data drawn from the task distribution D, so the analogous quantity for an algorithm should measure the expected loss of the hypothesis returned by the algorithm for future tasks drawn from the environmental distribution $\mathcal{E}$. A corresponding experiment involves the random draw of a task D from $\mathcal{E}$, training the algorithm with a sample S drawn randomly and independently from D, and applying the resulting hypothesis to data randomly drawn from D. Formally

$$\mathbf{R}(A,\mathcal{E}) = E_{D\sim\mathcal{E}}\left[E_{S\sim D^m}\left[R(A(S),D)\right]\right] = E_{D\sim\mathcal{E}}\left[E_{S\sim D^m}\left[E_{z\sim D}\left[l(A(S),z)\right]\right]\right]. \tag{4}$$

The *transfer risk* $\mathbf{R}(A,\mathcal{E})$ measures how well the algorithm A is adapted to the environment $\mathcal{E}$. If $\mathcal{E}$ is non-uniform the NFLT doesn't apply, and we may hope to optimize $\mathbf{R}(A,\mathcal{E})$ in A.

If the environment was known, we could in principle select A so as to minimize (4), but the only available information is the past experience or meta-sample $\mathbf{S}$. The situation is analogous to ordinary learning. Now suppose that $\mathbf{A}$ is a meta algorithm. The idea is to bound $\mathbf{R}(\mathbf{A}(\mathbf{S}),\mathcal{E})$ in terms of $\mathbf{S}$ with high probability in $\mathbf{S}$, as $\mathbf{S}$ is drawn from the environment $\mathcal{E}$ *for every environment* $\mathcal{E}$. Given $\mathbf{S}$ we can then reason that, regardless of $\mathcal{E}$, the bound is true with high probability. Formally we seek a function B such that, given a confidence parameter δ,

$$\forall\mathcal{E}, (\mathbf{D}_{\mathcal{E}})^n\left\{\mathbf{S} : \mathbf{R}(\mathbf{A}(\mathbf{S}),\mathcal{E}) \le B(\delta,\mathbf{S})\right\} \ge 1-\delta. \tag{5}$$

The principal contribution of this paper is a general method to prove bounds of this type for different classes of meta-algorithms.

The Method. Given an algorithm A, let $\mathbf{l}(A,S)$ be an *estimator* for the risk of $A(S)$ given the sample $S=(z_1,\dots,z_m)$. For example set $\mathbf{l}=l_{emp}$ with the empirical estimator

$$l_{emp}(A,S) = \sum_{i=1}^{m} l(A(S),z_i).$$

We then write, using $E_{S\sim\mathbf{D}_{\mathcal{E}}}[f(S)] = E_{D\sim\mathcal{E}}\left[E_{S\sim D^m}[f(S)]\right]$,

$$\begin{aligned}
&\mathbf{R}(\mathbf{A}(\mathbf{S}),\mathcal{E})\\
&= E_{S\sim\mathbf{D}_{\mathcal{E}}}\left[\mathbf{l}(\mathbf{A}(\mathbf{S}),S)\right] + E_{D\sim\mathcal{E}}\left[E_{S\sim D^m}\left[R(\mathbf{A}(\mathbf{S})(S),D) - \mathbf{l}(\mathbf{A}(\mathbf{S}),S)\right]\right]\\
&\le E_{S\sim\mathbf{D}_{\mathcal{E}}}\left[\mathbf{l}(\mathbf{A}(\mathbf{S}),S)\right] + \sup_{D,\mathbf{S}'}\left|E_{S\sim D^m}\left[R\left(\mathbf{A}\left(\mathbf{S}'\right)(S),D\right) - \mathbf{l}\left(\mathbf{A}\left(\mathbf{S}'\right),S\right)\right]\right|.
\end{aligned} \tag{6}$$

To control the first term in the last line it suffices to prove a bound of the type

$$\forall \mathbf{D} \in M_1 \left(Z^m\right), \mathbf{D}^n \left\{\mathbf{S} : E_{S \sim \mathbf{D}} \left[\mathbf{l}\left(\mathbf{A}\left(\mathbf{S}\right), S\right)\right] \leq \Pi\left(\delta, \mathbf{S}\right)\right\} \geq 1 - \delta, \tag{7}$$

where $\mathbf{D} \in M_1 \left(Z^m\right)$ refers to any probability distribution on the set Z^m of m-samples. Notice that (7) has exactly the same structure as an ordinary generalization error bound (3) where D has been repaced with $\mathbf{D}$, S with $\mathbf{S}$, A with $\mathbf{A}$, l with $\mathbf{l}$, and B with Π. We therefore propose to use established results of learning theory to obtain the statement (7). Because it controls future values of the estimator, a two-argument function Π satisfying (7) will be called an *estimator prediction bound* for $\mathbf{A}$ with respect to the estimator $\mathbf{l}$.

The simplest case, where a nontrivial estimator prediction bound can be found, occurs when $\mathbf{A}$ searches only a finite set of algorithms, but there are many other possibilities, some are listed in Section 3.

Suppose that we have established (7). To obtain (5) it will be sufficient to bound the second term in the last line of (6).

Methods for deriving ordinary generalization error bounds often use an intermediate bound on the estimation error

$$\left|R\left(A\left(S\right), D\right) - \mathbf{l}\left(A, S\right)\right|,$$

valid for all distributions with high probability in S, for example by bounding the complexity of a hypothesis space searched by A. Such bounds lead to a general method to control the second term in (6) and to prove (5). Theorem 5 states a corresponding result, which is applied in Section 5.2 to improve on the results in (Baxter, 2000).

A second method to bound the estimation error in (6) involves the notion of *algorithmic stability*. This method is less general but more elegant and often gives tighter bounds. Bousquet and Elisseeff (2002) have shown how generalization error bounds for learning algorithms can be obtained in an easy, elegant and direct way. Instead of measuring the size of the space which the algorithm searches, they concentrate directly on continuity properties of the algorithm in its dependence on the training sample. A learning algorithm is *uniformly* β*-stable* if the omission of a single example doesn't change the loss of the returned hypothesis by more than β, for any data point and training sample possible. Many algorithms are stable and stable algorithms have simple bounds on their estimation error. Corresponding theorems can be found in (Bousquet, Elisseeff, 2002). The requirement of stability has been weakened and the results have been extended by Kutin and Nyogi (2002).

If for some β and all $\mathbf{S}$ the algorithm $\mathbf{A}\left(\mathbf{S}\right)$ is uniformly β-stable, then the estimation term in (6) can be bounded in a particularly simple way, namely by 2β, as stated in Theorem 6.

Results. Algorithmic stability is also useful at a different level to prove that a meta-algorithm $\mathbf{A}$ has an estimator prediction bound. This can be done by appealing to Theorem 12 in (Bousquet, Elisseeff, 2002) (stated as Theorem 2 in Section 3). The following is an immediate consequence of this theorem in combination with our Theorem 6:

Theorem 1 *Suppose the meta-algorithm* $\mathbf{A}$ *satisfies the following two conditions:*

1. For every meta sample $\mathbf{S}=\left(S_1, \ldots, S_n\right)$*, let* $\mathbf{S}^{\setminus i}$ *be the same as* $\mathbf{S}$ *except that one of the* S_i *has been deleted. Then for every* $\mathbf{S}, \mathbf{S}^{\setminus i}$ *and every ordinary sample* S *we have*

$$\left|l_{emp}\left(\mathbf{A}\left(\mathbf{S}\right), S\right) - l_{emp}\left(\mathbf{A}\left(\mathbf{S}^{\setminus i}\right), S\right)\right| \leq \beta'.$$

2. For every ordinary sample $S = (z_1, ..., z_m)$, let $S^{\backslash i}$ is the same as S except that one of the z_i has been deleted. Then for every meta sample $\mathbf{S}$ and every S and $S^{\backslash i}$ we have

$$\left| l\left(\mathbf{A}(\mathbf{S})(S), z\right) - l\left(\mathbf{A}(\mathbf{S})\left(S^{\backslash i}\right), z\right) \right| \leq \beta.$$

Then for every environment $\mathcal{E}$ we have, with probability greater than $1-\delta$ in the meta-sample $\mathbf{S}=(S_1, ..., S_n)$ drawn from $(\mathbf{D}_{\mathcal{E}})^n$, the inequality

$$\mathbf{R}(\mathbf{A}(\mathbf{S}), \mathcal{E}) \leq \frac{1}{n}\sum_{i=1}^{n} l_{emp}(\mathbf{A}(\mathbf{S}), S_i) + 2\beta' + (4n\beta' + M)\sqrt{\frac{\ln(1/\delta)}{2n}} + 2\beta. \tag{8}$$

The left hand side of the last inequality measures the expected performance of the algorithm $\mathbf{A}(\mathbf{S})$ for all, and potentially yet unknown, tasks of the environment $\mathcal{E}$. The right side is composed of an empirical estimate and terms depending on the sample sizes n and m, the stability parameters β' and β and the confidence parameter δ. If $\beta' \approx 1/n^a$ and $\beta \approx 1/m^b$, with $a > 1/2$ and $b > 0$, the bound of the theorem becomes non-trivial.

We apply these results to a practical meta-algorithm for least squares regression. This meta-algorithm is related to the *Chorus of Prototypes* introduced by Edelman (1995), so we call it *CP-Regression*. CP-Regression takes the meta-sample $\mathbf{S}=(S_1, ..., S_n)$ and uses a primitive algorithm A_0 to compute a set of corresponding regression functions $h_1, ..., h_n$. For any new input object x the feature vector of x is then mixed with (or even replaced by) the vector $(h_1(x), ..., h_n(x))$. Finally $\mathbf{A}(\mathbf{S})$ is defined to be regularized least squares regression with this modified input representation. We show that Theorem 1 applies to this meta-algorithm, with $\beta' \approx 1/n$ and $\beta \approx 1/m$ as required.

CP-Regression can be implemented in practice and preliminary experiments seem to indicate that meta-learning gives a practical advantage over ordinary regularized least squares regression.

Outline of the Paper. In Section 2 we give a summary of the definitions and notation used in the paper. This section is intended as a reference for the reader. In Section 3 we show how to obtain estimator prediction bounds from standard results in learning theory. In Section 4 we derive transfer risk bounds for meta-algorithms. In Section 5 we attempt a comparison of our bounds to ordinary generalization error bounds and compare our method and results to the approach taken by J. Baxter (2000). In Section 6 we discuss regularized least squares regression, introduce CP-regression, analyse its properties and present some preliminary experimental results.

2. Definitions and Notation

This section is intended as a reference for the notation and definitions used in the paper.

Measurability. Any subset which we explicitely define on a measurable space will be assumed measurable, as will be any function. Thus for example '$F \subseteq \mathbb{R}$' is shorthand for the statement '$F \subseteq \mathbb{R}$ and F is Lebesgue-measurable'. $M_1(X)$ will always denote the space of probability measures on a measurable space X. We supply $M_1(X)$ with any σ-algebra containing the σ-algebra generated by the set of functions

$$\mu \in M_1(X) \mapsto E_{x \sim \mu}[f]$$

for all bounded measurable functions f and all singleton sets $\{\mu\}$ for $\mu \in M_1(X)$. In this way $M_1(X)$ becomes itself a measurable space and it makes sense to talk about $M_1(M_1(X))$.

Learning and Algorithms. Throughout Z will be a measurable space of *data-points* $z \in Z$, C a space of *hypotheses* or *concepts* $c \in C$ and $l : C \times Z \to [0,M]$ a *loss function*. *Samples* are polytuples $S \in \bigcup_{m=1}^{\infty} Z^m$, and *learning algorithms* are symmetric functions

$$A : \bigcup_{m=1}^{\infty} Z^m \to C.$$

Symmetry, which will be essential for our use of stability, means that for any permutation π on $\{1,...,m\}$ and any $S \in Z^m$ we have $A(\pi(S)) = A(S)$ where $\pi(S)$ refers to the permuted sample

$$\pi(z_1,...,z_m) = \left(z_{\pi(1)},...,z_{\pi(m)}\right).$$

The set of such algorithms depends only on C and Z and will be denoted by $\mathcal{A}(C,Z)$. The hypothesis $A(S)$ is what results when A is trained with S.

Learning Tasks and Risk. A *learning task* is specified by a probability measure $D \in M_1(Z)$. Given such a task D and a hypothesis $c \in C$ and a loss function l we use

$$R(c,D) = E_{z \sim D}[l(c,z)]$$

to denote the *risk* (=expected loss) of the hypothesis c in task D w.r.t. the loss function l.

Generalization Error Bounds. A function $B : (0,1] \times \bigcup_{m=1}^{\infty} Z^m \to [0,M]$ is a *generalization error bound* for the algorithm $A \in \mathcal{A}(C,Z)$ with respect to the loss function l iff

$$\forall D \in M_1(Z), \forall \delta > 0, D^m\{S : R(A(S),D) \leq B(\delta,S)\} \geq 1-\delta.$$

Estimators and Algorithmic Stability. The *leave-one-out estimator* l_{loo} and the *empirical estimator* l_{emp} are the functions (the notation is from Bousquet, Elisseeff, 2002)

$$l_{loo}, l_{emp} : \mathcal{A}(C,Z) \times (Z^m) \to [0,M]$$

defined for $A \in \mathcal{A}(C,Z)$ and $S = (z_1,...,z_m) \in Z^m$ by

$$l_{loo}(A,S) = \frac{1}{m}\sum_{i=1}^{m} l\left(A\left(S^{\setminus i}\right), z_i\right),$$

where $S^{\setminus i}$ generally denotes the sample S with the i-th element deleted, and

$$l_{emp}(A,S) = \frac{1}{m}\sum_{i=1}^{m} l(A(S), z_i).$$

For $\beta > 0$ an algorithm $A \in \mathcal{A}(C,Z)$ is called *uniformly* β*-stable w.r.t. the loss function* l if

$$|l(A(S),z) - l\left(A\left(S^{\setminus i}\right), z\right)| < \beta,$$

for every m, for every $S \in Z^m$, $z \in Z$ and $i \in \{1,...,m\}$.

Environments and Induced Distributions. A meta-learning task is specified by an *environment*

$$\mathcal{E} \in M_1(M_1(Z))$$

which models the drawing of learning tasks $D \sim \mathcal{E}$. The environment $\mathcal{E}$ defines an *induced distribution* $\mathbf{D}_{\mathcal{E}} \in M_1(Z^m)$, by

$$\mathbf{D}_{\mathcal{E}}(F) = E_{D\sim\mathcal{E}}[D^m(F)] \text{ for } F \subseteq Z^m \text{ measurable.} \tag{9}$$

The corresponding expectation for a measurable function f on Z^m is then

$$E_{S\sim\mathbf{D}_{\mathcal{E}}}[f] = E_{D\sim\mathcal{E}}[E_{S\sim D^m}[f(S)]].$$

The induced distribution $\mathbf{D}_{\mathcal{E}}$ models the probability $\mathbf{D}_{\mathcal{E}}(S)$ for an m-sample S to arise when a task D is drawn from the environment $\mathcal{E}$, followed by m independent draws of examples from the same distribution D. $\mathbf{D}_{\mathcal{E}}$ is not a product measure, but a mixture of symmetric product measures, and therefore itself symmetric. Repeated, independent draws from $\mathbf{D}_{\mathcal{E}}$ give rise to *meta-samples* (see below).

Transfer Risk. Given an environment $\mathcal{E} \in M_1(M_1(Z))$, an algorithm $A \in \mathcal{A}(C,Z)$ and a loss function $l : C \times Z \to [0,M]$ the *transfer risk* of A in the environment $\mathcal{E}$ w.r.t. the loss function l is given by

$$\mathbf{R}(A,\mathcal{E}) = E_{D\sim\mathcal{E}}[E_{S\sim D^m}[R(A(S),D)]].$$

It gives the expected risk of the hypothesis $A(S)$ for a task D randomly drawn from the environment and the sample S randomly drawn from this task. It measures how poorly the algorithm A is suited to the environment $\mathcal{E}$.

Meta-Samples and Meta-Algorithms. We use the letter $\mathbf{S}$ to denote a *meta-sample*, $\mathbf{S} = (S_1,...,S_n) \in (Z^m)^n$. Such can be generated by a sequence of n independent draws from some distribution $\mathbf{D} \in M_1(Z^m)$, typically the distribution $\mathbf{D}_{\mathcal{E}}$ induced by an environment $\mathcal{E}$, that is $\mathbf{S} \sim (\mathbf{D}_{\mathcal{E}})^n$.

$\mathcal{A}(\mathcal{A}(C,Z),Z^m)$ is the set of *meta algorithms*. That is for $\mathbf{A} \in \mathcal{A}(\mathcal{A}(C,Z),Z^m)$ and $\mathbf{S} \in \bigcup_{n=1}^{\infty}(Z^m)^n$ the object $\mathbf{A}(\mathbf{S})$ is the algorithm $A = \mathbf{A}(\mathbf{S}) \in \mathcal{A}(C,Z)$ which results from training $\mathbf{A}$ with the meta-sample $\mathbf{S}$. Given an m-sample S, the object $\mathbf{A}(\mathbf{S})(S)$ is the hypothesis returned by the algorithm $\mathbf{A}(\mathbf{S})$, when trained with an ordinary sample S.

Estimator Prediction Bounds. A function $\Pi : (0,1] \times \bigcup_{n=1}^{\infty}(Z^m)^n \to [0,M]$ is an *estimator prediction bound* for the meta-algorithm $\mathbf{A} \in \mathcal{A}(\mathcal{A}(C,Z),Z^m)$ with respect to the estimator $\mathbf{l} : \mathcal{A}(C,Z) \times (Z^m) \to [0,M]$ iff

$$\forall \mathbf{D} \in M_1(Z^m), \forall \delta > 0, \mathbf{D}^n\{\mathbf{S} : E_{S\sim\mathbf{D}}[\mathbf{l}(\mathbf{A}(\mathbf{S}),S)] \leq \Pi(\delta,\mathbf{S})\} \geq 1-\delta. \tag{10}$$

An estimator prediction bound is formally equivalent to an ordinary generalization bound under the identifications $Z \leftrightarrow Z^m$, $C \leftrightarrow \mathcal{A}(C,Z)$, $l \leftrightarrow \mathbf{l}$, $A \leftrightarrow \mathbf{A}$, $B \leftrightarrow \Pi$.

Meta-Estimators. Given an estimator $\mathbf{l} : \mathcal{A}(C,Z) \times (Z^m) \to [0,M]$ the *empirical meta-estimator* $\mathbf{l}_{emp}$ is the function

$$\mathbf{l}_{emp} : \mathcal{A}(\mathcal{A}(C,Z),Z^m) \times (Z^m)^n \to [0,M]$$

defined for $\mathbf{A} \in \mathcal{A}(\mathcal{A}(C,Z), Z^m)$ and $\mathbf{S} = (S_1, ..., S_n) \in (Z^m)^n$ by

$$\mathbf{l}_{emp}(\mathbf{A}, \mathbf{S}) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{l}(\mathbf{A}(\mathbf{S}), S_i).$$

The meta-estimator $\mathbf{l}_{loo}$ is defined analogously. These definitions depend on the choice of the estimator $\mathbf{l}$ itself. For example if $\mathbf{l}$=l_{loo} then

$$(l_{loo})_{emp}(\mathbf{A}, \mathbf{S}) = \frac{1}{n} \sum_{i=1}^{n} l_{loo}(\mathbf{A}(\mathbf{S}), S_i).$$

	Ordinary learning	Meta learning
Data	$z \in Z$	$S = (z_1, ..., z_m) \in Z^m$
Samples	$S = (z_1, ..., z_m) \in Z^m$	$\mathbf{S} = (S_1, ..., S_n) \in (Z^m)^n$
Hypotheses	$c \in C$	$A \in \mathcal{A}(C,Z)$
Algorithms	$A \in \mathcal{A}(C,Z)$	$\mathbf{A} \in \mathcal{A}(\mathcal{A}(C,Z), Z^m)$
Loss function	$l : C \times Z \to [0,M]$	$\mathbf{l} : \mathcal{A}(C,Z) \times Z^m \to [0,M]$, where $\mathbf{l} = l_{emp}$ or l_{loo}
Learning Task	$D \in M_1(Z)$	$\mathbf{D} \in M_1(Z^m)$, typically $\mathbf{D} = \mathbf{D}_{\mathcal{E}}$ where $\mathbf{D}_{\mathcal{E}}$ is induced by an environment $\mathcal{E} \in M_1(M_1(Z))$ (see(9))
Empirical estimator	$l_{emp}(A,S) =$ $= \frac{1}{m}\sum_{i=1}^{m} l(A(S), z_i)$	$\mathbf{l}_{emp}(\mathbf{A}, \mathbf{S}) =$ $= \frac{1}{n}\sum_{i=1}^{n} \mathbf{l}(\mathbf{A}(\mathbf{S}), S_i)$
Risk	$R(c,D) = E_{z \sim D}[l(c,z)]$	$E_{S \sim \mathbf{D}}[\mathbf{l}(A,S)]$
Bound	Generalization error	Estimator prediction

Table 1: This table relates the descriptions of ordinary and meta-learning tasks.

An important object which is *not* mapped is the transfer risk $\mathbf{R}(A, \mathcal{E})$. Correspondingly an estimator prediction bound is *not* a generalization error bound for the transfer risk.

Covering Numbers. These definitions are taken from (Anthony, Bartlett, 1999). Let X be a set, $X_0 \subseteq X$. For $\varepsilon > 0$ and a metric d on X the covering numbers $\mathcal{N}(\varepsilon, X_0, d)$ are defined by

$$\mathcal{N}(\varepsilon, X_0, d) = \min\left\{N \in \mathbb{N} : \exists (x_1, ..., x_N) \in X^N, \forall x \in X_0, \exists i, d(x, x_i) \leq \varepsilon\right\}.$$

For a class $\mathcal{F}$ of real functions on X and $S = (x_1, ..., x_n) \in X^n$ define $\mathcal{F} \restriction_S \subseteq \mathbb{R}^n$ by

$$\mathcal{F}|_S = \{(f(x_1), ..., f(x_n)) : f \in \mathcal{F}\},$$

and define, for $\varepsilon > 0$ and any given n,

$$\mathcal{N}_1(\varepsilon, \mathcal{F}, n) = \sup_{S \in X^n} \mathcal{N}(\varepsilon, \mathcal{F}|_S, d_1),$$

where d_1 is the metric on $\mathbb{R}^n$ defined by

$$d_1(x,y) = \frac{1}{n}\sum_{i=1}^{n} |x_i - y_i|.$$

Loss Function Classes. Let $\mathcal{H} \subseteq C$. The *loss function class* $\mathcal{F}(\mathcal{H}, l)$ is the family of real functions

$$\mathcal{F}(\mathcal{H}, l) = \left\{z \in Z \mapsto l(c, Z) : c \in \mathcal{H}\right\}.$$

For $\mathcal{F}(\mathcal{H}, l)$ we use the topology of pointwise convergence which it inherits as a subset of $[0,M]^Z$. A set $\mathcal{H} \subseteq C$ is called *closed* if $\mathcal{F}(\mathcal{H}, l)$ is closed in this topology (and therefore also compact by Tychonoffs theorem). If $\mathcal{H}$ is closed then any finite linear combination of functions $c \in \mathcal{H} \mapsto \sum_i \alpha_i l(c, z_i)$ attains minima and maxima in $\mathcal{H}$.

For $\mathbf{H} \subseteq \mathcal{A}(C,Z)$ and a given estimator $\mathbf{l} : \mathcal{A}(C,Z) \times Z^m \to [0,M]$ we define an analogous (meta-) loss function class

$$\mathcal{F}(\mathbf{H}, \mathbf{l}) = \{S \in Z^m \mapsto \mathbf{l}(A, S) : A \in \mathbf{H}\}.$$

3. Estimator Prediction Bounds

In this section we give examples of estimator prediction bounds obtained from established results of statistical learning theory.

Selection from a Finite Set. Set the bound on the loss function M to be equal to 1 for simplicity and suppose that there is a *finite* set of hypotheses $\mathcal{H} = \{c_1, ..., c_K\} \subseteq C$. Define the algorithm A for a sample $S = (z_1, ..., z_m) \in Z^m$ by

$$A(S) = \arg\min_{c \in \mathcal{H}} \frac{1}{m}\sum_{j=1}^{m} l(c, z_j).$$

A well known application of Hoeffdings inequality and a union bound (see e.g. Anthony, Bartlett, 1999) give, for any $\delta > 0$,

$$\forall D, D^m \left\{S : \sup_{c \in \mathcal{H}} \left| R(c,D) - \frac{1}{m}\sum_{j=1}^{m} l(c, z_j)\right| \le \sqrt{\frac{\ln(K/\delta)}{2m}}\right\} \ge 1 - \delta, \tag{11}$$

which gives the following generalization error bound for A:

$$\forall D \in M_1(Z), \forall \delta > 0, D^m\{S : R(A(S), D) \le B(\delta, S)\} \ge 1 - \delta$$

with

$$B(\delta, S) = l_{emp}(A, S) + \sqrt{\frac{\ln K + \ln(1/\delta)}{2m}}.$$

Note that this bound also holds for every algorithm searching a finite set of hypotheses of cardinality at most K, that is for every algorithm with $A(S) \in \mathcal{H}$ for all S and some set $\mathcal{H}$ with $|\mathcal{H}| \le K$.

We now use the table at the end of the previous section. Substituting Z^m for Z, $\mathcal{A}(C,Z)$ for C, $\mathbf{l} = l_{emp}$ or $\mathbf{l} = l_{loo}$ for l and a finite set of algorithms $\{A_1, ..., A_K\}$ for $\{c_1, ..., c_K\}$, we arrive at the following statement:

Every meta algorithm $\mathbf{A}$ that such $\mathbf{A}(\mathbf{S}) \in \{A_1, ..., A_K\}$ for all $\mathbf{S} = (S_1, ..., S_n)$ has the estimator prediction bound

$$\forall \mathbf{D} \in M_1(Z^m), \forall \delta > 0, \mathbf{D}^n \{\mathbf{S} : E_{S \sim \mathbf{D}} [\mathbf{l}(\mathbf{A}(\mathbf{S}), S)] \leq \Pi(\delta, \mathbf{S})\} \geq 1 - \delta$$

with

$$\Pi(\delta, \mathbf{S}) = \mathbf{l}_{emp}(\mathbf{A}, \mathbf{S}) + \sqrt{\frac{\ln K + \ln(1/\delta)}{2n}}. \tag{12}$$

Selection from a Set of Bounded Complexity. Again with $M = 1$ consider a subset $\mathcal{H} \subseteq C$. It follows from the analysis in chapter 17 in (Anthony, Bartlett, 1999) and Theorem 21.1 of the same reference, that the following holds for every $0 < \varepsilon < 1$ and every distribution D on Z:

$$\begin{aligned} &D^m \left\{ S \in Z^m : \forall c \in \mathcal{H}, \left| E_{z \sim D} [l(c,z)] - \frac{1}{m} \sum_{j=1}^{m} l(c, z_j) \right| \leq \varepsilon \right\} \\ &\geq 1 - 4\mathcal{N}_1 \left(\frac{\varepsilon}{8}, \mathcal{F}(\mathcal{H}, l), 2m \right) e^{\frac{-\varepsilon^2 m}{32}}. \end{aligned} \tag{13}$$

which implies the following generalization error bound, valid for every algorithm A searching only the hypothesis space $\mathcal{H}$:

$$B(\delta, S) = l_{emp}(A, S) + \inf \left\{ t : 4\mathcal{N}_1 \left(\frac{t}{8}, \mathcal{F}(\mathcal{H}, l), 2m \right) e^{\frac{-t^2 m}{32}} \leq \delta \right\}. \tag{14}$$

Suppose now that $\mathbf{H} \subseteq \mathcal{A}(C, Z)$ is a space of algorithms and fix an estimator $\mathbf{l} = l_{loo}$ or $\mathbf{l} = l_{emp}$. Substituting Z^m for Z, $\mathcal{A}(C, Z)$ for C, $\mathbf{l}$ for l and $\mathbf{H}$ for $\mathcal{H}$, and $\mathcal{F}(\mathbf{H}, \mathbf{l})$ for $\mathcal{F}(\mathcal{H}, l)$ in the above, we obtain analogous to (13):

For every $0 < \varepsilon < 1$ and every distribution $\mathbf{D}$ on Z^m:

$$\begin{aligned} &\mathbf{D}^m \left\{ \mathbf{S} \in (Z^m)^n : \forall A \in \mathbf{H}, \left| E_{S \sim \mathbf{D}} [\mathbf{l}(A, S)] - \frac{1}{n} \sum_{j=1}^{n} \mathbf{l}(A, S_j) \right| \leq \varepsilon \right\} \\ &\geq 1 - 4\mathcal{N}_1 \left(\frac{\varepsilon}{8}, \mathcal{F}(\mathbf{H}, \mathbf{l}), 2n \right) e^{\frac{-\varepsilon^2 n}{32}}. \end{aligned}$$

Every meta-algorithm $\mathbf{A}$ such $\mathbf{A}(\mathbf{S}) \in \mathbf{H}$ for all $\mathbf{S}$ has thus the estimator prediction bound

$$\Pi(\delta, \mathbf{S}) = \mathbf{l}_{emp}(\mathbf{A}, \mathbf{S}) + \inf \left\{ t : 4\mathcal{N}_1 \left(\frac{t}{8}, \mathcal{F}(\mathbf{H}, \mathbf{l}), 2n \right) e^{\frac{-t^2 n}{32}} \leq \delta \right\}. \tag{15}$$

Uniformly Stable Algorithms. Now let $M > 0$ be arbitrary. Bousquet and Elisseeff (2002) prove that uniformly β-stable algorithms have a generalization error bound with sample-independent bound on the estimation error:

Theorem 2 *Let $A \in \mathcal{A}(C,Z)$ be uniformly β-stable. Then for any learning task $D \in M_1(Z)$ and any positive integer m, with probability greater $1-\delta$ in a sample S drawn from D^m*

$$l(A(S),D) \leq l_{loo}(A,S) + \beta + (4m\beta + M)\sqrt{\frac{\ln\frac{1}{\delta}}{2m}}$$

and

$$l(A(S),D) \leq l_{emp}(A,S) + 2\beta + (4m\beta + M)\sqrt{\frac{\ln\frac{1}{\delta}}{2m}}.$$

These bounds are good if we can show uniform β-stability with $\beta \approx 1/m^a$, with $a > 1/2$. The notion of uniform stability easily transfers to meta-algorithms to give estimator prediction bounds. Fix an estimator $\mathbf{l} = l_{loo}$ or $\mathbf{l} = l_{emp}$ and suppose that the meta-algorithm satisfies the following condition:

For every meta sample $\mathbf{S} = (S_1, \ldots, S_n)$, if $\mathbf{S}'$ is the same as $\mathbf{S}$ except that one of the S_i has been deleted, and for every ordinary sample S we have

$$\left|\mathbf{l}(\mathbf{A}(\mathbf{S}),S) - \mathbf{l}\left(\mathbf{A}\left(\mathbf{S}'\right),S\right)\right| \leq \beta.$$

Theorem 2 then gives the estimator prediction bounds

$$\Pi_{loo}(\delta,\mathbf{S}) = \mathbf{l}_{loo}(\mathbf{A},\mathbf{S}) + \beta + (4n\beta + M)\sqrt{\frac{\ln\frac{1}{\delta}}{2n}} \tag{16}$$

and

$$\Pi_{emp}(\delta,\mathbf{S}) = \mathbf{l}_{emp}(\mathbf{A},\mathbf{S}) + 2\beta + (4n\beta + M)\sqrt{\frac{\ln\frac{1}{\delta}}{2n}}. \tag{17}$$

4. Transfer Risk Bounds for Meta Algorithms

To derive the results in this section we need the following simple lemma, which can also be found in (Bousquet, Elisseeff, 2002).

Lemma 3 *Let $A \in \mathcal{A}(C,Z)$. Then for any learning task $D \in M_1(Z)$*

1. *We have $E_{S\sim D^m}[l_{loo}(A,S)] = E_{S'\sim D^{m-1}}[R(A(S'),D)]$.*
2. *If A is uniformly β-stable then $|E_{S\sim D^m}[l_{emp}(A,S)] - E_{S\sim D^m}[l_{loo}(A,S)]| \leq \beta$.*

Proof Using the permutation symmetry of A and of the measure D^m we get

$$\begin{aligned} E_{S\sim D^m}[l_{loo}(A,S)] &= \frac{1}{m}\sum_{i=1}^{m} E_{S\sim D^m}\left[l\left(A\left(S^{\setminus i}\right),z_i\right)\right] \\ &= \frac{1}{m}\sum_{i=1}^{m} E_{S'\sim D^{m-1}}\left[E_{z\sim D}\left[l\left(A\left(S'\right),z\right)\right]\right] \\ &= E_{S'\sim D^{m-1}}\left[R\left(A\left(S'\right),D\right)\right]. \end{aligned}$$

Also

$$\begin{aligned}
&\left|E_{S\sim D^m}\left[l_{emp}(A,S)-l_{loo}(A,S)\right]\right| \\
&\leq \frac{1}{m}\sum_{i=1}^{m}\left|E_S\left[l(A(S),z_i)-l\left(A\left(S^{\setminus i}\right),z_i\right)\right]\right| \\
&\leq \frac{1}{m}\sum_{i=1}^{m}\left|E_S[\beta]\right| = \beta.
\end{aligned}$$

■

Suppose now that we have an estimator prediction bound Π for the meta-algorithm $\mathbf{A}$ with respect to the estimator $\mathbf{l}$, so that, for all $\delta > 0$,

$$\forall \mathbf{D} \in M_1(Z^m), \mathbf{D}^n\left\{\mathbf{S} : E_{S\sim\mathbf{D}}\left[\mathbf{l}(\mathbf{A}(\mathbf{S}),S)\right] \leq \Pi(\delta,\mathbf{S})\right\} \geq 1-\delta, \tag{18}$$

where the estimator $\mathbf{l} : \mathcal{A}(C,Z)\times Z^m \to [0,M]$ refers to either l_{emp} or l_{loo}. We have outlined several ways to obtain such bounds in Section 3.

When $\mathbf{l} = l_{loo}$ the bound (18) is already powerful by itself. By the definition of $\mathbf{D}_{\mathcal{E}}$ and the first conclusion of Lemma 3 we have

$$\begin{aligned}
E_{S\sim\mathbf{D}_{\mathcal{E}}}\left[l_{loo}(\mathbf{A}(\mathbf{S}),S)\right] &= E_{D\sim\mathcal{E}}\left[E_{S\sim D^m}\left[l_{loo}(\mathbf{A}(\mathbf{S}),S)\right]\right] \\
&= E_{D\sim\mathcal{E}}\left[E_{S'\sim D^{m-1}}\left[R\left(\mathbf{A}(\mathbf{S})\left(S'\right),D\right)\right]\right].
\end{aligned}$$

Substituting $\mathbf{D}_{\mathcal{E}}$ for $\mathbf{D}$ in (18) we conclude

Theorem 4 *If the meta-algorithm* $\mathbf{A}$ *satisfies the estimator prediction bound (18) with* $\mathbf{l} = l_{loo}$ *then for every environment* $\mathcal{E}$*, with probability greater than* $1-\delta$ *in the meta sample drawn from* $(\mathbf{D}_{\mathcal{E}})^n$ *we have*

$$E_{D\sim\mathcal{E}}\left[E_{S\sim D^{m-1}}\left[R(\mathbf{A}(\mathbf{S})(S),D)\right]\right] \leq \Pi(\delta,\mathbf{S}). \tag{19}$$

The left side of (19) is not quite equal to the transfer risk $\mathbf{R}(A,\mathcal{E})$. Here is a first application of this bound: Let $\{A_1,...,A_K\}$ be a finite collection of algorithms. For any meta sample $\mathbf{S} = (S_1,...,S_n)$ define $\mathbf{A}(\mathbf{S})$ to be

$$\mathbf{A}(\mathbf{S}) = \arg\min_{A\in\{A_1,...,A_K\}} \frac{1}{n}\sum_{i=1}^{n} l_{loo}(A,S_i).$$

The meta-algorithm $\mathbf{A}$ selects the algorithm with the lowest leave-one-out error on average over the meta-sample. Applying the estimator prediction bound (12) for this type of algorithm in combination with (19) above then gives, for any $\mathcal{E}$ and with probability greater than $1-\delta$ in the meta sample drawn from $(\mathbf{D}_{\mathcal{E}})^n$,

$$E_{D\sim\mathcal{E}}\left[E_{S\sim D^{m-1}}\left[R(\mathbf{A}(\mathbf{S})(S),D)\right]\right] \leq \frac{1}{n}\sum_{i=1}^{n} l_{loo}(\mathbf{A}(\mathbf{S}),S_i) + \sqrt{\frac{\ln(K/\delta)}{2n}}. \tag{20}$$

A similar result should hold if l_{loo} is replaced by any other, nearly unbiased estimator. A popular procedure, for example, is dividing the samples $S \in \mathbf{S}$ into training- and test-samples to estimate the generalization performance of an algorithm. If we chose from a finite set of candidates the algorithm $\mathbf{A}(\mathbf{S})$ which performs best on average over the test data in $\mathbf{S}$, when trained with the training data in $\mathbf{S}$, then we are implementing a version of the above meta-algorithm, and a corresponding version of (20) gives a probable performance guarantee for $\mathbf{A}(\mathbf{S})$ on future learning tasks drawn from the same environment as $\mathbf{S}$.

For more sophisticated meta-algorithms we need to consider the case $\mathbf{l} = l_{emp}$. In this case an estimator prediction bound only bounds the expected empirical error $l_{emp}(\mathbf{A}(\mathbf{S}), S)$ of $\mathbf{A}(\mathbf{S})$ for a sample S drawn from $\mathbf{D}_{\mathcal{E}}$, but it does not give any generalization guarantee for the hypothesis $\mathbf{A}(\mathbf{S})(S)$. For example $\mathbf{A}(\mathbf{S})$ could be some single-nearest-neighbour algorithm for which we would have $l_{emp}(\mathbf{A}(\mathbf{S}), S) = 0$ for almost all S, but $\mathbf{A}(\mathbf{S})$ would have poor generalization performance.

Recall the decomposition of the transfer risk (6) in the introduction:

$$\begin{aligned}&\mathbf{R}(\mathbf{A}(\mathbf{S}), \mathcal{E}) \\ &\leq E_{S\sim\mathbf{D}_{\mathcal{E}}}[\mathbf{l}(\mathbf{A}(\mathbf{S}), S)] + \sup_{D,\mathbf{S}'} \left| E_{S\sim D^m}\left[R\left(\mathbf{A}\left(\mathbf{S}'\right), D\right) - \mathbf{l}\left(\mathbf{A}\left(\mathbf{S}'\right), S\right)\right] \right|.\end{aligned}$$

The estimator prediction bound controls the first term above, so it remains to bound the second term which is independent of $\mathbf{S}$. We need to bound the expected estimation error of the estimator $\mathbf{l}$ uniformly for all distributions D and all algorithms $\mathbf{A}(\mathbf{S})$ for all meta-samples $\mathbf{S}$.

Theorem 5 *Suppose the meta-algorithm* $\mathbf{A}$ *has an estimator prediction bound* Π *with respect to the estimator* $\mathbf{l} = l_{emp}$ *, and that for every* $\eta > 0$ *there is a number* $B(\eta)$ *such that for every distribution* $D \in M_1(Z)$*, and every meta-sample* $\mathbf{S}$ *we have*

$$D^m\{S : |R(\mathbf{A}(\mathbf{S})(S), D) - l_{emp}(\mathbf{A}(\mathbf{S}), S)| \leq B(\eta)\} \geq 1 - \eta. \tag{21}$$

Let $\varepsilon = \inf_\eta (B(\eta) + M\eta)$*. Then for every environment* $\mathcal{E}$ *, with probability greater than* $1 - \delta$ *in* $\mathbf{S}$ *as drawn from* $(\mathbf{D}_{\mathcal{E}})^n$ *we have*

$$\mathbf{R}(\mathbf{A}(\mathbf{S}), \mathcal{E}) \leq \Pi(\delta, \mathbf{S}) + \varepsilon.$$

Proof For any D, $\mathbf{S}$ and arbitrary η we have

$$\begin{aligned}&E_{S\sim D^m}[R(\mathbf{A}(\mathbf{S})(S), D)] \\ &\leq E_{S\sim D^m}[l_{emp}(\mathbf{A}(\mathbf{S}), S)] + E_{S\sim D^m}[|R(\mathbf{A}(\mathbf{S})(S), D) - l_{emp}(\mathbf{A}(\mathbf{S}), S)|] \\ &\leq E_{S\sim D^m}[l_{emp}(\mathbf{A}(\mathbf{S}), S)] + B(\eta) + M\eta,\end{aligned}$$

where (21) was used in the last inequality. Taking the expectation $D \sim \mathcal{E}$ gives

$$\begin{aligned}\mathbf{R}(\mathbf{A}(\mathbf{S}), \mathcal{E}) &= E_{D\sim\mathcal{E}}[E_{S\sim D^m}[R(\mathbf{A}(\mathbf{S})(S), D)]] \\ &\leq E_{D\sim\mathcal{E}}[E_{S\sim D^m}[l_{emp}(\mathbf{A}(\mathbf{S}), S)]] + \varepsilon \\ &= E_{S\sim\mathbf{D}_{\mathcal{E}}}[l_{emp}(\mathbf{A}(\mathbf{S}), S)] + \varepsilon \\ &\leq \Pi(\delta, \mathbf{S}) + \varepsilon,\end{aligned}$$

where the last inequality holds with probability greater than $1 - \delta$ in the meta-sample $\mathbf{S}$ as drawn from $(\mathbf{D}_{\mathcal{E}})^n$ by virtue of the estimator prediction bound (18) applied with $\mathbf{D}_{\mathcal{E}}$ in place of $\mathbf{D}$. ■

The condition (21) is often satisfied, typically with $B(\delta)$ decreasing as $\ln(1/\delta)$ in δ and as $m^{-1/2}$ in m, so we should get a bound ε decreasing about as quickly as $\sqrt{\ln(m)/m}$. Using the results in Section 3, now on the level of ordinary learning, we see that the above theorem can be applied

- if every $\mathbf{A}(\mathbf{S})$ selects a hypothesis from a finite set $\mathcal{H}(\mathbf{S})$ of choices with $\left|\mathcal{H}(\mathbf{S})\right| \leq K$ for all $\mathbf{S}$. This follows from (11) The $\mathcal{H}(\mathbf{S})$ may of course be different for different $\mathbf{S}$..
- if every $\mathbf{A}(\mathbf{S})$ selects a hypothesis from a set $\mathcal{H}(\mathbf{S}) \subseteq C$ with uniformly bounded complexities. Here we use (13). An application is given in Section 5.2.
- if every $\mathbf{A}(\mathbf{S})$ is uniformly β-stable with $\beta \approx 1/m$. This follows from Theorem 2.

In the last case we can give a much better bound, where the additional error term ε is often of order $1/m$:

Theorem 6 *Suppose the meta-algorithm* $\mathbf{A}$ *has an estimator prediction bound* Π *with respect to the estimator* $\mathbf{l} = l_{emp}$ *, and that for some* β *the algorithms* $\mathbf{A}(\mathbf{S})$ *are uniformly* β*-stable for every meta-sample* $\mathbf{S}$. *Then for any environment* $\mathcal{E}$ *and* $\delta > 0$, *with probability greater than* $1-\delta$ *in* $\mathbf{S}$ *as drawn from* $(\mathbf{D}_{\mathcal{E}})^n$

$$\mathbf{R}(\mathbf{A}(\mathbf{S}), \mathcal{E}) \leq \Pi(\delta, \mathbf{S}) + 2\beta.$$

Proof We have

$$\begin{aligned} E_{S\sim D^m}\left[R(\mathbf{A}(\mathbf{S})(S), D)\right] &\leq E_{S'\sim D^{m-1}}\left[R\left(\mathbf{A}(\mathbf{S})\left(S'\right), D\right)\right] + \beta \\ &= E_{S\sim D^m}\left[l_{loo}(\mathbf{A}(\mathbf{S}), S)\right] + \beta \\ &\leq E_{S\sim D^m}\left[l_{emp}(\mathbf{A}(\mathbf{S}), S)\right] + 2\beta, \end{aligned}$$

where the first inequality follows directly from uniform stability and the next lines follow from Lemma 3. Taking the expectation $D \sim \mathcal{E}$ and using the estimator prediction bound (18) with $\mathbf{D}_{\mathcal{E}}$ in place of $\mathbf{D}$ gives the result in just as in the proof of the previous theorem. ■

Theorem 1 now follows immediately from Theorem 6 and from the estimator prediction bound (17) in Section 3. In Section 6 an application of this theorem to a practical meta-learning algorithm is discussed.

The estimator prediction bound $\Pi(\delta, \mathbf{S})$ will typically depend on the size n of the meta-sample $\mathbf{S} = (S_1, ..., S_n)$, and not on the size m of the constituting samples S_i. One may therefore wonder, how we can have an m-dependence of the estimation error as 2β (often order $1/m$), while in Theorem 2 (Bousquet, Elisseeff, 2002) it is $2\beta + O\left(\sqrt{1/m}\right)$. The reason for this difference is that to bound the transfer-risk in the above proof we only need to bound the expectation in S of the random variable $R(\mathbf{A}(\mathbf{S})(S), D)$, whereas the proof of Theorem 2 in (Bousquet, Elisseeff, 2002) needs to use McDiarmid's concentration inequality to bound this random variable itself with high probability in S, which is where the $O\left(\sqrt{1/m}\right)$ term comes from.

5. Comparison to Other Results

In this section we relate our results to others, beginning with a comparison to ordinary generalization bounds. Then we compare our method to the approach taken by J. Baxter (2000) where the generalization of meta-algorithms is also studied.

5.1 Comparison to Ordinary Generalization Error Bounds

Are our results better or worse than ordinary generalization error bounds? This question is at the same time very important and very imprecise, because the two kinds of results refer to different objects and situations.

The ordinary generalization error bound (examples in Section 3) applies to a situation where a sample S has already been drawn from an unknown task D and the estimator $l_{emp}(A,S)$ already has a definite value. It typically has the structure

$$\forall D, D^m \{S : R(A(S), D) \leq l_{emp}(A,S) + \varepsilon_0\} \geq 1 - \delta$$

where ε_0 is a bound on the estimation error. Often $\varepsilon_0 \approx \sqrt{1/m}$.

Our bounds on the other hand apply to a situation where only the meta-sample $\mathbf{S}$ is known, and typically have the structure

$$\forall \mathcal{E}, (\mathbf{D}_{\mathcal{E}})^n \left\{\mathbf{S} : \mathbf{R}(\mathbf{A}(\mathbf{S}), \mathcal{E}) \leq \Pi(\delta, \mathbf{S}) + \varepsilon_0'\right\} \geq 1 - \delta$$

where $\Pi(\delta, \mathbf{S})$ is the estimator prediction bound and ε_0' is again a bound on the estimation error, uniformly valid for all algorithms $A = \mathbf{A}(\mathbf{S})$ for any $\mathbf{S}$.

To get ε_0' our method always requires some condition (uniform bounds on estimation errors, β-stability) on the algorithms $\mathbf{A}(\mathbf{S})$, which is also sufficient to prove an ordinary generalization error bound for such algorithms $\mathbf{A}(\mathbf{S})$. The corresponding estimation errors are about the same in our bounds and in the ordinary generalization error bounds. In case of Theorem 5 our ε_0' is slightly worse than that of the ordinary bound (i.e. $\sqrt{\ln(m)/m}$ vs $\sqrt{1/m}$), in case of Theorem 6 it is actually better (2β vs $2\beta + O\left(\sqrt{1/m}\right)$). Let's ignore these differences and put $\varepsilon_0 = \varepsilon_0'$. Comparing the two bounds therefore involves a comparison of the estimator prediction bound $\Pi(\delta, \mathbf{S})$ to a 'generic' value of the estimator $l_{emp}(A,S)$.

Our bound $\Pi(\delta, \mathbf{S})$ has the disadvantage that it contains an additional error of meta-estimation. But as the size n of the meta-sample $\mathbf{S}$ becomes large, corresponding to an experienced meta-learner, this additional term tends to zero, and $\Pi(\delta, \mathbf{S})$ is likely to win over the 'generic' $l_{emp}(A,S)$, because $\mathbf{A}(\mathbf{S})$ is likely to outperform the 'generic' algorithm A on the meta-sample $\mathbf{S}$. To make this precise we have to give more meaning to the word 'generic'.

While it is easy to define a generic value of S (simply taking $S \sim \mathbf{D}_{\mathcal{E}}$ if some environment $\mathcal{E}$ is given), it is not so clear how we should pick a generic algorithm A. For simplicity consider a finite set of algorithms $\{A_1, ..., A_K\}$. We should select A uniformly at random from this set to obtain a generic algorithm. The generic value of $l_{emp}(A,S)$ is then

$$\Gamma = E_{S \sim \mathbf{D}_{\mathcal{E}}}\left[\frac{1}{K}\sum_{k=1}^{K} l_{emp}(A_k, S)\right].$$

The meta algorithm to consider for comparison is

$$\mathbf{A}(\mathbf{S}) = \arg \min_{A \in \{A_1,\ldots,A_K\}} \frac{1}{n} \sum_{S \in \mathbf{S}} l_{emp}(A,S)$$

with the estimator prediction bound

$$\begin{aligned} E_{S \sim \mathbf{D}_Q}\left[l_{emp}(\mathbf{A}(\mathbf{S}),S)\right] &\leq \min_{k=1}^{K} \frac{1}{n} \sum_{S_i \in \mathbf{S}} l_{emp}(A_k,S_i) + \sqrt{\frac{\ln(K/\delta_M)}{2n}} \\ &= \Pi(\delta_M,\mathbf{S}), \end{aligned} \tag{22}$$

where δ_M is the confidence parameter associated with the draw of the meta-sample $\mathbf{S}$. Now let

$$\Delta(\mathbf{S}) = \frac{1}{K}\sum_{k=1}^{K} \frac{1}{n} \sum_{S \in \mathbf{S}} l_{emp}(A_k,S) - \min_{k=1}^{K} \frac{1}{n} \sum_{S \in \mathbf{S}} l_{emp}(A_k,S).$$

$\Delta(\mathbf{S})$ will be positive unless all algorithms behave the same on the meta-sample, in which case it is zero and meta-learning is indeed pointless (essentially an empirical instantiation of the NFLT). With the bound M on the loss function equal to 1, an application of Hoeffding's inequality gives, with probability greater than $1-\delta_M$ in a meta sample $\mathbf{S}$ drawn from $(\mathbf{D}_{\mathcal{E}})^n$,

$$\frac{1}{n}\sum_{S \in \mathbf{S}} \frac{1}{K}\sum_{k=1}^{K} l_{emp}(A_k,S) \leq \Gamma + \sqrt{\frac{\ln(1/\delta_M)}{2n}},$$

so with probability greater than $1-2\delta_M$ in the meta-sample $\mathbf{S}$ we have

$$\Gamma - \Pi(\delta_M,\mathbf{S}) \geq \Delta(\mathbf{S}) - \frac{\sqrt{\ln(1/\delta_M)} + \sqrt{\ln K + \ln(1/\delta_M)}}{\sqrt{2n}}, \tag{23}$$

in addition to validitiy of our bound (22). So for large meta-samples $\mathbf{S}$ our bounds will very probably be true and better than the generic value of ordinary generalization bounds by a margin of roughly $\Delta(\mathbf{S})$.

For a practical perspective consider image recognition, when the tasks in the support of $\mathcal{E}$ share a certain invariance property (say image rotation), and there is only one algorithm in $\{A_1,\ldots,A_K\}$ having this invariance property. We can then expect the wrong algorithms to have fairly large losses for a given meta sample $\mathbf{S}$, so that $\Delta(\mathbf{S})$ will have order ≈ 1.

5.2 Comparison to the Bias Learning Model

The approach taken in Baxter (2000) can be partially reformulated in our framework. We will consider only ERM-algorithms in $\mathcal{A}(C,Z)$ which have the form

$$A_{\mathcal{H}}(S) = \arg\min_{c \in \mathcal{H}} \frac{1}{m} \sum_{z_i \in S} l(c,z_i), \tag{24}$$

for some closed set $\mathcal{H} \subseteq C$ (the assumption of closure ensures existence of the minimum). Actually Baxter (2000) allows any algorithm searching the set $\mathcal{H}$, such as regularized algorithms, but the analysis in (Baxter, 2000) does not exploit the advantages of regularisation and we stick to ERM for definiteness and motivation.

The traditional method to give generalization error bounds for such algorithms is described in (Anthony, Bartlett, 1999) or (Vapnik, 1995) and involves the study of the complexity of the function space $\mathcal{F}_{\mathcal{H}} = \left\{z \mapsto l(c,z) : c \in \mathcal{H}\right\}$ in terms of covering numbers or related quantities, and proceeds to prove a uniform bound on the estimation error, such as (13) in Section 3, valid for all $c \in \mathcal{H}$, and with high probability in the sample S. This leads to corresponding generalization error bounds. We have sketched a version of this approach which can be applied both to ordinary and to meta algorithms in Section 3.

The choice of the *hypothesis space* $\mathcal{H}$ completely defines the algorithm (24). A collection of such algorithms can therefore be viewed as a family $\mathbb{H}$ of closed subsets $\mathcal{H} \subseteq C$ which define the algorithms $A_{\mathcal{H}}$ by virtue of formula (24). A corresponding meta-algorithm takes a meta-sample $\mathbf{S}$, sampled from an environment $\mathcal{E}$ as usual, and returns an algorithm $\mathbf{A}(\mathbf{S}) = A_{\mathcal{H}(\mathbf{S})}$ for some hypothesis space $\mathcal{H}(\mathbf{S}) \in \mathbb{H}$. The meta-algorithm can thus be equivalently considered as a map $\mathbf{S} \to \mathcal{H}(\mathbf{S})$ or

$$\mathcal{H} : \bigcup_{n=1}^{\infty} (Z^m)^n \to \mathbb{H}.$$

Such a meta-algorithm effectively *learns the hypothesis space* $\mathcal{H}(\mathbf{S})$, and in (Baxter, 2000) it is called a *bias learner*. For the remainder of this section take $\mathbb{H}$ to be fixed and let $\mathbf{A}$ be any meta-algorithm defined by the ERM formula $\mathbf{A}(\mathbf{S}) = A_{\mathcal{H}(\mathbf{S})}$ for some map $\mathbf{S} \mapsto \mathcal{H}(\mathbf{S}) \in \mathbb{H}$. We also assume the bound M on the loss function to be equal to 1.

In our framework it is natural to study covering numbers for the space of algorithms

$$\mathbf{H}_{\mathbb{H}} = \left\{A_{\mathcal{H}} : \mathcal{H} \in \mathbb{H}\right\}$$

and use them to derive an estimator prediction bound (15) as outlined in Section 3. Imposing a uniform bound on the complexities of the hypothesis spaces in $\mathbb{H}$ then allows the application of Theorem 5. Putting together the estimator prediction bound (15), the uniform bound on the estimation error (13) and Theorem 5, we arrive at

Corollary 7 *Let*

$$\varepsilon_0 = \inf_{\gamma > 0} \left\{ \gamma + 4 \sup_{\mathcal{H} \in \mathbb{H}} \mathcal{N}_1 \left(\frac{\gamma}{8}, \mathcal{F}(\mathcal{H}, l), 2m \right) e^{-\gamma^2 m/32} \right\}$$

and, for $\delta > 0$,

$$\varepsilon_1 = \inf \left\{ t : 4\mathcal{N}_1 \left(\frac{t}{8}, \mathcal{F}(\mathbf{H}, \mathbf{l}), 2n \right) e^{\frac{-t^2 n}{32}} \leq \delta \right\}.$$

Then for any environment $\mathcal{E}$, with probability at least $1-\delta$ in the draw of a meta-sample $\mathbf{S}$ from $(\mathbf{D}_{\mathcal{E}})^n$, we have

$$\mathbf{R}\left(A_{\mathcal{H}(\mathbf{S})}, \mathcal{E}\right) \leq \frac{1}{n} \sum_{S_i \in \mathbf{S}} l_{emp}\left(A_{\mathcal{H}(\mathbf{S})}, S_i\right) + \varepsilon_1 + \varepsilon_0.$$

For convenience of comparison we give implicit bounds on the sample complexities, which are easily derived using $\varepsilon_0 = \varepsilon_1 = \varepsilon/2$ and $\gamma = \varepsilon/4$:

Corollary 8 *For any* $0 < \varepsilon < 1$, $\delta > 0$, *if*

$$n \geq \frac{128}{\varepsilon^2} \ln \left(\frac{4\mathcal{N}_1\left(\frac{\varepsilon}{16}, \mathcal{F}\left(\mathbf{H}_{\mathbb{H}}, l_{emp}\right), 2n\right)}{\delta} \right) \tag{25}$$

and

$$m \geq \frac{512}{\varepsilon^2} \ln \left(\frac{4 \sup_{\mathcal{H} \in \mathbb{H}} \mathcal{N}_1\left(\frac{\varepsilon}{32}, \mathcal{F}\left(\mathcal{H}, l\right), 2m\right)}{\varepsilon} \right), \tag{26}$$

then for any environment $\mathcal{E}$ *, with probability greater than* δ *in the draw of a meta-sample* $\mathbf{S}$ *from* $(\mathbf{D}_{\mathcal{E}})^n$*, we have*

$$\mathbf{R}\left(A_{\mathcal{H}(\mathbf{S})}, \mathcal{E}\right) \leq \frac{1}{n} \sum_{S_i \in \mathbf{S}} l_{emp}\left(A_{\mathcal{H}(\mathbf{S})}, S_i\right) + \varepsilon.$$

J. Baxter (2000) also defines capacities for $\mathbb{H}$, but aims at giving a bound on

$$\sup_{\mathcal{H} \in \mathbb{H}} \left| E_{D \sim \mathcal{E}} \left[\inf_{c \in \mathcal{H}} R(c, D) \right] - \frac{1}{n} \sum_{S_i \in \mathbf{S}} l_{emp}\left(A_{\mathcal{H}}, S_i\right) \right|$$

valid with high probability in $\mathbf{S}$ as drawn from $(\mathbf{D}_{\mathcal{E}})^n$ for any $\mathcal{E}$. A corresponding bound on

$$\mathrm{er}_{\mathcal{E}}\left(\mathcal{H}(\mathbf{S})\right) := E_{D \sim \mathcal{E}} \left[\inf_{c \in \mathcal{H}(\mathbf{S})} R(c, D) \right] \tag{27}$$

(which in Baxter, 2000, is called the *generalization error of the bias learner*), results. This is Theorem 2 in (Baxter, 2000). The expression (27) is the expected risk of the optimal hypothesis in $\mathcal{H}(\mathbf{S})$ as D is drawn from the environment.

The inequality

$$\begin{aligned} \mathrm{er}_{\mathcal{E}}\left(\mathcal{H}(\mathbf{S})\right) &= E_{D \sim \mathcal{E}} \left[E_{S \sim D^m} \left[\inf_{c \in \mathcal{H}(\mathbf{S})} R(c, D) \right] \right] \\ &\leq E_{D \sim \mathcal{E}} \left[E_{S \sim D^m} \left[R\left(A_{\mathcal{H}(\mathbf{S})}(S), D \right) \right] \right] \\ &= \mathbf{R}\left(A_{\mathcal{H}(\mathbf{S})}, \mathcal{E}\right) \end{aligned} \tag{28}$$

shows that our bounds on the transfer risk also provide bounds on (27). Note however that a bound on (27) does not itself guarantee generalization, because we may not find the optimal hypothesis from a finite future sample. This is similar to the estimator prediction bounds in our approach and contrary to our bounds on the transfer risk.

In Theorem 3 of Baxter (2000) the capacity of a given $\mathcal{H}$ is used to formulate a uniform bound on the estimation error of the hypotheses in $\mathcal{H}$ similar to (13). If corresponding capacity bounds held for *all* hypothesis spaces $\mathcal{H} \in \mathbb{H}$, a bound on the transfer risk $\mathbf{R}\left(A_{\mathcal{H}(\mathbf{S})}, \mathcal{E}\right)$ would result from the bound on (27) in a way parallel to our approach (in Baxter, 2000 a bound on the transfer risk comparable to our bounds is never stated). In this case the results become comparable and the bounds on the sample complexities look similar. This is not surprising since both derivations of bounds are rooted in the same classical method (see e.g. Vapnik, 1995).

The sample complexity bounds on the *m*-sample depending on the uniform capacity bound are then essentially the same in Baxter (2000) as in (26) (if we disregard that Baxter, 2000, imposes additional conditions on *m* in Theorem 2). For a comparison we therefore focus on the sample complexity bounds on the size *n* of the meta-sample. In Baxter (2000) Theorem 2, to get

$$\mathrm{er}_{\mathcal{E}}\left(\mathcal{H}\left(\mathbf{S}\right)\right) \leq \frac{1}{n}\sum_{S_i \in \mathbf{S}} l_{emp}\left(A_{\mathcal{H}(\mathbf{S})}, S_i\right) + \varepsilon$$

with probability at least $1-\delta$ in $\mathbf{S}$, it is required that

$$n \geq \frac{256}{\varepsilon^2} \ln \frac{8C\left(\frac{\varepsilon}{32}, \mathbb{H}^*\right)}{\delta}, \tag{29}$$

and there is an additional condition on *m*.

To compare (29) with our bound (25), we disregard the constants (which are better in (25)) and concentrate on a comparison of the complexity measures $C\left(\varepsilon, \mathbb{H}^*\right)$ and $\mathcal{N}_1\left(\varepsilon, \mathcal{F}\left(\mathbf{H}_{\mathbb{H}}, l_{emp}\right), n\right)$.

In (Baxter, 2000) the capacity $C\left(\varepsilon, \mathbb{H}^*\right)$ is defined as follows: For $\mathcal{H} \in \mathbb{H}$ define a real function $\mathcal{H}^*$ on $M_1(Z)$ by

$$\mathcal{H}^*(D) = \inf_{c \in \mathcal{H}} R(c, D).$$

In (Baxter, 2000) there are assumptions to guarantee that $\mathcal{H}^*$ is measurable on $M_1(Z)$, and since it is obviously bounded we have $\mathcal{H}^* \in L_1\left(M_1(Z), \mathbf{Q}\right)$ for any probability measure $\mathbf{Q} \in M_1\left(M_1(Z)\right)$. Use $d_{\mathbf{Q}}$ to denote the metric in $L_1\left(M_1(Z), \mathbf{Q}\right)$ and denote

$$\mathbb{H}^* = \left\{\mathcal{H}^* : \mathcal{H} \in \mathbb{H}\right\}.$$

Then

$$C\left(\varepsilon, \mathbb{H}^*\right) = \sup_{\mathbf{Q} \in M_1(M_1(Z))} \mathcal{N}\left(\varepsilon, \mathbb{H}^*, d_{\mathbf{Q}}\right).$$

It turns out that our complexity measures are bounded by those in Baxter (2000).

Proposition 9 *For all* ε, *n*

$$\mathcal{N}_1\left(\varepsilon, \mathcal{F}\left(\mathbf{H}_{\mathbb{H}}, l_{emp}\right), n\right) \leq C\left(\varepsilon, \mathbb{H}^*\right).$$

Proof For a sample $S = (z_1, ..., z_m) \in Z^m$ use D_S to denote the empirical distribution $D_S \in M_1(Z)$ induced by S:

$$D_S = \frac{1}{m}\sum_{i=1}^{m} \delta_{z_i},$$

where δ_z is the unit mass concentrated at $z \in Z$. Note that for $\mathcal{H} \in \mathbb{H}$ we have

$$\mathcal{H}^*(D_S) = \inf_{c \in \mathcal{H}} \frac{1}{m}\sum_{i=1}^{m} l(c, z_i) = l_{emp}\left(A_{\mathcal{H}}, S\right).$$

For a meta-sample $\mathbf{S} = (S_1, ..., S_n) \in (Z^m)^n$ use $\mathbf{Q_S}$ to denote the empirical distribution $\mathbf{Q_S} \in M_1\left(M_1(Z)\right)$ induced by $\mathbf{S}$:

$$\mathbf{Q_S} = \frac{1}{n}\sum_{i=1}^{n} \delta_{D_{S_i}},$$

where δ_D is the unit mass concentrated at $D \in M_1(Z)$.

Now take any meta-sample $\mathbf{S} = (S_1, ..., S_n) \in (Z^m)^n$ and let $N = \mathcal{N}(\varepsilon, \mathbb{H}^*, d_{\mathbf{Q_S}})$. Then there is a set of functions $\{\Psi_1, ..., \Psi_N\} \subseteq L_1(M_1(Z))$ such that for every $\mathcal{H} \in \mathbb{H}$ there is some i such that

$$\begin{aligned} \varepsilon &\geq d_{\mathbf{Q_S}}(\mathcal{H}^*, \Psi_i) \\ &= \frac{1}{n}\sum_{j=1}^n \left|\mathcal{H}^*\left(D_{S_j}\right) - \Psi_i\left(D_{S_j}\right)\right| \\ &= \frac{1}{n}\sum_{j=1}^n \left|l_{emp}(A_{\mathcal{H}}, S_j) - \Psi_i\left(D_{S_j}\right)\right|. \end{aligned} \tag{30}$$

On the other hand we have

$$\mathcal{F}(\mathbf{H}_{\mathbb{H}}, l_{emp})|_{\mathbf{S}} = \left\{(l_{emp}(A_{\mathcal{H}}, S_1), ..., l_{emp}(A_{\mathcal{H}}, S_n)) : \mathcal{H} \in \mathbb{H}\right\},$$

so, setting $x_i \in \mathbb{R}^n$ with $(x_i)_j = \Psi_i\left(D_{S_j}\right)$, we see from (30) that every member of $\mathcal{F}(\mathbf{H}_{\mathbb{H}}, l_{emp})|_{\mathbf{S}}$ is within d_1-distance ε of some x_i. It follows that

$$\mathcal{N}(\varepsilon, \mathcal{F}(\mathbf{H}_{\mathbb{H}}, l_{emp})|_{\mathbf{S}}, d_1) \leq \mathcal{N}(\varepsilon, \mathbb{H}^*, d_{\mathbf{Q_S}}),$$

whence

$$\begin{aligned} \mathcal{N}_1(\varepsilon, \mathcal{F}(\mathbf{H}_{\mathbb{H}}, l_{emp}), n) &= \sup_{\mathbf{S} \in (Z^m)^n} \mathcal{N}(\varepsilon, \mathcal{F}(\mathbf{H}_{\mathbb{H}}, l_{emp})|_{\mathbf{S}}, d_1) \\ &\leq \sup_{\mathbf{S} \in (Z^m)^n} \mathcal{N}(\varepsilon, \mathbb{H}^*, d_{\mathbf{Q_S}}) \\ &\leq \sup_{\mathbf{Q} \in M_1(M_1(Z))} \mathcal{N}(\varepsilon, \mathbb{H}^*, d_{\mathbf{Q}}) \\ &= \mathcal{C}(\varepsilon, \mathbb{H}^*) \end{aligned}$$

■

We can conclude that our bounds are normally applicable when those in (Baxter, 2000) are. It may however happen, that our covering numbers increase polynomially in n, in which case we still get tight bounds, but the capacities in (Baxter, 2000) are infinite.

6. A Meta-Algorithm for Regression

In this section we present a meta-learning algorithm for function estimation. The algorithm is based on *regularized least-squares regression*, or *ridge regression* (as in Bousquet, Elisseeff, 2002, or Christianini, Shawe-Taylor, 2000) and preliminary experiments appear promising.

To implicitly also define a 'kernelized' version of the algorithm, we describe it in a setting where the *input space* is a subset $\mathcal{X}$ of the unit ball $\{\|x\| \leq 1\}$ in a separable, possibly infinite dimensional Hilbert space H, with an appropriately defined inner product.

The *output space* $\mathcal{Y}$ is the interval $[0,1]$, the data space Z is given by $Z = \mathcal{X} \times \mathcal{Y} \subseteq \{\|x\| \leq 1\} \times [0,1]$ and a learning task is given by a distribution $D \in M_1(\mathcal{X} \times \mathcal{Y})$. Then $D(x,y)$ is interpreted as the probability of finding the input value x associated with the output value y in the context of the task D.

As a hypothesis or concept space we consider the bounded linear functionals h on H which can be identified with members $h \in H$ via the action of the inner product $h(x) = \langle h, x \rangle$ in H.

As a loss function we use $l : H \times Z \to \mathbb{R}_+$ given by

$$l(h,(x,y)) = (\langle h, x \rangle - y)^2.$$

This loss function is unbounded contrary to what is generally required in this paper. It will however turn out that the effective hypothesis space searched by the algorithms in this section is the ball $\left\{ \|h\| \leq \lambda^{-1/2} \right\}$ where λ is the regularization parameter introduced below.

6.1 Regularized least squares Regression

A standard algorithm $A \in A(H,Z)$ for this type of problem is defined as follows: Let $S = (z_1, ..., z_m) = ((x_1, y_1), ..., (x_m, y_m)) \in Z^m$ be a sample. We write, for $h \in H$,

$$L(h) = \frac{1}{m} \sum_{i=1}^{m} (\langle h, x_i \rangle - y_i)^2 + \lambda \|h\|^2$$

and define

$$A(S) = \arg\min_{h \in H} L(h). \tag{31}$$

Note that $\lambda \|A(S)\|^2 \leq L(A(S)) \leq L(0) \leq 1$ so $\|A(S)\| \leq \lambda^{-1/2}$. The effective hypothesis space is then $\left\{ \|h\| \leq \lambda^{-1/2} \right\}$, as claimed above. Thus $|\langle h, x \rangle| \leq \lambda^{-1/2}$ and the loss function is bounded by λ^{-1}.

Any component of h perpendicular to all the x_i will only increase L, so we may assume that $A(S)$ is in the subspace generated by $\{x_1, ..., x_m\}$, in other words

$$A(S) = \sum_{i=1}^{m} \alpha_i x_i \tag{32}$$

for some (possibly non-unique) vector $\alpha \in \mathbb{R}^m$. To find α we substitute (32) in L and equate the gradient to zero. The result of this well known computation is the formula

$$(G + m\lambda I)\alpha = y \tag{33}$$

where $G_{ij} = \langle x_i, x_j \rangle$ is the *Gramian matrix*, here considered as an operator on $\mathbb{R}^m$, $I = \delta_{ij}$ is the identity, and $y = (y_1, ..., y_m)$ the set of target values in the sample, here considered as a vector $y \in \mathbb{R}^m$. Equation (33) can be efficiently solved for α using the Cholesky decomposition method. The formula for the empirical loss of $A(S)$ is, using (32) and (33)

$$\begin{aligned} l_{emp}(A,S) &= \frac{1}{m} \sum_{i=1}^{m} \left((G\alpha)_i - y_i \right)^2 \\ &= \frac{1}{m} \sum_{i=1}^{m} \left(((G + m\lambda I)\alpha)_i - y_i - m\lambda\alpha_i \right)^2 \\ &= \frac{1}{m} \sum_{i=1}^{m} (-m\lambda\alpha_i)^2 \\ &= m\lambda^2 \sum_{i=1}^{m} \alpha_i^2. \end{aligned} \tag{34}$$

It follows from example 3 in (Bousquet, Elisseeff, 2002) that the algorithm A so defined is β-stable with $\beta = 2/(\lambda m)$.

6.2 A Meta-Algorithm

Consider now a meta sample $\mathbf{S} = (S_1, ..., S_n)$, drawn from $(\mathbf{D}_{\mathcal{E}})^n$ for some environment $\mathcal{E}$, and suppose that we have used some 'primer' algorithm A_0 (for example the regression algorithm above for an appropriate value of $\lambda = \lambda_0$) to train corresponding regression functions $h_k = A_0(S_k) \in H$. The sequence of vectors $(A_0(S_1), ..., A_0(S_n)) = (h_1, ..., h_n)$ in some way contains our experiences with the environment $\mathcal{E}$. The idea of the meta-algorithm is now to use the h_k as *additional features* to describe a given new data-point x. We do this by combining the n-dimensional vector $(h_1(x), ..., h_n(x))$ with the existing description $x \in H$.

The intuitive motivation is that we expect the h_i to already describe relevant properties (symmetries, elimination of irrelevant features) of the environment, that we rely on, in particular if the sample-sizes are rather small. Imagine the classification (by thresholding of a regression functions) of character-images of a new character set, say the greek characters, after having learnt other character sets (roman, gothic etc). We could attempt to describe the image of the character α by saying that 'it looks *a little bit like* an x and *a lot like* an $\mathbf{a}$, but rather *unlike* an l'. On the basis of this description a person might recognize the character α, without any previous *visual* training data for α.

The terms *a little bit like*, *a lot like* and *unlike* are quantifications given by previously learnt regression functions for x, $\mathbf{a}$ and l, which may already have a certain robustness relative to deformations, changes in scaling or variations in line thickness. If the sample-size m is large we can derive such robustness more directly and reliably from the training data for α itself, but for a very small sample-size we expect the new features to be helpful. The whole idea is strongly related to the *Chorus of Prototypes* introduced by Edelman (1995), so we will call our algorithm *CP-Regression*.

To formally define the algorithm, consider a 'primer' algorithm $A_0 \in \mathcal{A}(H,Z)$ such that $\|A_0(S)\| \leq \kappa$ for all $S \in Z^m$. For example we could take for A_0 the regularized least squares regression, as defined above, with a regularization parameter λ_0, in which case we would have $\kappa = \lambda_0^{-1/2}$. Fix a mixture parameter $\mu \in [0,1]$ which will be used to interpolate between the old and the new features and a regularization parameter $\lambda > 0$.

Now let the meta-sample $\mathbf{S} = (S_1, ..., S_n)$ be given. We have to define an algorithm $\mathbf{A}(\mathbf{S}) \in \mathcal{A}(H,Z)$. On the vectorspace H we define a new inner product $\langle ., . \rangle_{\mathbf{S}}$ by

$$\langle x_1, x_2 \rangle_{\mathbf{S}} = (1-\mu)\langle x_1, x_2 \rangle + \frac{\mu}{\kappa^2 n} \sum_{k=1}^{n} \langle A_0(S_k), x_1 \rangle \langle A_0(S_k), x_2 \rangle, \tag{35}$$

which is positive definite for $0 \leq \mu < 1$ (in the case $\mu = 1$ we can use a quotient construction to replace H, which then becomes n'-dimensional with $n' \leq n$). We will use $\|.\|_{\mathbf{S}}$ to denote the norm corresponding to $\langle ., . \rangle_{\mathbf{S}}$.

Let $S \in Z^n$ be any sample, $S = (z_1, ..., z_m) = ((x_1, y_1), ..., (x_m, y_m))$ with $x_i \in H$, $\|x_i\| \leq 1$, $y_i \in [0,1]$. We define

$$\mathbf{A}(\mathbf{S})(S) = \arg\min_{h \in H} \frac{1}{m} \sum_{i=1}^{m} \left(\langle h, x_i \rangle_{\mathbf{S}} - y_i\right)^2 + \lambda \|h\|_{\mathbf{S}}^2$$

and the corresponding regression function

$$\mathbf{A}(\mathbf{S})(S)(x) = \langle \mathbf{A}(\mathbf{S})(S), x \rangle_{\mathbf{S}}.$$

Note that

$$\begin{aligned} \|x\|_{\mathbf{S}}^2 &= (1-\mu)\|x\|^2 + \frac{\mu}{\kappa^2 n}\sum_{k=1}^{n}\langle A_0(S_k), x\rangle^2 \\ &\leq (1-\mu)\|x\|^2 + \frac{\mu}{\kappa^2 n}\sum_{i=1}^{n}\kappa^2\|x\|^2 = \|x\|^2, \end{aligned}$$

so $\mathcal{X} \subseteq \{\|x\|_{\mathbf{S}} \leq 1\}$. Therefore $\mathbf{A}(\mathbf{S})$ is ordinary regularized least squares regression with the modified inner product $\langle .,.\rangle_{\mathbf{S}}$. It follows from the analysis in (Bousquet, Elisseeff, 2002) that the algorithms $\mathbf{A}(\mathbf{S})$ are uniformly β-stable with $\beta = 2/(m\lambda)$, for every meta-sample $\mathbf{S}$, with respect to the square loss function we use.

The implementation of $\mathbf{A}$ is straightforward: Given $\mathbf{S} = (S_1, ..., S_n)$ one computes the vectors $h_k = A_0(S_k)$. Now for any new m-sample S the Gramian

$$(G_{\mathbf{S}})_{ij} = \langle x_i, x_j\rangle_{\mathbf{S}} = (1-\mu)\langle x_i, x_j\rangle + \frac{\mu}{\kappa^2 n}\sum_{k=1}^{n}\langle h_k, x_i\rangle\langle h_k, x_2\rangle$$

is determined, and the equation $(G_{\mathbf{S}} + m\lambda I)\alpha = y$ is solved for α using Cholesky decomposition. We then get the regression function

$$\begin{aligned} x &\mapsto \sum_{i=1}^{m}\alpha_i\langle x_i, x\rangle_{\mathbf{S}} = \\ &= (1-\mu)\sum_{i=1}^{m}\alpha_i\langle x_i, x\rangle + \mu\sum_{k=1}^{n}\gamma_k\langle h_k, x\rangle \end{aligned}$$

with

$$\gamma_k = \frac{1}{\kappa^2 n}\sum_{i=1}^{m}\alpha_i\langle h_k, x_i\rangle.$$

In a nonlinear case, when the inner product in H is defined by a complicated kernel, this regression function may be cumbersome to compute since all the computations of $\langle h_k, x\rangle$ will each again involve m computations of the kernel. Also the entire meta-sample $\mathbf{S}$ has then to be present in memory. In a linear case, when the vectorspace operations in H can be performed explicitly, the computational burden is significantly reduced to the computation of a single inner product $\langle h, x\rangle$ of x with the vector

$$h = (1-\mu)\sum_{i=1}^{m}\alpha_i x_i + \mu\sum_{k=1}^{n}\gamma_k h_k$$

which is determined once during training.

6.3 Analysis of CP-Regression

As already noted the algorithms $\mathbf{A}(\mathbf{S})$ are uniformly β-stable with $\beta = 2/(m\lambda)$, for every meta-sample $\mathbf{S}$, with respect to square loss. This gives condition 2 for the application of Theorem 1.

The first condition, essential for the estimator prediction bound, is satisfied by virtue of the following proposition which is proven in the next subsection:

Corollary 10 *The algorithm* $\mathbf{A}$ *is uniformly* β'*-stable w.r.t.* l_{emp} *in the sense that, if* $\mathbf{S} = (S_1, ., S_k, .., S_n)$ *is a meta sample and* $\mathbf{S}' = (S_1, ., S_{k-1}, S_{k+1} .., S_n)$ *is the same as* $\mathbf{S}$*, with only some* S_k *deleted, then*

$$\left| l_{emp}\left(\mathbf{A}\left(\mathbf{S}\right), S\right) - l_{emp}\left(\mathbf{A}\left(\mathbf{S}'\right), S\right)\right| \leq \beta'$$

for every sample $S \in Z^m$*, with*

$$\beta' = \frac{4\mu}{\lambda\left(n-1\right)}.$$

Substitution in Theorem 1 gives, for every environment $\mathcal{E}$ with probability at least $1-\delta$ in a meta-sample $\mathbf{S}$ drawn from $\left(\mathbf{D}_{\mathcal{E}}\right)^n$,

$$\begin{aligned}\mathbf{R}\left(\mathbf{A}\left(\mathbf{S}\right), \mathcal{E}\right) \leq \frac{1}{n}\sum_{S_i \in \mathbf{S}} l_{emp}\left(\mathbf{A}\left(\mathbf{S}\right), S_i\right) + \\ + \frac{8\mu}{\lambda\left(n-1\right)} + \left(\frac{16\mu n}{\lambda\left(n-1\right)} + \frac{1}{\lambda}\right)\sqrt{\frac{\ln\left(1/\delta\right)}{2n}} + \frac{4}{m\lambda}.\end{aligned} \tag{36}$$

The bound gives a performance guarantee of the algorithm applied to future tasks on the basis of the empirical term

$$\left(l_{emp}\right)_{emp}\left(\mathbf{A}, \mathbf{S}\right) = \frac{1}{n}\sum_{S_i \in \mathbf{S}} l_{emp}\left(\mathbf{A}\left(\mathbf{S}\right), S_i\right). \tag{37}$$

If $\mu = 0$, corresponding to no meta-learning at all, the bound (36) becomes more attractive to look at, but we expect the empirical term to be larger. For small n it is better to take small μ, while for very large values of n the value of μ which results in the smallest empirical term is best. It is tempting to minimize the bound with respect to μ. Unfortunately (36) applies only if the parameters of $\mathbf{A}$ have been fixed in advance, it does not justify the selection of the parameters λ, μ or the choice of the primer algorithm A_0 which enters the bound only indirectly through the term (37)). Although this problem can be partially eliminated (see the method of sieves as used in Anthony, 1999), it remains a major weakness of our algorithm. A more principled approach would involve the direct minimization of

$$\frac{1}{n}\sum_{S_i \in \mathbf{S}} l_{emp}\left(A, S_i\right) + N\left(A\right)$$

where $N\left(A\right)$ would be some meta-regularizer. Our algorithm attempts to decrease the quantity (37) only indirectly by the passage to (presumably) more reliable features.

6.4 Stability of CP-Regression

In this subsection we prove Proposition 10. For a bounded operator T on a real Hilbert space H we use $\|T\|_\infty$ to denote its operator norm

$$\|T\|_\infty = \sup_{\|x\| \leq 1}\|Tx\| = \sup_{\|x\|, \|y\| \leq 1}\left|\langle Tx, y\rangle\right|$$

and use T^t, $Ker\left(T\right)$ and $Ran\left(T\right)$ to denote its transpose, nullspace and range respectively. A symmetric operator satisfies $\langle Tx, y\rangle = \langle x, Ty\rangle$ for all x and y (i.e. $T = T^t$), and a positive operator is a symmetric operator also satisfying $\langle Tx, x\rangle \geq 0$ for all x.

Lemma 11 *Let G_1 and G_2 be positive operators and $\lambda > 0$. Then*

1. $G_i + \lambda I$ is invertible,

2. $\left\| (G_i + \lambda I)^{-1} \right\|_\infty \leq 1/\lambda$ and

3. we have

$$\left\| (G_1 + \lambda I)^{-1} - (G_2 + \lambda I)^{-1} \right\|_\infty \leq \frac{1}{\lambda^2} \|G_1 - G_2\|_\infty .$$

4. Let x_1 and x_2 satisfy $(G_i + \lambda I) x_i = y$. Then

$$\left| \|x_1\|^2 - \|x_2\|^2 \right| \leq 2\lambda^{-3} \|G_1 - G_2\|_\infty \|y\|^2 .$$

Proof 1. If $(G_i + \lambda I) x = 0$ then $-\lambda \|x\| = \langle G_i x, x \rangle \geq 0$ so $x = 0$. Thus $G_i + \lambda I$ is 1-1, and since $Ran\,(G_i + \lambda I) = Ran\left((G_i + \lambda I)^t\right) = Ker\,(G_i + \lambda I)^\perp = \{0\}^\perp$ it is also onto.

2. Suppose $(G_i + \lambda I) x = y$. Then

$$\begin{aligned} \lambda^2 \|x\|^2 &= \|y - G_i x\|^2 = \|y\|^2 - 2 \langle G_i x, y \rangle + \|G_i x\|^2 \\ &= \|y\|^2 - 2 \langle G_i x, G_i x + \lambda x \rangle + \|G_i x\|^2 \\ &= \|y\|^2 - \|G_i x\|^2 - 2\lambda \langle x, G_i x \rangle \leq \|y\|^2 , \end{aligned}$$

which proves the second conclusion.

3. We have

$$\begin{aligned} &\left((G_1 + \lambda I)^{-1} - (G_2 + \lambda I)^{-1} \right) (G_2 + \lambda I) \\ &= (G_1 + \lambda I)^{-1} (G_1 + \lambda I + G_2 - G_1) - (G_2 + \lambda I)^{-1} (G_2 + \lambda I) \\ &= (G_1 + \lambda I)^{-1} (G_2 - G_1) , \end{aligned}$$

so, using the second conclusion,

$$\begin{aligned} &\left\| (G_1 + \lambda I)^{-1} - (G_2 + \lambda I)^{-1} \right\|_\infty \\ &= \left\| (G_1 + \lambda I)^{-1} (G_2 - G_1) (G_2 + \lambda I)^{-1} \right\|_\infty \\ &\leq \left\| (G_1 + \lambda I)^{-1} \right\|_\infty \|G_2 - G_1\|_\infty \left\| (G_2 + \lambda I)^{-1} \right\|_\infty \\ &\leq \lambda^{-2} \|G_1 - G_2\|_\infty . \end{aligned}$$

Finally, using the first three conclusions, if $x_i = (G_i + \lambda I)^{-1} y$, then

$$\begin{aligned} \left| \|x_1\|^2 - \|x_2\|^2 \right| &= |\langle x_1 + x_2, x_1 - x_2 \rangle| \\ &\leq (\|x_1\| + \|x_2\|) \|x_1 - x_2\| \\ &\leq \left(2\lambda^{-1} \|y\|\right) \left(\lambda^{-2} \|G_1 - G_2\|_\infty \|y\|\right) . \end{aligned}$$

■

Proof of Proposition 10. Suppose $\mathbf{S} = (S_1, ., S_{k_0}, .., S_n)$ is a meta sample and that $\mathbf{S}' = (S_1, ., S_{k_0-1}, S_{k_0+1} .., S_n)$ is the same as $\mathbf{S}$, with only some S_{k_0} deleted. We have to show that

$$\left| l_{emp}\left(\mathbf{A}(\mathbf{S}), S\right) - l_{emp}\left(\mathbf{A}\left(\mathbf{S}'\right), S\right) \right| \leq \frac{4\mu}{\lambda(n-1)}$$

for every sample $S = (z_1, ..., z_m) = ((x_1, y_1), ..., (x_m, y_m)) \in Z^m$.

Let G and G' be the gramian matrices arising from the vectors x_i and the inner products $\langle ., . \rangle_{\mathbf{S}}$ and $\langle ., . \rangle_{\mathbf{S}'}$ respectively, that is

$$G_{ij} = \left\langle x_i, x_j \right\rangle_{\mathbf{S}} \text{ and } G'_{ij} = \left\langle x_i, x_j \right\rangle_{\mathbf{S}'}.$$

We regard G and G' as operators on $\mathbb{R}^m$ and use $\|.\|_m$ and $\langle ., . \rangle_m$ for the canonical norm and inner product in $\mathbb{R}^m$ respectively.

We have, using (35) and denoting $h_k = A_0(S_k)$,

$$G_{ij} - G'_{ij} = \frac{-\mu}{\kappa^2 n(n-1)} \sum_{k \neq k_0} \langle h_k, x_i \rangle \left\langle h_k, x_j \right\rangle + \frac{\mu}{\kappa^2 n} \langle h_{k_0}, x_i \rangle \left\langle h_{k_0}, x_j \right\rangle$$

so, if η and γ are any two unit vectors in $\mathbb{R}^m$, we have, with $v = \sum_{i=1}^m \eta_i x_i$ and $w = \sum_{i=1}^m \gamma_j x_j$,

$$\begin{aligned} \left| \left\langle \left(G - G'\right) \eta, \gamma \right\rangle_m \right| &= \frac{-\mu}{\kappa^2 n(n-1)} \sum_{k \neq k_0} \langle h_k, v \rangle \langle h_k, w \rangle + \frac{\mu}{\kappa^2 n} \langle h_{k_0}, v \rangle \langle h_{k_0}, w \rangle \\ &\leq \frac{\mu}{\kappa^2 n(n-1)} \sum_{k \neq k_0} \|h_k\|^2 \|v\| \|w\| + \frac{\mu}{\kappa^2 n} \|h_{k_0}\|^2 \|v\| \|w\| \\ &\leq \frac{2\mu}{n-1} \|v\| \|w\| \end{aligned}$$

Now using the triangle and Cauchy Schwarz inequalities

$$\|v\| = \left\| \sum_{i=1}^m \eta_i x_i \right\| \leq \sum_{i=1}^m |\eta_i| \, \|x_i\| \leq \|\eta\|_m \left(\sum_{i=1}^m \|x_i\|^2 \right)^{1/2} \leq m^{1/2}$$

and similarly

$$\|w\| \leq m^{1/2},$$

so that $|\langle (G - G') \eta, \gamma \rangle_m| \leq (2\mu m) / (n-1)$. Since η and γ were arbitrary unit vectors we have

$$\left\| G - G' \right\|_\infty \leq \frac{2\mu m}{n-1}. \tag{38}$$

Now if α and α' are vectors in $\mathbb{R}^m$ which are solutions of $(G - m\lambda I)\alpha = y$ and $(G' - m\lambda I)\alpha' = y$ respectively, and $y \in \mathbb{R}^m$ is a vector with $|y_i| \leq 1$, then, using the last conclusion of Lemma 11 together with (38),

$$\begin{aligned} \left| \|\alpha\|_m^2 - \left\|\alpha'\right\|_m^2 \right| &\leq 2(m\lambda)^{-3} \left\| G - G' \right\|_\infty \|y\|_m^2 \\ &\leq 4m^{-2} \lambda^{-3} \mu \|y\|_m^2 / (n-1) \\ &\leq 4m^{-1} \lambda^{-3} \mu / (n-1). \end{aligned}$$

Using the formula (34) for the empirical error in regularised least squares regression then gives

$$\begin{aligned}\left|l_{emp}\left(\mathbf{A}\left(\mathbf{S}\right),S\right)-l_{emp}\left(\mathbf{A}\left(\mathbf{S}'\right),S\right)\right| &= m\lambda^2\left|\|\alpha\|_m^2-\left\|\alpha'\right\|_m^2\right| \\ &\leq \frac{4\mu}{\lambda\left(n-1\right)}.\end{aligned}$$

■

7. Conclusion

We have employed established analytical tools of statistical learning theory to analyze transfer learning. The notion of uniform algorithmic stability has proven to be particularly useful. Many interesting problems remain, of which we mention only two:

1. The unnatural requirement, that all sample-sizes be equal to the meta-learner, should be eliminated.
2. CP-Regression could be implemented and systematically tested with a nonlinear kernel.

References

M. Anthony, P. Bartlett, *Learning in Neural Networks: Theoretical Foundations*, Cambridge University Press 1999.

J. Baxter, Theoretical Models of Learning to Learn, in *Learning to Learn*, S. Thrun, L. Pratt Eds. Springer 1998.

J. Baxter, A Model of Inductive Bias Learning, *Journal of Artificial Intelligence Research* 12: 149-198, 2000.

O. Bousquet, A. Elisseeff, “Stability and Generalization”, *Journal of Machine Learning Research*, 2: 499-526, 2002.

R. Caruana, Multitask Learning, in *Learning to Learn*, S. Thrun, L. Pratt Eds. Springer 1998.

N. Christianini, J. Shawe-Taylor, *Support Vector Machines*, Cambridge University Press 2000.

L. Devroye, L. Györfi, G. Lugosi, *A Probabilistic Theory of Pattern Recognition*. Springer, 1996.

S. Edelman, Representation, similarity and the chorus of prototypes. *Minds and Machines*, 45-68, 1995.

W. Hoeffding, “Probability inequalities for sums of bounded random variables”, *Journal of the American Statistical Association*, 58:13-30, 1963.

S. Kutin, P. Niyogi, Almost-everywhere algorithmic stability and generalization performance, Technical report , Department of Computer Science, University of Chicago, 2002.

D. McAllester, “Some PAC-Bayesian Theorems”, *Proceedings of the Eleventh Annual Conference In Computational Learning Theory*, 230-234, 1998.

C. McDiarmid, “Concentration”, in *Probabilistic Methods of Algorithmic Discrete Mathematics*, p. 195-248. Springer, Berlin, 1998.

A. Robins, Transfer in Congnition, in *Learning to Learn*, S. Thrun, L. Pratt Eds. Springer 1998.

S. Thrun, *Explanation-Based Neural Network Learning*, Kluwer 1996.

S.Thrun, Lifelong Learning Algorithms, in *Learning to Learn*, S.Thrun, L.Pratt Eds. Springer 1998.

V. Vapnik, *The Nature of Statistical Learning Theory*, Springer 1995.

D. H. Wolpert, *The Mathematics of Generalization*, Addison Wesley, 1995.

Journal of Machine Learning Research 6 (2005) 995–1018 Submitted 11/04; Revised 4/05; Published 6/05

Matrix Exponentiated Gradient Updates for On-line Learning and Bregman Projection

Koji Tsuda KOJI.TSUDA@TUEBINGEN.MPG.DE
Max Planck Institute for Biological Cybernetics
Spemannstrasse 38
72076 Tübingen, Germany, and
Computational Biology Research Center
National Institute of Advanced Science and Technology (AIST)
2-42 Aomi, Koto-ku, Tokyo
135-0064, Japan

Gunnar Rätsch GUNNAR.RAETSCH@TUEBINGEN.MPG.DE
Friedrich Miescher Laboratory of the Max Planck Society
Spemannstrasse 35
72076 Tübingen, Germany

Manfred K. Warmuth MANFRED@CSE.UCSC.EDU
Computer Science Department
University of California
Santa Cruz, CA 95064, USA

Editor: Yoram Singer

Abstract

We address the problem of learning a symmetric positive definite matrix. The central issue is to design parameter updates that preserve positive definiteness. Our updates are motivated with the *von Neumann* divergence. Rather than treating the most general case, we focus on two key applications that exemplify our methods: on-line learning with a simple square loss, and finding a symmetric positive definite matrix subject to linear constraints. The updates generalize the exponentiated gradient (EG) update and AdaBoost, respectively: the parameter is now a symmetric positive definite matrix of trace one instead of a probability vector (which in this context is a diagonal positive definite matrix with trace one). The generalized updates use matrix logarithms and exponentials to preserve positive definiteness. Most importantly, we show how the derivation and the analyses of the original EG update and AdaBoost generalize to the non-diagonal case. We apply the resulting *matrix exponentiated gradient* (MEG) update and *DefiniteBoost* to the problem of learning a kernel matrix from distance measurements.

1. Introduction

Most learning algorithms have been developed to learn a *vector* of parameters from data. However, an increasing number of papers are now dealing with more structured parameters. More specifically, when learning a similarity or a distance function among objects, the parameters are defined as a *symmetric positive definite matrix* that serves as a kernel (e.g., Xing et al., 2003; Shai-Shwartz et al., 2004; Tsang and Kwok, 2003; Tsuda and Noble, 2004). Learning is typically formulated as a parameter updating procedure to optimize a *loss function*. The gradient descent update is one of the most commonly used algorithms, but it is not appropriate when the parameters form a positive definite matrix, because the updated parameter matrix does not necessarily stay positive definite. Xing et al. (2003) solved this problem by always correcting the updated matrix to be positive definite. However no bound has been proven for this update-and-correct approach. Also, Shai-Shwartz et al. (2004) proposed an on-line algorithm for learning a kernel matrix when only some of the class labels of the examples are provided. This algorithm is also based on the update-and-correction approach, but since the update step performs rank-one modification, the correction step can be efficiently implemented. They have shown a generalization bound inspired by similar previously known bounds for the perceptron.

In this paper, we introduce the *matrix exponentiated gradient update* which works as follows: First, the matrix logarithm of the current parameter matrix is computed. Then a step is taken in the direction of the steepest descent of the loss function. Finally, the parameter matrix is updated to the exponential of the modified log-matrix. Our update preserves symmetry and positive definiteness because the matrix exponential maps any symmetric matrix to a symmetric positive definite matrix.

Bregman divergences play a central role in the motivation and the analysis of *on-line learning algorithms* (Kivinen and Warmuth, 1997). A learning problem is essentially defined by a loss function and a divergence that measures the discrepancy between parameters. More precisely, the updates are motivated by minimizing the sum of the loss function and the Bregman divergence, where the loss function is multiplied by a positive learning rate. Different divergences lead to radically different updates (Kivinen and Warmuth, 1997, 2001). For example, the gradient descent update is derived from the squared Euclidean distance, and the exponentiated gradient update from the Kullback-Leibler divergence (relative entropy). In this work we use the *von Neumann* divergence (also called quantum relative entropy) for measuring the discrepancy between two positive definite matrices (Nielsen and Chuang, 2000). We derive a new *matrix exponentiated gradient update* from this divergence (which is a Bregman divergence for symmetric positive definite matrices). Finally we prove *relative loss bounds* using the *von Neumann* divergence as a measure of progress.

We apply our techniques to solve the following related key problem that has received a lot of attention recently (Xing et al., 2003; Shai-Shwartz et al., 2004; Tsang and Kwok, 2003; Tsuda and Noble, 2004). Find a symmetric positive definite matrix that satisfies a number of linear inequality constraints. The new *DefiniteBoost* algorithm greedily chooses a violated linear constraint and performs an approximated Bregman projection. In the diagonal case, we recover AdaBoost (Schapire and Singer, 1999). We also show how the convergence proof of AdaBoost generalizes to the non-diagonal case.

2. Preliminaries

In this section, we first present mathematical definitions and basic lemmas.

2.1 Matrix Basics

We denote matrices by capital bold letters and restrict ourselves to square matrices with real entries in this paper. For any such matrix $\boldsymbol{A} \in \mathbb{R}^{d\times d}$, $\mathbf{exp}\,\boldsymbol{A}$ and $\mathbf{log}\,\boldsymbol{A}$ denote the matrix exponential and logarithm, respectively. The matrix exponential is defined as the following power series,

$$\mathbf{exp}(\boldsymbol{A}) := \boldsymbol{I} + \boldsymbol{A} + \frac{1}{2!}\boldsymbol{A}^2 + \frac{1}{3!}\boldsymbol{A}^3 + \cdots . \tag{2.1}$$

In the case of symmetric matrices, the matrix exponential operation can be computed using the eigenvalue decomposition $\boldsymbol{A} = \boldsymbol{V}\Lambda\boldsymbol{V}^\top$, where $\boldsymbol{V}$ is an orthonormal matrix with the eigenvectors of $\boldsymbol{A}$ as columns and Λ the diagonal matrix of eigenvalues. Thus, $\mathbf{exp}\,\boldsymbol{A} = \boldsymbol{V}(\mathbf{exp}\,\Lambda)\boldsymbol{V}^\top$, where $(\mathbf{exp}\,\Lambda)_{i,i} = \exp(\Lambda_{i,i})$. The matrix logarithm $\mathbf{log}\,\boldsymbol{A}$ is defined as the inverse function of $\mathbf{exp}\,\boldsymbol{A}$, which does not always exist for arbitrary $\boldsymbol{A}$. However, when $\boldsymbol{A}$ is symmetric and strictly positive definite, $\mathbf{log}\,\boldsymbol{A}$ is computed as $\mathbf{log}\,\boldsymbol{A} := \boldsymbol{V}(\mathbf{log}\,\Lambda)\boldsymbol{V}^\top$, where $(\mathbf{log}\,\Lambda)_{i,i} = \log \Lambda_{i,i}$. Throughout the paper $\log a$ and $\exp a$ denote the natural logarithm and exponential of scalar "a".

A square matrix is positive definite if all its eigenvalues are strictly positive. Positive semi-definiteness only requires the non-negativity of the eigenvalues. For two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, $\boldsymbol{A} \preceq \boldsymbol{B}$ iff $\boldsymbol{B} - \boldsymbol{A}$ is positive semi-definite. Similarly, $\boldsymbol{A} \prec \boldsymbol{B}$ iff $\boldsymbol{B} - \boldsymbol{A}$ is (strictly) positive definite.

The trace of a matrix is the sum of its diagonal elements, i.e. $\mathrm{tr}(\boldsymbol{A}) = \sum_i A_{i,i}$ and thus $\mathrm{tr}(\boldsymbol{AB}) = \sum_{i,j} A_{i,j}B_{j,i} = \mathrm{tr}(\boldsymbol{BA})$. In matrix algebra, $\mathrm{tr}(\boldsymbol{AB})$ plays a similar role as the dot product for vectors. Furthermore, $\mathrm{tr}(\boldsymbol{A}) = \sum_i \lambda_i$, where λ_i are the eigenvalues of $\boldsymbol{A}$ and the determinant $\det(\boldsymbol{A}) = \prod_i \lambda_i$.

If $\mathrm{F}(\boldsymbol{W}) : \mathbb{R}^{d\times d} \to \mathbb{R}$ is a real-valued function on matrices, then $\nabla_{\boldsymbol{W}}\mathrm{F}(\boldsymbol{W})$ denotes the *gradient* with respect to matrix $\boldsymbol{W}$:

$$\nabla_{\boldsymbol{W}}\mathrm{F}(\boldsymbol{W}) = \begin{pmatrix} \frac{\partial \mathrm{F}}{\partial W_{11}} & \cdots & \frac{\partial \mathrm{F}}{\partial W_{1d}} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathrm{F}}{\partial W_{d1}} & \cdots & \frac{\partial \mathrm{F}}{\partial W_{dd}} \end{pmatrix}.$$

For example, it is easy to see that $\nabla_{\boldsymbol{A}}\mathrm{tr}(\boldsymbol{AB}) = \boldsymbol{B}^\top$. More examples of computing gradients are given in Appendix A.

For a square matrix $\boldsymbol{X}$, $\mathbf{sym}(\boldsymbol{X}) = (\boldsymbol{X} + \boldsymbol{X}^\top)/2$ denotes the symmetric part of $\boldsymbol{X}$. If $\boldsymbol{W}$ is symmetric and $\boldsymbol{X}$ an arbitrary matrix, then

$$\mathrm{tr}(\boldsymbol{WX}) = \mathrm{tr}\left(\boldsymbol{W}\frac{\boldsymbol{X}+\boldsymbol{X}^\top}{2}\right) + \mathrm{tr}\left(\boldsymbol{W}\frac{\boldsymbol{X}-\boldsymbol{X}^\top}{2}\right) = \mathrm{tr}(\boldsymbol{W}\,\mathbf{sym}(\boldsymbol{X})). \tag{2.2}$$

Our analysis requires the use of the Golden-Thompson inequality (Golden, 1965):

$$\mathrm{tr}(\mathbf{exp}(\boldsymbol{A}+\boldsymbol{B})) \leq \mathrm{tr}(\mathbf{exp}(\boldsymbol{A})\,\mathbf{exp}(\boldsymbol{B})), \tag{2.3}$$

which holds for arbitrary *symmetric* matrices $\boldsymbol{A}$ and $\boldsymbol{B}$.

We also need the following two basic inequalities for symmetric matrices. The first one generalizes the following simple inequality, which is a realization of Jensen's inequality for the convex function $\exp(x)$: For any $0 \leq a \leq 1$ and $\rho_1, \rho_2 \in \mathbb{R}$,

$$\exp(a\rho_1 + (1-a)\rho_2) \leq a\exp(\rho_1) + (1-a)\exp(\rho_2).$$

In the below generalization, the distribution $(a, 1-a)$ is replaced by $(\boldsymbol{A}, \boldsymbol{I} - \boldsymbol{A})$, where $\boldsymbol{A}$ is any symmetric matrix for which $\mathbf{0} \preceq \boldsymbol{A} \preceq \boldsymbol{I}$.

Lemma 2.1 *For any symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{d\times d}$ such that $\mathbf{0} \preceq \boldsymbol{A} \preceq \boldsymbol{I}$, and any $\rho_1, \rho_2 \in \mathbb{R}$,*

$$\mathbf{exp}(\boldsymbol{A}\rho_1 + (\boldsymbol{I}-\boldsymbol{A})\rho_2) \preceq \boldsymbol{A}\exp(\rho_1) + (\boldsymbol{I}-\boldsymbol{A})\exp(\rho_2).$$

Proof Assume $\boldsymbol{A}$ is eigen-decomposed as $\boldsymbol{A} = \boldsymbol{V}\Lambda\boldsymbol{V}^\top$, where Λ is the diagonal matrix of eigenvalues and $\boldsymbol{V}$ is an orthogonal matrix with the eigenvectors of $\boldsymbol{A}$ as columns. By assumption, $0 \le \lambda_k \le 1$. Let θ_k be the k-th eigenvalue of the left hand side of the inequality that we are to prove. Clearly $\theta_k = \exp(\lambda_k\rho_1 + (1-\lambda_k)\rho_2)$ and by Jensen's inequality, $\theta_k \le \lambda_k \exp(\rho_1) + (1-\lambda_k)\exp(\rho_2)$. Let Θ be the diagonal matrix with entries θ_k. Then $\Theta \preceq \Lambda\exp(\rho_1) + (\boldsymbol{I} - \Lambda)\exp(\rho_2)$, and by multiplying both sides by $\boldsymbol{V}$ from left and by $\boldsymbol{V}^\top$ from right, we obtain the desired inequality. ■

Lemma 2.2 *For any positive semi-definite symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{d\times d}$ and any two symmetric matrices $\boldsymbol{B}, \boldsymbol{C} \in \mathbb{R}^{d\times d}$, $\boldsymbol{B} \preceq \boldsymbol{C}$ implies* $\mathrm{tr}(\boldsymbol{AB}) \le \mathrm{tr}(\boldsymbol{AC})$.

Proof Let $\boldsymbol{D} = \boldsymbol{C} - \boldsymbol{B}$, then $\boldsymbol{D} \succeq \boldsymbol{0}$ by assumption. Suffices to show that $\mathrm{tr}(\boldsymbol{AD}) \ge 0$. Let us eigen-decompose $\boldsymbol{A}$ as $\boldsymbol{V}\Lambda\boldsymbol{V}^\top$. Since $\boldsymbol{V}\boldsymbol{V}^\top = \boldsymbol{V}^\top\boldsymbol{V} = \boldsymbol{I}$, $\boldsymbol{D} = \boldsymbol{V}P\boldsymbol{V}^\top$ where $P = \boldsymbol{V}^\top\boldsymbol{D}\boldsymbol{V} \succeq \boldsymbol{0}$. Then $\mathrm{tr}(\boldsymbol{AD}) = \mathrm{tr}(\boldsymbol{V}\Lambda\boldsymbol{V}^\top\boldsymbol{V}P\boldsymbol{V}^\top) = \mathrm{tr}(\Lambda P) = \sum_{i=1}^n \lambda_i P_{ii}$. Since P is positive semi-definite, the diagonal elements P_{ii} are nonnegative. Also by assumption the eigenvalues λ_i of $\boldsymbol{A}$ are nonnegative. Thus we conclude that $\mathrm{tr}(\boldsymbol{AD}) \ge 0$. ■

2.2 Von Neumann Divergence or Quantum Relative Entropy

If F is a real-valued strictly convex differentiable function on the parameter domain (a subset of matrices in $\mathbb{R}^{d\times d}$) and $\boldsymbol{f}(\boldsymbol{W}) := \nabla_{\boldsymbol{W}}\mathrm{F}(\boldsymbol{W})$, then the Bregman divergence between two parameters $\widetilde{\boldsymbol{W}}$ and $\boldsymbol{W}$ is defined as

$$\Delta_{\mathrm{F}}(\widetilde{\boldsymbol{W}}, \boldsymbol{W}) := \mathrm{F}(\widetilde{\boldsymbol{W}}) - \mathrm{F}(\boldsymbol{W}) - \mathrm{tr}((\widetilde{\boldsymbol{W}} - \boldsymbol{W})\boldsymbol{f}(\boldsymbol{W})^\top).$$

Since F is strictly convex, $\Delta_{\mathrm{F}}(\widetilde{\boldsymbol{W}}, \boldsymbol{W})$ is also strictly convex in its first argument. Furthermore, the gradient in the first argument has the following simple form:

$$\nabla_{\widetilde{\boldsymbol{W}}}\,\Delta_{\mathrm{F}}(\widetilde{\boldsymbol{W}}, \boldsymbol{W}) = \boldsymbol{f}(\widetilde{\boldsymbol{W}}) - \boldsymbol{f}(\boldsymbol{W}),$$

since $\nabla_{\boldsymbol{A}}\mathrm{tr}(\boldsymbol{AB}) = \boldsymbol{B}^\top$ (cf. Section 2.1).

For the divergences used in this paper, we restrict ourselves to the domain of symmetric positive definite matrices. Our main choice of F is $\mathrm{F}(\boldsymbol{W}) = \mathrm{tr}(\boldsymbol{W}\,\mathbf{log}\,\boldsymbol{W} - \boldsymbol{W})$, which is called *von Neumann entropy* or *quantum entropy*. The strict convexity of this function is well known (Nielsen and Chuang, 2000). Furthermore we show in Appendix A that $\nabla_{\boldsymbol{W}}\mathrm{F}(\boldsymbol{W}) = \boldsymbol{f}(\boldsymbol{W}) = \mathbf{log}\,\boldsymbol{W}$.

The Bregman divergence corresponding to this choice of F is the *von Neumann divergence* or *quantum relative entropy* (e.g., Nielsen and Chuang, 2000):

$$\Delta_{\mathrm{F}}(\widetilde{\boldsymbol{W}}, \boldsymbol{W}) = \mathrm{tr}(\widetilde{\boldsymbol{W}}\,\mathbf{log}\,\widetilde{\boldsymbol{W}} - \widetilde{\boldsymbol{W}}\,\mathbf{log}\,\boldsymbol{W} - \widetilde{\boldsymbol{W}} + \boldsymbol{W}).$$

In this paper, we are primarily interested in the case when the parameters are normalized in the sense that $\mathrm{tr}(\boldsymbol{W}) = \mathrm{tr}(\widetilde{\boldsymbol{W}}) = 1$. Symmetric positive definite matrices of trace one are related to density matrices commonly used in Statistical Physics. For normalized parameters the divergence simplifies to

$$\Delta_{\mathrm{F}}(\widetilde{\boldsymbol{W}}, \boldsymbol{W}) = \mathrm{tr}(\widetilde{\boldsymbol{W}}\,\mathbf{log}\,\widetilde{\boldsymbol{W}} - \widetilde{\boldsymbol{W}}\,\mathbf{log}\,\boldsymbol{W}).$$

If $\boldsymbol{W} = \sum_i \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^\top$ is our notation for the eigenvalue decomposition, then the von Neumann entropy[1] becomes $\mathrm{F}(\boldsymbol{W}) = \sum_i \lambda_i \log \lambda_i$. We can rewrite the normalized divergence[2] as

$$\Delta_{\mathrm{F}}(\widetilde{\boldsymbol{W}}, \boldsymbol{W}) = \sum_i \tilde{\lambda}_i \log \tilde{\lambda}_i - \sum_{i,j} \tilde{\lambda}_i \log \lambda_j (\tilde{\boldsymbol{v}}_i^\top \boldsymbol{v}_j)^2. \tag{2.4}$$

This divergence quantifies the difference in the eigenvalues as well as the eigenvectors. When both eigen systems are the same (i.e., $\tilde{\boldsymbol{v}}_i = \boldsymbol{v}_i$), then the divergence becomes the usual relative entropy between the eigenvalues $\Delta_{\mathrm{F}}(\widetilde{\boldsymbol{W}}, \boldsymbol{W}) = \sum_i \tilde{\lambda}_i \log \frac{\tilde{\lambda}_i}{\lambda_i}$.

2.3 Rotation Invariance

One can visualize a symmetric positive definite matrix $\boldsymbol{W} = \sum_i \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^\top = \boldsymbol{V} \Lambda \boldsymbol{V}^\top$ as an ellipse, where the eigenvectors $\boldsymbol{v}_i$ are the axes of the ellipse and the square-roots of the eigenvalues (i.e. $\sqrt{\lambda_i}$) are the lengths of the corresponding axes. Thus the von Neumann divergence quantifies the "discrepancy" between two ellipses and is invariant under a simultaneous rotation of both eigen systems. That is, for any orthonormal matrix $\boldsymbol{U}$, the von Neumann divergence has the property that

$$\Delta_{\mathrm{F}}(\widetilde{\boldsymbol{W}}, \boldsymbol{W}) = \Delta_{\mathrm{F}}(\boldsymbol{U}\widetilde{\boldsymbol{W}}\boldsymbol{U}^\top, \boldsymbol{U}\boldsymbol{W}\boldsymbol{U}^\top). \tag{2.5}$$

This follows from (2.4) and

$$\Delta_{\mathrm{F}}(\widetilde{\boldsymbol{V}}\widetilde{\Lambda}\widetilde{\boldsymbol{V}}^\top, \boldsymbol{V}\Lambda\boldsymbol{V}^\top) = \Delta_{\mathrm{F}}(\boldsymbol{U}\widetilde{\boldsymbol{V}}\widetilde{\Lambda}(\boldsymbol{U}\widetilde{\boldsymbol{V}})^\top, \boldsymbol{U}\boldsymbol{V}\Lambda(\boldsymbol{U}\boldsymbol{V})^\top).$$

However, the divergence is decidedly not invariant under the unitary rotation of both parameters, i.e. typically $\Delta_{\mathrm{F}}(\widetilde{\boldsymbol{W}}, \boldsymbol{W}) \neq \Delta_{\mathrm{F}}(\boldsymbol{U}\widetilde{\boldsymbol{W}}, \boldsymbol{U}\boldsymbol{W})$ for an orthonormal matrix $\boldsymbol{U}$. This is because such rotations can change the sign of the eigenvalues. Also rotating symmetric matrices typically produces non-symmetric matrices.

There is a second important divergence between symmetric positive definite matrices that is invariant under the simultaneous rotation of both eigen systems (2.5). It is a Bregman divergence based on the strictly convex function $\mathrm{F}(\boldsymbol{W}) = -\log\det(\boldsymbol{W})$ (e.g., Boyd and Vandenberghe (2004)) over the cone of positive definite matrices. Note that $\mathrm{F}(\boldsymbol{W}) = -\sum_i \log \lambda_i$, where the λ_i denote the eigenvalues of $\boldsymbol{W}$. Also since $\boldsymbol{f}(\boldsymbol{W}) = \nabla_{\boldsymbol{W}} F(\boldsymbol{W}) = (\boldsymbol{W}^{-1})^\top = \boldsymbol{W}^{-1}$, the Bregman divergence becomes:

$$\begin{aligned}\Delta_{\mathrm{F}}(\widetilde{\boldsymbol{W}}, \boldsymbol{W}) &= \log \frac{\det(\boldsymbol{W})}{\det(\widetilde{\boldsymbol{W}})} + \mathrm{tr}(\boldsymbol{W}^{-1}\widetilde{\boldsymbol{W}}) - d \\ &= \sum_i \log \frac{\lambda_i}{\tilde{\lambda}_i} + \mathrm{tr}(\boldsymbol{W}^{-1}\widetilde{\boldsymbol{W}}) - d,\end{aligned}$$

where d is the dimension of the parameter matrices. We call this the *LogDet* divergence. Notice that in this case, $\mathrm{F}(\boldsymbol{W})$ is essentially minus the log of the volume of the ellipse $\boldsymbol{W}$, and the LogDet divergence is the relative entropy between two multidimensional Gaussians with fixed mean and covariance matrices $\widetilde{\boldsymbol{W}}$ and $\boldsymbol{W}$, respectively (see Singer and Warmuth, 1999). At the end of Section 3.1 we will also briefly discuss the updates derived from the LogDet divergence. Note that for this divergence $\Delta_{\mathrm{F}}(\widetilde{\boldsymbol{W}}, \boldsymbol{W}) = \Delta_{\mathrm{F}}(\boldsymbol{U}\widetilde{\boldsymbol{W}}, \boldsymbol{U}\boldsymbol{W})$ for any orthonormal matrix $\boldsymbol{U}$ and parameter matrices in the domain of F.

1. $\mathrm{F}(\boldsymbol{W})$ can be extended to symmetric positive semi-definite matrices by using the convention $0 \log 0 = 0$.
2. The domain of the first argument can be extended to symmetric positive semi-definite matrices.

3. On-line Learning

In this section we present a natural extension of the *exponentiated gradient* (EG) update (Kivinen and Warmuth, 1997) to an update for symmetric positive definite matrices.

3.1 Motivation of the Updates

On-line learning proceeds in trials. In the most basic form, the on-line algorithm produces a parameter $\boldsymbol{W}_t$ at trial t and then incurs a loss $L_t(\boldsymbol{W}_t)$. In this paper, the parameters are square matrices in $\mathbb{R}^{d\times d}$.

In a refined form, the algorithm aims to predict a label and several actions occur in each trial: The algorithm first receives an *instance* $\boldsymbol{X}_t$ in some instance domain $\mathcal{X}$. It then produces a prediction $\hat{y}_t$ for the instance $\boldsymbol{X}_t$ based on the algorithm's current parameter matrix $\boldsymbol{W}_t$ and receives a label y_t. (The prediction $\hat{y}_t$ and the label y_t lie some labeling domain $\mathcal{Y}$.) Finally the algorithm incurs a real valued loss $L(\hat{y}_t, y_t)$ and updates its parameter matrix to $\boldsymbol{W}_{t+1}$.

For example in Section 3.3 we consider a case where the labeling domain $\mathcal{Y}$ is the real line. The on-line algorithm we analyze for this case predicts with $\hat{y}_t = \operatorname{tr}(\boldsymbol{W}_t\boldsymbol{X}_t)$ and is based on the loss $L_t(\boldsymbol{W}_t) = L(\hat{y}_t, y_t) = (\hat{y}_t - y_t)^2$.

In this section we only discuss updates at a high level and only consider the basic form of the on-line algorithm. We assume that $L_t(\boldsymbol{W})$ is convex in the parameter $\boldsymbol{W}$ (for all t) and that the gradient $\nabla_{\boldsymbol{W}} L_t(\boldsymbol{W})$ is a well defined matrix in $\mathbb{R}^{d\times d}$. In the update, we aim to solve the following problem (see Kivinen and Warmuth, 1997, 2001):

$$\boldsymbol{W}_{t+1} = \underset{\boldsymbol{W}}{\operatorname{argmin}} \quad \Delta_{\mathrm{F}}(\boldsymbol{W}, \boldsymbol{W}_t) + \eta L_t(\boldsymbol{W}), \tag{3.1}$$

where the convex function F defines the Bregman divergence and η is a non-negative learning rate. The update balances two conflicting goals: staying close to the old parameter $\boldsymbol{W}_t$ (as quantified by the divergence) and achieving small loss on the current labeled instance. The learning rate becomes a trade-off parameter.

We can eliminate the argmin by setting the gradient (with respect to $\boldsymbol{W}$) of its objective to zero:

$$\boldsymbol{W}_{t+1} = \boldsymbol{f}^{-1}\left(\boldsymbol{f}(\boldsymbol{W}_t) - \eta \nabla_{\boldsymbol{W}} L_t(\boldsymbol{W}_{t+1})\right). \tag{3.2}$$

If we assume that $\boldsymbol{f}$ and $\boldsymbol{f}^{-1}$ preserve symmetry, then constraining $\boldsymbol{W}$ in (3.1) to be symmetric changes the update to (cf. Appendix B for details):

$$\boldsymbol{W}_{t+1} = \boldsymbol{f}^{-1}\left(\boldsymbol{f}(\boldsymbol{W}_t) - \eta\, \mathbf{sym}(\nabla_{\boldsymbol{W}} L_t(\boldsymbol{W}_{t+1}))\right). \tag{3.3}$$

The above *implicit* update is usually not solvable in closed form. A common way to avoid this problem (Kivinen and Warmuth, 1997) is to approximate $\nabla_{\boldsymbol{W}} L_t(\boldsymbol{W}_{t+1})$ by $\nabla_{\boldsymbol{W}} L_t(\boldsymbol{W}_t)$, leading to the following *explicit* update for the constraint case:

$$\boldsymbol{W}_{t+1} = \boldsymbol{f}^{-1}\left(\boldsymbol{f}(\boldsymbol{W}_t) - \eta\, \mathbf{sym}(\nabla_{\boldsymbol{W}} L_t(\boldsymbol{W}_t))\right).$$

In the case of the von Neumann divergence, the functions $\boldsymbol{f}(\boldsymbol{W}) = \mathbf{log}\, \boldsymbol{W}$ and $\boldsymbol{f}^{-1}(\boldsymbol{Q}) = \mathbf{exp}\, \boldsymbol{Q}$ clearly preserve symmetry. When using this divergence we arrive at the following (explicit) update:

$$\boldsymbol{W}_{t+1} = \underbrace{\mathbf{exp}\left(\underbrace{\mathbf{log}\overbrace{\boldsymbol{W}_t}^{\text{sym.pos.def.}} - \eta\,\mathbf{sym}(\overbrace{\nabla_{\boldsymbol{W}} L_t(\boldsymbol{W}_t)}^{\text{pos. semi. def.}})}_{\text{symmetric}}\right)}_{\text{symmetric positive definite}}. \tag{3.4}$$

We call this update the *unnormalized matrix exponentiated gradient update*. Note that $\boldsymbol{f}(\boldsymbol{W}) = \mathbf{log}\,\boldsymbol{W}$ maps symmetric positive definite matrices to arbitrary symmetric matrices, and after adding a scaled symmetrized gradient, the function $\boldsymbol{f}^{-1}(\boldsymbol{Q}) = \mathbf{exp}\,\boldsymbol{Q}$ maps the symmetric exponent back to a symmetric positive definite matrix.

When the parameters are constrained to trace one, then we arrive at the *Matrix Exponentiated Gradient (MEG) update*, which generalizes the exponentiated gradient (EG) update of Kivinen and Warmuth (1997) to non-diagonal matrices:

$$\boldsymbol{W}_{t+1} = \frac{1}{Z_t}\mathbf{exp}\left(\mathbf{log}\,\boldsymbol{W}_t - \eta\,\mathbf{sym}(\nabla_{\boldsymbol{W}} L_t(\boldsymbol{W}_t))\right), \tag{3.5}$$

where $Z_t = \mathrm{tr}\left(\mathbf{exp}\left(\mathbf{log}\,\boldsymbol{W}_t - \eta\,\mathbf{sym}(\nabla_{\boldsymbol{W}} L_t(\boldsymbol{W}_t))\right)\right)$ is the normalizing constant (See Appendix B for details.)

Finally, observe that for the LogDet divergence $\boldsymbol{f}(\boldsymbol{W}) = \nabla_{\boldsymbol{W}} F = -\boldsymbol{W}^{-1}$ and $\boldsymbol{f}^{-1}(\boldsymbol{Q}) = -\boldsymbol{Q}^{-1}$. Thus both $\boldsymbol{f}$ and $\boldsymbol{f}^{-1}$ negate and invert all eigenvalues. Both functions also preserve symmetry. However, $\boldsymbol{f}^{-1}$ does not map an arbitrary symmetric matrix back to a symmetric positive definite matrix. Note that for this divergence update (3.3) becomes

$$\boldsymbol{W}_{t+1} = \underbrace{-\left(\underbrace{-(\overbrace{\boldsymbol{W}_t}^{\text{sym.pos.def.}})^{-1} - \eta\,\mathbf{sym}(\overbrace{\nabla_{\boldsymbol{W}} L_t(\boldsymbol{W}_{t+1})}^{\text{pos.semi.def.}})}_{\text{symmetric negative definite}}\right)^{-1}}_{\text{symmetric positive definite}}.$$

This update also preserves symmetric positive definiteness of the parameter matrix under the assumption that the gradient $\nabla_{\boldsymbol{W}} L_t(\boldsymbol{W}_{t+1})$ is positive semi-definite: If $\boldsymbol{W}_t$ is symmetric positive definite, then $\boldsymbol{f}(\boldsymbol{W}_t)$ is symmetric negative definite. Using this assumption, we have that the argument of $\boldsymbol{f}^{-1}$ is symmetric negative definite and therefore $\boldsymbol{W}_{t+1}$ is again symmetric positive definite.

In this paper we prove a certain type of relative loss bound for the MEG update which generalize the analogously known bounds for the EG algorithm to the non-diagonal case. To our knowledge, no relative loss bounds have been proven for the above update that is derived from the LogDet divergence. For this update, such bounds are not even known for the diagonal case. Also, if the gradients of the loss are only known to be symmetric then η must be small in order to guarantee that $\boldsymbol{W}_{t+1}$ stays in the positive definite cone.

3.2 Numerically Stable MEG Update

The MEG update (3.5) is numerically unstable when the eigenvalues of $\boldsymbol{W}_t$ are around zero. However we can "unwrap" this update to the following:

$$\boldsymbol{W}_{t+1} = \frac{1}{\tilde{Z}_t}\mathbf{exp}\left(c_t\boldsymbol{I} + \mathbf{log}\,\boldsymbol{W}_1 - \eta\sum_{s=1}^{t}\mathbf{sym}(\nabla_{\boldsymbol{W}} L_s(\boldsymbol{W}_s))\right),$$

where the constant $\tilde{Z}_t$ normalizes the trace of $\boldsymbol{W}_{t+1}$ to one. As long as the eigenvalues of $\boldsymbol{W}_1$ are not too small, the computation of $\mathbf{log}\,\boldsymbol{W}_1$ is stable. Note that the update is independent of the choice of $c_t \in \mathbb{R}$. We incrementally maintain an eigenvalue decomposition of the matrix in the exponent ($O(n^3)$ per iteration):

$$\boldsymbol{V}_t \Lambda_t \boldsymbol{V}_t^\top = c_t \boldsymbol{I} + \mathbf{log}\,\boldsymbol{W}_1 - \eta \sum_{s=1}^{t} \mathbf{sym}(\nabla_{\boldsymbol{W}} L_s(\boldsymbol{W}_s))$$

where the constant c_t is chosen so that the maximum eigenvalue of the above is zero. Now $\boldsymbol{W}_{t+1} = \boldsymbol{V}_t\,\mathbf{exp}(\Lambda_t)\boldsymbol{V}_t^\top/\mathrm{tr}(\mathbf{exp}(\Lambda_t))$. The pseudo-code is given in Algorithm 1.

Algorithm 1 Pseudo-code of the matrix exponentiated gradient (MEG) algorithm for quadratic Loss

```
Choose $\boldsymbol{W}_1$ and $\eta$
Initialize $\boldsymbol{G}_0 = \mathbf{log}\,\boldsymbol{W}_1$
for $t = 1, 2, \ldots$ do
  Obtain instance matrix $\boldsymbol{X}_t$
  Predict $\hat{y}_t = \mathrm{tr}(\boldsymbol{W}_t \boldsymbol{X}_t)$
  Obtain label $y_t$ and determine the loss $L_t = (y_t - \hat{y}_t)^2$
  Update $\boldsymbol{G}_t = \boldsymbol{G}_{t-1} - 2\eta(\hat{y}_t - y_t)\,\mathbf{sym}(\boldsymbol{X}_t)$
  Compute spectral decomposition: $\boldsymbol{G}_t = \boldsymbol{V}_t \Lambda_t \boldsymbol{V}_t^\top$
  Update $\boldsymbol{W}_{t+1} = \boldsymbol{V}_t\,\mathbf{exp}(\Lambda_t - c_t\boldsymbol{I})\boldsymbol{V}_t^\top/\mathrm{tr}(\mathbf{exp}(\Lambda_t - c_t\boldsymbol{I}))$, where $c_t = \max_s(\Lambda_t)_{s,s}$
end for
```

3.3 Relative Loss Bounds

For the sake of simplicity we now restrict ourselves to the case when the algorithm predicts with $\hat{y}_t = \mathrm{tr}(\boldsymbol{W}_t \boldsymbol{X}_t)$ and the loss function is quadratic: $L_t(\boldsymbol{W}_t) = L(\hat{y}_t, y_t) := (\hat{y}_t - y_t)^2$.

We begin with the definitions needed for the relative loss bounds. Let $S = (\boldsymbol{X}_1, y_1), \ldots, (\boldsymbol{X}_T, y_T)$ denote a sequence of examples, where the instance matrices $\boldsymbol{X}_t \in \mathbb{R}^{d\times d}$ and the labels $y_t \in \mathbb{R}$. The total loss of the on-line algorithm on the entire sequence S is $L_{MEG}(S) = \sum_{t=1}^{t}(\mathrm{tr}(\boldsymbol{W}_t \boldsymbol{X}_t) - y_t)^2$. We prove a bound on the *relative loss* $L_{MEG}(S) - L_{\boldsymbol{U}}(S)$ that holds for any comparator parameter $\boldsymbol{U}$. Such a comparator parameter is any symmetric positive semi-definite matrix $\boldsymbol{U}$ with trace one, and its total loss is defined as $L_{\boldsymbol{U}}(S) = \sum_{t=1}^{T}(\mathrm{tr}(\boldsymbol{U}\boldsymbol{X}_t) - y_t)^2$. The relative loss bound is derived in two steps: Lemma 3.1 upper bounds the relative loss for an individual trial in terms of the progress towards the comparator parameter $\boldsymbol{U}$ (as measured by the divergence). In the second Lemma 3.2, the bound for individual trials is summed to obtain a bound for a whole sequence. These two lemmas generalize similar lemmas previously proven for the exponentiated gradient update (Lemmas 5.8 and 5.9 of Kivinen and Warmuth, 1997).

Lemma 3.1 *Let $\boldsymbol{W}_t$ be any symmetric positive definite matrix. Let $\boldsymbol{X}_t$ be any square matrix for which the eigenvalues of $\mathbf{sym}(\boldsymbol{X}_t)$ have range at most r, i.e.*

$$\lambda^{\max}(\mathbf{sym}(\boldsymbol{X}_t)) - \lambda^{\min}(\mathbf{sym}(\boldsymbol{X}_t)) \le r.$$

Assume $\boldsymbol{W}_{t+1}$ is produced from $\boldsymbol{W}_t$ by the MEG update with learning rate η, and let $\boldsymbol{U}$ be any symmetric positive semi-definite matrix. Then for any $b > 0$ and $a = \eta = 2b/(2 + r^2 b)$:

$$a \underbrace{(y_t - \mathrm{tr}(\boldsymbol{W}_t \boldsymbol{X}_t))^2}_{\text{MEG-loss}} - b \underbrace{(y_t - \mathrm{tr}(\boldsymbol{U}\boldsymbol{X}_t))^2}_{\boldsymbol{U}\text{-loss}} \le \underbrace{\Delta_F(\boldsymbol{U}, \boldsymbol{W}_t) - \Delta_F(\boldsymbol{U}, \boldsymbol{W}_{t+1})}_{\text{progress towards } \boldsymbol{U}}. \qquad (3.6)$$

The above type of inequality is central to all relative loss bounds (Kivinen and Warmuth, 1997). If the loss of the algorithm is small, then the inequality becomes vacuous. However, if the algorithm incurs a large loss, then its parameter $\boldsymbol{W}_t$ must make progress towards any parameter vector $\boldsymbol{U}$ that has small loss on the current example (if such parameters exist).

The proof of this inequality is given in Appendix C. It has the same structure as the corresponding previous lemma proven for the exponentiated gradient algorithm, but now we apply the various matrix inequalities given at the end of Section 2.1 (in particular the Golden-Thompson inequality (2.3) and the approximation of the matrix exponential (Lemma 2.1)). These inequalities will also be essential for the analysis of *DefiniteBoost* in the next section.

Lemma 3.2 *Let S be any sequence of examples with square real matrices as instances and real labels, and let r be an upper bound on the range of eigenvalues of the symmetric part of each instance matrix of S. Let the initial parameter $\boldsymbol{W}_1$ and comparison parameter $\boldsymbol{U}$ be arbitrary symmetric positive definite matrices of trace one. Then for any c such that $\eta = 2c/(r^2(2+c))$,*

$$L_{MEG}(S) \leq \left(1+\frac{c}{2}\right) L_{\boldsymbol{U}}(S) + \left(\frac{1}{2}+\frac{1}{c}\right) r^2 \Delta_F(\boldsymbol{U}, \boldsymbol{W}_1). \tag{3.7}$$

Proof For the maximum tightness of (3.6), a should be chosen as $a = \eta = 2b/(2+r^2 b)$. Let $b = c/r^2$, and thus $a = 2c/(r^2(2+c))$. Then (3.6) is rewritten as

$$\frac{2c}{2+c}(y_t - \operatorname{tr}(\boldsymbol{W}_t \boldsymbol{X}_t))^2 - c(y_t - \operatorname{tr}(\boldsymbol{U}\boldsymbol{X}_t))^2 \leq r^2(\Delta_F(\boldsymbol{U}, \boldsymbol{W}_t) - \Delta_F(\boldsymbol{U}, \boldsymbol{W}_{t+1}))$$

Adding the bounds for $t = 1, \cdots, T$, we get

$$\frac{2c}{2+c} L_{MEG}(S) - c L_{\boldsymbol{U}}(S) \leq r^2(\Delta_F(\boldsymbol{U}, \boldsymbol{W}_1) - \Delta_F(\boldsymbol{U}, \boldsymbol{W}_{t+1})) \leq r^2 \Delta_F(\boldsymbol{U}, \boldsymbol{W}_1),$$

which is equivalent to (3.7). ■

Assuming $L_{\boldsymbol{U}}(S) \leq L_{\max}$ and $\Delta_F(\boldsymbol{U}, \boldsymbol{W}_1) \leq d_{\max}$, then the bound (3.7) is tightest when $c = r\sqrt{2d_{\max}/L_{\max}}$. With this choice of c, we have

$$L_{MEG}(S) - L_{\boldsymbol{U}}(S) \leq r\sqrt{2L_{\max} d_{\max}} + \frac{r^2}{2}\Delta_F(\boldsymbol{U}, \boldsymbol{W}_1).$$

In particular, if $\boldsymbol{W}_1 = \frac{1}{d}\boldsymbol{I}$, then $\Delta_F(\boldsymbol{U}, \boldsymbol{W}_1) = \log d - \sum_i \lambda_i \log \frac{1}{\lambda_i} \leq \log d$. Additionally, when $L_{\max} = 0$, then the total loss of the algorithm is bounded by $\frac{r^2 \log d}{2}$.

Note that the MEG algorithm generalizes the EG algorithm of Kivinen and Warmuth (1997). In the case of linear regression, a square of a product of dual norms appears in the bounds for the EG algorithm: $||\boldsymbol{u}||_1^2 X_\infty^2$. Here $\boldsymbol{u}$ is a parameter *vector* and X_∞ is an upper bound on the infinity norm of the instance vectors $\boldsymbol{x}_t$. Note the correspondence with the above bound (which generalizes the bounds for EG to the non-diagonal case): the one norm of the parameter vector is replaced by the trace and the infinity norm by the maximum range of the eigenvalues.

4. Bregman Projection and *DefiniteBoost*

Using the von Neumann divergence, we will generalize the boosting algorithms for matrix parameters.

4.1 Preliminaries

In this section, we address the following Bregman projection problem of finding a positive semi-definite symmetric matrix $\boldsymbol{W} \in \mathbb{R}^{d\times d}$ of trace one satisfying a set of linear constraints:[3]

$$\begin{aligned} \boldsymbol{W}^* = \operatorname*{argmin}_{\boldsymbol{W}} \quad & \Delta_F(\boldsymbol{W}, \boldsymbol{W}_1) \\ \text{s.t.} \quad & \boldsymbol{W} = \boldsymbol{W}^\top, \operatorname{tr}(\boldsymbol{W}) = 1 \\ & \operatorname{tr}(\boldsymbol{W}\boldsymbol{C}_j) \leq 0, \text{ for } j = 1, \dots, n, \end{aligned} \tag{4.1}$$

where the symmetric positive definite matrix $\boldsymbol{W}_1$ of trace one is the initial parameter matrix and $\boldsymbol{C}_1, \dots, \boldsymbol{C}_n$ are arbitrary matrices. Note that we do not explicitly constrain $\boldsymbol{W}$ to be positive semi-definite because when the von Neumann divergence is used, then the solution $\boldsymbol{W}^*$ will always be positive semi-definite. Prior knowledge about $\boldsymbol{W}$ is encoded in the constraints, and the matrix closest to $\boldsymbol{W}_1$ is chosen among the matrices satisfying all constraints. Tsuda and Noble (2004) employed this approach for learning a kernel matrix among graph nodes, and this method can be potentially applied to learn a kernel matrix in other settings (e.g., Xing et al., 2003; Tsang and Kwok, 2003). In the previous work by (Tsuda and Noble, 2004), an algorithm was developed that processes a batch of constraints. The problem was converted to a dual unconstraint problem (as done below) and an iterative gradient descent algorithm was given. However, no convergence proofs were provided previously. In this paper we give on-line algorithms with strong convergence proofs.[4]

The problem (4.1) is a projection of $\boldsymbol{W}_1$ to the intersection of convex regions defined by the constraints. It is well known that the Bregman projection into the intersection of convex regions can be solved by sequential projections to each region (Bregman, 1967; Censor and Lent, 1981). In the original papers only asymptotic convergence was shown. More recently a connection (Kivinen and Warmuth, 1999; Lafferty, 1999) was made to the AdaBoost algorithm which has an improved convergence analysis (Freund and Schapire, 1997; Schapire and Singer, 1999). We generalize the latter algorithm and its analysis to symmetric positive definite matrices and call the new algorithm *DefiniteBoost*. As in the original setting, only *approximate* projections (Figure 1) are required to show fast convergence.

Before presenting the algorithm, let us describe the dual problem of minimizing the von Neumann divergence subject to linear constraints (4.1). The dual variables are the Lagrange multipliers $\boldsymbol{\alpha} \in \mathbb{R}^n$ ($\boldsymbol{\alpha} \geq \mathbf{0}$) associated with this optimization problem:

$$\boldsymbol{\alpha}^* = \operatorname*{argmax}_{\boldsymbol{\alpha}\geq\mathbf{0}} \quad -\log\left\{\operatorname{tr}\left(\mathbf{exp}(\mathbf{log}\,\boldsymbol{W}_1 - \sum_{j=1}^n \alpha_j\,\mathbf{sym}(\boldsymbol{C}_j))\right)\right\}. \tag{4.2}$$

See Appendix D for a detailed derivation of the dual problem that handles the case when the constraint matrix $\boldsymbol{C}_j$ is allowed to be an arbitrary square matrix. Previous derivations required symmetric $\boldsymbol{C}_j$ (Tsuda and Noble, 2004). When (4.1) is feasible, the optimal solution is described as

$$\boldsymbol{W}^* = \frac{1}{\mathrm{Z}(\boldsymbol{\alpha}^*)}\,\mathbf{exp}(\mathbf{log}\,\boldsymbol{W}_1 - \sum_{j=1}^n \alpha_j^*\,\mathbf{sym}(\boldsymbol{C}_j)),$$

3. Note that if η is large then the on-line update (3.1) becomes a Bregman projection subject to a single equality constraint $\operatorname{tr}(\boldsymbol{W}\boldsymbol{X}_t) = y_t$.
4. The methodology employed in this paper is not limited to on-line learning. For example in Littlestone et al. (1992), cf. Corollary 15, the EG algorithm was used for solving a system of linear equations and fast convergence was shown.

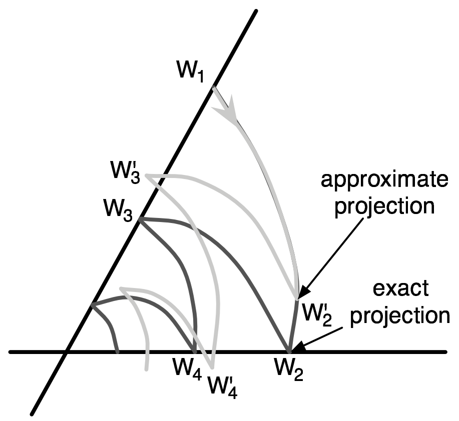


Figure 1: The intersection of two convex sets (here two straight lines) can be found by projecting back and forth between the two sets with exact Bregman projections ($\boldsymbol{W}_1, \boldsymbol{W}_2, \ldots$). In this paper we use certain approximate projections ($\boldsymbol{W}_1, \boldsymbol{W}_2', \ldots$). Now each projection may over or undershoot the alternating target set. Nevertheless, global convergence to the optimal solution is still guaranteed via our proofs.

where $\mathrm{Z}(\boldsymbol{\alpha}^*) = \mathrm{tr}\left(\mathbf{exp}(\mathbf{log}\, \boldsymbol{W}_1 - \sum_{j=1}^n \alpha_j^* \,\mathbf{sym}(\boldsymbol{C}_j))\right)$ and $\boldsymbol{\alpha}^*$ is the optimal dual solution.

4.2 Exact Bregman Projections

Problem (4.1) can be solved with the following algorithm: Start from some initial parameter $\boldsymbol{W}_1$ (for instance $\boldsymbol{W}_1 = \frac{1}{d}\boldsymbol{I}$). At the t-th step, choose an unsatisfied constraint j_t, i.e. $\mathrm{tr}(\boldsymbol{W}_t \boldsymbol{C}_{j_t}) > 0$.[5] Then solve the following Bregman projection with respect to the chosen constraint:

$$\begin{aligned} \boldsymbol{W}_{t+1} = \underset{\boldsymbol{W}}{\mathrm{argmin}} \quad & \Delta_F(\boldsymbol{W}, \boldsymbol{W}_t) \\ \text{s.t.} \quad & \boldsymbol{W} = \boldsymbol{W}^\top, \mathrm{tr}(\boldsymbol{W}) = 1, \\ & \mathrm{tr}(\boldsymbol{W}\boldsymbol{C}_{j_t}) \le 0. \end{aligned} \tag{4.3}$$

By means of a Lagrange multiplier α, the dual problem is described as (cf. Appendix D)

$$\alpha_t^* = \underset{\alpha \ge 0}{\mathrm{argmin}} \quad \mathrm{tr}\left(\mathbf{exp}(\mathbf{log}\, \boldsymbol{W}_t - \alpha\, \mathbf{sym}(\boldsymbol{C}_{j_t}))\right). \tag{4.4}$$

Using the solution of the dual problem, $\boldsymbol{W}_t$ is updated as

$$\boldsymbol{W}_{t+1} = \frac{1}{\mathrm{Z}_t(\alpha_t^*)} \mathbf{exp}(\mathbf{log}\, \boldsymbol{W}_t - \alpha_t^* \,\mathbf{sym}(\boldsymbol{C}_{j_t})) \tag{4.5}$$

where the normalization factor is $\mathrm{Z}_t(\alpha_t^*) = \mathrm{tr}\left(\mathbf{exp}(\mathbf{log}\, \boldsymbol{W}_t - \alpha_t^* \,\mathbf{sym}(\boldsymbol{C}_{j_t}))\right)$. If $\boldsymbol{W}_t$ is symmetric positive definite, then $\boldsymbol{W}_{t+1}$ is as well. Note that we can use the same numerically stable reformulation of the update as discussed in Section 3.2.

5. For instance, the most unsatisfied constraint, i.e. $j_t = \mathrm{argmax}_{j=1,\cdots,n} \mathrm{tr}(\boldsymbol{W}_t \boldsymbol{C}_j)$, can be chosen.

4.3 Approximate Bregman Projections

The solution of (4.4) cannot be obtained in closed form. However, one can use the following approximate choice of α_t:

$$\widehat{\alpha}_t = \frac{1}{\lambda_t^{\max} - \lambda_t^{\min}} \log\left(\frac{1 + r_t/\lambda_t^{\max}}{1 + r_t/\lambda_t^{\min}}\right), \tag{4.6}$$

when the eigenvalues of $\mathbf{sym}(\boldsymbol{C}_{j_t})$ lie in the interval $[\lambda_t^{\min}, \lambda_t^{\max}]$ and $r_t = \mathrm{tr}(\boldsymbol{W}_t \boldsymbol{C}_{j_t})$. Since the most unsatisfied constraint is chosen, $r_t \geq 0$ and thus $\widehat{\alpha}_t \geq 0$. We call this approximate Bregman projection algorithm *DefiniteBoost*. It may be seen as a natural extension of AdaBoost (cf. Section 4.5), where probability distributions are replaced by symmetric positive definite matrices of trace one. The pseudo-code of DefiniteBoost is given in Algorithm 2.

Algorithm 2 Pseudo-code of the DefiniteBoost algorithm; $\lambda_t^{\min}$ and $\lambda_t^{\max}$ are lower and upper bounds on the eigenvalues of $\mathbf{sym}(\boldsymbol{C}_t)$.

```
Choose $\boldsymbol{W}_1$
Initialize $\boldsymbol{G}_0 = \mathbf{log}\,\boldsymbol{W}_1$
**for** $t = 1, 2, \ldots$ **do**
  Choose an unsatisfied constraint $j_t$ (i.e. $\mathrm{tr}(\boldsymbol{W}_t \boldsymbol{C}_{j_t}) > 0$) or stop when all constraints satisfied
  Compute constraint violation $r_t = \mathrm{tr}(\boldsymbol{W}_t \boldsymbol{C}_{j_t})$
  Compute approximate step size $\widehat{\alpha}_t = \frac{1}{\lambda_t^{\max} - \lambda_t^{\min}} \log\left(\frac{1 + r_t/\lambda_t^{\max}}{1 + r_t/\lambda_t^{\min}}\right)$
  Update $\boldsymbol{G}_t = \boldsymbol{G}_{t-1} - \widehat{\alpha}_t\,\mathbf{sym}(\boldsymbol{C}_{j_t})$
  Compute spectral decomposition: $\boldsymbol{G}_t = \boldsymbol{V}_t \Lambda_t \boldsymbol{V}_t$
  Update $\boldsymbol{W}_{t+1} = \boldsymbol{V}_t\,\mathbf{exp}(\Lambda_t - c_t \boldsymbol{I}) \boldsymbol{V}_t^\top / \mathrm{tr}(\mathbf{exp}(\Lambda_t - c_t \boldsymbol{I}))$, where $c_t = \max_s (\Lambda_t)_{s,s}$
**end for**
```

Although the projection is done only approximately,[6] the convergence of the dual objective (4.2) can be shown using the following upper bound of the negative dual objective , i.e.

$$\mathrm{tr}\left(\mathbf{exp}(\mathbf{log}\,\boldsymbol{W}_1 - \sum_{j=1}^{n} \alpha_j\,\mathbf{sym}(\boldsymbol{C}_j))\right).$$

Theorem 4.1 *The negative exponentiated dual objective is bounded from above by*

$$\mathrm{tr}\left(\mathbf{exp}\left(\mathbf{log}\,\boldsymbol{W}_1 - \sum_{t=1}^{T} \widehat{\alpha}_t\,\mathbf{sym}(\boldsymbol{C}_{j_t})\right)\right) \leq \prod_{t=1}^{T} \rho(r_t), \tag{4.7}$$

where

$$\widehat{\alpha}_t = \frac{1}{\lambda_t^{\max} - \lambda_t^{\min}} \log\left(\frac{1 + r_t/\lambda_t^{\max}}{1 + r_t/\lambda_t^{\min}}\right),\ r_t = \mathrm{tr}(\boldsymbol{W}_t \boldsymbol{C}_{j_t}),$$

and

$$\rho(r_t) = \left(1 - \frac{r_t}{\lambda_t^{\max}}\right)^{\frac{\lambda_t^{\max}}{\lambda_t^{\max} - \lambda_t^{\min}}} \left(1 - \frac{r_t}{\lambda_t^{\min}}\right)^{\frac{-\lambda_t^{\min}}{\lambda_t^{\max} - \lambda_t^{\min}}}.$$

6. The approximate Bregman projection (with α_t as in (4.6)) can also be motivated as an on-line algorithm based on an entropic loss and learning rate one (following Section 3 and Kivinen and Warmuth (1999)).

The proof of this inequality for our setting is given in Appendix E. The bound (4.7) is monotonically decreasing, because $\rho(r_t) \leq 1$. Also, since we always chose a violated constraint (if there is one), we have $r_t > 0$ and therefore $\rho(r_t) < 1$ (or we stop). Thus the dual objective (4.2) continues to increase until all constraints are satisfied.

4.4 Convergence Speed

Next we determine the maximal number of iterations needed to find a matrix $\boldsymbol{W}$ which satisfies all constraints up to the predetermined accuracy ε, i.e. $\operatorname{tr}(\boldsymbol{W}\boldsymbol{C}_j) \leq \varepsilon$, for $1 \leq j \leq n$. The algorithm selects in each iteration an constraint j_t that is violated by at least ε (i.e. $r_t = \operatorname{tr}(\boldsymbol{W}_t\boldsymbol{C}_{j_t}) \geq \varepsilon$), or stops if no such constraint exists. Assuming the algorithm stops at $(T+1)$-th step, we derive an upper bound on T as a function of ε.

For simplicity, let us assume $\boldsymbol{W}_1 = \frac{1}{d}\boldsymbol{I}$, $\lambda_j^{\min} = -\lambda$, and $\lambda_j^{\max} = \lambda$ (for all j). Denote by $h_{primal}(\boldsymbol{W})$ and $h_{dual}(\boldsymbol{\alpha})$ the primal and dual objective functions in (4.1) and (4.2), respectively.

$$\begin{aligned} h_{primal}(\boldsymbol{W}) &= \Delta_F(\boldsymbol{W}, \boldsymbol{W}_1) && (4.8)\\ h_{dual}(\boldsymbol{\alpha}) &= -\log\operatorname{tr}\left(\mathbf{exp}\left(\mathbf{log}\,\boldsymbol{W}_1 - \sum_{j=1}^{n} \alpha_j\,\mathbf{sym}(\boldsymbol{C}_j)\right)\right) && (4.9)\end{aligned}$$

The primal objective is upper-bounded by $\log d$, since $\Delta_F(\boldsymbol{W}, \boldsymbol{W}_1) = \sum_i \lambda_i \log\lambda_i + \log d \leq \log d$. Since the algorithm stops at the $(T+1)$-th iteration (with $r_t \geq \varepsilon$ for $t = 1, \ldots, T$), we get from Theorem 4.1:

$$\exp(-h_{dual}(\tilde{\boldsymbol{\alpha}})) = \operatorname{tr}\left(\mathbf{exp}\left(\mathbf{log}\,\boldsymbol{W}_1 - \sum_{t=1}^{T} \widehat{\alpha}_t\,\mathbf{sym}(\boldsymbol{C}_{j_t})\right)\right) \leq \left(\frac{\lambda^2 - \varepsilon^2}{\lambda^2}\right)^{T/2},$$

where $\tilde{\boldsymbol{\alpha}}$ is the cumulative coefficient vector for the constraints, i.e. $\widetilde{\alpha}_j = \sum_{t=1}^{T} \widehat{\alpha}_t \delta(j_t = j)$, for $1 \leq j \leq n$.

Thus the objective in (4.2) is lower bounded by $\frac{1}{2}T\frac{\varepsilon^2}{\lambda^2}$, since

$$\begin{aligned} h_{dual}(\widetilde{\boldsymbol{\alpha}}) &\geq -\log\left(\frac{\lambda^2 - \varepsilon^2}{\lambda^2}\right)^{T/2}\\ &\geq \frac{T\varepsilon^2}{2\lambda^2}, && (4.10)\end{aligned}$$

where the last inequality follows by convexity of $-\log\left(\frac{\lambda^2-\varepsilon^2}{\lambda^2}\right)$ with respect to ε. At the optimal solution $\boldsymbol{W}^*$ and $\boldsymbol{\alpha}^*$, the values of the objective functions coincide, i.e. $h_{dual}(\boldsymbol{\alpha}^*) = h_{primal}(\boldsymbol{W}^*)$. Finally, we obtain

$$\frac{T\varepsilon^2}{2\lambda^2} \leq h_{dual}(\widetilde{\boldsymbol{\alpha}}) \leq h_{dual}(\boldsymbol{\alpha}^*) = h_{primal}(\boldsymbol{W}^*) \leq \log d,$$

and the upper bound $T \leq \frac{2\lambda^2 \log d}{\varepsilon^2}$. In summary, we have proven the following:

Corollary 4.2 *Suppose we are solving problem* (4.1) *with DefiniteBoost, where $\boldsymbol{C}_j$ ($j = 1, \ldots, n$) are arbitrary matrices with $\lambda_{\min}(\boldsymbol{C}_j) \geq -\lambda$ and $\lambda_{\max}(\boldsymbol{C}_j) \leq \lambda$ and $\boldsymbol{W}_1 = \frac{1}{d}\boldsymbol{I}$. Assume an optimal solution $\boldsymbol{W}^*$ to* (4.1) *exists and the algorithm selects in each iteration an ε-violated constraint, i.e.*

$r_t = \mathrm{tr}(\boldsymbol{W}_t \boldsymbol{C}_{j_t}) \geq \varepsilon$, or stops if no such constraint exists. Then after at most $T = \frac{2\lambda^2 \log d}{\varepsilon^2}$ iterations, DefiniteBoost stops and the resulting $\boldsymbol{W}$ satisfies all linear constraints up to accuracy ε, i.e.

$$\mathrm{tr}(\boldsymbol{W}\boldsymbol{C}_j) \leq \varepsilon \quad \textit{for all } j = 1, \ldots, n.$$

This result implies that we can solve (4.1) with accuracy ε in $O(d^3 \log d / \varepsilon^2)$ operations (excluding the cost of identifying violated constraints). Similar bounds on the number of iterations for solving a system of linear equations with the EG algorithm were first proven in (Littlestone et al., 1992, Corollary 15). Observe that if (4.1) is not feasible, then one may continue finding ε-violated constraints and the primal objective can become unbounded, i.e. $\sum_t \widehat{\alpha}_t$ may become unbounded.

4.5 Relation to Boosting

When all matrices are diagonal, then DefiniteBoost specializes to the AdaBoost algorithm (Schapire and Singer, 1999). Let $\{\boldsymbol{x}_i, y_i\}_{i=1}^d$ be the training samples, where $\boldsymbol{x}_i \in \mathbb{R}^m$ and $y_i \in \{-1, 1\}$. Let $h_1(x), \ldots, h_n(x) \in [-1, 1]$ be the weak hypotheses. For the j-th hypothesis $h_j(x)$, let us define $\boldsymbol{C}_j = \mathrm{diag}(y_1 h_j(x_1), \ldots, y_d h_j(x_d))$. Since $|y h_j(x)| \leq 1$, we may choose $\lambda_t^{\max} = 1$ and $\lambda_t^{\min} = -1$ for any t. Setting $\boldsymbol{W}_1 = \frac{1}{d}\boldsymbol{I}$, the dual objective (4.7) is rewritten as

$$-\log\left(\frac{1}{d}\sum_{i=1}^{d} \exp\left(-y_i \sum_{j=1}^{n} \alpha_j h_j(x_i)\right)\right),$$

which is equivalent to the exponential loss function used in AdaBoost. Since $\boldsymbol{C}_j$ and $\boldsymbol{W}_1$ are diagonal, the matrix $\boldsymbol{W}_t$ stays diagonal after the update. If $w_{t,i} = (\boldsymbol{W}_t)_{i,i}$, the updating formula (4.5) becomes the AdaBoost update: $w_{t+1,i} = w_{t,i} \exp(-\alpha_t y_i h_t(x_i))/Z_t(\alpha_t)$. The approximate solution of α_t (4.6) is described as $\alpha_t = \frac{1}{2} \log \frac{1+r_t}{1-r_t}$, where r_t is the weighted training error of the t-th hypothesis, i.e. $r_t = \sum_{i=1}^d w_{t,i} y_i h_t(x_i)$.

4.6 Solving Semi-definite Programs

Suppose we aim to solve the following semi-definite programming problem:

$$\begin{aligned} \boldsymbol{W}^* = \operatorname*{argmin}_{\boldsymbol{W}, \theta} \quad & \theta \\ \text{s.t.} \quad & \mathrm{tr}(\boldsymbol{W}) = 1, \boldsymbol{W} \succeq \boldsymbol{0}, \boldsymbol{W} = \boldsymbol{W}^\top \\ & \mathrm{tr}(\boldsymbol{W}\boldsymbol{C}_j) \leq \theta, \text{ for } j = 1, \ldots, n. \end{aligned} \tag{4.11}$$

If one would know the optimal θ^* beforehand, then following problem would lead to an optimal solution of (4.11):

$$\begin{aligned} \boldsymbol{W}^* = \operatorname*{argmin}_{\boldsymbol{W}} \quad & \Delta_F(\boldsymbol{W}, \frac{1}{d}\boldsymbol{I}) \\ \text{s.t.} \quad & \mathrm{tr}(\boldsymbol{W}) = 1, \boldsymbol{W} = \boldsymbol{W}^\top \\ & \mathrm{tr}(\boldsymbol{W}(\boldsymbol{C}_j - \theta^* \boldsymbol{I})) \leq 0, \text{ for } j = 1, \ldots, n. \end{aligned} \tag{4.12}$$

Running DefiniteBoost on the above problem with matrices $\widetilde{\boldsymbol{C}}_j = (\boldsymbol{C}_j - \theta^* \boldsymbol{I})$ can approximate the solution of (4.12) rather efficiently and, hence, it is only left to determine the optimal value

θ^*. If it is chosen too small, then no feasible solution to (4.12) exists and DefiniteBoost will not terminate after $2\lambda^2 \log d/\varepsilon^2$ iterations with accuracy ε,[7] where $\lambda^{\min}(\widetilde{C}_j) \geq -\lambda$ and $\lambda^{\max}(\widetilde{C}_j) \leq \lambda$. If it is chosen too large, then a feasible solution exists and DefiniteBoost terminates in a bounded number of iterations. Hence one has a way of identifying when $\theta < \theta^*$ and also $\theta > \theta^*$. This allows the design of a binary search procedure to approximate θ^* in a few steps. Based on this idea we previously proposed a margin maximizing version of AdaBoost (Rätsch and Warmuth, 2002). For this algorithm we could show that after $O(\log d \log(1/\varepsilon)/\varepsilon^2)$ iterations the algorithm achieved an optimal solution within accuracy ε. We claim that the outlined binary search procedure can also be applied in combination with DefiniteBoost for solving the semi-definite problem (4.11) in time $O(d^3 \log d \log(1/\varepsilon)/\varepsilon^2)$ (excluding the cost of identifying violated constraints). Additionally we assert that a slightly more advanced adaptation of θ during the optimization (as was done by Rätsch, 2001; Rätsch and Warmuth, 2005, for the diagonal case) will yield the reduced time complexity of $O(d^3 \log d/\varepsilon^2)$. Rigorous proofs of these conjectures go beyond the scope of this paper.

5. Experiments on Learning Kernels

In this section, our technique is applied to learning a kernel matrix from a set of distance measurements. This application is not on-line *per se*, but it shows nevertheless that the theoretical bounds can be reasonably tight on natural data.

When $\boldsymbol{K}$ is a $d \times d$ kernel matrix among d objects, then the K_{ij} characterizes the similarity between objects i and j. In the feature space, K_{ij} corresponds to the inner product between object i and j, and thus the Euclidean distance can be computed from the entries of the kernel matrix (Schölkopf and Smola, 2002). In some cases, the kernel matrix is not given explicitly, but only a set of distance measurements is available. The data are represented either as (i) quantitative distance values (e.g., the distance between i and j is 0.75), or (ii) qualitative evaluations (e.g., the distance between i and j is small) (Xing et al., 2003; Tsuda and Noble, 2004). Our task is to obtain a positive definite kernel matrix which fits well to the given distance data.

5.1 On-line Kernel Learning

In the first experiment, we consider the on-line learning scenario in which only one distance example is shown to the learner at each time step. The distance example at time t is described as $\{a_t, b_t, y_t\}$, which indicates that the squared Euclidean distance between objects a_t and b_t is y_t. Let us define a time-developing sequence of kernel matrices as $\{\boldsymbol{W}_t\}_{t=1}^T$, and the corresponding points in the feature space as $\{\boldsymbol{x}_{ti}\}_{i=1}^d$ (i.e. $(\boldsymbol{W}_t)_{ab} = \boldsymbol{x}_{ta}^\top \boldsymbol{x}_{tb}$). Then, the total loss incurred by this sequence is

$$\sum_{t=1}^T \left(\|\boldsymbol{x}_{ta_t} - \boldsymbol{x}_{tb_t}\|^2 - y_t\right)^2 = \sum_{t=1}^T (\mathrm{tr}(\boldsymbol{W}_t \boldsymbol{X}_t) - y_t)^2,$$

where $\boldsymbol{X}_t$ is a symmetric matrix whose (a_t, a_t) and (b_t, b_t) elements are 0.5, (a_t, b_t) and (b_t, a_t) elements are -0.5, and all the other elements are zero. We consider a controlled experiment in which the distance examples are created from a known *target kernel matrix*. We used a 52×52 kernel matrix among `gyrB` proteins of bacteria ($d = 52$). This data contains three bacteria species (see Tsuda et al., 2003, for details). Each distance example is created by randomly choosing one element of the target kernel. The initial parameter was set as $\boldsymbol{W}_1 = \frac{1}{d}\boldsymbol{I}$. When the comparison matrix U is set

7. This statement is slightly simplified. Please check Rätsch and Warmuth (2002) for details.

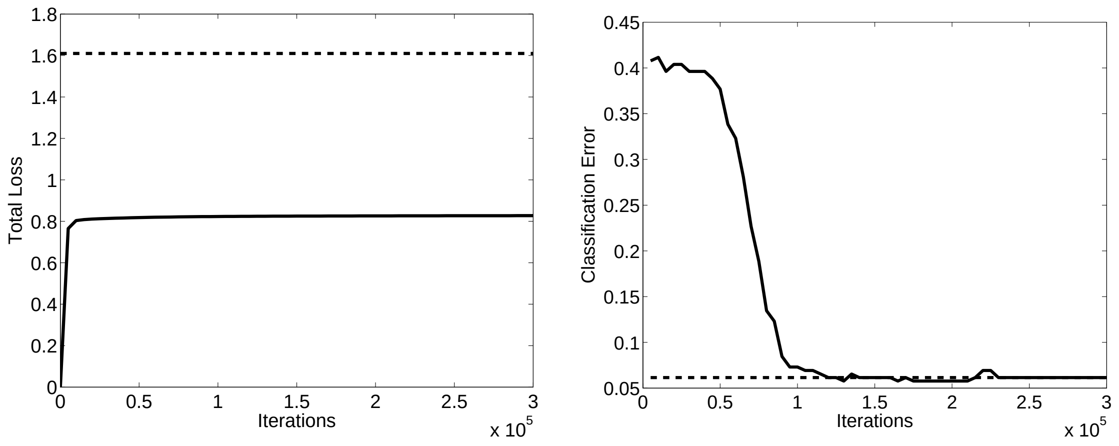


Figure 2: Numerical results of on-line learning. (Left) total loss against the number of iterations. The dashed line shows the loss bound. (Right) classification error of the nearest neighbor classifier using the learned kernel. The dashed line shows the error by the target kernel.

to the target matrix, then because all the distance examples are derived from this matrix, $L_U(S) = 0$ and $L_{\max} = 0$. Therefore we choose learning rate $\eta = 2$, which minimizes the relative loss bound of Lemma 3.2. The total loss of the kernel matrix sequence obtained by the matrix exponential update is shown in Figure 2 (left). In the plot, we have also shown the relative loss bound. The bound seems to give a reasonably tight performance guarantee—it is about twice the actual total loss. To evaluate the learned kernel matrix, the prediction accuracy of bacteria species by the nearest neighbor classifier is calculated (Figure 2, right), where the 52 proteins are randomly divided into 50% training and 50% testing data. The value shown in the plot is the test error averaged over 10 different divisions. It took a large number of iterations ($\sim 2 \times 10^5$) for the error rate to converge to the level of the target kernel. In practice one can often increase the learning rate for faster convergence, but here we chose the small rate suggested by our analysis to check the tightness of the bound.

5.2 Kernel Learning by Bregman Projection

Next, let us consider a batch learning scenario where we have a set of qualitative distance evaluations (i.e. inequality constraints). Given n pairs of similar objects $\{a_j, b_j\}_{j=1}^n$, the inequality constraints are constructed as $\|\boldsymbol{x}_{a_j} - \boldsymbol{x}_{b_j}\| \leq \gamma$, $j = 1, \dots, n$, where γ is a predetermined constant. If $\boldsymbol{X}_j$ is defined as in the previous section and $\boldsymbol{C}_j = \boldsymbol{X}_j - \gamma \boldsymbol{I}$, the inequalities are then rewritten as $\mathrm{tr}(\boldsymbol{W}\boldsymbol{C}_j) \leq 0, j = 1, \dots, n$. The largest and smallest eigenvalues of any $\boldsymbol{C}_j$ are $1 - \gamma$ and $-\gamma$, respectively. As in the previous section, distance examples are randomly generated from the target kernel matrix between `gyrB` proteins. Setting $\gamma = 0.2/d$, we collected all object pairs whose distance in the feature space is less than γ to yield 980 inequalities ($n = 980$). Figure 3 (left) shows the convergence of the dual objective function as proven in Theorem 4.1. The convergence was much faster than the previous experiment, because in the batch setting, one can choose the most unsatisfied constraint and optimize the step size as well. Figure 3 (right) shows the classification error of the nearest

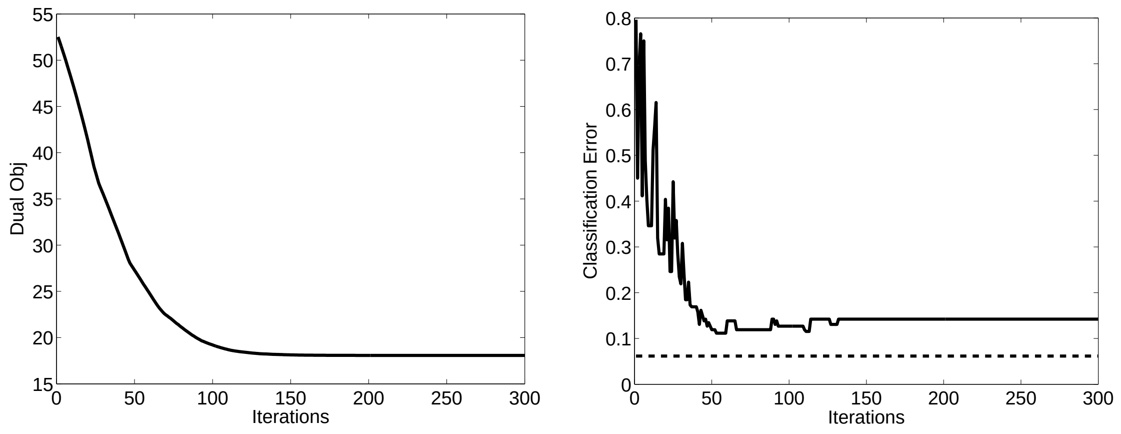


Figure 3: Numerical results of Bregman projection. (Left) convergence of the dual objective function. (Right) classification error of the nearest neighbor classifier using the learned kernel.

neighbor classifier. As opposed to the previous experiment, the error rate is higher than that of the target kernel matrix, because a substantial amount of information is lost by the conversion to inequality constraints.

6. Summary and Discussion

We motivated and analyzed a new update for symmetric positive matrices using the *von Neumann* divergence. We showed that the standard bounds for on-line learning and boosting generalize to the case when the parameters are symmetric positive definite matrices of trace one instead of a probability vector. As in quantum physics, the eigenvalues act as probabilities. In addition to the applications suggested by the experiments, our algorithm can be straightforwardly applied to learning a covariance matrix. It would also be interesting to use a robust loss $L_t(\boldsymbol{W})$ for the purpose of ignoring outliers (Huber, 1981) and investigate possible applications of our learning algorithms to quantum statistical inference problems (Barndorff-Nielsen et al., 2003).

Our method is designed for learning a positive definite parameter matrix of fixed size. It is not straightforward to extend it to the case where the size of the parameter matrix grows on-line as more examples are seen. Our methods immediately generalize to the Hermitian matrices, i.e. square matrices in $\mathbb{C}^{d\times d}$ for which $\boldsymbol{A} = \bar{\boldsymbol{A}}^\top = \boldsymbol{A}^*$. The spectral decomposition of these matrices becomes $\boldsymbol{A} = \boldsymbol{U}\Lambda\boldsymbol{U}^*$, where $\boldsymbol{U}$ is a unitary matrix (i.e. $\boldsymbol{U}\boldsymbol{U}^* = \boldsymbol{I}$) and Λ is a diagonal matrix of *real* eigenvalues. In the case when all entries of the matrix are real, then Hermitian is equivalent to symmetric. All algorithms of this paper (and their analyzes) immediately generalize to the case when symmetric is replaced by Hermitian and symmetric positive definite by positive Hermitian (i.e. Hermitian with positive eigenvalues). In particular, the Golden-Thompson inequality, Jensen's inequality for the matrix exponential (Lemma 2.1) and Lemma 2.2 all hold for Hermitian matrices. Note that density matrices (as used in Statistical Physics) are positive Hermitian matrices of trace one.

Acknowledgments

We dedicate this paper to Nick Littlestone who first proved relative loss bounds for an algorithm in the EG family—his well known *Winnow* algorithm for learning disjunctions (Littlestone, 1988, 1989).

K.T. and G.R. gratefully acknowledge partial support from the PASCAL Network of Excellence (EU #506778). M.W. was supported by NSF grant CCR 9821087 and UC Discovery grant LSIT02-10110. This work was partially done while G.R. was still at Fraunhofer FIRST in Berlin and Max Planck Institute for Biological Cybernetics in Tübingen. Part of this work was done while all three authors were visiting the National ICT Australia in Canberra.

Appendix A. Derivatives of Matrix Functions

The matrix functions considered in this paper are mostly trace functions e.g. $\mathrm{tr}(\mathbf{exp}(\boldsymbol{W}))$ and $\mathrm{tr}(\boldsymbol{W}\,\mathbf{log}\,\boldsymbol{W})$, which we will expand into power series. Thus we begin with computing the gradient of $\mathrm{F}(\boldsymbol{W}) = \mathrm{tr}(\boldsymbol{W}^k)$. The partial derivative with respect to (i, j) element is described as

$$\frac{\partial \mathrm{tr}(\boldsymbol{W}^k)}{\partial W_{ij}} = \lim_{\lambda \to 0} \frac{\mathrm{tr}((\boldsymbol{W} + \lambda \boldsymbol{E}_{ij})^k) - \mathrm{tr}(\boldsymbol{W}^k)}{\lambda},$$

where $\boldsymbol{E}_{ij}$ is the sparse matrix whose (i, j) element is one and all the others are zero. For example, when $k = 3$,

$$(\boldsymbol{W} + \lambda \boldsymbol{E}_{ij})^3 = (\boldsymbol{W}^3 + \lambda \boldsymbol{E}_{ij}\boldsymbol{W}\boldsymbol{W} + \lambda \boldsymbol{W}\boldsymbol{E}_{ij}\boldsymbol{W} + \lambda \boldsymbol{W}\boldsymbol{W}\boldsymbol{E}_{ij}) + O(\lambda^2).$$

The trace is simply described as

$$\begin{aligned} \mathrm{tr}((\boldsymbol{W} + \lambda \boldsymbol{E}_{ij})^3) &= \mathrm{tr}(\boldsymbol{W}^3) + 3\lambda \mathrm{tr}(\boldsymbol{E}_{ij}\boldsymbol{W}^2) + O(\lambda^2) \\ &= \mathrm{tr}(\boldsymbol{W}^3) + 3\lambda [\boldsymbol{W}^2]_{j,i} + O(\lambda^2). \end{aligned}$$

Therefore, $\nabla_{\boldsymbol{W}} \mathrm{tr}(\boldsymbol{W}^3) = 3(\boldsymbol{W}^2)^\top$. For general k, we get

$$\nabla_{\boldsymbol{W}} \mathrm{tr}(\boldsymbol{W}^k) = k(\boldsymbol{W}^{k-1})^\top. \tag{A.1}$$

The matrix exponential is defined as

$$\mathbf{exp}(\boldsymbol{W}) = \boldsymbol{I} + \boldsymbol{W} + \frac{1}{2!}\boldsymbol{W}^2 + \frac{1}{3!}\boldsymbol{W}^3 + \cdots.$$

Applying (A.1) to all terms, we get $\nabla_{\boldsymbol{W}} \mathrm{tr}(\mathbf{exp}(\boldsymbol{W})) = \mathbf{exp}(\boldsymbol{W})^\top$. Next, let us calculate the gradient of $\mathrm{tr}(\boldsymbol{W}\,\mathbf{log}\,\boldsymbol{W} - \boldsymbol{W})$. Using the expansion

$$\mathbf{log}\,\boldsymbol{W} = \sum_{i=1}^{\infty} \frac{(-1)^{i-1}}{i} (\boldsymbol{W} - \boldsymbol{I})^i,$$

we get

$$\boldsymbol{W}\,\mathbf{log}\,\boldsymbol{W} - \boldsymbol{W} = \sum_{i=2}^{\infty} \frac{(-1)^i}{i(i-1)} (\boldsymbol{W} - \boldsymbol{I})^i - \boldsymbol{I}.$$

Applying the shifted version of (A.1), i.e. $\nabla_{\boldsymbol{W}} \mathrm{tr}((\boldsymbol{W} - \boldsymbol{I})^k) = k((\boldsymbol{W} - \boldsymbol{I})^{k-1})^\top$, to all terms, the gradient is obtained as $\nabla_{\boldsymbol{W}} \mathrm{tr}(\boldsymbol{W}\,\mathbf{log}\,\boldsymbol{W} - \boldsymbol{W}) = (\mathbf{log}\,\boldsymbol{W})^\top$. When $\boldsymbol{W}$ is symmetric, then one can drop the transposition. Thus in in this case $\nabla_{\boldsymbol{W}} \mathrm{tr}(\mathbf{exp}\,\boldsymbol{W}) = \mathbf{exp}\,\boldsymbol{W}$.

Appendix B. Derivation of the MEG Update

In this appendix we derive parameter updates when the parameter must meet some linear constraints. One method is to incorporate such constraints into the strictly convex function F defining the Bregman divergence. The modified function F is then only defined when the constraints are met. The updates always have the simple form (3.2). However this method often leads to difficult forms of F and $\boldsymbol{f} = \nabla \mathrm{F}$. Here we choose the alternate method of keeping the linear constraints on the side. We begin by discussing how to enforce symmetry. Consider the following optimization problem, where $\boldsymbol{X}_t$ is an arbitrary matrix in $\mathbb{R}^{d\times d}$, $\boldsymbol{W}_t$ an arbitrary symmetric matrix in $\mathbb{R}^{d\times d}$ and $y_t \in \mathbb{R}$:

$$\begin{aligned} \boldsymbol{W}_{t+1} = \underset{\boldsymbol{W}}{\operatorname{argmin}} \qquad & \Delta_{\mathrm{F}}(\boldsymbol{W}, \boldsymbol{W}_t) + \eta L_t(\boldsymbol{W}) \\ \text{s.t. } \boldsymbol{W} = \boldsymbol{W}^\top. & \end{aligned}$$

We assume that $\nabla_{\boldsymbol{W}} L_t(\boldsymbol{W})$ is always a well defined matrix in $\mathbb{R}^{d\times d}$.

We introduce one Lagrange multiplier $\boldsymbol{\Gamma}_{i,j}$ for the each of the constraints $\boldsymbol{W}_{i,j} = \boldsymbol{W}_{j,i}$. This contributes the term $\boldsymbol{\Gamma}_{i,j}(\boldsymbol{W}_{i,j} - \boldsymbol{W}_{j,i})$ to the Lagrangian. In matrix form these constraints can be summarized as $\mathrm{tr}(\boldsymbol{\Gamma}(\boldsymbol{W}^\top - \boldsymbol{W})) = \mathrm{tr}((\boldsymbol{\Gamma}^\top - \boldsymbol{\Gamma})\boldsymbol{W})$. This gives us the Lagrangian

$$\mathcal{L}(\boldsymbol{W}, \boldsymbol{\Gamma}) = \Delta_{\mathrm{F}}(\boldsymbol{W}, \boldsymbol{W}_t) + \eta L_t(\boldsymbol{W}) + \mathrm{tr}((\boldsymbol{\Gamma}^\top - \boldsymbol{\Gamma})\boldsymbol{W}).$$

for $\boldsymbol{\Gamma} \in \mathbb{R}^{d\times d}$. Setting the gradient with respect to $\boldsymbol{W}$ to zero yields:

$$\boldsymbol{W}_{t+1} = \boldsymbol{f}^{-1}\left(\boldsymbol{f}(\boldsymbol{W}_t) - \eta \nabla_{\boldsymbol{W}} L_t(\boldsymbol{W}_{t+1}) - (\boldsymbol{\Gamma} - \boldsymbol{\Gamma}^\top)\right).$$

Since the objective is convex, it suffices to exhibit a choice of $\boldsymbol{\Gamma}$ such that the symmetry constraint is satisfied. Under the assumption that $\boldsymbol{f}$ and $\boldsymbol{f}^{-1}$ preserve symmetry, $\boldsymbol{\Gamma} = -\eta \nabla_{\boldsymbol{W}} L_t(\boldsymbol{W}_{t+1})/2$ achieves this and the update becomes (3.3):

$$\boldsymbol{W}_{t+1} = \boldsymbol{f}^{-1}\left(\boldsymbol{f}(\boldsymbol{W}_t) - \eta\, \mathbf{sym}(\nabla_{\boldsymbol{W}} L_t(\boldsymbol{W}_{t+1}))^\top)\right).$$

For the normalized case we still need to enforce the trace one constraint on $\boldsymbol{W}_{t+1}$. This adds a term $\delta(\mathrm{tr}(\boldsymbol{W}) - 1)$ to the Lagrangian and the update now has the form

$$\boldsymbol{W}_{t+1} = \mathbf{exp}\left(\mathbf{log}\, \boldsymbol{W}_t - \eta \nabla_{\boldsymbol{W}} L_t(\boldsymbol{W}_{t+1}) - (\boldsymbol{\Gamma} - \boldsymbol{\Gamma}^\top) - \delta \boldsymbol{I}\right).$$

Choosing $\boldsymbol{\Gamma} = -\eta \nabla_{\boldsymbol{W}} L_t(\boldsymbol{W}_{t+1})/2$ and

$$\delta = -\log\left(\mathrm{tr}(\mathbf{exp}(\mathbf{log}\, \boldsymbol{W}_t - \eta\, \mathbf{sym}(\nabla_{\boldsymbol{W}} L_t(\boldsymbol{W}_{t+1}))))\right)$$

enforces the symmetry and trace constraints and after approximating the gradient we arrive at the explicit MEG update (3.5).

Appendix C. Proof of Lemma 3.1

Let $\delta_t = -2\eta(\mathrm{tr}(\boldsymbol{X}\boldsymbol{W}_t) - y_t)$, then the right hand side of (3.6) can be reformulated as

$$\Delta_F(\boldsymbol{U}, \boldsymbol{W}_t) - \Delta_F(\boldsymbol{U}, \boldsymbol{W}_{t+1}) = \delta_t \mathrm{tr}(\boldsymbol{U}\boldsymbol{X}_t) - \log \mathrm{tr}(\mathbf{exp}(\mathbf{log}\, \boldsymbol{W}_t + \delta_t\, \mathbf{sym}(\boldsymbol{X}_t))).$$

Therefore, (3.6) is equivalent to $f \le 0$, where

$$f = \log \operatorname{tr}(\mathbf{exp}(\mathbf{log}\, \boldsymbol{W}_t + \delta_t\, \mathbf{sym}(\boldsymbol{X}_t))) - \delta_t \operatorname{tr}(\boldsymbol{U}\boldsymbol{X}_t) + a(y_t - \operatorname{tr}(\boldsymbol{W}_t\boldsymbol{X}_t))^2 - b(y_t - \operatorname{tr}(\boldsymbol{U}\boldsymbol{X}_t))^2.$$

Let us bound the first term. Due to Golden-Thompson inequality (2.3), we have

$$\operatorname{tr}\left(\mathbf{exp}(\mathbf{log}\, \boldsymbol{W}_t + \delta_t\, \mathbf{sym}(\boldsymbol{X}_t))\right) \le \operatorname{tr}\left(\boldsymbol{W}_t\, \mathbf{exp}(\delta_t\, \mathbf{sym}(\boldsymbol{X}_t))\right). \tag{C.1}$$

The right hand side can be rewritten as

$$\mathbf{exp}(\delta_t\, \mathbf{sym}(\boldsymbol{X}_t)) = \exp(r_0\delta_t)\, \mathbf{exp}(\delta_t(\mathbf{sym}(\boldsymbol{X}_t) - r_0\boldsymbol{I})).$$

Let r_0 be a lower bound of the eigenvalues of $\mathbf{sym}(\boldsymbol{X}_t)$. By assumption, the range of the eigenvalues of $\mathbf{sym}(\boldsymbol{X}_t)$ is at most r, i.e.

$$r_0\boldsymbol{I} \preceq \mathbf{sym}(\boldsymbol{X}_t) \preceq (r_0 + r)\boldsymbol{I}.$$

Thus $\mathbf{0} \preceq \boldsymbol{A} \preceq \boldsymbol{I}$, for $\boldsymbol{A} = (\mathbf{sym}(\boldsymbol{X}_t) - r_0\boldsymbol{I})/r$. Applying Lemma 2.1 with this choice of $\boldsymbol{A}$ and $\rho_1 = r\delta_t$, $\rho_2 = 0$, we obtain

$$\mathbf{exp}(\delta_t(\mathbf{sym}(\boldsymbol{X}_t) - r_0\boldsymbol{I})) \preceq \boldsymbol{I} - \frac{\mathbf{sym}(\boldsymbol{X}_t) - r_0\boldsymbol{I}}{r}(1 - \exp(r\delta_t)).$$

Since $\boldsymbol{W}_t$ is symmetric positive definite and both sides of the above inequality are symmetric, we can apply Lemma 2.2 by pre-multiplying the inequality by $\boldsymbol{W}_t$ and taking a trace of both sides:

$$\operatorname{tr}\left(\boldsymbol{W}_t\, \mathbf{exp}(\delta_t\, \mathbf{sym}(\boldsymbol{X}_t))\right) \le \exp(r_0\delta_t)\left(1 - \frac{\operatorname{tr}(\boldsymbol{W}_t\boldsymbol{X}_t) - r_0}{r}(1 - \exp(r\delta_t))\right).$$

Note that we used the assumption that $\operatorname{tr}(\boldsymbol{W}_t) = 1$. The above gives an upper bound on the right hand side of inequality (C.1) We now plug this upper bound into the first term of f and obtain $f \le g$, where

$$\begin{aligned} g = \quad & r_0\delta_t + \log(1 - \tfrac{\operatorname{tr}(\boldsymbol{W}_t\boldsymbol{X}_t) - r_0}{r}(1 - \exp(r\delta_t))) - \operatorname{tr}(\boldsymbol{U}\boldsymbol{X}_t)\delta_t \\ & + a(y_t - \operatorname{tr}(\boldsymbol{W}_t\boldsymbol{X}_t))^2 - b(y_t - \operatorname{tr}(\boldsymbol{U}\boldsymbol{X}_t))^2. \end{aligned} \tag{C.2}$$

Let us define $z = \operatorname{tr}(\boldsymbol{U}\boldsymbol{X}_t)$ and maximize the upper bound (C.2) with respect to z. Solving $\frac{\partial g}{\partial z} = 0$, we have $z = y_t - \delta_t/(2b) = y_t + \eta(\operatorname{tr}(\boldsymbol{X}_t\boldsymbol{W}_t) - y_t)/b$. Substituting this into (C.2), we have the upper bound $g \le h$ where

$$\begin{aligned} h = \quad & 2\eta r_0(y_t - \operatorname{tr}(\boldsymbol{X}_t\boldsymbol{W}_t)) + \log\left(1 - \tfrac{\operatorname{tr}(\boldsymbol{X}_t\boldsymbol{W}_t) - r_0}{r}(1 - \mathbf{exp}(2\eta r(y - \operatorname{tr}(X_t\boldsymbol{W}_t))))\right) \\ & -2\eta y_t(y_t - \operatorname{tr}(\boldsymbol{X}_t\boldsymbol{W}_t)) + (a + \tfrac{\eta^2}{b}(y - \operatorname{tr}(\boldsymbol{X}_t\boldsymbol{W}_t))^2. \end{aligned}$$

We now upper bound the second term using the inequality $\log(1 - p(1 - \exp q)) \le pq + q^2/8$, for $0 \le q \le 1$ and $q \in \mathbb{R}$ (Helmbold et al., 1997):

$$h \le \frac{(y_t - \operatorname{tr}(\boldsymbol{X}_t\boldsymbol{W}_t))^2}{2b}((2 + r^2b)\eta^2 - 4b\eta + 2ab).$$

It remains to show $q = (2 + r^2b)\eta^2 - 4b\eta + 2ab \le 0$. We easily see that q is minimized for $\eta = 2b/(2 + r^2b)$ and that for this value of η we have $q \le 0$ if and only if $a \le 2b/(2 + r^2b)$.

Appendix D. Derivation of the DefiniteBoost Dual Problem

For the sake of brevity we assume that the primal problem has one inequality constraint (note that (4.1) has multiple constraints):

$$\begin{aligned} \boldsymbol{W}^* = \operatorname*{argmin}_{\boldsymbol{W}} \quad & \operatorname{tr}(\boldsymbol{W}(\mathbf{log}\,\boldsymbol{W} - \mathbf{log}\,\boldsymbol{W}_1) + \boldsymbol{W}_1 - \boldsymbol{W} \\ \text{s.t.} \quad & \operatorname{tr}(\boldsymbol{W}\boldsymbol{C}) \leq 0 \\ & \operatorname{tr}(\boldsymbol{W}) = 1 \\ & \boldsymbol{W} = \boldsymbol{W}^\top. \end{aligned}$$

Following Appendix B we arrive at the Lagrangian

$$\begin{aligned} \mathcal{L}(\boldsymbol{W}, \alpha, \beta, \boldsymbol{\Gamma}) \quad := \quad & \operatorname{tr}(\boldsymbol{W}(\mathbf{log}\,\boldsymbol{W} - \mathbf{log}\,\boldsymbol{W}_1) + \boldsymbol{W}_1 - \boldsymbol{W} + \alpha \operatorname{tr}(\boldsymbol{W}\boldsymbol{C}) + \\ & + \beta\,(\operatorname{tr}(\boldsymbol{W}) - 1) + \operatorname{tr}((\boldsymbol{\Gamma}^\top - \boldsymbol{\Gamma})\boldsymbol{W}), \end{aligned} \tag{D.1}$$

which is minimized w.r.t. $\boldsymbol{W}$ and maximized w.r.t. $\alpha \geq 0$, $\beta \in \mathbb{R}$ and $\boldsymbol{\Gamma} \in \mathbb{R}^{d\times d}$. Setting the gradient w.r.t. $\boldsymbol{W}$ to zero we obtain

$$\begin{aligned} \boldsymbol{W}^* \quad &= \quad \mathbf{exp}(\mathbf{log}\,\boldsymbol{W}_1 - \alpha\boldsymbol{C} - \beta\boldsymbol{I} - (\boldsymbol{\Gamma} - \boldsymbol{\Gamma}^\top) \\ &= \quad \exp(-\beta)\,\mathbf{exp}(\mathbf{log}\,\boldsymbol{W}_1 - \alpha\boldsymbol{C} - (\boldsymbol{\Gamma} - \boldsymbol{\Gamma}^\top). \end{aligned}$$

We now enforce the symmetry constraint, giving us $\boldsymbol{\Gamma} = -\alpha(\boldsymbol{C} - \boldsymbol{C}^\top)/2$, and plug this choice into the above

$$\boldsymbol{W}^* = \exp(-\beta)\,\mathbf{exp}(\mathbf{log}\,\boldsymbol{W}_1 - \alpha\,\mathbf{sym}(\boldsymbol{C})).$$

Similarly, $\beta = \log\operatorname{tr}(\mathbf{exp}(\mathbf{log}\,\boldsymbol{W}_1 - \alpha\ \mathbf{sym}(\boldsymbol{C})))$ enforces the trace constraint. Now

$$\boldsymbol{W}^* = \mathbf{exp}(\mathbf{log}\,\boldsymbol{W}_1 - \alpha\,\mathbf{sym}(\boldsymbol{C})/Z(\alpha),$$

where $Z(\alpha) = -\log\operatorname{tr}(\mathbf{exp}(\mathbf{log}\,\boldsymbol{W}_1 - \alpha\,\mathbf{sym}(\boldsymbol{C})))$. Plugging $\boldsymbol{W}^*$ into in the Lagrangian, we obtain the dual optimization problem for one constraint:

$$\alpha^* = \operatorname*{argmax}_{\alpha\geq 0} \ -\log \mathrm{Z}_t(\alpha).$$

One can easily verify that the solution of the problem with n constraints is of the form:

$$\boldsymbol{\alpha}^* = \operatorname*{argmax}_{\boldsymbol{\alpha}\geq\mathbf{0}} \ -\log\operatorname{tr}(\mathbf{exp}(\mathbf{log}\,\boldsymbol{W}_1 - \sum_{j=1}^{n} \alpha_j\,\mathbf{sym}(\boldsymbol{C}_j))).$$

Appendix E. Proof of Theorem 4.1

Recall the definition of the normalization factor $\mathrm{Z}_t(\alpha) = \operatorname{tr}(\mathbf{exp}(\mathbf{log}\,\boldsymbol{W}_t - \alpha\,\mathbf{sym}(\boldsymbol{C}_{j_t})))$ of Definite-Boost. By the Golden-Thompson inequality,

$$\mathrm{Z}_t(\alpha) \leq \operatorname{tr}(\boldsymbol{W}_t\,\mathbf{exp}(-\alpha\,\mathbf{sym}(\boldsymbol{C}_{j_t}))). \tag{E.1}$$

Similarly to the proof of Lemma 3.1, we now upper bound the right hand side of this inequality by applying lemmas 2.1 and 2.2. We choose $\boldsymbol{A}$ as $(\lambda_t^{\min}\boldsymbol{I}+\mathbf{sym}(\boldsymbol{C}_{j_t}))/(\lambda_t^{\max}+\lambda_t^{\min})$. Then $\mathbf{sym}(\boldsymbol{C}_{j_t})$ can be expressed as $\lambda_t^{\max}\boldsymbol{A}-\lambda_t^{\min}(\boldsymbol{I}-\boldsymbol{A})$ and $\mathbf{0}\preceq\boldsymbol{A}\preceq\boldsymbol{I}$. Thus by Lemma 2.1,

$$\mathbf{exp}(-\alpha\,\mathbf{sym}(\boldsymbol{C}_{j_t}))\preceq\exp(-\alpha\lambda_t^{\max})\boldsymbol{A}+\exp(\alpha\lambda_t^{\min})(\boldsymbol{I}-\boldsymbol{A}).$$

Since $\boldsymbol{W}_t$ is positive definite and both sides of the above inequality are symmetric, we can apply Lemma 2.2 by multiplying this inequality by $\boldsymbol{W}_t$ and taking a trace of both sides:

$$\mathrm{tr}(\boldsymbol{W}_t\,\mathbf{exp}(-\alpha\,\mathbf{sym}(\boldsymbol{C}_{j_t})))\leq\exp(-\alpha\lambda_t^{\max})\mathrm{tr}(\boldsymbol{W}_t\boldsymbol{A})+\exp(\alpha\lambda_t^{\min})\mathrm{tr}\left(\boldsymbol{W}_t(\boldsymbol{I}-\boldsymbol{A})\right).$$

By expanding $\boldsymbol{A}$ and using the shorthand $r_t=\mathrm{tr}(\boldsymbol{W}_t\boldsymbol{C}_{j_t})$, we obtain

$$\mathrm{Z}_t(\alpha)\leq\exp(-\alpha\lambda_t^{\max})\frac{\lambda_t^{\min}+r_t}{\lambda_t^{\max}+\lambda_t^{\min}}+\exp(\alpha\lambda^{\min})\frac{\lambda_t^{\max}-r_t}{\lambda_t^{\max}+\lambda_t^{\min}}.$$

We now choose the α that minimizes the right hand side of the above inequality (which is the $\widehat{\alpha}_t$ given in equation (4.6)). With this choice, the inequality becomes

$$\mathrm{Z}_t(\widehat{\alpha}_t)\leq(1-\frac{r_t}{\lambda_t^{\max}})^{\frac{\lambda_t^{\max}}{\lambda_t^{\max}+\lambda_t^{\min}}}(1+\frac{r_t}{\lambda_t^{\min}})^{\frac{\lambda_t^{\min}}{\lambda_t^{\max}+\lambda_t^{\min}}}. \tag{E.2}$$

Applying the update rule (4.5) T times, we have

$$\boldsymbol{W}_{T+1}=\frac{\mathbf{exp}(\mathbf{log}\,\boldsymbol{W}_1-\sum_{t=1}^T\widehat{\alpha}_t\,\mathbf{sym}(\boldsymbol{C}_{j_t}))}{\prod_t\mathrm{Z}_t(\widehat{\alpha}_t)}.$$

Taking the trace of both sides and rearranging terms, we get

$$\mathrm{tr}\left(\mathbf{exp}(\mathbf{log}\,\boldsymbol{W}_1-\sum_{t=1}^T\widehat{\alpha}_t\,\mathbf{sym}(\boldsymbol{C}_{j_t}))\right)=\prod_{t=1}^T\mathrm{Z}_t(\widehat{\alpha}_t).$$

By using the bound (E.2) for each $\mathrm{Z}_t(\alpha_t)$, the inequality of the theorem readily follows.

References

O. E. Barndorff-Nielsen, R. D. Gill, and P. E. Jupp. On quantum statistical inference. *J. R. Statist. Soc. B*, 65(4):775–816, 2003.

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

L. M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Physics*, 7:200–217, 1967.

Y. Censor and A. Lent. An iterative row-action method for interval convex programming. *Journal of Optimization Theory and Applications*, 34(3):321–353, July 1981.

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

S. Golden. Lower bounds for the Helmholtz function. *Phys. Rev.*, 137:B1127–B1128, 1965.

D. Helmbold, R. E. Schapire, Y. Singer, and M. K. Warmuth. A comparison of new and old algorithms for amixture estimation problem. *Machine Learning*, 27(1):97–119, 1997.

P. J. Huber. *Robust Statistics*. John Wiley and Sons, New York, 1981.

J. Kivinen and M. K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, 1997.

J. Kivinen and M. K. Warmuth. Boosting as entropy projection. In *Proceedings of the 12th Annual Conference on Computational Learning Theory*, pages 134–144. ACM Press, New York, NY, 1999.

J. Kivinen and M. K. Warmuth. Relative loss bounds for multidimensional regression problems. *Machine Learning*, 45(3):301–329, 2001.

J. Lafferty. Additive models, boosting, and inference for generalized divergences. In *Proceedings of the 12th Annual Conference on Computational Learning Theory*, pages 125–133. ACM Press, New York, NY, 1999.

N. Littlestone. Learning when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

N. Littlestone. *Mistake Bounds and Logarithmic Linear-threshold Learning Algorithms*. PhD thesis, Technical Report UCSC-CRL-89-11, University of California, Santa Cruz, 1989.

N. Littlestone, P. M. Long, and M. K. Warmuth. On-line learning of linear functions. Technical Report UCSC-CRL-91-29, University of California, Santa Cruz, May 1992.

M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

G. Rätsch. *Robust Boosting via Convex Optimization*. PhD thesis, University of Potsdam, Potsdam, Germany, October 2001.

G. Rätsch and M. K. Warmuth. Maximizing the margin with boosting. In *Proceedings of the 15th Annual Conference on Computational Learning Theory*, pages 319–333. Springer, Sydney, Australia, 2002.

G. Rätsch and M. K. Warmuth. Efficient margin maximization with boosting. submitted to Journal of Machine Learning Research, 2005.

R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37:297–336, 1999.

B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

S. Shai-Shwartz, Y. Singer, and A. Y. Ng. Online and batch learning of pseudo-metrics. In C. E. Brodley, editor, *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004)*. ACM Press, New York, NY, 2004.

Y. Singer and M. K. Warmuth. Batch and on-line parameter estimation of Gaussian mixtures based on the joint entropy. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11 (NIPS'98)*, pages 578–584. MIT Press, 1999.

I. W. Tsang and J. T. Kwok. Distance metric learning with kernels. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN'03)*, pages 126–129. Springer Verlag, New York, NY, 2003.

K. Tsuda, S. Akaho, and K. Asai. The em algorithm for kernel matrix completion with auxiliary data. *Journal of Machine Learning Research*, 4:67–81, May 2003.

K. Tsuda and W. S. Noble. Learning kernels from biological networks by maximizing entropy. *Bioinformatics*, 20(Suppl. 1):i326–i333, 2004.

E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning with application to clustering with side-information. In S. Thrun S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 505–512. MIT Press, Cambridge, MA, 2003.

Journal of Machine Learning Research 6 (2005) 1019–1041 Submitted 11/04; Revised 3/05; Published 7/05

Gaussian Processes for Ordinal Regression

Wei Chu CHUWEI@GATSBY.UCL.AC.UK
Zoubin Ghahramani ZOUBIN@GATSBY.UCL.AC.UK
Gatsby Computational Neuroscience Unit
University College London
London, WC1N 3AR, UK

Editor: Christopher K. I. Williams

Abstract

We present a probabilistic kernel approach to ordinal regression based on Gaussian processes. A threshold model that generalizes the *probit* function is used as the likelihood function for ordinal variables. Two inference techniques, based on the Laplace approximation and the expectation propagation algorithm respectively, are derived for hyperparameter learning and model selection. We compare these two Gaussian process approaches with a previous ordinal regression method based on support vector machines on some benchmark and real-world data sets, including applications of ordinal regression to collaborative filtering and gene expression analysis. Experimental results on these data sets verify the usefulness of our approach.

Keywords: Gaussian processes, ordinal regression, approximate Bayesian inference, collaborative filtering, gene expression analysis, feature selection

1. Introduction

Practical applications of supervised learning frequently involve situations exhibiting an order among the different categories, e.g. a teacher always rates his/her students by giving grades on their overall performance. In contrast to metric regression problems, the grades are usually discrete and finite. These grades are also different from the class labels in classification problems due to the existence of ranking information. For example, grade labels have the ordering $F < D < C < B < A$. This is a learning task of predicting variables of ordinal scale, a setting bridging between metric regression and classification referred to as *ranking learning* or *ordinal regression*.

There is some literature about ordinal regression in the domain of machine learning. Kramer et al. (2001) investigated the use of a regression tree learner by mapping the ordinal variables into numeric values. However there might be no principled way of devising an appropriate mapping function. Frank and Hall (2001) converted an ordinal regression problem into nested binary classification problems that encode the ordering of the original ranks, and then the results of standard binary classifiers can be organized for prediction. Har-Peled et al. (2003) proposed a constraint classification approach for ranking problems based on binary classifiers. Cohen et al. (1999) considered general ranking problems in the form of preference judgements. Herbrich et al. (2000) applied the principle of Structural Risk Minimization (Vapnik, 1995) to ordinal regression leading to a new distribution-independent learning algorithm based on a loss function between pairs of ranks. Shashua and Levin (2003) generalized the formulation of support vector machines to ordinal regression and the numerical results they presented shows a significant improvement on the performance compared with the on-line algorithm proposed by Crammer and Singer (2002).

In the statistics literature, most of the approaches are based on generalized linear models (McCullagh and Nelder, 1983). The cumulative model (McCullagh, 1980) is well-known in classical statistical approaches for ordinal regression, in which they rely on a specific distributional assumption on the unobservable latent variables and a stochastic ordering of the input space. Johnson and Albert (1999) described Bayesian inference on parametric models for ordinal data using sampling techniques. Tutz (2003) presented a general framework for semiparametric models that extends generalized additive models (Hastie and Tibshirani, 1990) by incorporating nonparametric parts. The nonparametric components of the regression model are fitted by maximizing penalized log likelihood, and model selection is carried out using AIC.

Gaussian processes (O'Hagan, 1978; Neal, 1997) have provided a promising non-parametric Bayesian approach to metric regression (Williams and Rasmussen, 1996) and classification problems (Williams and Barber, 1998). The important advantage of Gaussian process models (GPs) over other non-Bayesian models is the explicit probabilistic formulation. This not only provides probabilistic predictions but also gives the ability to infer model parameters such as those that control the kernel shape and the noise level. The GPs are also different from the semiparametric approach of Tutz (2003) in several ways. First, the additive models (Fahrmeir and Tutz, 2001) are defined by functions in each input dimension, whereas the GPs can have more general non-additive covariance functions; second, the kernel trick allows to use infinite basis function expansions; third, the GPs perform Bayesian inference in the space of the latent functions.

In this paper, we present a probabilistic kernel approach to ordinal regression in Gaussian processes. We impose a Gaussian process prior distribution on the latent functions, and employ an appropriate likelihood function for ordinal variables which can be regarded as a generalization of the *probit* function. Two Bayesian inference techniques are applied to implement model adaptation by using the Laplace approximation (MacKay, 1992) and the expectation propagation (Minka, 2001) respectively. Comparisons of the generalization performance against the support vector approach (Shashua and Levin, 2003) on some benchmark and real-world data sets, such as movie ranking and gene expression analysis, verify the usefulness of this approach.

The paper is organized as follows: in Section 2, we describe the Bayesian framework in Gaussian processes for ordinal regression; in Section 3, we discuss the Bayesian techniques for hyperparameter inference; in Section 4, we present the predictive distribution for probabilistic prediction; in Section 5, we give some extensive discussion on these techniques; in Section 6, we report the results of numerical experiments on some benchmark and real-world data sets; we conclude this paper in Section 7.

2. Bayesian Framework

Consider a data set composed of n samples. Each of the samples is a pair of input vector $x_i \in \mathcal{R}^d$ and the corresponding target $y_i \in \mathcal{Y}$ where $\mathcal{Y}$ is a finite set of r ordered categories. Without loss of generality, these categories are denoted as consecutive integers $\mathcal{Y} = \{1, 2, \dots, r\}$ that keep the known ordering information. The main idea is to assume an unobservable latent function $f(x_i) \in \mathcal{R}$ associated with x_i in a Gaussian process, and the ordinal variable y_i dependent on the latent function $f(x_i)$ by modelling the ranks as intervals on the real line. A Bayesian framework is described with more details in the following.

2.1 Gaussian Process Prior

The latent functions $\{f(x_i)\}$ are usually assumed as the realizations of random variables indexed by their input vectors in a zero-mean Gaussian process. The Gaussian process can then be fully specified by giving the covariance matrix for any finite set of zero-mean random variables $\{f(x_i)\}$. The covariance between the functions corresponding to the inputs x_i and x_j can be defined by Mercer kernel functions (Wahba, 1990; Schölkopf and Smola, 2001), e.g. Gaussian kernel which is defined as

$$\mathrm{Cov}[f(x_i), f(x_j)] = \mathcal{K}(x_i, x_j) = \exp\left(-\frac{\kappa}{2}\sum_{\varsigma=1}^{d}(x_i^\varsigma - x_j^\varsigma)^2\right) \tag{1}$$

where $\kappa > 0$ and x_i^ς denotes the ς-th element of x_i.[1] Thus, the prior probability of these latent functions $\{f(x_i)\}$ is a multivariate Gaussian

$$\mathcal{P}(f) = \frac{1}{\mathcal{Z}_f}\exp\left(-\frac{1}{2}f^T\Sigma^{-1}f\right) \tag{2}$$

where $f = [f(x_1), f(x_2), \ldots, f(x_n)]^T$, $\mathcal{Z}_f = (2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}$, and Σ is the $n \times n$ covariance matrix whose ij-th element is defined as in (1).

2.2 Likelihood for Ordinal Variables

The likelihood is the joint probability of observing the ordinal variables given the latent functions, denoted as $\mathcal{P}(\mathcal{D}|f)$ where $\mathcal{D}$ denotes the target set $\{y_i\}$. Generally, the likelihood can be evaluated as a product of the likelihood function on individual observation:

$$\mathcal{P}(\mathcal{D}|f) = \prod_{i=1}^{n}\mathcal{P}(y_i|f(x_i)) \tag{3}$$

where the likelihood function $\mathcal{P}(y_i|f(x_i))$ could be intuitively defined as

$$\mathcal{P}_{\text{ideal}}(y_i|f(x_i)) = \begin{cases} 1 & \text{if } b_{y_i-1} < f(x_i) \le b_{y_i}, \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where $b_0 = -\infty$ and $b_r = +\infty$ are defined subsidiarily, $b_1 \in \mathcal{R}$ and the other threshold variables can be further defined as $b_j = b_1 + \sum_{\iota=2}^{j}\Delta_\iota$ with positive padding variables Δ_ι and $\iota = 2, \ldots, r-1$. The role of $b_1 < b_2 < \ldots < b_{r-1}$ is to divide the real line into r contiguous intervals; these intervals map the real function value $f(x_i)$ into the discrete variable y_i while enforcing the ordinal constraints. The likelihood function (4) is used for ideally noise-free cases. In the presence of noise from inputs or targets, we may explicitly assume that the latent functions are contaminated by a Gaussian noise with zero mean and unknown variance σ^2.[2] $\mathcal{N}(\delta; \mu, \sigma^2)$ is used to denote a Gaussian random variable δ with mean μ and variance σ^2 henceforth. Then the ordinal likelihood function becomes

$$\mathcal{P}(y_i|f(x_i)) = \int \mathcal{P}_{\text{ideal}}(y_i|f(x_i)+\delta_i)\mathcal{N}(\delta_i; 0, \sigma^2)\,d\delta_i = \Phi\left(z_1^i\right) - \Phi\left(z_2^i\right) \tag{5}$$

1. Other Mercer kernel functions, such as polynomial kernels and spline kernels etc., can also be used in the covariance function.
2. In principle, any distribution rather than a Gaussian can be assumed for the noise on the latent functions.

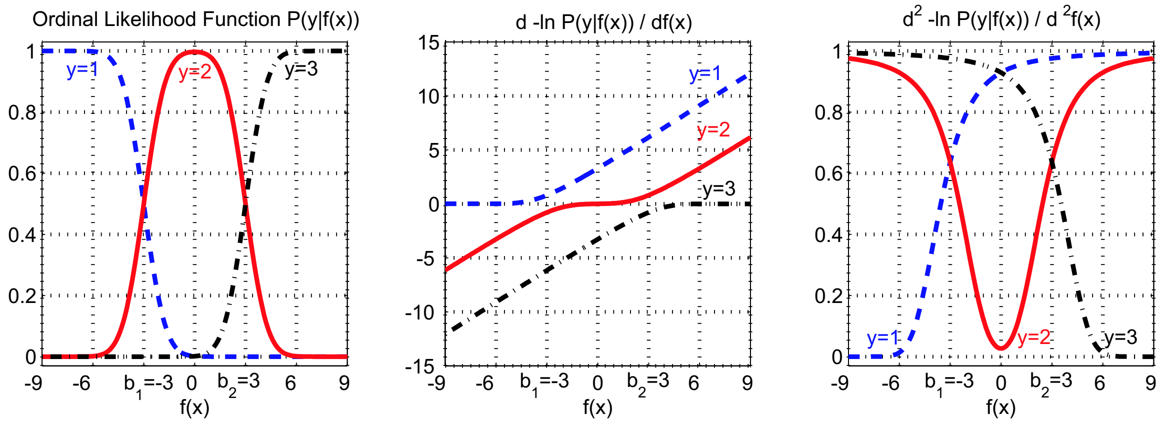


Figure 1: The graph of the likelihood function for an ordinal regression problem with $r = 3$, along with the first and second order derivatives of the loss function (negative logarithm of the likelihood function), where the noise variance $\sigma^2 = 1$, and the two thresholds are $b_1 = -3$ and $b_2 = +3$.

where $z_1^i = \frac{b_{y_i} - f(x_i)}{\sigma}$, $z_2^i = \frac{b_{y_i-1} - f(x_i)}{\sigma}$, and $\Phi(z) = \int_{-\infty}^{z} \mathcal{N}(\varsigma; 0, 1) d\varsigma$. Note that binary classification is a special case of ordinal regression when $r = 2$, and in this case the likelihood function (5) becomes the *probit* function. The quantity $-\ln \mathcal{P}(y_i | f(x_i))$ is usually referred to as the loss function $\ell(y_i, f(x_i))$. The derivatives of the loss function with respect to $f(x_i)$ are needed in some approximate Bayesian inference methods. The first order derivative of the loss function can be written as

$$\frac{\partial \ell(y_i, f(x_i))}{\partial f(x_i)} = \frac{1}{\sigma} \frac{\mathcal{N}(z_1^i; 0, 1) - \mathcal{N}(z_2^i; 0, 1)}{\Phi(z_1^i) - \Phi(z_2^i)} \tag{6}$$

and the second order derivative can be given as

$$\frac{\partial^2 \ell(y_i, f(x_i))}{\partial^2 f(x_i)} = \frac{1}{\sigma^2} \left(\frac{\mathcal{N}(z_1^i; 0, 1) - \mathcal{N}(z_2^i; 0, 1)}{\Phi(z_1^i) - \Phi(z_2^i)} \right)^2 + \frac{1}{\sigma^2} \frac{z_1^i \mathcal{N}(z_1^i; 0, 1) - z_2^i \mathcal{N}(z_2^i; 0, 1)}{\Phi(z_1^i) - \Phi(z_2^i)}. \tag{7}$$

We present graphs of the ordinal likelihood function (5) and the derivatives of the loss function in Figure 1 as an illustration. Note that the first order derivative (6) is a monotonically increasing function of $f(x_i)$, and the second order derivative (7) is always a positive value between 0 and $\frac{1}{\sigma^2}$. Given the facts that $\mathcal{P}_{\text{ideal}}(y_i | f(x_i) + \delta_i)$ is log-concave in $(f(x_i), \delta_i)$ and $\mathcal{N}(\delta_i; 0, \sigma^2)$ is also log-concave, as pointed out by Pratt (1981), the convexity of the loss function follows, because the integral of a log-concave function with respect to some of its arguments is a log-concave function of its remaining arguments (Brascamp and Lieb, 1976, Cor. 3.5).

2.3 Posterior Probability

Based on Bayes' theorem, the posterior probability can then be written as

$$\mathcal{P}(f | \mathcal{D}) = \frac{1}{\mathcal{P}(\mathcal{D})} \prod_{i=1}^{n} \mathcal{P}(y_i | f(x_i)) \mathcal{P}(f) \tag{8}$$

where the prior probability $\mathcal{P}(f)$ is defined as in (2), the likelihood function $\mathcal{P}(y_i | f(x_i))$ is defined as in (5), and $\mathcal{P}(\mathcal{D}) = \int \mathcal{P}(\mathcal{D} | f) \mathcal{P}(f) df$.

The Bayesian framework we described above is conditional on the model parameters including the kernel parameters κ in the covariance function (1) that control the kernel shape, the threshold parameters $\{b_1, \Delta_2, \ldots, \Delta_{r-1}\}$ and the noise level σ in the likelihood function (5). All these parameters can be collected into θ, which is the hyperparameter vector. The normalization factor $\mathcal{P}(\mathcal{D})$ in (8), more exactly $\mathcal{P}(\mathcal{D}|\theta)$, is known as the evidence for θ, a yardstick for model selection. In the next section, we discuss techniques for hyperparameter learning.

3. Model Adaptation

In a full Bayesian treatment, the hyperparameters θ must be integrated over the θ-space. Monte Carlo methods (Neal, 1997) can be adopted here to approximate the integral effectively. However these might be prohibitively expensive to use in practice. Alternatively, we consider model selection by determining an optimal setting for θ. The optimal values of hyperparameters θ can be *simply* inferred by maximizing the posterior probability $\mathcal{P}(\theta|\mathcal{D})$, where $\mathcal{P}(\theta|\mathcal{D}) \propto \mathcal{P}(\mathcal{D}|\theta)\mathcal{P}(\theta)$. The prior distribution on the hyperparameters $\mathcal{P}(\theta)$ can be specified by domain knowledge, or alternatively some vague uninformative distribution. The evidence is given by a high dimensional integral, $\mathcal{P}(\mathcal{D}|\theta) = \int \mathcal{P}(\mathcal{D}|f)\mathcal{P}(f)\,df$. A popular idea for computing the evidence is to approximate the posterior distribution $\mathcal{P}(f|\mathcal{D})$ as a Gaussian, and then the evidence can be calculated by an explicit formula (MacKay, 1992; Csató et al., 2000; Minka, 2001). In this section, we describe two Bayesian techniques for model adaptation by using the Laplace approximation and the expectation propagation respectively.

3.1 MAP Approach with Laplace Approximation

The evidence can be calculated analytically after applying the Laplace approximation at the maximum a posteriori (MAP) estimate, and gradient-based optimization methods can then be used to infer the optimal hyperparameters by maximizing the evidence. The MAP estimate on the latent functions is referred to $f_{\text{MAP}} = \arg\max_f \mathcal{P}(f|\mathcal{D})$, which is equivalent to the minimizer of negative logarithm of $\mathcal{P}(f|\mathcal{D})$, i.e.

$$\mathcal{S}(f) = \sum_{i=1}^{n} \ell(y_i, f(x_i)) + \frac{1}{2} f^T \Sigma^{-1} f \tag{9}$$

where $\ell(y_i, f(x_i)) = -\ln \mathcal{P}(y_i|f(x_i))$ is known as the loss function. Note that $\frac{\partial^2 \mathcal{S}(f)}{\partial f \partial f^T} = \Sigma^{-1} + \Lambda$ is a positive definite matrix, where Λ is a diagonal matrix whose *ii*-th entry is $\frac{\partial^2 \ell(y_i, f(x_i))}{\partial^2 f(x_i)}$ given as in (7). Thus, this is a convex programming problem with a unique solution.[3] The Laplace approximation of $\mathcal{S}(f)$ refers to carrying out the Taylor expansion at the MAP point and retaining the terms up to the second order (MacKay, 1992). Since the first order derivative with respect to f vanishes at f_{MAP}, $\mathcal{S}(f)$ can also be written as

$$\mathcal{S}(f) \approx \mathcal{S}(f_{\text{MAP}}) + \frac{1}{2}(f - f_{\text{MAP}})^T \left(\Sigma^{-1} + \Lambda_{\text{MAP}}\right)(f - f_{\text{MAP}}) \tag{10}$$

where Λ_{MAP} denotes the matrix Λ at the MAP estimate. This is equivalent to approximating the posterior distribution $\mathcal{P}(f|\mathcal{D})$ as a Gaussian distribution centered on f_{MAP} with the covariance matrix

3. The Newton-Raphson formula can be used to find the solution for simple cases.

$(\Sigma^{-1}+\Lambda_{\text{MAP}})^{-1}$, i.e. $\mathcal{P}(f|\mathcal{D}) \approx \mathcal{N}(f; f_{\text{MAP}}, (\Sigma^{-1}+\Lambda_{\text{MAP}})^{-1})$. Using the Laplace approximation (10) and $\mathcal{Z}_f$ defined as in (2), the evidence can be computed analytically as follows

$$\mathcal{P}(\mathcal{D}|\theta) = \frac{1}{\mathcal{Z}_f}\int \exp(-\mathcal{S}(f))\,df \approx \exp(-\mathcal{S}(f_{\text{MAP}}))|\mathbf{I}+\Sigma\Lambda_{\text{MAP}}|^{-\frac{1}{2}} \tag{11}$$

where $\mathbf{I}$ is an $n\times n$ identity matrix. The gradients of the logarithm of the evidence (11) with respect to the hyperparameters θ can be derived analytically. Then gradient-based optimization methods can be employed to search for the maximizer of the evidence. Refer to Appendix A for the detailed gradient formulae and the outline of our algorithm for model adaptation.

3.2 Expectation Propagation with Variational Methods

The expectation propagation algorithm (EP) is an approximate Bayesian inference method (Minka, 2001), which can be regarded as an extension of assumed-density-filter (ADF). The EP algorithm has been applied in Gaussian process classification along with variational methods for model selection (Seeger, 2002; Kim and Ghahramani, 2003). In the setting of Gaussian processes, EP attempts to approximate $\mathcal{P}(f|\mathcal{D})$ as a product distribution in the form of $Q(f)=\prod_{i=1}^{n}\tilde{t}_i(f(x_i))\mathcal{P}(f)$ where $\tilde{t}_i(f(x_i)) = s_i\exp(-\frac{1}{2}p_i(f(x_i)-m_i)^2)$. The parameters $\{s_i,m_i,p_i\}$ in $\{\tilde{t}_i\}$ are successively optimized by minimizing the following Kullback-Leibler divergence,

$$\tilde{t}_i^{\text{new}} = \arg\min_{\tilde{t}_i}\mathbf{KL}\left(\frac{Q(f)}{\tilde{t}_i^{\text{old}}}\mathcal{P}(y_i|f(x_i))\,\middle\|\,\frac{Q(f)}{\tilde{t}_i^{\text{old}}}\tilde{t}_i\right). \tag{12}$$

Since $Q(f)$ is in the exponential family, this minimization can be simply solved by moment matching up to the second order. A detailed updating scheme can be found in Appendix B. At the equilibrium of $Q(f)$, we obtain an approximate posterior distribution as $\mathcal{P}(f|\mathcal{D}) \approx \mathcal{N}(f;(\Sigma^{-1}+\Pi)^{-1}\Pi m,(\Sigma^{-1}+\Pi)^{-1})$ where Π is a diagonal matrix whose ii-th entry is p_i and $m=[m_1,m_2,\ldots,m_n]^T$.

Variational methods can be used to optimize the hyperparameters θ by maximizing the lower bound on the logarithm of the evidence. By applying Jensen's inequality, we have

$$\begin{aligned}\log\mathcal{P}(\mathcal{D}|\theta) &= \log\int\tfrac{\mathcal{P}(\mathcal{D}|f)\mathcal{P}(f)}{Q(f)}Q(f)df \geq \int Q(f)\log\tfrac{\mathcal{P}(\mathcal{D}|f)\mathcal{P}(f)}{Q(f)}df\\ &= \int Q(f)\log\mathcal{P}(\mathcal{D}|f)df+\int Q(f)\log\mathcal{P}(f)df-\int Q(f)\log Q(f)df = \mathcal{F}(\theta).\end{aligned} \tag{13}$$

The lower bound $\mathcal{F}(\theta)$ can be written as an explicit expression at the equilibrium of $Q(f)$, and then the gradients with respect to θ can be derived by neglecting the possible dependency of $Q(f)$ on θ. The detailed formulation can be found in Appendix C.

4. Prediction

We have described two techniques, the MAP approach and the EP approach, to infer the optimal model. At the optimal hyperparameters we inferred, denoted as θ^*, let us take a test case x for which the target y_x is unknown. The latent variable $f(x)$ and the column vector f containing the n zero-mean random variables $\{f(x_i)\}_{i=1}^n$ have the prior joint multivariate Gaussian distribution, i.e.

$$\begin{bmatrix} f \\ f(x)\end{bmatrix} \sim \mathcal{N}\left[\begin{pmatrix}0\\0\end{pmatrix},\begin{pmatrix}\Sigma & \boldsymbol{k}\\ \boldsymbol{k}^T & \mathcal{K}(x,x)\end{pmatrix}\right]$$

where $\boldsymbol{k} = [\mathcal{K}(x,x_1), \mathcal{K}(x,x_2), \ldots, \mathcal{K}(x,x_n)]^T$. The conditional distribution of $f(x)$ given f is a Gaussian too, denoted as $\mathcal{P}(f(x)|f,\theta^*)$ with mean $f^T\Sigma^{-1}\boldsymbol{k}$ and variance $\mathcal{K}(x,x) - \boldsymbol{k}^T\Sigma^{-1}\boldsymbol{k}$. The predictive distribution of $\mathcal{P}(f(x)|\mathcal{D},\theta^*)$ can be computed as an integral over f-space, which can be written as

$$\mathcal{P}(f(x)|\mathcal{D},\theta^*) = \int \mathcal{P}(f(x)|f,\theta^*)\mathcal{P}(f|\mathcal{D},\theta^*)\,df. \tag{14}$$

The posterior distribution $\mathcal{P}(f|\mathcal{D},\theta^*)$ can be approximated as a Gaussian by the MAP approach or the EP approach (refer to Section 3). The predictive distribution (14) can then be simplified as a Gaussian $\mathcal{N}(f(x);\mu_x,\sigma_x^2)$ with mean μ_x and variance σ_x^2. In the MAP approach, we reach

$$\mu_x = k^T\Sigma^{-1}f_{\text{MAP}} \quad \text{and} \quad \sigma_x^2 = \mathcal{K}(x,x) - k^T(\Sigma + \Lambda_{\text{MAP}}^{-1})^{-1}k. \tag{15}$$

While in the EP approach, we get

$$\mu_x = k^T(\Sigma + \Pi^{-1})^{-1}m \quad \text{and} \quad \sigma_x^2 = \mathcal{K}(x,x) - k^T(\Sigma + \Pi^{-1})^{-1}k. \tag{16}$$

The predictive distribution over ordinal targets y_x is

$$\begin{aligned}\mathcal{P}(y_x|x,\mathcal{D},\theta^*) &= \int \mathcal{P}(y_x|f(x),\theta^*)\mathcal{P}(f(x)|\mathcal{D},\theta^*)\,df(x)\\ &= \Phi\left(\frac{b_{y_x}-\mu_x}{\sqrt{\sigma^2+\sigma_x^2}}\right) - \Phi\left(\frac{b_{y_x-1}-\mu_x}{\sqrt{\sigma^2+\sigma_x^2}}\right).\end{aligned}$$

The predictive ordinal scale can be decided as $\arg\max_i \mathcal{P}(y_x = i|x,\mathcal{D},\theta^*)$.

5. Discussion

In the MAP approach, the mean of the predictive distribution depends on the MAP estimate f_{MAP}, which is unique and can be found by solving a convex programming problem. Evidence maximization is useful if the Laplace approximation around the mode point f_{MAP} gives a good summary of the posterior distribution $\mathcal{P}(f|\mathcal{D})$. While in the approach of expectation propagation, the mean of the predictive distribution depends on the approximate mean of the posterior distribution. When the true shape of $\mathcal{P}(f|\mathcal{D})$ is far from a Gaussian centered on the mode, the EP approach can have a great advantage over the Laplace approximation. However the EP algorithm cannot guarantee convergence, though it usually works well in practice.

The gradient-based optimization method usually requests evidence evaluation at tens of different settings of θ before the minimum is found. For each θ, the inversion of the matrix Σ is required that costs time at $O(n^3)$, where n is the number of training samples. Recently, Csató and Opper (2002) proposed a fast training algorithm for Gaussian processes in which the set of basis vectors are determined on-line for sparse representation. Lawrence et al. (2003) proposed a greedy selection with criteria based on information-theoretic principles for sparse Gaussian processes (Seeger, 2003). Tresp (2000) proposed the Bayesian committee machines to divide and conquer large data sets, while using infinite mixtures of Gaussian Processes (Rasmussen and Ghahramani, 2002) is another promising technique. These algorithms can be applied directly in the settings of ordinal regression for speedup.

Feature selection is an essential part in modelling. In Gaussian processes, the automatic relevance determination (ARD) method proposed by MacKay (1994) and Neal (1996) can be embedded into the covariance function (1) as follows:

$$Cov[f(x_i), f(x_j)] = \mathcal{K}(x_i, x_j) = \exp\left(-\frac{1}{2}\sum_{\varsigma=1}^{d} \kappa_\varsigma (x_i^\varsigma - x_j^\varsigma)^2\right) \tag{17}$$

where $\kappa_\varsigma > 0$ is the ARD parameter.[4] The gradients with respect to the variables $\{\ln \kappa_\varsigma\}$ can also be derived analytically for model adaptation. The optimal value of the ARD parameter κ_ς indicates the relevance of the ς-th input feature to the target. The form of feature selection we use here results in a type of feature weighting. Furthermore, the linear combination of heterogeneous kernels with positive coefficients is still a valid covariance function. Lanckriet et al. (2004) suggest to learn the kernel matrix with semidefinite programming. In the Bayesian framework, these positive coefficients for kernels could be treated as hyperparameters, and optimized using the evidence as a criterion for optimization.

Note that binary classification is a special case of ordinal regression with $r = 2$, and the likelihood function (5) becomes the *probit* function when $r = 2$. Both of the *probit* function and the logistic function can be used as the likelihood function in binary classification, while they have different origins. Due to the dichotomous nature in the classes of multi-classification, discriminant functions are constructed for each class and then compete again others via the *softmax* function to determine the likelihood. The logistic function, as a special case of the *softmax* function, comes from general classification problems.

In metric regression, warped Gaussian processes (Snelson et al., 2004) assume that there is a nonlinear, monotonic, and continuous warping function relating the observed targets and some latent variables in a Gaussian process. The warping function, which is learned from the data, can be thought of as a pre-processing transformation applied before modelling with a Gaussian process. A different (and very common) approach to dealing with this preprocessing is to *discretize* the target values into r different bins. These discrete values are clearly ordinal, and applying ordinal regression to these discrete values seems the natural choice. Interestingly, as the number of discretization bins r is increased, the ordinal regression model becomes very similar to the warped Gaussian processes model. In particular, by varying the thresholds in our ordinal regression model, it can approximate any continuous warping function.

6. Numerical Experiments

We start this section with a simple synthetic data set to visualize the behavior of these algorithms, and report the experimental results on sixteen benchmark data sets.[5] Then we perform experiments on a collaborative filtering problem using the "EachMovie" data, and on Gleason score prediction from gene microarray data related to prostate cancer. Shashua and Levin (2003) generalized the support vector formulation by finding multiple thresholds to define parallel discriminant hyperplanes for ordinal scales, and reported that the performance of the support vector approach is better than that of the on-line algorithm (Crammer and Singer, 2002). The problem size in the large-margin ranking algorithm of Herbrich et al. (2000) is a quadratic function of the training data size making the algorithmic complexity $O(n^4)$–$O(n^6)$. This makes the experiments on large data sets computationally difficult. Thus, we decide to limit our comparisons to the support vector approach (SVM)

4. These ARD parameters control the covariance length-scale of the Gaussian process along each input dimension.
5. These data sets are publicly available at http://www.liacc.up.pt/∼ltorgo/Regression/DataSets.html.

of Shashua and Levin (2003) and the two versions of our approach, the MAP approach with Laplace approximation (MAP) and the EP algorithm with variational methods (EP). In our implementation,[6] we used the routine L-BFGS-B (Byrd et al., 1995) as the gradient-based optimization package, and started from the initial values of hyperparameters to infer the optimal values in the criterion of the approximate evidence (11) for MAP or the variational lower bound (13) for EP respectively.[7] The improved SMO algorithm (Keerthi et al., 2001) was adapted to implement the SVM approach (refer to Chu and Keerthi (2005) for detailed description and extensive discussion),[8] and 5-fold cross validation was used to determine the optimal values of model parameters (the kernel parameter κ and the regularization factor C) involved in the problem formulations. The initial search was done on a 7×7 coarse grid linearly spaced in the region $\{(\log_{10} C, \log_{10}\kappa) | -3 \le \log_{10} C \le 3, -3 \le \log_{10}\kappa \le 3\}$, followed by a fine search on a 9×9 uniform grid linearly spaced by 0.2 in the $(\log_{10} C, \log_{10}\kappa)$ space. We have utilized two evaluation metrics which quantify the accuracy of predictive ordinal scales $\{\hat{y}_1, \ldots, \hat{y}_t\}$ with respect to true targets $\{y_1, \ldots, y_t\}$:

- *Mean absolute error* is the average deviation of the prediction from the true target, i.e. $\frac{1}{t}\sum_{i=1}^{t} |\hat{y}_i - y_i|$, in which we treat the ordinal scales as consecutive integers;

- *Mean zero-one error* gives an error of 1 to every incorrect prediction that is the fraction of incorrect predictions.

6.1 Artificial Data

Figure 2 presents the behavior of the three algorithms using the Gaussian kernel (1) on a synthetic 2D data with three ordinal scales. In the support vector approach, the optimal thresholds were determined by the SMO algorithm and 5-fold cross validation was used to decide the optimal values of the kernel parameter and the regularization factor. As for the Gaussian process algorithms, model adaptation (see Section 3) was used to determine the optimal values of the kernel parameter, the noise level and the thresholds automatically. The figure shows that all the algorithms are working reasonably well on this task.

6.2 Benchmark Data

We collected nine benchmark data sets (Set I in Table 1) that were used for metric regression problems. The target values were discretized into ordinal quantities using equal-length binning. These bins divide the range of target values into a given number of intervals that are of same length. The resulting rank values are ordered, representing these intervals of the original metric quantities. For each data set, we generated two versions by discretizing the target values into five and ten intervals respectively. We randomly partitioned each data set into training/test splits as specified in Table 1. The partition was repeated 20 times independently. The Gaussian kernel (1) was used in these three algorithms. The test results are recorded in Tables 2 and 3. The performance of the MAP and EP approaches are closely matching. Our Gaussian process algorithms often yield better results than

6. The two versions of our proposed approach were implemented in ANSI C, and the source code is accessible at http://www.gatsby.ucl.ac.uk/∼chuwei/code/gpor.tar.
7. In numerical experiments, the initial values of the hyperparameters were usually chosen as $\sigma^2 = 1$, $\kappa = 1/d$ for Gaussian kernel, the threshold $b_1 = -1$ and $\Delta_\iota = 2/r$. We suggest to try several starting points in practice, and then choose the best model by the objective functional.
8. The source code in ANSI C is available at http://www.gatsby.ucl.ac.uk/∼chuwei/code/svorim.tar.

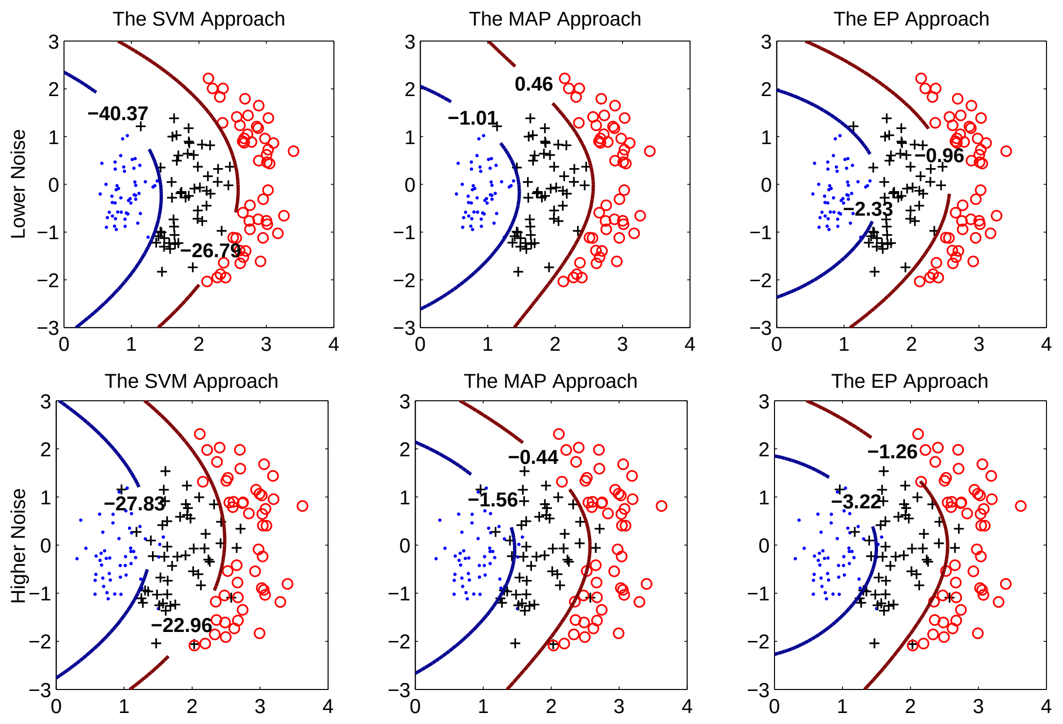


Figure 2: The performance of the three algorithms on a synthetic three-rank ordinal regression problem. The discriminant function values of the SVM approach, and the predictive mean values of the two Gaussian process approaches are presented as contour graphs indexed by the two thresholds. The upper graphs are for the case of lower noise level, while the lower graphs are for the case of higher noise level. The training samples we used are presented in these graphs. The dots denote the training samples of rank 1, the crosses denote the training samples of rank 2 and the circles denote the training samples of rank 3.

the support vector approach on the average value, especially when the number of training samples is small.

In the next experiment, we selected seven very large metric regression data sets (Set II in Table 1). The input vectors were normalized to zero mean and unit variance coordinate-wise. The target values of these data sets were discretized into 10 ordinal quantities using equal-frequency binning. For each data set, a small subset was randomly selected for training and then tested on the remaining samples, as specified in Table 1. The partition was repeated 100 times independently. To show the advantage of explicitly modelling the ordinal nature of the targets, we also employed the standard Gaussian process algorithm (Williams and Rasmussen, 1996) for metric regression (GPR)[9] to tackle these ordinal regression tasks, where the ordinal targets were naively treated as continuous values and the predictions for test cases were rounded to the nearest ordinal scale. The Gaussian kernel (1) was used in the four algorithms. From the test results in Table 4, the ordinal regression algo-

9. In the GPR, the type-II maximum likelihood was used for model selection.

	Data Sets	Attributes(Numeric,Nominal)	Training Instances	Instances for Test
	Diabetes	2(2,0)	30	13
	Pyrimidines	27(27,0)	50	24
	Triazines	60(60,0)	100	86
	Wisconsin Breast Cancer	32(32,0)	130	64
Set I	Machine CPU	6(6,0)	150	59
	Auto MPG	7(4,3)	200	192
	Boston Housing	13(12,1)	300	206
	Stocks Domain	9(9,0)	600	350
	Abalone	8(7,1)	1000	3177
	Bank Domains(1)	8(8,0)	50	8142
	Bank Domains(2)	32(32,0)	75	8117
	Computer Activity(1)	12(12,0)	100	8092
Set II	Computer Activity(2)	21(21,0)	125	8067
	California Housing	8(8,0)	150	15490
	Census Domains(1)	8(8,0)	175	16609
	Census Domains(2)	16(16,0)	200	16584

Table 1: Data sets and their characteristics. “Attributes” state the number of numerical and nominal attributes. “Training Instances” and “Instances for Test” specify the size of training/test partition. The partitions we generated and the test results on individual partitions can be accessed at http://www.gatsby.ucl.ac.uk/∼chuwei/ordinalregression.html.

	Mean zero-one error			Mean absolute error		
Data	SVM	MAP	EP	SVM	MAP	EP
Diabetes	57.31±12.09%	**54.23±13.78%**	54.23±13.78%	0.7462±0.1414	**0.6615±0.1376**	0.6654±0.1373
Pyrimidines	41.46±8.49%	39.79±7.21%	**36.46±6.47%**	0.4500±0.1136	0.4271±0.0906	**0.3917±0.0745**
Triazines	54.19±1.48%	52.91±2.15%	**52.62±2.66%**	0.6977±0.0259	**0.6872±0.0229**	0.6878±0.0295
Wisconsin	*70.78±3.73%	**65.00±4.71%**	65.16±4.65%	1.0031±0.0727	**1.0102±0.0937**	1.0141±0.0932
Machine	17.37±3.56%	**16.53±3.56%**	16.78±3.88%	0.1915±0.0423	**0.1847±0.0404**	0.1856±0.0424
Auto MPG	*25.73±2.24%	23.78±1.85%	**23.75±1.74%**	0.2596±0.0230	0.2411±0.0189	**0.2411±0.0186**
Boston	25.56±1.98%	24.88±2.02%	**24.49±1.85%**	0.2672±0.0190	0.2604±0.0206	**0.2585±0.0200**
Stocks	**10.81±1.70%**	11.99±2.34%	12.00±2.06%	**0.1081±0.0170**	0.1199±0.0234	0.1200±0.0206
Abalone	21.58±0.32%	**21.50±0.22%**	21.56±0.36%	**0.2293±0.0038**	0.2322±0.0025	*0.2337±0.0072

Table 2: Test results of the three algorithms using a Gaussian kernel. The targets of these benchmark data sets were discretized by 5 equal-length bins. The results are the averages over 20 trials, along with the standard deviation. We use the bold face to indicate the cases in which the average value is the lowest in the results of the three algorithms. The symbols $\star$ are used to indicate the cases in which the indicated entry is significantly worse than the winning entry; A p-value threshold of 0.01 in Wilcoxon rank sum test was used to decide statistical significance.

rithms are clearly superior to the naive approach of applying standard metric regression. We also observed that the performance of Gaussian process algorithms are significantly better than that of the support vector approach on six of the seven data sets. This verifies our judgement in the previous experiment that our Gaussian process algorithms yield better performance than the support vector approach on small data sets. Although the EP approach often yields better results of mean zero-one error than the MAP approach on these tasks, we have not detected any statistically significant difference on their performance. In Table 4 we also report their negative logarithm of the likelihood in prediction (NLL). The performance of the MAP and EP approaches are closely matching too with no statistically significant difference.

	Mean zero-one error			Mean absolute error		
Data	SVM	MAP	EP	SVM	MAP	EP
Diabetes	*90.38±7.00%	83.46±5.73%	**83.08±5.91%**	2.4577±0.4369	**2.1385±0.3317**	2.1423±0.3314
Pyrimidines	59.37±7.63%	55.42±8.01%	**54.38±7.70%**	0.9187±0.1895	0.8771±0.1749	**0.8292±0.1338**
Triazines	*67.91±3.63%	**63.72±4.34%**	64.01±3.78%	1.2308±0.0874	**1.1994±0.0671**	1.2012±0.0680
Wisconsin	*85.86±3.78%	**78.52±3.58%**	78.52±3.51%	2.1250±0.1500	**2.1391±0.1797**	2.1437±0.1790
Machine	**32.63±3.84%**	33.81±3.91%	33.73±3.64%	**0.4398±0.0688**	0.4746±0.0727	0.4686±0.0763
Auto MPG	44.01±2.30%	43.96±2.81%	**43.88±2.60%**	0.5081±0.0263	0.4990±0.0352	**0.4979±0.0340**
Boston	42.06±2.49%	41.53±2.77%	**41.26±2.86%**	0.4971±0.0305	0.4920±0.0330	**0.4896±0.0346**
Stocks	**17.74±2.15%**	*19.90±1.72%	*19.44±1.91%	**0.1804±0.0213**	*0.2006±0.0166	*0.1960±0.0184
Abalone	42.84±0.86%	42.60±0.91%	**42.27±0.46%**	0.5160±0.0087	0.5140±0.0075	**0.5113±0.0053**

Table 3: Test results of the three algorithms using a Gaussian kernel. The targets of these benchmark data sets were discretized by 10 equal-length bins. The results are the averages over 20 trials, along with the standard deviation. We use the bold face to indicate the cases in which the average value is the lowest in the results of the three algorithms. The symbols $\star$ are used to indicate the cases in which the indicated entry is significantly worse than the winning entry; A p-value threshold of 0.01 in Wilcoxon rank sum test was used to decide statistical significance.

	Mean zero-one error				NLL	
Data	GPR	SVM	MAP	EP	MAP	EP
Bank(1)	*59.43 ± 2.80 %	49.07 ± 2.69 %	48.65 ± 1.93 %	**48.35 ± 1.91 %**	1.14 ± 0.07	1.14 ± 0.07
Bank(2)	*86.37 ± 1.49 %	*82.26 ± 2.06 %	80.96 ± 1.51 %	**80.89 ± 1.52 %**	2.20 ± 0.09	2.20 ± 0.09
CompAct(1)	*65.52 ± 2.31 %	*59.87 ± 2.25 %	58.52 ± 1.73 %	**58.51 ± 1.53 %**	1.65 ± 0.16	1.64 ± 0.14
CompAct(2)	*59.30 ± 2.27 %	*54.79 ± 2.10 %	**53.80 ± 1.84 %**	53.92 ± 1.68 %	1.49 ± 0.11	1.48 ± 0.09
California	*76.13 ± 1.27 %	*70.63 ± 1.40 %	69.60 ± 1.12 %	**69.58 ± 1.11 %**	1.89 ± 0.08	1.89 ± 0.09
Census(1)	*78.06 ± 0.81 %	*74.69 ± 0.94 %	**73.71 ± 0.77 %**	73.71 ± 0.77 %	2.04 ± 0.08	2.05 ± 0.08
Census(2)	*78.02 ± 0.85 %	*76.01 ± 1.03 %	74.53 ± 0.81 %	**74.48 ± 0.84 %**	2.03 ± 0.06	2.03 ± 0.07

Table 4: Test results of the four algorithms using a Gaussian kernel. The targets of these benchmark data sets were discretized by 10 equal-frequency bins. The results are the average over 100 trials, along with the standard deviation. "GPR" denotes the standard algorithm of Gaussian process metric regression that treats the ordinal scales as continuous values. "NLL" denotes the negative logarithm of the likelihood in prediction. We use the bold face to indicate the cases in which the average value is the lowest mean zero-one error of the four algorithms. The symbols $\star$ are used to indicate the cases in which the indicated entry is significantly worse than the winning entry; A p-value threshold of 0.01 in Wilcoxon rank sum test was used to decide statistical significance.

For these data sets, the overall training time of MAP and EP approaches was substantially less than that of the SVM approach. This is because the MAP and EP approaches can tune the model parameters by gradient descent that usually required evidence evaluations at tens of different settings of θ, whereas k-fold cross validation for the SVM approach required evaluations at 130 different nodes of θ on the grid for every fold. For larger data sets, the SVM approach may still have an advantage on training time due to the sparseness property in its computation.

6.3 Collaborative Filtering

Collaborative filtering exploits correlations between ratings across a population of users. The goal is to predict a person's rating on new items given the person's past ratings on similar items and the ratings of other people on all the items (including the new item). The ratings are ordered, making collaborative filtering an ordinal regression problem. We carried out ordinal regression on a subset of the EachMovie data (Compaq, 2001).[10] The rates given by the user with ID number "52647" on 449 movies were used as the targets, in which the numbers of zero-to-five star are 40, 20, 57, 113, 145 and 74 respectively. We selected 1500 users who contributed the most ratings on these 449 movies as the input features. The ratings given by the 1500 users on each movie were used as the input vector accordingly. In the 449×1500 input matrix, about 40% elements were observed. We randomly selected a subset with size $\{50, 100, \ldots, 300\}$ of the 449 movies for training, and then tested on the remaining movies. At each size, the random selection was carried out 20 times independently.

Pearson correlation coefficient is the most popular correlation measure (Basilico and Hofmann, 2004), which corresponds to a dot product between normalized rating vectors. For instance, if applied to the movies, we can define the so-called z-scores as

$$z(v,u) = \frac{r(v,u) - \mu(v)}{\sigma(v)}$$

where u indexes users, v indexes movies, and $r(v,u)$ is the rating on the movie v given by the user u. $\mu(v)$ and $\sigma(v)$ are the movie-specific mean and standard deviation respectively. This correlation coefficient, defined as

$$\mathcal{K}(v,v') = \sum_u z(v,u) z(v',u)$$

where $\sum_u$ denotes summing over all the users, was used as the covariance/kernel function in our experiments for the three algorithms. As not all ratings are observed in the input vectors, we consider two *ad hoc* strategies to deal with missing values: mean imputation and weighted low-rank approximation. In the first case, unobserved values are identified with the mean value, that means their corresponding z-score is zero. In the second case, we applied the EM procedure described by Srebro and Jaakkola (2003) to fill in the missing data with the estimate. In the input matrix, observed elements were weighted by one and missing data were given weight zero. The low rank was fixed at 2. In Figure 3, we present the test results of the two cases at different training data size. Using mean imputation, SVM produced a bit more accurate results than Gaussian processes on mean absolute error. In the cases with low rank approximation as preprocessing, the performance of the three algorithms are highly competitive, and more interestingly, we observed about 0.08 improvement on mean absolute error for all the three algorithms. A serious treatment on the missing data could be an interesting research topic for future work.

6.4 Gene Expression Analysis

Singh et al. (2002) carried out microarray expression analysis on 12600 genes to identify genes that might anticipate the clinical behavior of prostate cancer. Fifty-two samples of prostate tumor were investigated. For each sample, the Gleason score ranging from 6 to 10, was given by the

10. The Compaq System Research Center ran the EachMovie service for 18 months. 72916 users entered a total of 2811983 numeric ratings on 1628 movies, i.e. about 2.4% are rated by zero-to-five star.

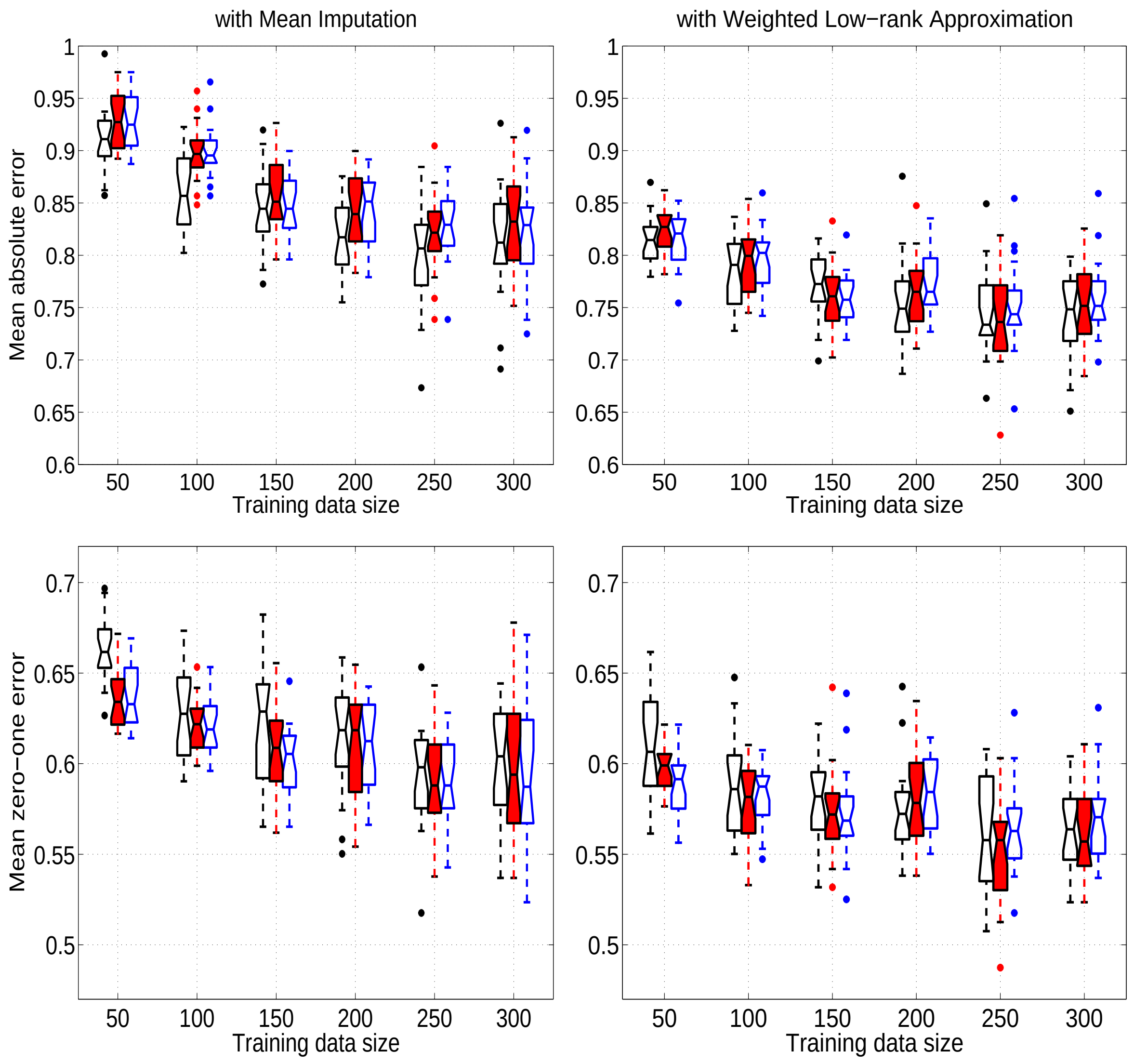


Figure 3: The test results of the three algorithms on the subset of EachMovie data over 20 trials. The grouped boxes represent the results of SVM (left), MAP (middle) and EP (right) respectively at different training data size. The notched-boxes have lines at the lower quartile, median, and upper quartile values. The whiskers are lines extending from each end of the box to the most extreme data value within 1.5·IQR(Interquartile Range) of the box. Outliers are data with values beyond the ends of the whiskers, which are displayed by dots. The higher graphs are for the results of mean absolute error and the lower graphs are for mean zero-one error. The cases of mean imputation are presented in the left graphs, and the cases with weighted low-rank approximation as preprocessing are presented in the right graphs.

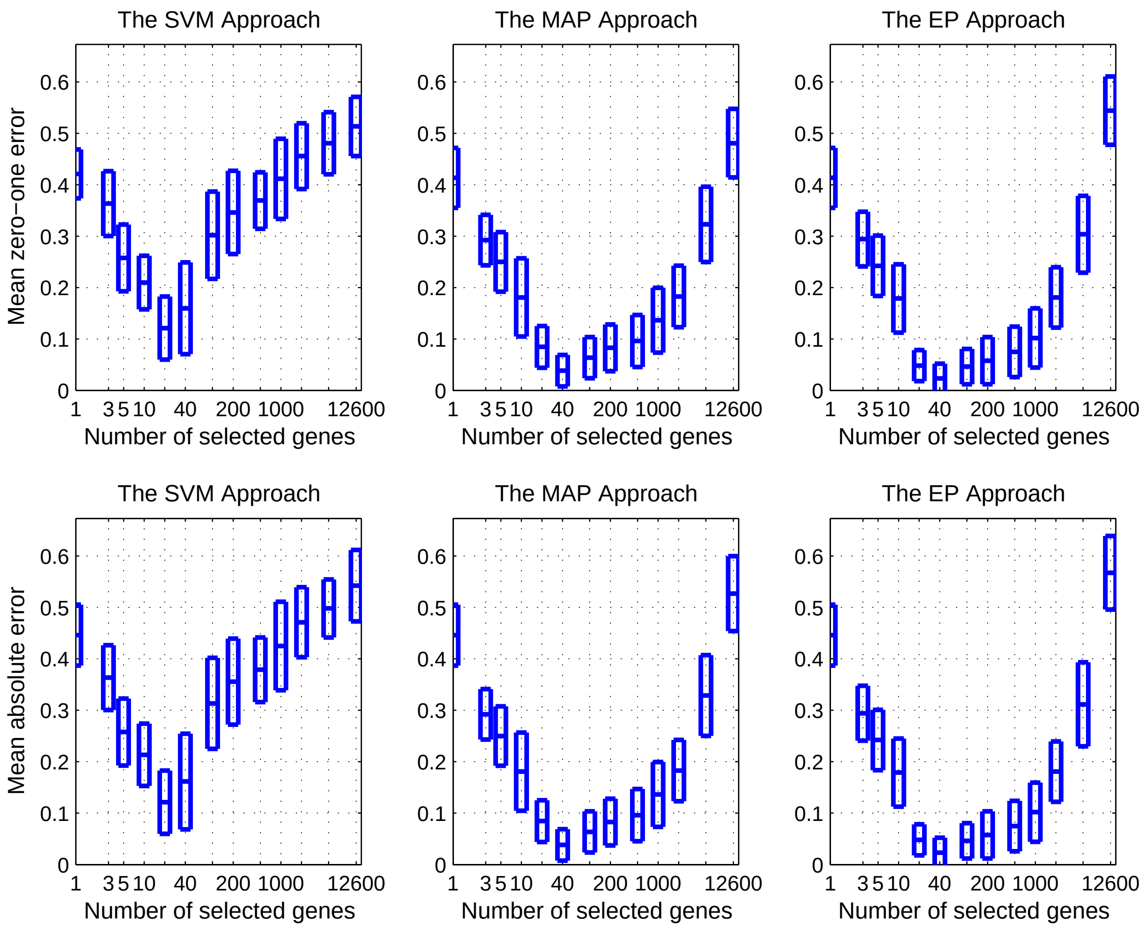


Figure 4: The test results of the three algorithms using a linear kernel on the prostate cancer data of selected genes. The horizonal axes are indexed on $\log_2$ scale. The rungs in these boxes indicate the mean values, and the heights of these vertical boxes indicate the standard deviations over the 20 trials.

pathologist reflecting the level of differentiation of the glands in the prostate tumor. Predicting the Gleason score from the gene expression data is thus a typical ordinal regression problem. Since only 6 samples had a score greater than 7, we merged them as the top level, leading to three levels $\{=6, =7, \geq 8\}$ with 26, 20 and 6 samples respectively. We randomly partitioned the data into 2 folds for training and test and repeated this partitioning 20 times independently. An ARD linear kernel $\mathcal{K}(x_i, x_j) = \sum_{\varsigma=1}^{d} \kappa_\varsigma x_i^\varsigma x_j^\varsigma$ was used to evaluate feature relevance. These ARD parameters $\{\kappa_\varsigma\}$ were optimized by evidence maximization. According to the optimal values of these ARD parameters, the genes were ranked from irrelevant to relevant. We then removed the irrelevant genes gradually based on the rank list. The gene number was reduced from 12600 to 1. At each number of selected genes, a linear kernel $\mathcal{K}(x_i, x_j) = \sum_{\varsigma=1}^{d} x_i^\varsigma x_j^\varsigma$ was used in the three algorithms for a fair comparison. Figure 4 presents the test results of the three algorithms for different numbers of selected genes. We observed great and steady improvement using the subset of genes selected by the ARD technique. The best validation output is achieved around 40 top-ranked features. In this case, with only 26 training samples, the Bayesian approaches perform much better than the SVM, and the EP approach is generally better than the MAP approach but the difference is not statistically significant.

7. Conclusion

Ordinal regression is an important supervised learning problem with properties of both metric regression and classification. In this paper, we proposed a simple yet novel nonparametric Bayesian approach to ordinal regression based on a generalization of the *probit* likelihood function for Gaussian processes. Two approximate inference procedures were derived in detail for evidence evaluation and model adaptation. The approach intrinsically incorporates ARD feature selection and provides probabilistic prediction. The existent fast algorithms for Gaussian processes can be adapted directly to tackle relatively large data sets. Experiments on benchmark and real-world data sets show that the generalization performance is competitive and often better than support vector methods.

Acknowledgments

The main part of this work was carried out at Institute for Pure and Applied Mathematics (IPAM) of UCLA. We thank David L. Wild for stimulating this work and for many discussions. We also thank David J. C. MacKay for valuable comments. Wei Chu was supported by the National Institutes of Health and its National Institute of General Medical Sciences division under Grant Number 1 P01 GM63208. Zoubin Ghahramani was partially supported from CMU by DARPA under the CALO project. The reviewers' thoughtful comments are gratefully appreciated.

Appendix A. Gradient Formulae for Evidence Maximization

Evidence maximization is equivalent to finding the minimizer of the negative logarithm of the evidence which can be written in an explicit expression as follows

$$-\ln \mathcal{P}(\mathcal{D}|\theta) \approx \sum_{i=1}^{n} \ell(y_i, f_{\text{MAP}}(x_i)) + \frac{1}{2} f_{\text{MAP}}^T \Sigma^{-1} f_{\text{MAP}} + \frac{1}{2} \ln |\mathbf{I} + \Sigma \Lambda_{\text{MAP}}|.$$

Initialization	choose a favorite gradient-descent optimization package select the starting point θ for the optimization package
Looping	while the optimization package requests evidence/gradient evaluation at θ 1. find the MAP estimate by solving the convex programming problem (9) 2. evaluate the negative logarithm of the evidence (18) at the MAP 3. calculate the gradients with respect to θ (18)–(18) 4. feed the evidence and gradients to the optimization package
Exit	Return the optimal θ found by the optimization package

Table 5: The outline of our algorithm for model adaptation using the MAP approach with Laplace approximation.

We usually collect $\{\ln\kappa, \ln\sigma, b_1, \ln\Delta_2, \ldots, \ln\Delta_{r-1}\}$ as the set of variables to tune. This definition of tunable variables is helpful to convert the constrained optimization problem into an unconstrained optimization problem. The outline of our algorithm for model adaptation is described in Table 5.

The derivatives of $-\ln\mathcal{P}(\mathcal{D}|\theta)$ with respect to these variables can be derived as follows:

$$\begin{aligned}\frac{\partial-\ln\mathcal{P}(\mathcal{D}|\theta)}{\partial\ln\kappa} \quad = \quad & \frac{\kappa}{2}\mathrm{trace}\left[(\Lambda_{\mathrm{MAP}}^{-1}+\Sigma)^{-1}\frac{\partial\Sigma}{\partial\kappa}\right]-\frac{\kappa}{2}f_{\mathrm{MAP}}^T\Sigma^{-1}\frac{\partial\Sigma}{\partial\kappa}\Sigma^{-1}f_{\mathrm{MAP}}\\ & +\frac{\kappa}{2}\mathrm{trace}\left[\Lambda_{\mathrm{MAP}}^{-1}(\Lambda_{\mathrm{MAP}}^{-1}+\Sigma)^{-1}\Sigma\frac{\partial\Lambda_{\mathrm{MAP}}}{\partial\kappa}\right];\end{aligned}$$

$$\frac{\partial-\ln\mathcal{P}(\mathcal{D}|\theta)}{\partial\ln\sigma}=\sigma\sum_{i=1}^n\frac{\partial\ell(y_i,f_{\mathrm{MAP}}(x_i))}{\partial\sigma}+\frac{\sigma}{2}\mathrm{trace}\left[\Lambda_{\mathrm{MAP}}^{-1}(\Lambda_{\mathrm{MAP}}^{-1}+\Sigma)^{-1}\Sigma\frac{\partial\Lambda_{\mathrm{MAP}}}{\partial\sigma}\right];$$

$$\frac{\partial-\ln\mathcal{P}(\mathcal{D}|\theta)}{\partial b_1}=\sum_{i=1}^n\frac{\partial\ell(y_i,f_{\mathrm{MAP}}(x_i))}{\partial b_1}+\frac{1}{2}\mathrm{trace}\left[\Lambda_{\mathrm{MAP}}^{-1}(\Lambda_{\mathrm{MAP}}^{-1}+\Sigma)^{-1}\Sigma\frac{\partial\Lambda_{\mathrm{MAP}}}{\partial b_1}\right];$$

$$\frac{\partial-\ln\mathcal{P}(\mathcal{D}|\theta)}{\partial\ln\Delta_\iota}=\Delta_\iota\sum_{i=1}^n\frac{\partial\ell(y_i,f_{\mathrm{MAP}}(x_i))}{\partial\Delta_\iota}+\frac{\Delta_\iota}{2}\mathrm{trace}\left[\Lambda_{\mathrm{MAP}}^{-1}(\Lambda_{\mathrm{MAP}}^{-1}+\Sigma)^{-1}\Sigma\frac{\partial\Lambda_{\mathrm{MAP}}}{\partial\Delta_\iota}\right].$$

Note that at the MAP estimate $\Sigma^{-1}f_{\mathrm{MAP}}=-\sum_{i=1}^n\left.\frac{\partial\ell(y_i,f(x_i))}{\partial f}\right|_{f=f_{\mathrm{MAP}}}$. For more details, let us define

$$s_\rho=\frac{(z_1^i)^\rho\mathcal{N}(z_1^i;0,1)}{\Phi(z_1^i)-\Phi(z_2^i)}$$

and

$$v_\rho=\frac{(z_1^i)^\rho\mathcal{N}(z_1^i;0,1)-(z_2^i)^\rho\mathcal{N}(z_2^i;0,1)}{\Phi(z_1^i)-\Phi(z_2^i)}$$

where ρ runs from 0 to 3, $z_1^i=\frac{b_{y_i}-f(x_i)}{\sigma}$ and $z_2^i=\frac{b_{y_i-1}-f(x_i)}{\sigma}$. The ii-th entry of the diagonal matrix Λ is denoted as Λ_{ii}, which is defined as in (7), i.e. $\Lambda_{ii}=\frac{1}{\sigma^2}(v_0)^2+\frac{1}{\sigma^2}v_1$. The detailed derivatives are given in the following:

- $\frac{\partial\Lambda_{ii}}{\partial\kappa}=\frac{\partial\Lambda_{ii}}{\partial f^T}\frac{\partial f}{\partial\kappa}$.

- $\frac{\partial\Lambda_{ii}}{\partial f(x_i)}=\frac{1}{\sigma^3}(2(v_0)^3+3v_0v_1+v_2-v_0)$.

- $\frac{\partial f}{\partial \kappa} = \Lambda^{-1}(\Lambda^{-1}+\Sigma)^{-1}\frac{\partial \Sigma}{\partial \kappa}\Sigma^{-1}f$.
- $\frac{\partial \ell(y_i,f(x_i))}{\partial \sigma} = \frac{v_1}{\sigma}$.
- $\frac{\partial \Lambda_{ii}}{\partial \sigma} = -\frac{2}{\sigma}\Lambda_{ii} + \frac{1}{\sigma^3}(2v_0v_2 + 2(v_0)^2v_1 - v_1 + (v_1)^2 + v_3) + \frac{\partial \Lambda_{ii}}{\partial f^T}\frac{\partial f}{\partial \sigma}$.
- $\frac{\partial f}{\partial \sigma} = \Lambda^{-1}(\Lambda^{-1}+\Sigma)^{-1}\Sigma\psi_\sigma$, where ψ_σ is a column vector whose i-th element is $\frac{1}{\sigma^2}(v_0 - v_0v_1 - v_2)$.
- $\frac{\partial \Lambda_{ii}}{\partial b_1} = -\frac{\partial \Lambda_{ii}}{\partial f(x_i)} + \frac{\partial \Lambda_{ii}}{\partial f^T}\frac{\partial f}{\partial b_1}$.
- $\frac{\partial f}{\partial b_1} = \Lambda^{-1}(\Lambda^{-1}+\Sigma)^{-1}\Sigma\psi_b$, where ψ_b is a column vector whose i-th element is Λ_{ii}.
- $\frac{\partial \ell(y_i,f(x_i))}{\partial \Delta_\iota} = \begin{cases} -\frac{v_0}{\sigma} & \text{if } y_i > \iota; \\ -\frac{s_0}{\sigma} & \text{if } y_i = \iota; \\ 0 & \text{otherwise.} \end{cases}$
- $\frac{\partial \Lambda_{ii}}{\partial \Delta_\iota} = \begin{cases} -\frac{\partial \Lambda_{ii}}{\partial f(x_i)} + \frac{\partial \Lambda_{ii}}{\partial f^T}\frac{\partial f}{\partial \Delta_\iota} & \text{if } y_i > \iota; \\ \varphi_i + \frac{\partial \Lambda_{ii}}{\partial f}\frac{\partial f^T}{\partial \Delta_\iota} & \text{if } y_i = \iota; \\ \frac{\partial \Lambda_{ii}}{\partial f}\frac{\partial f^T}{\partial \Delta_\iota} & \text{otherwise.} \end{cases}$
- $\varphi_i = \frac{\partial \Lambda_{ii}}{\partial \Delta_\iota} = \frac{1}{\sigma^3}(s_0 - 2v_0s_1 - 2(v_0)^2s_0 - s_2 - v_1s_0)$.
- $\frac{\partial f}{\partial \Delta_\iota} = \Lambda^{-1}(\Lambda^{-1}+\Sigma)^{-1}\Sigma\psi_\Delta$, where ψ_Δ is a column vector whose i-th element is defined as $\psi_\Delta^i = \begin{cases} \Lambda_{ii} \text{ i.e. } \frac{1}{\sigma^2}((v_0)^2 + v_1) & \text{if } y_i > \iota; \\ \frac{1}{\sigma^2}(v_0s_0 + s_1) & \text{if } y_i = \iota; \\ 0 & \text{otherwise.} \end{cases}$

Appendix B. Approximate Posterior Distribution by EP

The expectation propagation algorithm attempts to approximate $\mathcal{P}(f|\mathcal{D})$ in form of a product of Gaussian distributions $Q(f) = \prod_{i=1}^n \tilde{t}(f(x_i))\mathcal{P}(f)$ where $\tilde{t}(f(x_i)) = s_i \exp(-\frac{1}{2}p_i(f(x_i)-m_i)^2)$. The updating scheme is given as follows.

The initial states:

- individual mean $m_i = 0 \ \forall i$;
- individual inverse variance $p_i = 0 \ \forall i$;
- individual amplitude $s_i = 1 \ \forall i$;
- posterior covariance $\mathcal{A} = (\Sigma^{-1}+\Pi)^{-1}$, where $\Pi = \text{diag}(p_1, p_2, \ldots, p_n)$;
- posterior mean $h = \mathcal{A}\Pi m$, where $m = [m_1, m_2, \ldots, m_n]^T$.

Looping i from 1 to n until there is no significant change in $\{m_i, p_i, s_i\}_{i=1}^n$:

- $\tilde{t}(f(x_i))$ is removed from $Q(f)$ to get a leave-one-out posterior distribution $Q^{\setminus i}(f)$ having

  - variance of $f(x_i)$: $\lambda_i^{\setminus i} = \frac{\mathcal{A}_{ii}}{1-\mathcal{A}_{ii}p_i}$;
  - mean of $f(x_i)$: $h_i^{\setminus i} = h_i + \lambda_i^{\setminus i} p_i (h_i - m_i)$;
  - others with $j \neq i$: $\lambda_j^{\setminus i} = \mathcal{A}_{jj}$ and $h_j^{\setminus i} = h_j$.

- $\tilde{t}(f(x_i))$ in $Q(f)$ is updated by incorporating the message $\mathcal{P}(y_i|f(x_i))$ into $Q^{\setminus i}(f)$:
  - $\mathcal{Z}_i = \int \mathcal{P}(y_i|f(x_i))\mathcal{N}(f(x_i); h_i^{\setminus i}, \lambda_i^{\setminus i}) df(x_i) = \Phi(\tilde{z}_1) - \Phi(\tilde{z}_2)$
    where $\tilde{z}_1 = \frac{b_{y_i} - h_i^{\setminus i}}{\sqrt{\lambda_i^{\setminus i}+\sigma^2}}$ and $\tilde{z}_2 = \frac{b_{y_i-1} - h_i^{\setminus i}}{\sqrt{\lambda_i^{\setminus i}+\sigma^2}}$.
  - $$\beta_i = \frac{\partial \log \mathcal{Z}_i}{\partial \lambda_i^{\setminus i}} = -\frac{1}{2(\lambda_i^{\setminus i}+\sigma^2)} \left(\frac{\tilde{z}_1 \mathcal{N}(\tilde{z}_1;0,1) - \tilde{z}_2 \mathcal{N}(\tilde{z}_2;0,1)}{\Phi(\tilde{z}_1) - \Phi(\tilde{z}_2)} \right).$$
    $$\gamma_i = \frac{\partial \log \mathcal{Z}_i}{\partial h_i^{\setminus i}} = -\frac{1}{\sqrt{\lambda_i^{\setminus i}+\sigma^2}} \left(\frac{\mathcal{N}(\tilde{z}_1;0,1) - \mathcal{N}(\tilde{z}_2;0,1)}{\Phi(\tilde{z}_1) - \Phi(\tilde{z}_2)} \right). \tag{18}$$
  - $\upsilon_i = \gamma_i^2 - 2\beta_i$.
  - $h_i^{new} = h_i^{\setminus i} + \lambda_i^{\setminus i}\gamma_i$.
  - $p_i^{new} = \frac{\upsilon_i}{1-\lambda_i^{\setminus i}\upsilon_i}$.
  - $m_i^{new} = h_i^{\setminus i} + \frac{\gamma_i}{\upsilon_i}$.
  - $s_i^{new} = \mathcal{Z}_i \sqrt{\lambda_i^{\setminus i} p_i^{new} + 1} \exp\left(\frac{\gamma_i^2}{2\upsilon_i}\right)$.

- Note that $p_i^{new} > 0$ all the time, because $0 < \upsilon_i < \frac{1}{\lambda_i^{\setminus i}+\sigma^2}$ and then $\lambda_i^{\setminus i}\upsilon_i < 1$.

- if $p_i^{new} \approx p_i$, skip this sample and this updating; otherwise update $\{p_i, m_i, s_i\}$, the posterior mean h and covariance $\mathcal{A}$ as follows:
  - $\mathcal{A}^{new} = \mathcal{A} - \rho a_i a_i^T$ where $\rho = \frac{p_i^{new} - p_i}{1+(p_i^{new}-p_i)\mathcal{A}_{ii}}$ and a_i is the i-th column of $\mathcal{A}$.
  - $h^{new} = h + \eta a_i$ where $\eta = \frac{\gamma_i + p_i(h_i - m_i)}{1-\mathcal{A}_{ii}p_i}$ and γ_i is defined as in (18).

As a byproduct, we can get the approximate evidence $\mathcal{P}(\mathcal{D}|\theta)$ at the EP solution, which can be written as

$$\prod_{i=1}^{n} s_i \frac{\det^{\frac{1}{2}}(\Pi^{-1})}{\det^{\frac{1}{2}}(\Sigma + \Pi^{-1})} \exp\left(\frac{B}{2}\right)$$

where $B = \sum_{ij} \mathcal{A}_{ij}(m_i p_i)(m_j p_j) - \sum_i p_i m_i^2$.

Appendix C. Gradient Formulae for Variational Bound

At the equilibrium of $Q(f)$, the variational bound $\mathcal{F}(\theta)$ can be analytically calculated as follows:

$$\begin{aligned} \mathcal{F}(\theta) = \; & \sum_{i=1}^{n} \int \mathcal{N}(f(x_i); h_i, \mathcal{A}_{ii}) \ln(\mathcal{P}(y_i|f(x_i))) df(x_i) - \frac{1}{2} \ln |\mathrm{I} + \Sigma\Pi| \\ & -\frac{1}{2} \mathrm{trace}((\mathrm{I} + \Sigma\Pi)^{-1}) - \frac{1}{2} m^T (\Sigma + \Pi^{-1})^{-1} \Sigma (\Sigma + \Pi^{-1})^{-1} m + \frac{n}{2} \; . \end{aligned}$$

Note that $(\Sigma+\Pi^{-1})^{-1}m$ can be directly obtained by $\{\gamma_i\}$ defined as in (18). The gradient of $\mathcal{F}(\theta)$ with respect to the variables $\{\ln\kappa, \ln\sigma, b_1, \ln\Delta_2, \ldots, \ln\Delta_{r-1}\}$ can be given in the following:

$$\begin{aligned}
\frac{\partial\mathcal{F}(\theta)}{\partial\ln\kappa} &= \kappa\int Q(f)\frac{\partial\log\mathcal{P}(f)}{\partial\kappa}df \\
&= -\frac{\kappa}{2}\text{trace}\left(\Sigma^{-1}\frac{\partial\Sigma}{\partial\kappa}\right)+\frac{\kappa}{2}h^T\Sigma^{-1}\frac{\partial\Sigma}{\partial\kappa}\Sigma^{-1}h+\frac{\kappa}{2}\text{trace}\left(\Sigma^{-1}\frac{\partial\Sigma}{\partial\kappa}\Sigma^{-1}\mathcal{A}\right) \\
&= -\frac{\kappa}{2}\text{trace}\left((\Pi^{-1}+\Sigma)^{-1}\frac{\partial\Sigma}{\partial\kappa}\right)+\frac{\kappa}{2}m^T(\Pi^{-1}+\Sigma)^{-1}\frac{\partial\Sigma}{\partial\kappa}(\Pi^{-1}+\Sigma)^{-1}m\,,
\end{aligned}$$

$$\begin{aligned}
\frac{\partial\mathcal{F}(\theta)}{\partial\ln\sigma} &= \sigma\textstyle\sum_{i=1}^n\int\mathcal{N}(f(x_i);h_i,\mathcal{A}_{ii})\frac{\partial\ln\mathcal{P}(y_i|f(x_i))}{\partial\sigma}df(x_i) \\
&= -\textstyle\sum_{\{1\le y_i<r\}}\int\mathcal{N}\left(f(x_i);\frac{h_i\sigma^2+\mathcal{A}_{ii}b_{y_i}}{\sigma^2+\mathcal{A}_{ii}},\frac{\sigma^2\mathcal{A}_{ii}}{\sigma^2+\mathcal{A}_{ii}}\right)\frac{\frac{b_{y_i}-f(x_i)}{\sqrt{2\pi(\sigma^2+\mathcal{A}_{ii})}}\exp\left(-\frac{(h_i-b_{y_i})^2}{2(\sigma^2+\mathcal{A}_{ii})}\right)}{\mathcal{P}(y_i|f(x_i))}df(x_i) \\
&\quad+\textstyle\sum_{\{1<y_i\le r\}}\int\mathcal{N}\left(f(x_i);\frac{h_i\sigma^2+\mathcal{A}_{ii}b_{y_i-1}}{\sigma^2+\mathcal{A}_{ii}},\frac{\sigma^2\mathcal{A}_{ii}}{\sigma^2+\mathcal{A}_{ii}}\right)\frac{\frac{b_{y_i-1}-f(x_i)}{\sqrt{2\pi(\sigma^2+\mathcal{A}_{ii})}}\exp\left(-\frac{(h_i-b_{y_i-1})^2}{2(\sigma^2+\mathcal{A}_{ii})}\right)}{\mathcal{P}(y_i|f(x_i))}df(x_i)\,,
\end{aligned}$$

$$\begin{aligned}
\frac{\partial\mathcal{F}(\theta)}{\partial b_1} &= \textstyle\sum_{i=1}^n\int\mathcal{N}(f(x_i);h_i,\mathcal{A}_{ii})\frac{\partial\ln\mathcal{P}(y_i|f(x_i))}{\partial b_1}df(x_i) \\
&= \textstyle\sum_{\{1\le y_i<r\}}\int\mathcal{N}(f(x_i);\frac{h_i\sigma^2+\mathcal{A}_{ii}b_{y_i}}{\sigma^2+\mathcal{A}_{ii}},\frac{\sigma^2\mathcal{A}_{ii}}{\sigma^2+\mathcal{A}_{ii}})\frac{\frac{1}{\sqrt{2\pi(\sigma^2+\mathcal{A}_{ii})}}\exp\left(-\frac{(h_i-b_{y_i})^2}{2(\sigma^2+\mathcal{A}_{ii})}\right)}{\mathcal{P}(y_i|f(x_i)}df(x_i) \\
&\quad-\textstyle\sum_{\{1<y_i\le r\}}\int\mathcal{N}(f(x_i);\frac{h_i\sigma^2+\mathcal{A}_{ii}b_{y_i-1}}{\sigma^2+\mathcal{A}_{ii}},\frac{\sigma^2\mathcal{A}_{ii}}{\sigma^2+\mathcal{A}_{ii}})\frac{\frac{1}{\sqrt{2\pi(\sigma^2+\mathcal{A}_{ii})}}\exp\left(-\frac{(h_i-b_{y_i-1})^2}{2(\sigma^2+\mathcal{A}_{ii})}\right)}{\mathcal{P}(y_i|f(x_i))}df(x_i)\,,
\end{aligned}$$

$$\begin{aligned}
\frac{\partial\mathcal{F}(\theta)}{\partial\ln\Delta_\iota} &= \Delta_\iota\textstyle\sum_{i=1}^n\int\mathcal{N}(f(x_i);h_i,\mathcal{A}_{ii})\frac{\partial\ln\mathcal{P}(y_i|f(x_i))}{\partial\Delta_\iota}df(x_i) \\
&= \Delta_\iota\textstyle\sum_{\{\iota\le y_i<r\}}\int\mathcal{N}(f(x_i);\frac{h_i\sigma^2+\mathcal{A}_{ii}b_{y_i}}{\sigma^2+\mathcal{A}_{ii}},\frac{\sigma^2\mathcal{A}_{ii}}{\sigma^2+\mathcal{A}_{ii}})\frac{\frac{1}{\sqrt{2\pi(\sigma^2+\mathcal{A}_{ii})}}\exp\left(-\frac{(h_i-b_{y_i})^2}{2(\sigma^2+\mathcal{A}_{ii})}\right)}{\mathcal{P}(y_i|f(x_i)}df(x_i) \\
&\quad-\Delta_\iota\textstyle\sum_{\{\iota<y_i\le r\}}\int\mathcal{N}(f(x_i);\frac{h_i\sigma^2+\mathcal{A}_{ii}b_{y_i-1}}{\sigma^2+\mathcal{A}_{ii}},\frac{\sigma^2\mathcal{A}_{ii}}{\sigma^2+\mathcal{A}_{ii}})\frac{\frac{1}{\sqrt{2\pi(\sigma^2+\mathcal{A}_{ii})}}\exp\left(-\frac{(h_i-b_{y_i-1})^2}{2(\sigma^2+\mathcal{A}_{ii})}\right)}{\mathcal{P}(y_i|f(x_i))}df(x_i)\,,
\end{aligned}$$

where $\sum_{\{\iota<y_i\le r\}}$ means summing over all the samples whose targets satisfy $\iota<y_i\le r$, and these one-dimensional integrals can be approximated using Gaussian quadrature or calculated by Romberg integration at some appropriate accuracy.

References

J. Basilico and T. Hofmann. Unifying collaborative and content-based filtering. In *Proceedings of the 21th International Conference on Machine Learning*, pages 65–72, 2004.

H. J. Brascamp and E. H. Lieb. On extensions of the Brunn-Minkowski and Prekopa-Leindler theorems, including inequalities for log concave functions, and with an application to the diffusion equation. *Journal of Functional Analysis*, 22:366–389, 1976.

R. H. Byrd, P. Lu, and J. Nocedal. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific and Statistical Computing*, 16(5):1190–1208, 1995.

W. Chu and S. S. Keerthi. New approaches to support vector ordinal regression. Technical report, Yahoo! Research Labs, 2005.

W. W. Cohen, R. E. Schapire, and Y. Singer. Learning to order things. *Journal of artificial intelligence research*, 10:243–270, 1999.

Compaq. EachMovie. *http://research.compaq.com/SRC/eachmovie/*, 2001.

K. Crammer and Y. Singer. Pranking with ranking. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 641–647, Cambridge, MA, 2002. MIT Press.

L. Csató, E. Fokoué, M. Opper, B. Schottky, and O. Winther. Efficient approaches to Gaussian process classification. In Sara A. Solla, Todd K. Leen, and Klaus-Robert Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 251–257, 2000.

L. Csató and M. Opper. Sparse online Gaussian processes. *Neural Computation, The MIT Press*, 14:641–668, 2002.

L. Fahrmeir and G. Tutz. *Multivariate Statistical Modelling Based on Generalized Linear Models*. New York, Springer-Verlag, 2nd edition, 2001.

E. Frank and M. Hall. A simple approach to ordinal classification. In *Proceedings of the European Conference on Machine Learning*, pages 145–165, 2001.

S. Har-Peled, D. Roth, and D. Zimak. Constraint classification: A new approach to multiclass classification and ranking. In S. Thrun S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 785–792, 2003.

T. Hastie and R. Tibshirani. *Generalized Additive Models*. Chapman and Hall, London, 1990.

R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. In *Advances in Large Margin Classifiers*, pages 115–132. MIT Press, 2000.

V. E. Johnson and J. H. Albert. *Ordinal Data Modeling (Statistics for Social Science and Public Policy)*. Springer-Verlag, 1999.

S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. Improvements to Platt's SMO algorithm for SVM classifier design. *Neural Computation*, 13:637–649, March 2001.

H. Kim and Z. Ghahramani. The EM-EP algorithm for Gaussian process classification. In *Proc. of the Workshop on Probabilistic Graphical Models for Classification (at ECML)*, 2003.

S. Kramer, G. Widmer, B. Pfahringer, and M. DeGroeve. Prediction of ordinal classes using regression trees. *Fundamenta Informaticae*, 47:1–13, 2001.

G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

N. D. Lawrence, M. Seeger, and R. Herbrich. Fast sparse Gaussian process methods: The informative vector machine. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 609–616, 2003.

D. J. C. MacKay. A practical Bayesian framework for back propagation networks. *Neural Computation*, 4(3):448–472, 1992.

D. J. C. MacKay. Bayesian methods for backpropagation networks. In J. L. van Hemmen, E. Domany, and K. Schulten, editors, *Models of Neural Networks III*, pages 211–254, New York, 1994. Springer-Verlag.

P. McCullagh. Regression models for ordinal data. *Journal of the Royal Statistical Society B*, 42 (2):109–142, 1980.

P. McCullagh and J. A. Nelder. *Generalized Linear Models*. Chapman & Hall, London, 1983.

T. P. Minka. *A family of algorithms for approximate Bayesian inference*. PhD thesis, Massachusetts Institute of Technology, January 2001.

R. M. Neal. *Bayesian Learning for Neural Networks*. Lecture Notes in Statistics, No. 118. Springer-Verlag, New York, 1996.

R. M. Neal. Monte Carlo implementation of Gaussian process models for Bayesian regression and classification. Technical Report No. 9702, Department of Statistics, University of Toronto, 1997.

A. O'Hagan. Curve fitting and optimal design for prediction (with discussion). *Journal of the Royal Statistical Society B*, 40(1):1–42, 1978.

J. W. Pratt. Concavity of the log likelihood. *Journal of the American Statistical Association*, 76 (373):103–106, 1981.

C. E. Rasmussen and Z. Ghahramani. Infinite mixtures of Gaussian process experts. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 881–888, 2002.

B. Schölkopf and A. J. Smola. *Learning with Kernels – Support Vector Machines, Regularization, Optimization and Beyond*. Adaptive Computation and Machine Learning. The MIT Press, December 2001.

M. Seeger. Notes on Minka's expectation propagation for Gaussian process classification. Technical report, University of Edinburgh, 2002.

M. Seeger. *Bayesian Gaussian process models: PAC-Bayesian generalisation error bounds and sparse approximations*. PhD thesis, University of Edinburgh, July 2003.

A. Shashua and A. Levin. Ranking with large margin principle: two approaches. In S. Thrun S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 937–944. MIT Press, 2003.

D. Singh, P. G. Febbo, K. Ross, D. G. Jackson, J. Manola, C. Ladd, P. Tamayo, A. A. Renshaw, A. V. D'Amico, J. P. Richie, E. S. Lander, M. Loda, P. W. Kantoff, T. R. Golub, and W. R. Sellers. Gene expression correlates of clinical prostate cancer behavior. *Cancer Cell*, 1:203–209, 2002. www.genome.wi.mit.edu/MPR/prostate.

E. Snelson, Z. Ghahramani, and C. Rasmussen. Warped Gaussian processes. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 337–344, 2004.

N. Srebro and T. Jaakkola. Weighted low-rank approximations. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 720–727, 2003.

V. Tresp. A Bayesian committee machine. *Neural Computation*, 12(11):2719–2741, November 2000.

G. Tutz. Generalized semiparametrically structured ordinal models. *Biometrics*, 59:263–273, June 2003.

V. N. Vapnik. *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995.

G. Wahba. *Spline Models for Observational Data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM, 1990.

C. K. I. Williams and D. Barber. Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351, 1998.

C. K. I. Williams and C. E. Rasmussen. Gaussian processes for regression. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 598–604, 1996. MIT Press.

Journal of Machine Learning Research 6 (2005) 1043–1071 Submitted 5/03; Revised 10/04; Published 7/05

Learning the Kernel with Hyperkernels

Cheng Soon Ong[*] CHENGSOON.ONG@TUEBINGEN.MPG.DE
Max Planck Institute for Biological Cybernetics and
Friedrich Miescher Laboratory
Spemannstrasse 35
72076 Tübingen, Germany

Alexander J. Smola ALEX.SMOLA@NICTA.COM.AU

Robert C. Williamson BOB.WILLIAMSON@NICTA.COM.AU
National ICT Australia
Locked Bag 8001
Canberra ACT 2601, Australia
and Australian National University

Editor: Ralf Herbrich

Abstract

This paper addresses the problem of choosing a kernel suitable for estimation with a support vector machine, hence further automating machine learning. This goal is achieved by defining a reproducing kernel Hilbert space on the space of kernels itself. Such a formulation leads to a statistical estimation problem similar to the problem of minimizing a regularized risk functional.

We state the equivalent representer theorem for the choice of kernels and present a semidefinite programming formulation of the resulting optimization problem. Several recipes for constructing hyperkernels are provided, as well as the details of common machine learning problems. Experimental results for classification, regression and novelty detection on UCI data show the feasibility of our approach.

Keywords: learning the kernel, capacity control, kernel methods, support vector machines, representer theorem, semidefinite programming

1. Introduction

Kernel methods have been highly successful in solving various problems in machine learning. The algorithms work by implicitly mapping the inputs into a feature space, and finding a suitable hypothesis in this new space. In the case of the support vector machine (SVM), this solution is the hyperplane which maximizes the margin in the feature space. The feature mapping in question is defined by a kernel function, which allows us to compute dot products in feature space using only objects in the input space. For an introduction to SVMs and kernel methods, the reader is referred to numerous tutorials such as Burges (1998) and books such as Schölkopf and Smola (2002).

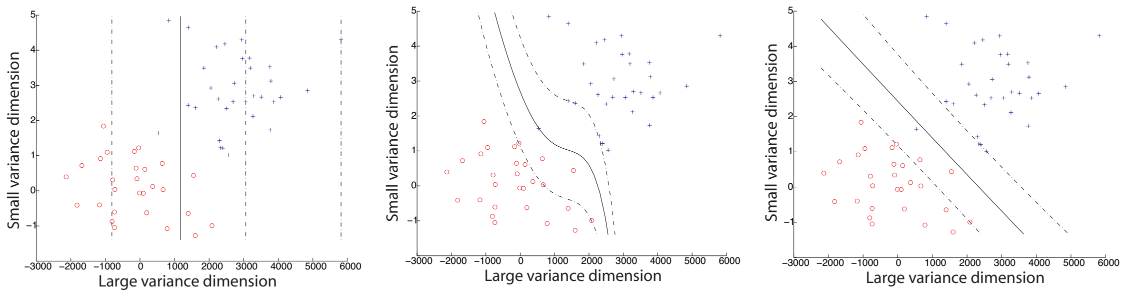
Choosing a suitable kernel function, and therefore a feature mapping, is imperative to the success of this inference process. This paper provides an inference framework for learning the kernel from training data using an approach akin to the regularized quality functional.

[*]. This work was done when the author was at the Australian National University.

1.1 Motivation

As motivation for the need for methods to learn the kernel, consider Figure 1, which shows the separating hyperplane, the margin and the training data for a synthetic data set. Figure 1(a) shows the classification function for a support vector machine using a Gaussian radial basis function (RBF) kernel. The data has been generated using two Gaussian distributions with standard deviation 1 in one dimension and 1000 in the other. This difference in scale creates problems for the Gaussian RBF kernel, since it is unable to find a kernel width suitable for both directions. Hence, the classification function is dominated by the dimension with large variance. Increasing the value of the regularization parameter, C, and hence decreasing the smoothness of the function results in a hyperplane which is more complex, and equally unsatisfactory (Figure 1(b)). The traditional way to handle such data is to normalize each dimension independently.

Instead of normalising the input data, we make the kernel adaptive to allow independent scales for each dimension. This allows the kernel to handle unnormalised data. However, the resulting kernel would be difficult to hand-tune as there may be numerous free variables. In this case, we have a free parameter for each dimension of the input. We 'learn' this kernel by defining a quantity analogous to the risk functional, called the quality functional, which measures the 'badness' of the kernel function. The classification function for the above mentioned data is shown in Figure 1(c). Observe that it captures the scale of each dimension independently. In general, the solution does not consist of only a single kernel but a linear combination of them.



(a) Standard Gaussian RBF kernel (C=10)

(b) Standard Gaussian RBF kernel (C=10^8)

(c) RBF-Hyperkernel with adaptive widths

Figure 1: For data with highly non-isotropic variance, choosing one scale for all dimensions leads to unsatisfactory results. Plot of synthetic data, showing the separating hyperplane and the margins given for a uniformly chosen length scale (left and middle) and an automatic width selection (right).

1.2 Related Work

We analyze some recent approaches to learning the kernel by looking at the objective function that is being optimized and the class of kernels being considered. We will see later (Section 2) that this objective function is related to our definition of a quality functional. Cross validation has been used to select the parameters of the kernels and SVMs (Duan et al., 2003, Meyer et al., 2003), with varying degrees of success. The objective function is the cross validation risk, and the class of kernels is a finite subset of the possible parameter settings. Duan et al. (2003) and Chapelle et al. (2002)

test various simple approximations which bound the leave one out error, or some measure of the capacity of the SVM. The notion of Kernel Target Alignment (Cristianini et al., 2002) uses the objective function $tr(Kyy^\top)$ where y are the training labels, and K is from the class of kernels spanned by the eigenvectors of the kernel matrix of the combined training and test data. The semidefinite programming (SDP) approach (Lanckriet et al., 2004) uses a more general class of kernels, namely a linear combination of positive semidefinite matrices. They minimize the margin of the resulting SVM using a SDP for kernel matrices with constant trace. Similar to this, Bousquet and Herrmann (2002) further restricts the class of kernels to the convex hull of the kernel matrices normalized by their trace. This restriction, along with minimization of the complexity class of the kernel, allows them to perform gradient descent to find the optimum kernel. Using the idea of boosting, Crammer et al. (2002) optimize $\sum_t \beta_t K_t$, where β_t are the weights used in the boosting algorithm. The class of base kernels $\{K_t\}$ is obtained from the normalized solution of the generalized eigenvector problem. In principle, one can learn the kernel using Bayesian methods by defining a suitable prior, and learning the hyperparameters by optimizing the marginal likelihood (Williams and Barber, 1998, Williams and Rasmussen, 1996). As an example of this, when other information is available, an auxiliary matrix can be used with the EM algorithm for learning the kernel (Tsuda et al., 2003). Table 1 summarizes these approaches. The notation $K \succeq 0$ means that K is positive semidefinite, that is for all $a \in \mathbb{R}^n$, $a^\top K a \geqslant 0$.

Approach	Objective	Kernel class ($\mathcal{K}$)
Cross Validation	CV Risk	Finite set of kernels
Alignment	$y^\top K y$	$\{\sum_{i=1}^m \beta_i v_i v_i^\top$ such that v_i are eigenvectors of $K\}$
SDP	margin	$\{\sum_{i=1}^m \beta_i K_i$ such that $K_i \succeq 0, \mathrm{tr} K_i = c\}$
Complexity Bound	margin	$\{\sum_{i=1}^m \beta_i K_i$ such that $K_i \succeq 0, \mathrm{tr} K_i = c, \beta_i \geqslant 0\}$
Boosting	Exp/LogLoss	Base kernels from generalized eigenvector problem
Bayesian	neg. log-post.	dependent on prior
EM Algorithm	KL Divergence	linear combination of auxiliary matrix

Table 1: Summary of recent approaches to kernel learning.

1.3 Outline of the Paper

The contribution of this paper is a theoretical framework for learning the kernel. Using this framework, we analyze the regularized risk functional. Motivated by the ideas of Cristianini et al. (2003), we show (Section 2) that for most kernel-based learning methods there exists a functional, the *quality functional*, which plays a similar role to the empirical risk functional. We introduce a kernel on the space of kernels itself, a *hyperkernel* (Section 3), and its regularization on the associated hyper reproducing kernel Hilbert space (Hyper-RKHS). This leads to a systematic way of parameterizing kernel classes while managing overfitting (Ong et al., 2002). We give several examples of hyperkernels and recipes to construct others (Section 4). Using this general framework, we consider the specific example of using the regularized risk functional in the rest of the paper. The positive definiteness of the kernel function is ensured using the positive definiteness of the kernel matrix (Section 5), and the resulting optimization problem is a semidefinite program. The semidefinite programming approach follows that of Lanckriet et al. (2004), with a different constraint due to a difference in regularization (Ong and Smola, 2003). Details of the specific optimization problems associated with the C-SVM, ν-SVM, Lagrangian SVM, ν-SVR and one class SVM are defined in Section 6. Experimental results for classification, regression and novelty detection (Section 7) are shown. Finally some issues and open problems are discussed (Section 8).

2. Kernel Quality Functionals

We denote by $\mathcal{X}$ the space of input data and $\mathcal{Y}$ the space of labels (if we have a supervised learning problem). Denote by $X_{\text{train}} := \{x_1, \ldots, x_m\}$ the training data and with $Y_{\text{train}} := \{y_1, \ldots, y_m\}$ a set of corresponding labels, jointly drawn independently and identically from some probability distribution $\Pr(x,y)$ on $\mathcal{X} \times \mathcal{Y}$. We shall, by convenient abuse of notation, generally denote Y_{train} by the vector y, when writing equations in matrix notation. We denote by K the kernel matrix given by $K_{ij} := k(x_i, x_j)$ where $x_i, x_j \in X_{\text{train}}$ and k is a positive semidefinite kernel function. We also use $\operatorname{tr}K$ to mean the trace of the matrix and $|K|$ to mean the determinant.

We begin by introducing a new class of functionals Q on data which we will call *quality functionals*. Note that by quality we actually mean *badness* or lack of quality, as we would like to minimize this quantity. Their purpose is to indicate, given a kernel k and the training data, how suitable the kernel is for explaining the training data, or in other words, the *quality* of the kernel for the estimation problem at hand. Such quality functionals may be the Kernel Target Alignment, the negative log posterior, the minimum of the regularized risk functional, or any luckiness function for kernel methods. We will discuss those functionals after a formal definition of the quality functional itself.

2.1 Empirical and Expected Quality

Definition 1 (Empirical Quality Functional) *Given a kernel k, and data X,Y, we define $Q_{\text{emp}}(k,X,Y)$ to be an* empirical quality functional *if it depends on k only via $k(x_i,x_j)$ where $x_i,x_j \in X$ for $1 \leqslant i,j \leqslant m$.*

By this definition, Q_{emp} is a function which tells us how well matched k is to a specific data set X,Y. Typically such a quantity is used to adapt k in such a manner that Q_{emp} is optimal (for example, optimal Kernel Target Alignment, greatest luckiness, smallest negative log-posterior), based on this one *single* data set X,Y. Provided a sufficiently rich class of kernels $\mathcal{K}$ it is in general possible to find a kernel $k^* \in \mathcal{K}$ that attains the minimum of any such Q_{emp} regardless of the data. However, it is very unlikely that $Q_{\text{emp}}(k^*,X,Y)$ would be similarly small for other X,Y, for such a k^*. To measure the overall quality of k we therefore introduce the following definition:

Definition 2 (Expected Quality Functional) *Denote by $Q_{\text{emp}}(k,X,Y)$ an empirical quality functional, then*

$$Q(k) := \mathbf{E}_{X,Y}\left[Q_{\text{emp}}(k,X,Y)\right]$$

is defined to be the expected quality functional. Here the expectation is taken over X,Y, where all x_i,y_i are drawn from $\Pr(x,y)$.

Observe the similarity between the empirical quality functional, $Q_{\text{emp}}(k,X,Y)$, and the empirical risk of an estimator, $R_{\text{emp}}(f,X,Y) = \frac{1}{m}\sum_{i=1}^{m} l(x_i,y_i,f(x_i))$ (where l is a suitable loss function); in both cases we compute the value of a functional which depends on some sample X,Y drawn from $\Pr(x,y)$ and a function and in both cases we have

$$Q(k) = \mathbf{E}_{X,Y}\left[Q_{\text{emp}}(k,X,Y)\right] \text{ and } R(f) = \mathbf{E}_{X,Y}\left[R_{\text{emp}}(f,X,Y)\right].$$

Here $R(f)$ denotes the expected risk. However, while in the case of the empirical risk we can interpret R_{emp} as the empirical estimate of the expected loss $R(f) = \mathbf{E}_{x,y}[l(x,y,f(x))]$, due to the general form of Q_{emp}, no such analogy is available for quality functionals. Finding a general-purpose bound of the expected error in terms of $Q(k)$ is difficult, since the definition of Q depends heavily on the algorithm under consideration. Nonetheless, it provides a general framework within which such bounds can be derived.

To obtain a generalization error bound, it is sufficient that Q_{emp} is concentrated around its expected value. Furthermore, one would require the deviation of the empirical risk to be upper bounded by Q_{emp} and possibly other terms. In other words, we assume a) we have given a concentration inequality on quality functionals, such as

$$\Pr\left\{|Q_{\text{emp}}(k,X,Y) - Q(k)| \geqslant \varepsilon_Q\right\} < \delta_Q,$$

and b) we have a bound on the deviation of the empirical risk in terms of the quality functional

$$\Pr\left\{|R_{\text{emp}}(f,X,Y) - R(f)| \geqslant \varepsilon_R\right\} < \delta(Q_{\text{emp}}).$$

Then we can chain both inequalities together to obtain the following bound

$$\Pr\left\{|R_{\text{emp}}(f,X,Y) - R(f)| \geqslant \varepsilon_R\right\} < \delta_Q + \delta(Q + \varepsilon_Q).$$

This means that the bound now becomes independent of the particular value of the quality functional obtained *on* the data, rather than the expected value of the quality functional. Bounds of this type have been derived for Kernel Target Alignment (Cristianini et al., 2003, Theorem 9) and the Algorithmic Luckiness framework (Herbrich and Williamson, 2002, Theorem 17).

2.2 Examples of Q_{emp}

Before we continue with the derivations of a regularized quality functional and introduce a corresponding reproducing kernel Hilbert space, we give some examples of quality functionals and present their exact minimizers, whenever possible. This demonstrates that given a rich enough feature space, we can arbitrarily minimize the empirical quality functional Q_{emp}. The difference here from traditional kernel methods is the fact that we allow the kernel to change. This extra degree of freedom allows us to overfit the training data. In many of the examples below, we show that given a feature mapping which can model the labels of the training data precisely, overfitting occurs. That is, if we use the training labels as the kernel matrix, we arbitrarily minimize the quality functional. The reader who is convinced that one can arbitrarily minimize Q_{emp}, by optimizing over a suitably large class of kernels, may skip the following examples.

Example 1 (Regularized Risk Functional) *These are commonly used in SVMs and related kernel methods (see Wahba (1990), Vapnik (1995), Schölkopf and Smola (2002)). They take on the general form*

$$R_{\text{reg}}(f,X_{\text{train}},Y_{\text{train}}) := \frac{1}{m}\sum_{i=1}^{m} l(x_i,y_i,f(x_i)) + \frac{\lambda}{2}\|f\|_{\mathcal{H}}^2 \tag{1}$$

where $\|f\|^2_{\mathcal{H}}$ is the RKHS norm of f and l is a loss function such that for $f(x_i) = y_i$, $l(x_i, y_i, y_i) = 0$. By virtue of the representer theorem (see Section 3) we know that the minimizer of (1) can be written as a kernel expansion. This leads to the following definition of a quality functional, for a particular loss functional l:

$$Q^{\text{regrisk}}_{\text{emp}}(k, X_{\text{train}}, Y_{\text{train}}) := \min_{\alpha \in \mathbb{R}^m} \left[\frac{1}{m} \sum_{i=1}^{m} l(x_i, y_i, [K\alpha]_i) + \frac{\lambda}{2} \alpha^\top K \alpha \right]. \tag{2}$$

The minimizer of (2) is somewhat difficult to find, since we have to carry out a double minimization over K and α. However, we know that $Q^{\text{regrisk}}_{\text{emp}}$ is bounded from below by 0. Hence, it is sufficient if we can find a (possibly) suboptimal pair (α, k) for which $Q^{\text{regrisk}}_{\text{emp}} \leq \varepsilon$ for any $\varepsilon > 0$:

- *Note that for $K = \beta y y^\top$ and $\alpha = \frac{1}{\beta \|y\|^2} y$ we have $K\alpha = y$ and $\alpha^\top K \alpha = \beta^{-1}$. This leads to $l(x_i, y_i, f(x_i)) = 0$ and therefore $Q^{\text{regrisk}}_{\text{emp}}(k, X_{\text{train}}, Y_{\text{train}}) = \frac{\lambda}{2\beta}$. For sufficiently large β we can make $Q^{\text{regrisk}}_{\text{emp}}(k, X_{\text{train}}, Y_{\text{train}})$ arbitrarily close to 0.*

- *Even if we disallow setting K arbitrarily close to zero by setting $\operatorname{tr} K = 1$, finding the minimum of (2) can be achieved as follows: let $K = \frac{1}{\|z\|^2} z z^\top$, where $z \in \mathbb{R}^m$, and $\alpha = z$. Then $K\alpha = z$ and we obtain*

$$\frac{1}{m} \sum_{i=1}^{m} l(x_i, y_i, [K\alpha]_i) + \frac{\lambda}{2} \alpha^\top K \alpha = \sum_{i=1}^{m} l(x_i, y_i, z_i) + \frac{\lambda}{2} \|z\|^2_2. \tag{3}$$

  Choosing each $z_i = \operatorname{argmin}_\zeta l(x_i, y_i, \zeta(x_i)) + \frac{\lambda}{2} \zeta^2$, where ζ are the possible hypothesis functions obtained from the training data, yields the minimum with respect to z. Since (3) tends to zero and the regularized risk is lower bounded by zero, we can still arbitrarily minimize $Q^{\text{regrisk}}_{\text{emp}}$. This is not surprising since the set of allowable K is huge.

Example 2 (Cross Validation) *Cross validation is a widely used method for estimating the generalization error of a particular learning algorithm. Specifically, the leave-one-out cross validation is an almost unbiased estimate of the generalization error (Luntz and Brailovsky, 1969). The quality functional for classification using kernel methods is given by:*

$$Q^{\text{loo}}_{\text{emp}}(k, X_{\text{train}}, Y_{\text{train}}) := \min_{\alpha \in \mathbb{R}^m} \left[\frac{1}{m} \sum_{i=1}^{m} -y^i \operatorname{sign}([K\alpha^i]_i) \right],$$

which is optimized in Duan et al. (2003), Meyer et al. (2003).

Choosing $K = y y^\top$ and $\alpha^i = \frac{1}{\|y^i\|^2} y^i$, where α^i and y^i are the vectors α and y with the ith element set to zero, we have $K\alpha^i = y^i$. Hence we can match the training data perfectly. For a validation set of larger size, i.e. k-fold cross validation, the same result can be achieved by defining a corresponding α.

Example 3 (Kernel Target Alignment) *This quality functional was introduced by Cristianini et al. (2002) to assess the alignment of a kernel with training labels. It is defined by*

$$Q^{\text{alignment}}_{\text{emp}}(k, X_{\text{train}}, Y_{\text{train}}) := 1 - \frac{\operatorname{tr}(K y y^\top)}{\|y\|^2_2 \|K\|_F}. \tag{4}$$

Here $\|y\|_2$ denotes the ℓ_2 norm of the vector of observations and $\|K\|_F$ is the Frobenius norm, i.e., $\|K\|_F^2 := \operatorname{tr}(KK^\top) = \sum_{i,j}(K_{ij})^2$. This quality functional was optimized in Lanckriet et al. (2004). By decomposing K into its eigensystem one can see that (4) is minimized, if $K = yy^\top$, in which case

$$Q_{\text{emp}}^{\text{alignment}}(k^*, X_{\text{train}}, Y_{\text{train}}) = 1 - \frac{\operatorname{tr}(y^\top yy^\top y)}{\|y\|_2^2 \|yy^\top\|_F} = 1 - \frac{\|y\|_2^4}{\|y\|_2^2\|y\|_2^2} = 0.$$

We cannot expect that $Q_{\text{emp}}^{\text{alignment}}(k^*, X, Y) = 0$ for data other than that chosen to determine k^*, in other words, a restriction of the class of kernels is required. This was also observed in Cristianini et al. (2003).

The above examples illustrate how existing methods for assessing the quality of a kernel fit within the quality functional framework. We also saw that given a rich enough class of kernels $\mathcal{K}$, optimization of Q_{emp} over $\mathcal{K}$ would result in a kernel that would be useless for prediction purposes, in the sense that they can be made to look arbitrarily good in terms of Q_{emp} but with the result that the generalization performance will be poor. This is yet another example of the danger of optimizing too much and overfitting – there is (still) no free lunch.

3. Hyper Reproducing Kernel Hilbert Spaces

We now propose a conceptually simple method to optimize quality functionals over classes of kernels by introducing a reproducing kernel Hilbert space *on the kernel k itself*, so to say, a Hyper-RKHS. We first review the definition of a RKHS (Aronszajn, 1950).

Definition 3 (Reproducing Kernel Hilbert Space) *Let $\mathcal{X}$ be a nonempty set (the index set) and denote by $\mathcal{H}$ a Hilbert space of functions $f : \mathcal{X} \to \mathbb{R}$. $\mathcal{H}$ is called a reproducing kernel Hilbert space endowed with the dot product $\langle\cdot,\cdot\rangle$ (and the norm $\|f\| := \sqrt{\langle f, f\rangle}$) if there exists a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ with the following properties.*

1. *k has the reproducing property*

$$\langle f, k(x,\cdot)\rangle = f(x) \text{ for all } f \in \mathcal{H}, x \in \mathcal{X};$$

in particular, $\langle k(x,\cdot), k(x',\cdot)\rangle = k(x,x')$ for all $x, x' \in \mathcal{X}$.

2. *k spans $\mathcal{H}$, i.e. $\mathcal{H} = \overline{\operatorname{span}\{k(x,\cdot) | x \in \mathcal{X}\}}$ where $\overline{X}$ is the completion of the set X.*

In the rest of the paper, we use the notation k to represent the kernel function and $\mathcal{H}$ to represent the RKHS. In essence, $\mathcal{H}$ is a Hilbert space of functions, which has the special property of being generated by the kernel function k.

The advantage of optimization in an RKHS is that under certain conditions the optimal solutions can be found as the linear combination of a finite number of basis functions, regardless of the dimensionality of the space $\mathcal{H}$ the optimization is carried out in. The theorem below formalizes this notion (see Kimeldorf and Wahba (1971), Cox and O'Sullivan (1990)).

Theorem 4 (Representer Theorem) *Denote by $\Omega : [0,\infty) \to \mathbb{R}$ a strictly monotonic increasing function, by $\mathcal{X}$ a set, and by $l : (\mathcal{X} \times \mathbb{R}^2)^m \to \mathbb{R} \cup \{\infty\}$ an arbitrary loss function. Then each minimizer $f \in \mathcal{H}$ of the general regularized risk*

$$l\left((x_1, y_1, f(x_1)), \ldots, (x_m, y_m, f(x_m))\right) + \Omega\left(\|f\|_{\mathcal{H}}\right)$$

admits a representation of the form

$$f(x) = \sum_{i=1}^{m} \alpha_i k(x_i, x), \tag{5}$$

where $\alpha_i \in \mathbb{R}$ *for all* $1 \leqslant i \leqslant m$.

3.1 Regularized Quality Functional

To learn the kernel, we need to define a function space of kernels, a method to regularize them and a practical optimization procedure. We will address each of these issues in the following. We define an RKHS on kernels $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, simply by introducing the compounded index set, $\underline{\mathcal{X}} := \mathcal{X} \times \mathcal{X}$ and by treating k as a function $k : \underline{\mathcal{X}} \to \mathbb{R}$:

Definition 5 (Hyper Reproducing Kernel Hilbert Space) *Let* $\mathcal{X}$ *be a nonempty set. and denote by* $\underline{\mathcal{X}} := \mathcal{X} \times \mathcal{X}$ *the compounded index set. The Hilbert space* $\underline{\mathcal{H}}$ *of functions* $k : \underline{\mathcal{X}} \to \mathbb{R}$, *endowed with a dot product* $\langle \cdot, \cdot \rangle$ *(and the norm* $\|k\| = \sqrt{\langle k, k \rangle}$*) is called a hyper reproducing kernel Hilbert space if there exists a hyperkernel* $\underline{k} : \underline{\mathcal{X}} \times \underline{\mathcal{X}} \to \mathbb{R}$ *with the following properties:*

1. $\underline{k}$ *has the reproducing property* $\langle k, \underline{k}(\underline{x}, \cdot) \rangle = k(\underline{x})$ *for all* $k \in \underline{\mathcal{H}}$; *in particular,* $\langle \underline{k}(\underline{x}, \cdot), \underline{k}(\underline{x}', \cdot) \rangle = \underline{k}(\underline{x}, \underline{x}')$.
2. $\underline{k}$ *spans* $\underline{\mathcal{H}}$, *i.e.* $\underline{\mathcal{H}} = \overline{\operatorname{span}\{\underline{k}(\underline{x}, \cdot) | \underline{x} \in \underline{\mathcal{X}}\}}$.
3. $\underline{k}(x, y, s, t) = \underline{k}(y, x, s, t)$ *for all* $x, y, s, t \in \mathcal{X}$.

This is a RKHS with the additional requirement of symmetry in its first two arguments (in fact, we can have a recursive definition of an RKHS of an RKHS ad infinitum, with suitable restrictions on the elements). We define the corresponding notations for elements, kernels, and RKHS by underlining it. What distinguishes $\underline{\mathcal{H}}$ from a normal RKHS is the particular form of its index set ($\underline{\mathcal{X}} = \mathcal{X}^2$) and the additional condition on $\underline{k}$ to be symmetric in its first two arguments, and therefore in its second two arguments as well.

This approach of defining a RKHS on the space of symmetric functions of two variables leads us to a natural regularization method. By analogy with the definition of the regularized risk functional (1), we proceed to define the regularized quality functional.

Definition 6 (Regularized Quality Functional) *Let* X, Y *be the combined training and test set of examples and labels respectively. For a positive semidefinite kernel matrix* K *on* X, *the* regularized quality functional *is defined as*

$$Q_{\text{reg}}(k, X, Y) := Q_{\text{emp}}(k, X, Y) + \frac{\lambda_Q}{2} \|k\|^2_{\underline{\mathcal{H}}}, \tag{6}$$

where $\lambda_Q \geqslant 0$ *is a regularization constant and* $\|k\|^2_{\underline{\mathcal{H}}}$ *denotes the RKHS norm in* $\underline{\mathcal{H}}$.

Note that although we have possibly non positive kernels in $\underline{\mathcal{H}}$, we define the regularized quality functional only on positive semidefinite kernel matrices. This is a slightly weaker condition than requiring a positive semidefinite kernel k, since we only require positivity on the data. Since Q_{emp} depends on k only via the data, this is sufficient for the above definition. Minimization of Q_{reg} is less prone to overfitting than minimizing Q_{emp}, since the regularization term $\frac{\lambda_Q}{2}\|k\|^2_{\underline{\mathcal{H}}}$ effectively controls the complexity of the class of kernels under consideration. Bousquet and Herrmann (2002) provide a generalization error bound by estimating the Rademacher complexity of the kernel classes in the transduction setting. Regularizers other than $\|k\|^2_{\underline{\mathcal{H}}}$ are possible, such as ℓ_p penalties. In this paper, we restrict ourselves to the ℓ_2 norm (6). The advantage of (6) is that its minimizer satisfies the representer theorem.

Lemma 7 (Representer Theorem for Hyper-RKHS) *Let $\mathcal{X}$ be a set, Q_{emp} an arbitrary empirical quality functional, and X,Y the combined training and test set, then each minimizer $k \in \underline{\mathcal{H}}$ of the regularized quality functional $Q_{\text{reg}}(k,X,Y)$ admits a representation of the form*

$$k(x,x') = \sum_{i,j}^{m} \beta_{ij}\underline{k}((x_i,x_j),(x,x')) \text{ for all } x,x' \in X, \tag{7}$$

where $\beta_{ij} \in \mathbb{R}$, for each $1 \leqslant i,j \leqslant m$.

Proof All we need to do is rewrite (6) so that it satisfies the conditions of Theorem 4. Let $\underline{x}_{ij} := (x_i,x_j)$. Then $Q_{\text{emp}}(k,X,Y)$ has the properties of a loss function, as it only depends on k via its values at $\underline{x}_{ij}$. Note too that the kernel matrix K also only depends on k via its values at $\underline{x}_{ij}$. Furthermore, $\frac{\lambda_Q}{2}\|k\|^2_{\underline{\mathcal{H}}}$ is an RKHS regularizer, so the representer theorem applies and (7) follows. ■

Lemma 7 implies that the solution of the regularized quality functional is a linear combination of hyperkernels on the input data. This shows that even though the optimization takes place over an entire Hilbert space of kernels, one can find the optimal solution by choosing among a finite number.

Note that the minimizer (7) is not necessarily positive semidefinite. In practice, this is not what we want, since we require a positive semidefinite kernel but we do not have any guarantees for examples in the test set. Therefore we need to impose additional constraints of the type $K \succeq 0$ or k is a Mercer Kernel. While the latter is almost impossible to enforce directly, the former could be verified directly, hence imposing a constraint only on the values of the kernel matrix $k(x_i,x_j)$ rather than on the kernel function k itself. This means that the conditions of the Representer Theorem apply and (7) applies (with suitable constraints on the coefficients β_{ij}).

Another option is to be somewhat more restrictive and require that all expansion coefficients $\beta_{i,j} \geqslant 0$ and all the functions be positive semidefinite kernels. This latter requirement can be formally stated as follows: For any fixed $\underline{x} \in \underline{\mathcal{X}}$ the hyperkernel $\underline{k}$ is a kernel in its second argument; that is for any fixed $\underline{x} \in \underline{\mathcal{X}}$, the function $k(x,x') := \underline{k}(\underline{x},(x,x'))$, with $x,x' \in \mathcal{X}$, is a positive semidefinite kernel.

Proposition 8 *Given a hyperkernel, $\underline{k}$ with elements such that for any fixed $\underline{x} \in \underline{\mathcal{X}}$, the function $k(x_p,x_q) := \underline{k}(\underline{x},(x_p,x_q))$, with $x_p,x_q \in \mathcal{X}$, is a positive semidefinite kernel, and $\beta_{ij} \geqslant 0$ for all $i,j = 1,\dots,m$, then the kernel*

$$k(x_p,x_q) := \sum_{i,j=1}^{m} \beta_{ij}\underline{k}(x_i,x_j,x_p,x_q)$$

is positive semidefinite.

Proof The result is obtained by observing that positive combinations of positive semidefinite kernels are positive semidefinite. ■

While this may prevent us from obtaining the minimizer of the objective function, it yields a much more amenable optimization problem in practice, in particular if the resulting cone spans a large enough space (as happens with increasing m). In the subsequent derivations of optimization problems, we choose this restriction as it provides a more tractable problem in practice. In Section 4, we give examples and recipes for constructing hyperkernels. Before that, we relate our framework defined above to Bayesian inference.

3.2 A Bayesian Perspective

A generative Bayesian approach to inference encodes all knowledge we might have about the problem setting into a prior distribution. Hence, the choice of the prior distribution determines the behaviour of the inference, as once we have the data, we condition on the prior distribution we have chosen to obtain the posterior, and then marginalize to obtain the label that we are interested in. One popular choice of prior is the normal distribution, resulting in a Gaussian process (GP). All prior knowledge we have about the problem is then encoded in the covariance of the GP. There exists a GP analog to the support vector machine (for example Opper and Winther (2000), Seeger (1999)), which is essentially obtained (ignoring normalizing terms) by exponentiating the regularized risk functional used in SVMs.

In this section, we derive the prior and hyperprior implied by our framework of hyperkernels. This is obtained by exponentiating Q_{reg}, again ignoring normalization terms. Given the regularized quality functional (Equation 6), with the Q_{emp} set to the SVM with squared loss, we obtain the following equation.

$$Q_{\text{reg}}(k,X,Y) := \frac{1}{m}\sum_{i=1}^{m}(y_i - f(x_i))^2 + \frac{\lambda}{2}\|f\|^2_{\mathcal{H}} + \frac{\lambda_Q}{2}\|k\|^2_{\underline{\mathcal{H}}}.$$

Exponentiating the negative of the above equation gives,

$$\begin{aligned}&\exp(-Q_{\text{reg}}(k,X,Y)) = \\ &\exp\left(-\frac{1}{m}\sum_{i=1}^{m}(y_i - f(x_i))^2\right)\exp\left(-\frac{\lambda}{2}\|f\|^2_{\mathcal{H}}\right)\exp\left(-\frac{\lambda_Q}{2}\|k\|^2_{\underline{\mathcal{H}}}\right).\end{aligned} \tag{8}$$

We compare Equation (8) to Gaussian process estimation. The general scheme is known in Bayesian estimation as hyperpriors (Bishop, 1995, Chapter 10), which determine the distribution of the priors (here the GP with covariance k). Figure 2 describes the model of an ordinary GP, where f is drawn from a Gaussian distribution with covariance matrix K and y is conditionally independent given f. For hyperprior estimation, we draw the prior K from a distribution instead of setting it.

$$\begin{tikzcd}\text{Gaussian Process} & ? \arrow[r, "k \text{ chosen by user}"] & K \arrow[r] & f \arrow[r] & y\end{tikzcd}$$

Figure 2: Generative model for Gaussian process estimation

To determine the distribution from which we draw the prior, we compute the hyperprior explicitly. For given data $Z = \{X, Y\}$ and applying Bayes' Rule, the posterior is given by

$$p(f|Z,k) = \frac{p(Z|f,k)p(f|k)p(k)}{p(k|Z)p(Z)}. \tag{9}$$

We have the directed graphical model shown in Figure 3 for a Hyperkernel-GP, where we assume that the covariance matrix of the Gaussian process K is drawn according to a distribution before performing further steps of dependency calculation. We shall now explicitly compute the terms in the numerator of Equation (9).

Hyperkernel GP
$$\begin{tikzcd} k_0, \underline{k} \arrow[r, "p(k|k_0,\underline{k})"] & k \arrow[r, "p(f|k)"] & f \arrow[r, "p(y|f,x)"] & y \end{tikzcd}$$

Figure 3: Generative model for Gaussian process estimation using hyperpriors on k defined by $\underline{k}$.

In the following derivations, we assume that we are dealing with finite dimensional objects, to simplify the calculations of the normalizing constants in the expressions for the distributions. Given that we have additive Gaussian noise, that is $\varepsilon \sim \mathcal{N}(0, \frac{1}{\gamma_\varepsilon}\mathbf{I})$, then,

$$p(y|f,x) \propto \exp\left(-\frac{\gamma_\varepsilon}{2}(y - f(x))^2\right).$$

Therefore, for the whole data set (assumed to be i.i.d.),

$$p(Y|f,X) = \prod_{i=1}^{m} p(y_i|f,x_i) = \left(\frac{2\pi}{\gamma_\varepsilon}\right)^{-\frac{m}{2}} \exp\left(-\frac{\gamma_\varepsilon}{2}\sum_{i=1}^{m}(y_i - f(x_i))^2\right).$$

We assume a Gaussian prior on the function f, with covariance function k. The positive semidefinite function, k, defines an inner product $\langle\cdot,\cdot\rangle_{\mathcal{H}_k}$ in the RKHS denoted by $\mathcal{H}_k$. Then,

$$p(f|k) = \left(\frac{2\pi}{\gamma_f}\right)^{-\frac{F}{2}} \exp\left(-\frac{\gamma_f}{2}\langle f, f\rangle_{\mathcal{H}_k}\right)$$

where F is the dimension of f and γ_f is a constant.

We assume a Wishart distribution (Lauritzen, 1996, Appendix C), with p degrees of freedom and covariance k_0, for the prior distribution of the covariance function k, that is $k \sim \mathcal{W}_m(p, k_0)$. This is a hyperprior used in the Gaussian process literature:

$$p(k|k_0) = \frac{|k|^{\frac{p-(m+1)}{2}} \exp\left(-\frac{1}{2}\mathrm{tr}(kk_0)\right)}{\Gamma_m(p)|k|^{\frac{p}{2}}}$$

where $\Gamma_m(p)$ denotes the Gamma distribution, $\Gamma_m(p) = 2^{\frac{pm}{2}}\pi^{\frac{m(m-1)}{4}}\prod_{i=1}^{m}\Gamma\left(\frac{p-i+1}{2}\right)$. For more details of the Wishart distribution, the reader is referred to Lauritzen (1996).

Observe that $\mathrm{tr}(kk_0)$ is an inner product between two matrices. We can define a general inner product between two matrices, as the inner product defined in the RKHS denoted by $\underline{\mathcal{H}}$:

$$p(k|k_0,\underline{k}) = \frac{|k|^{\frac{p-(m+1)}{2}} \exp\left(-\frac{1}{2}\langle k, k_0\rangle_{\underline{\mathcal{H}}}\right)}{\Gamma_m(p)|k|^{\frac{p}{2}}}.$$

We can interpret the above equation as measuring the similarity between the covariance matrix that we obtain from data and the expected covariance matrix (given by the user). This similarity is measured by a dot product defined by $\underline{k}$. Substituting the expressions for $p(Y|X,f), p(f|k)$ and $p(k|k_0,\underline{k})$ into the posterior (Equation 9), we get Equation (10) which is of the same form as the exponentiated negative quality (Equation 8):

$$\exp\left(-\frac{\gamma_\varepsilon}{2}\sum_{i=1}^{m}(y_i - f(x_i))^2\right)\exp\left(-\frac{\gamma_f}{2}\langle f,f\rangle_{\mathcal{H}_k}\right)\exp\left(-\frac{1}{2}\langle k,k_0\rangle_{\underline{\mathcal{H}}}\right). \tag{10}$$

In a nutshell, we assume that the covariance function of the GP k, is distributed according to a Wishart distribution. In other words, we have two nested processes, a Gaussian and a Wishart process, to model the data generation scheme. Hence we are studying a mixture of Gaussian processes. Note that the maximum likelihood (ML-II) estimator (MacKay, 1994, Williams and Barber, 1998, Williams and Rasmussen, 1996) in Bayesian estimation leads to the same optimization problems as those arising from minimizing the regularized quality functional.

4. Hyperkernels

Having introduced the theoretical basis of the Hyper-RKHS, it is natural to ask whether hyperkernels, $\underline{k}$, exist which satisfy the conditions of Definition 5. We address this question by giving a set of general recipes for building such kernels.

4.1 Power Series Construction

Suppose k is a kernel such that $k(x,x') \geq 0$ for all $x,x' \in \mathcal{X}$, and suppose $g : \mathbb{R} \to \mathbb{R}$ is a function with positive Taylor expansion coefficients, that is $g(\xi) = \sum_{i=0}^{\infty} c_i \xi^i$ for basis functions ξ, $c_i \geqslant 0$ for all $i = 0,\ldots,\infty$, and convergence radius R. Then for pointwise positive $k(x,x') \leq \sqrt{R}$,

$$\underline{k}(\underline{x},\underline{x}') := g(k(\underline{x})k(\underline{x}')) = \sum_{i=0}^{\infty} c_i (k(\underline{x})k(\underline{x}'))^i \tag{11}$$

is a hyperkernel. For $\underline{k}$ to be a hyperkernel, we need to check that first, $\underline{k}$ is a kernel, and second, for any fixed pair of elements of the input data, $\underline{x}$, the function $\underline{k}(\underline{x},(x,x'))$ is a kernel, and third that is satisfies the symmetry condition. Here, the symmetry condition follows from the symmetry of k. To see this, observe that for any fixed $\underline{x}$, $\underline{k}(\underline{x},(x,x'))$ is a sum of kernel functions, hence it is a kernel itself (since $k^p(x,x')$ is a kernel if k is, for $p \in \mathbb{N}$). To show that $\underline{k}$ is a kernel, note that $\underline{k}(\underline{x},\underline{x}') = \langle \underline{\Phi}(\underline{x}), \underline{\Phi}(\underline{x}')\rangle$, where $\underline{\Phi}(\underline{x}) := (\sqrt{c_0}, \sqrt{c_1}k^1(\underline{x}), \sqrt{c_2}k^2(\underline{x}),\ldots)$. Note that we require pointwise positivity, so that the coefficients of the sum in Equation (11) are always positive. The Gaussian RBF kernel satisfies this condition, but polynomial kernels of odd degree are not always pointwise positive. In the following example, we use the Gaussian kernel to construct a hyperkernel.

Example 4 (Harmonic Hyperkernel) *Suppose k is a kernel with range* $[0,1]$, *(RBF kernels satisfy this property), and set* $c_i := (1-\lambda_h)\lambda_h^i$, $i \in \mathbb{N}$, *for some* $0 < \lambda_h < 1$. *Then we have*

$$\underline{k}(\underline{x},\underline{x}') = (1-\lambda_h)\sum_{i=0}^{\infty}\left(\lambda_h k(\underline{x})k(\underline{x}')\right)^i = \frac{1-\lambda_h}{1-\lambda_h k(\underline{x})k(\underline{x}')}. \tag{12}$$

For $k(x,x') = \exp(-\sigma^2\|x-x'\|^2)$ this construction leads to

$$\underline{k}((x,x'),(x'',x''')) = \frac{1-\lambda_h}{1-\lambda_h \exp\left(-\sigma^2(\|x-x'\|^2+\|x''-x'''\|^2)\right)}. \tag{13}$$

As one can see, for $\lambda_h \to 1$, $\underline{k}$ converges to $\delta_{\underline{x},\underline{x}'}$, and thus $\|k\|^2_{\underline{\mathcal{H}}}$ converges to the Frobenius norm of k on $X \times X$.

It is straightforward to find other hyperkernels of this sort, simply by consulting tables on power series of functions. Table 2 contains a short list of suitable expansions.

$g(\xi)$	Power series expansion	Radius of Convergence
$\exp\xi$	$1+\frac{1}{1!}\xi+\frac{1}{2!}\xi^2+\frac{1}{3!}\xi^3+\ldots+\frac{1}{n!}\xi^n+\ldots$	∞
$\sinh\xi$	$\frac{1}{1!}\xi+\frac{1}{3!}\xi^3+\frac{1}{5!}\xi^5+\ldots+\frac{1}{(2n+1)!}\xi^{(2n+1)}+\ldots$	∞
$\cosh\xi$	$1+\frac{1}{2!}\xi^2+\frac{1}{4!}\xi^4+\ldots+\frac{1}{(2n)!}\xi^{(2n)}+\ldots$	∞
$\operatorname{arctanh}\xi$	$\frac{\xi}{1}+\frac{\xi^3}{3}+\frac{\xi^5}{5}+\ldots+\frac{\xi^{2n+1}}{2n+1}+\ldots$	1
$-\ln(1-\xi)$	$\frac{\xi}{1}+\frac{\xi^2}{2}+\frac{\xi^3}{3}+\ldots+\frac{\xi^n}{n}+\ldots$	1

Table 2: Hyperkernels by Power Series Construction.

However, if we want the kernel to adapt automatically to different widths for each dimension, we need to perform the summation that led to (12) for each dimension in its arguments separately. Such a hyperkernel corresponds to ideas developed in automatic relevance determination (ARD) (MacKay, 1994, Neal, 1996).

Example 5 (Hyperkernel for ARD) *Let $k_\Sigma(x,x') = \exp(-d_\Sigma(x,x'))$, where $d_\Sigma(x,x') = (x-x')^\top\Sigma(x-x')$, and Σ is a diagonal covariance matrix. Take sums over each diagonal entry $\sigma_j = \Sigma_{jj}$ separately to obtain*

$$\begin{aligned}\underline{k}((x,x'),(x'',x''')) &= (1-\lambda_h)\sum_{j=1}^{d}\sum_{i=0}^{\infty}\left(\lambda_h k_\Sigma(x,x')k_\Sigma(x'',x''')\right)^i \\ &= \prod_{j=1}^{d}\frac{1-\lambda_h}{1-\lambda_h\exp\left(-\sigma_j((x_j-x'_j)^2+(x''_j-x'''_j)^2)\right)}.\end{aligned} \tag{14}$$

Eq. (14) holds since $k(\underline{x})$ factorizes into its coordinates. A similar definition also allows us to use a distance metric $d(x,x')$ which is a generalized radial distance as defined by Haussler (1999).

4.2 Hyperkernels Invariant to Translation

Another approach to constructing hyperkernels is via an extension of a result due to Smola et al. (1998) concerning the Fourier transform of translation invariant kernels.

Theorem 9 (Translation Invariant Hyperkernel) *Suppose $\underline{k}((x_1-x'_1),(x_2-x'_2))$ is a function which depends on its arguments only via $x_1-x'_1$ and $x_2-x'_2$. Let $\mathcal{F}_1\underline{k}(\omega,(x_2-x'_2))$ denote the Fourier transform with respect to $(x_1-x'_1)$.*

The function $\underline{k}$ is a hyperkernel if $\underline{k}(\tau,\tau')$ is a kernel in τ,τ' and $\mathcal{F}_1\underline{k}(\omega,(x''-x''')) \geq 0$ for all $(x''-x''')$ and ω.

Proof From (Smola et al., 1998) we know that for $\underline{k}$ to be a kernel in one of its arguments, its Fourier transform has to be nonnegative. This yields the second condition. Next, we need to show that $\underline{k}$ is a kernel in its own right. Mercer's condition requires that for arbitrary f the following is positive:

$$\begin{aligned}
& \int f(x_1,x_1')f(x_2,x_2')\underline{k}((x_1-x_1'),(x_2-x_2'))dx_1dx_1'dx_2dx_2' \\
= \quad & \int f(\tau_1+x_1',x_1')f(\tau_2+x_2',x_2')dx_{1,2}\underline{k}(\tau_1,\tau_2)d\tau_1d\tau_2 \\
= \quad & \int g(\tau_1)g(\tau_2)\underline{k}(\tau_1,\tau_2)d\tau_1d\tau_2,
\end{aligned}$$

where $\tau_1 = x_1 - x_1'$ and $\tau_2 = x_2 - x_2'$. Here g is obtained by integration over x_1 and x_2 respectively. The latter is exactly Mercer's condition on $\underline{k}$, when viewed as a function of two variables only. ■

This means that we can check whether a radial basis function (for example Gaussian RBF, exponential RBF, damped harmonic oscillator, generalized B_n spline), can be used to construct a hyperkernel by checking whether its Fourier transform is positive.

4.3 Explicit Expansion

If we have a finite set of kernels that we want to choose from, we can generate a hyperkernel which is a finite sum of possible kernel functions. This setting is similar to that of Lanckriet et al. (2004).

Suppose $k_i(x,x')$ is a kernel for each $i = 1,\dots,n$ (for example the RBF kernel or the polynomial kernel), then

$$\underline{k}(\underline{x},\underline{x}') := \sum_{i=1}^{n} c_i k_i(\underline{x})k_i(\underline{x}'), k_i(\underline{x}) \geqslant 0, \forall \underline{x} \tag{15}$$

is a hyperkernel, as can be seen by an argument similar to that of section 4.1. $\underline{k}$ is a kernel since $\underline{k}(\underline{x},\underline{x}') = \langle \underline{\Phi}(\underline{x}), \underline{\Phi}(\underline{x}')\rangle$, where $\underline{\Phi}(\underline{x}) := (\sqrt{c_1}k_1(\underline{x}), \sqrt{c_2}k_2(\underline{x}), \dots, \sqrt{c_n}k_n(\underline{x}))$.

Example 6 (Polynomial and RBF combination) *Let $k_1(x,x') = (\langle x,x'\rangle + b)^{2p}$ for some choice of $b \in \mathbb{R}^+$ and $p \in \mathbb{N}$, and $k_2(x,x') = \exp(-\sigma^2\|x-x'\|^2)$. Then,*

$$\begin{aligned}
\underline{k}((x_1,x_1'),(x_2,x_2')) \quad & = c_1(\langle x_1,x_1'\rangle + b)^{2p}(\langle x_2,x_2'\rangle + b)^{2p} \\
& + c_2 \exp(-\sigma^2\|x_1-x_1'\|^2)\exp(-\sigma^2\|x_2-x_2'\|^2)
\end{aligned} \tag{16}$$

is a hyperkernel.

5. Optimization Problems for Regularized Risk based Quality Functionals

We will now consider the optimization of the quality functionals utilizing hyperkernels. We choose the regularized risk functional as the empirical quality functional; that is we set $Q_{\text{emp}}(k,X,Y) := R_{\text{reg}}(f,X,Y)$. It is possible to utilize other quality functionals, such as the Kernel Target Alignment (Example 12). We focus our attention on the regularized risk functional, which is commonly used in SVMs. Furthermore, we will only consider positive semidefinite kernels. For a particular loss function $l(x_i,y_i,f(x_i))$, we obtain the regularized quality functional.

$$\min_{k\in\underline{\mathcal{H}}}\min_{f\in\mathcal{H}_k} \frac{1}{m}\sum_{i=1}^{m} l(x_i,y_i,f(x_i)) + \frac{\lambda}{2}\|f\|^2_{\mathcal{H}_k} + \frac{\lambda_Q}{2}\|k\|^2_{\underline{\mathcal{H}}}. \tag{17}$$

By the representer theorem (Theorem 4 and Corollary 7) we can write the regularizers as quadratic terms. Using the soft margin loss, we obtain

$$\min_{\beta}\min_{\alpha}\frac{1}{m}\sum_{i=1}^{m}\max(0,1-y_i f(x_i))+\frac{\lambda}{2}\alpha^{\top}K\alpha+\frac{\lambda_Q}{2}\beta^{\top}\underline{K}\beta \text{ subject to } \beta\geqslant 0 \tag{18}$$

where $\alpha\in\mathbb{R}^m$ are the coefficients of the kernel expansion (5), and $\beta\in\mathbb{R}^{m^2}$ are the coefficients of the hyperkernel expansion (7).

For fixed k, the problem can be formulated as a constrained minimization problem in f, and subsequently expressed in terms of the Lagrange multipliers α. However, this minimum depends on k, and for efficient minimization we would like to compute the derivatives with respect to k. The following lemma tells us how (it is an extension of a result in Chapelle et al. (2002)):

Lemma 10 *Let $x\in\mathbb{R}^m$ and denote by $f(x,\theta),c_i:\mathbb{R}^m\to\mathbb{R}$ convex functions, where f is parameterized by θ. Let $R(\theta)$ be the minimum of the following optimization problem (and denote by $x(\theta)$ its minimizer):*

$$\underset{x\in\mathbb{R}^m}{\textit{minimize}}\ f(x,\theta)\ \textit{subject to}\ c_i(x)\leq 0\ \textit{for all}\ 1\leq i\leq n.$$

Then $\partial_\theta^j R(\theta)=D_2^j f(x(\theta),\theta)$, where $j\in\mathbb{N}$ and D_2 denotes the derivative with respect to the second argument of f.

Proof At optimality we have a saddlepoint in the Lagrangian

$$\partial_x\mathcal{L}(x,\alpha)=\partial_x f(x,\theta)+\sum_{i=1}^{n}\alpha_i\partial_x c_i(x)=0. \tag{19}$$

Furthermore, for all θ the Kuhn-Tucker conditions have to hold, and in particular also $\sum_{i=1}^{n}\alpha_i\partial_\theta c_i(x(\theta))=0$, since for all $\alpha_i>0$ the condition $c_i(x)=0$ and therefore also $\partial_\theta c_i(x(\theta))=0$ has to be satisfied. Taking higher order derivatives with respect to θ yields

$$0=\partial_\theta^j\left[\sum_{i=1}^{n}\alpha_i\partial_x c_i(x(\theta))\frac{\partial x}{\partial\theta}\right]=\partial_\theta^j\left[-\partial_x f(x,\theta)\frac{\partial x}{\partial\theta}\right]. \tag{20}$$

Here the last equality follows from (19). Next we use

$$\partial_\theta^{j+1}f(x,\theta)=\partial_\theta^j\left[D_2 f(x,\theta)+\partial_x f(x,\theta)\frac{\partial x}{\partial\theta}\right]=\partial_\theta^j D_2 f(x,\theta).$$

Repeated application then proves the claim. ■

Instead of directly minimizing Equation (18), we derive the dual formulation. Using the approach in Lanckriet et al. (2004), the corresponding optimization problems can be expressed as a SDP. In general, solving a SDP would be take longer than solving a quadratic program (a traditional SVM is a quadratic program). This reflects the added cost incurred for optimizing over a class of kernels.

Semidefinite programming (Vandenberghe and Boyd, 1996) is the optimization of a linear objective function subject to constraints which are linear matrix inequalities and affine equalities.

Definition 11 (Semidefinite Program) *A semidefinite program (SDP) is a problem of the form:*

$$\begin{array}{ll} \min\limits_{x} & c^\top x \\ \textit{subject to} & F_0 + \sum_{i=1}^{q} x_i F_i \succeq 0 \textit{ and } Ax = b \end{array}$$

where $x \in \mathbb{R}^p$ are the decision variables, $A \in \mathbb{R}^{p\times q}$, $b \in \mathbb{R}^p$, $c \in \mathbb{R}^q$, and $F_i \in \mathbb{R}^{r\times r}$ are given.

In general, linear constraints $Ax + a \geqslant 0$ can be expressed as a semidefinite constraint $diag(Ax + a) \succeq 0$, and a convex quadratic constraint $(Ax+b)^\top(Ax+b) - c^\top x - d \leqslant 0$ can be written as

$$\begin{bmatrix} I & Ax+b \\ (Ax+b)^\top & c^\top x + d \end{bmatrix} \succeq 0.$$

When $t \in \mathbb{R}$, we can write the quadratic constraint $a^\top Aa \leqslant t$ as $\|A^{\frac{1}{2}}a\| \leqslant t$. In practice, linear and quadratic constraints are simpler and faster to implement in a convex solver.

We derive the corresponding SDP for Equation (17). The following proposition allows us to derive a SDP from a class of general convex programs. It follows the approach in Lanckriet et al. (2004), with some care taken with Schur complements of positive semidefinite matrices (Albert, 1969), and its proof is omitted for brevity.

Proposition 12 (Quadratic Minimax) *Let $m, n, M \in \mathbb{N}$, $H : \mathbb{R}^n \to \mathbb{R}^{m\times m}$, $c : \mathbb{R}^n \to \mathbb{R}^m$, be linear maps. Let $A \in \mathbb{R}^{M\times m}$ and $a \in \mathbb{R}^M$. Also, let $d : \mathbb{R}^n \to \mathbb{R}$ and $G(\xi)$ be a function and the further constraints on ξ. Then the optimization problem*

$$\begin{array}{ll} \underset{\xi\in\mathbb{R}^n}{\text{minimize}}\,\underset{x\in\mathbb{R}^m}{\text{maximize}} & -\frac{1}{2}x^\top H(\xi)x - c(\xi)^\top x + d(\xi) \\ \textit{subject to} & H(\xi) \succeq 0 \\ & Ax + a \geqslant 0 \\ & G(\xi) \succeq 0 \end{array} \tag{21}$$

can be rewritten as

$$\begin{array}{ll} \underset{t,\xi,\gamma}{\text{minimize}} & \frac{1}{2}t + a^\top\gamma + d(\xi) \\ \textit{subject to} & \begin{bmatrix} diag(\gamma) & 0 & 0 & 0 \\ 0 & G(\xi) & 0 & 0 \\ 0 & 0 & H(\xi) & (A^\top\gamma - c(\xi)) \\ 0 & 0 & (A^\top\gamma - c(\xi))^\top & t \end{bmatrix} \succeq 0 \end{array} \tag{22}$$

in the sense that the ξ which solves (22) also solves (21).

Specifically, when we have the regularized quality functional, $d(\xi)$ is quadratic, and hence we obtain an optimization problem which has a mix of linear, quadratic and semidefinite constraints.

Corollary 13 *Let H, c, A and a be as in Proposition 12, and $\Sigma \succeq 0$. Then the solution ξ^* to the optimization problem*

$$\begin{array}{ll} \underset{\xi}{\text{minimize}}\,\underset{x}{\text{maximize}} & -\frac{1}{2}x^\top H(\xi)x - c(\xi)^\top x + \frac{1}{2}\xi^\top\Sigma\xi \\ \textit{subject to} & H(\xi) \succeq 0 \\ & Ax + a \geqslant 0 \\ & \xi \geqslant 0 \end{array} \tag{23}$$

can be found by solving the semidefinite programming problem

$$\begin{aligned} \underset{t,t',\xi,\gamma}{\text{minimize}} \quad & \tfrac{1}{2}t + \tfrac{1}{2}t' + a^\top \gamma \\ \textit{subject to} \quad & \gamma \geqslant 0 \\ & \xi \geqslant 0 \\ & \|\Sigma^{\frac{1}{2}}\xi\|^2 \leqslant t' \\ & \begin{bmatrix} H(\xi) & (A^\top\gamma - c(\xi)) \\ (A^\top\gamma - c(\xi))^\top & t \end{bmatrix} \succeq 0 \end{aligned} \tag{24}$$

Proof By applying proposition 12, and introducing an auxiliary variable t' which upper bounds the quadratic term of ξ, the claim is proved. ■

Comparing the objective function in (21) with (18), we observe that $H(\xi)$ and $c(\xi)$ are linear in ξ. Let $\xi' = \varepsilon\xi$. As we vary ε the constraints are still satisfied, but the objective function scales with ε. Since ξ is the coeffient in the hyperkernel expansion, this implies that we have a set of possible kernels which are just scalar multiples of each other. To avoid this, we add an additional constraint on ξ which is $\mathbf{1}^\top\xi = c$, where c is a constant. This breaks the scaling freedom of the kernel matrix. As a side-effect, the numerical stability of the SDP problems improves considerably. We chose a linear constraint so that it does not add too much overhead to the optimization problem We make one additional simplification of the optimization problem, which is to replace the upper bound of the squared norm ($\|\Sigma^{\frac{1}{2}}\xi\|^2 \leqslant t'$) with and upper bound on the norm ($\|\Sigma^{\frac{1}{2}}\xi\| \leqslant t'$).

In our setting, the regularizer for controlling the complexity of the kernel is taken to be the squared norm of the kernel in the Hyper-RKHS. By looking at the constraints of Equation (24), this is expressed as a bound on the norm ($\|\Sigma^{\frac{1}{2}}\xi\| \leqslant t'$). Comparing this result to the SDP obtained in Lanckriet et al. (2004, Theorem 16), we see that the corresponding regularizer in their setting is $\operatorname{tr}(K) = c$, where c is a constant. Hence the main difference between the two SDPs is the choice of the regularizer for the kernel. However, the motivations of the two methods are different. This paper sets out an induction framework for learning the kernel, and for a particular choice of Q_{emp}, namely the regularized risk functional, we obtain an SDP which has similarities to the approach of Lanckriet et al. (2004). On the other hand, they start out with a transduction problem and derive the optimization problem directly. It is unclear at this point which is the better approach.

From the general framework above (Corollary 13, we derive several examples of machine learning problems, specifically binary classification, regression, and single class (also known as novelty detection) problems. The following examples illustrate our method for simultaneously optimizing over the class of kernels induced by the hyperkernel, as well as the hypothesis class of the machine learning problem. We consider machine learning problems based on kernel methods which are derived from (17). The derivation is essentially by application of Corollary 13 with the two additional conditions above.

6. Examples of Hyperkernel Optimization Problems

In this section, we define the following notation. For $p,q,r \in \mathbb{R}^n, n \in \mathbb{N}$ let $r = p \circ q$ be defined as element by element multiplication, $r_i = p_i \times q_i$ (the Hadamard product, or the $.*$ operation in Matlab). The pseudo-inverse (also known as the Moore-Penrose inverse) of a matrix K is denoted $K^\dagger$. Let $\vec{K}$ be the m^2 by 1 vector formed by concatenating the columns of an m by m matrix.

We define the hyperkernel Gram matrix $\underline{K}$ by putting together m^2 of these vectors, that is we set $\underline{K} = [\vec{K}_{pq}]_{p,q=1}^m$. Other notations include: the kernel matrix $K = \text{reshape}(\underline{K}\beta)$ (reshaping a m^2 by 1 vector, $\underline{K}\beta$, to a m by m matrix), $Y = diag(y)$ (a matrix with y on the diagonal and zero everywhere else), $G(\beta) = YKY$ (the dependence on β is made explicit), $\mathbf{I}$ the identity matrix, $\mathbf{1}$ a vector of ones and $\mathbf{1}_{m\times m}$ a matrix of ones. Let w be the weight vector and b_{offset} the bias term in feature space, that is the hypothesis function in feature space is defined as $g(x) = w^\top\phi(x) + b_{offset}$ where $\phi(\cdot)$ is the feature mapping defined by the kernel function k.

The number of training examples is assumed to be m, that is $X_{\text{train}} = \{x_1,\ldots,x_m\}$ and $Y_{\text{train}} = y = \{y_1,\ldots,y_m\}$. Where appropriate, γ and χ are Lagrange multipliers, while η and ξ are vectors of Lagrange multipliers from the derivation of the Wolfe dual for the SDP, β are the hyperkernel coefficients, t_1 and t_2 are the auxiliary variables. When $\eta \in \mathbb{R}^m$, we define $\eta \geqslant 0$ to mean that each $\eta_i \geqslant 0$ for $i = 1,\ldots,m$.

We derive the corresponding SDP for the case when Q_{emp} is a C-SVM (Example 7). Derivations of the other examples follow the same reasoning, and are omitted.

Example 7 (Linear SVM (C-parameterization)) *A commonly used support vector classifier, the C-SVM (Bennett and Mangasarian, 1992, Cortes and Vapnik, 1995) uses an ℓ_1 soft margin, $l(x_i,y_i,f(x_i)) = \max(0, 1-y_if(x_i))$, which allows errors on the training set. The parameter C is given by the user. Setting the quality functional $Q_{\text{emp}}(k,X,Y) = \min_{f\in\mathcal{H}} \frac{C}{m}\sum_{i=1}^m l(x_i,y_i,f(x_i)) + \frac{1}{2}\|w\|_{\mathcal{H}}^2$,*

$$\begin{aligned}\min_{k\in\underline{\mathcal{H}}}\min_{f\in\mathcal{H}_k} \quad & \frac{C}{m}\sum_{i=1}^m \zeta_i + \frac{1}{2}\|f\|_{\mathcal{H}_k}^2 + \frac{\lambda_Q}{2}\|k\|_{\underline{\mathcal{H}}}^2 \\ \textit{subject to} \quad & y_if(x_i) \geqslant 1-\zeta_i \\ & \zeta_i \geqslant 0\end{aligned} \tag{25}$$

Recall the dual form of the C-SVM,

$$\begin{aligned}\max_{\alpha\in\mathbb{R}^m} \quad & \textstyle\sum_{i=1}^m \alpha_i - \frac{1}{2}\sum_{i=1}^m \alpha_i\alpha_jy_iy_jk(x_i,x_j) \\ \textit{subject to} \quad & \textstyle\sum_{i=1}^m \alpha_iy_i = 0 \\ & 0 \leqslant \alpha_i \leqslant \tfrac{C}{m} \textit{ for all } i = 1,\ldots,m.\end{aligned}$$

By considering the optimization problem dependent on f in (25), we can use the derivation of the dual problem of the standard C-SVM. Observe that we can rewrite $\|k\|_{\underline{\mathcal{H}}}^2 = \beta^\top\underline{K}\beta$ due to the representer theorem for hyperkernels. Substituting the dual C-SVM problem into (25), we get the following matrix equation,

$$\begin{aligned}\min_{\beta}\max_{\alpha} \quad & \mathbf{1}^\top\alpha - \tfrac{1}{2}\alpha^\top G(\beta)\alpha + \tfrac{\lambda_Q}{2}\beta^\top\underline{K}\beta \\ \textit{subject to} \quad & \alpha^\top y = 0 \\ & 0 \leqslant \alpha \leqslant \tfrac{C}{m} \\ & \beta \geqslant 0\end{aligned} \tag{26}$$

This is of the quadratic form of Corollary 13 where $x = \alpha$, $\theta = \beta$, $H(\theta) = G(\beta)$, $c(\theta) = -\mathbf{1}$, $\Sigma = C\lambda_Q\underline{K}$, the constraints are $A = \begin{bmatrix} y & -y & \mathbf{I} & -\mathbf{I}\end{bmatrix}^\top$ and $a = \begin{bmatrix} 0 & 0 & \mathbf{0} & \frac{C}{m}\mathbf{1}\end{bmatrix}^\top$. Applying Corollary 13, we obtain the corresponding SDP.

The proof of Proposition 12 uses the Lagrange method. As an illustration of how this proof proceeds, we derive it for this special case of the C-SVM. The Lagrangian associated with (26) is

$$\mathfrak{L}(\alpha,\beta,\gamma,\eta,\xi) = \mathbf{1}^\top\alpha - \frac{1}{2}\alpha^\top G(\beta)\alpha + \frac{\lambda_Q}{2}\beta^\top \underline{K}\beta + \gamma y^\top\alpha + \eta^\top\alpha - \xi^\top(\alpha - \frac{C}{m}\mathbf{1}),$$

where $\beta \geqslant 0, \eta \geqslant 0, \xi \geqslant 0$*. The minimum is achieved at*

$$\alpha = G(\beta)^\dagger(\gamma y + \mathbf{1} + \eta - \xi),$$

and the corresponding dual optimization problem is

$$\underset{\beta,\gamma,\eta,\xi}{\text{minimize}} \frac{1}{2}z^\top G(\beta)^\dagger z + \frac{C}{m}\xi^\top\mathbf{1} + \frac{\lambda_Q}{2}\beta^\top\underline{K}\beta,$$

where $z = \gamma y + \mathbf{1} + \eta - \xi$*. From this point, we replace the quadratic terms with auxiliary variables* t_1 *and* t_2*, and apply the Schur complement lemma (Albert, 1969). The resulting SDP after replacing* $\|\underline{K}^{\frac{1}{2}}\beta\|^2 \leqslant t_2$ *by* $\|\underline{K}^{\frac{1}{2}}\beta\| \leqslant t_2$*, and introducing the scale breaking constraint* $\mathbf{1}^\top\beta = 1$ *is*

$$\begin{array}{ll} \underset{\beta,\gamma,\eta,\xi}{\text{minimize}} & \frac{1}{2}t_1 + \frac{C}{m}\xi^\top\mathbf{1} + \frac{\lambda_Q}{2}t_2 \\ \textit{subject to} & \eta \geqslant 0, \xi \geqslant 0, \beta \geqslant 0 \\ & \|\underline{K}^{\frac{1}{2}}\beta\| \leqslant t_2, \mathbf{1}^\top\beta = 1 \\ & \begin{bmatrix} G(\beta) & z \\ z^\top & t_1 \end{bmatrix} \succeq 0. \end{array} \tag{27}$$

Note that the value of the support vector coefficients, α*, which optimizes the corresponding Lagrange function is* $G(\beta)^\dagger z$*, and the classification function,* $f = sign(K(\alpha\circ y) - b_{offset})$*, is given by* $f = sign(KG(\beta)^\dagger(y\circ z) - \gamma)$*.*

Example 8 (Linear SVM (ν-parameterization)) *An alternative parameterization of the* ℓ_1 *soft margin was introduced by Schölkopf et al. (2000), where the user defined parameter* $\nu \in [0,1]$ *controls the fraction of margin errors and support vectors. Using ν-SVM as* Q_{emp}*, that is, for a given* ν*,* $Q_{\text{emp}}(k,X,Y) = \min_{f\in\mathcal{H}} \frac{1}{m}\sum_{i=1}^m \zeta_i + \frac{1}{2}\|w\|^2_{\mathcal{H}} - \nu\rho$ *subject to* $y_i f(x_i) \geqslant \rho - \zeta_i$ *and* $\zeta_i \geqslant 0$ *for all* $i = 1,\dots,m$*, the corresponding SDP is given by*

$$\begin{array}{ll} \underset{\beta,\gamma,\eta,\xi,\chi}{\text{minimize}} & \frac{1}{2}t_1 - \chi\nu + \xi^\top\frac{1}{m} + \frac{\lambda_Q}{2}t_2 \\ \textit{subject to} & \chi \geqslant 0, \eta \geqslant 0, \xi \geqslant 0, \beta \geqslant 0 \\ & \|\underline{K}^{\frac{1}{2}}\beta\| \leqslant t_2, \mathbf{1}^\top\beta = 1 \\ & \begin{bmatrix} G(\beta) & z \\ z^\top & t_1 \end{bmatrix} \succeq 0 \end{array} \tag{28}$$

where $z = \gamma y + \chi\mathbf{1} + \eta - \xi$*.*

The value of α *which optimizes the corresponding Lagrange function is* $G(\beta)^\dagger z$*, and the classification function,* $f = sign(K(\alpha\circ y) - b_{offset})$*, is given by* $f = sign(KG(\beta)^\dagger(y\circ z) - \gamma)$*.*

Example 9 (Quadratic SVM or Lagrangian SVM) *Instead of using an ℓ_1 loss class, Mangasarian and Musicant (2001) use an ℓ_2 loss class,*

$$l(x_i,y_i,f(x_i)) = \begin{cases} 0 & \textit{if } y_i f(x_i) \geqslant 1 \\ (1-y_i f(x_i))^2 & \textit{otherwise} \end{cases},$$

and regularized the weight vector as well as the bias term. The empirical quality functional derived from this is $Q_{\mathrm{emp}}(k,X,Y) = \min_{f\in\mathcal{H}} \frac{1}{m}\sum_{i=1}^m \zeta_i^2 + \frac{1}{2}(\|w\|_{\mathcal{H}}^2 + b_{offset}^2)$ subject to $y_i f(x_i) \geqslant 1-\zeta_i$ and $\zeta_i \geqslant 0$ for all $i = 1,\ldots,m$. The resulting dual SVM problem has fewer constraints, as is evidenced by the smaller number of Lagrange multipliers needed in the corresponding SDP below.

$$\begin{array}{ll} \underset{\beta,\eta}{\text{minimize}} & \frac{1}{2}t_1 + \frac{\lambda_Q}{2}t_2 \\ \textit{subject to} & \eta \geqslant 0, \beta \geqslant 0 \\ & \|\underline{K}^{\frac{1}{2}}\beta\| \leqslant t_2, \mathbf{1}^\top\beta = 1 \\ & \begin{bmatrix} H(\beta) & (\eta+\mathbf{1}) \\ (\eta+\mathbf{1})^\top & t_1 \end{bmatrix} \succeq 0 \end{array} \tag{29}$$

where $H(\beta) = Y(K + \mathbf{1}_{m\times m} + \lambda m\mathbf{I})Y$, and $z = \gamma\mathbf{1} + \eta - \xi$.

The value of α which optimizes the corresponding Lagrange function is $H(\beta)^\dagger(\eta+\mathbf{1})$, and the classification function, $f = sign(K(\alpha\circ y) - b_{offset})$, is given by $f = sign(KH(\beta)^\dagger((\eta+\mathbf{1})\circ y) + y^\top(H(\beta)^\dagger(\eta+\mathbf{1})))$.

Example 10 (Single class SVM or Novelty Detection) *For unsupervised learning, the single class SVM computes a function which captures regions in input space where the probability density is in some sense large (Schölkopf et al., 2001). A suitable quality functional $Q_{\mathrm{emp}}(k,X,Y) = \min_{f\in\mathcal{H}} \frac{1}{\nu m}\sum_{i=1}^m \zeta_i + \frac{1}{2}\|w\|_{\mathcal{H}}^2 - \rho$ subject to $f(x_i) \geqslant \rho - \zeta_i$, and $\zeta_i \geqslant 0$ for all $i = 1,\ldots,m$, and $\rho \geqslant 0$. The corresponding SDP for this problem is*

$$\begin{array}{ll} \underset{\beta,\gamma,\eta,\xi}{\text{minimize}} & \frac{1}{2}t_1 + \xi^\top\frac{1}{\nu m} - \gamma + \frac{\lambda_Q}{2\nu}t_2 \\ \textit{subject to} & \eta \geqslant 0, \xi \geqslant 0, \beta \geqslant 0 \\ & \|\underline{K}^{\frac{1}{2}}\beta\| \leqslant t_2 \\ & \begin{bmatrix} K & z \\ z^\top & t_1 \end{bmatrix} \succeq 0 \end{array} \tag{30}$$

where $z = \gamma\mathbf{1} + \eta - \xi$, and $\nu \in [0,1]$ is a user selected parameter controlling the proportion of the data to be classified as novel.

The score to be used for novelty detection is given by $f = K\alpha - b_{offset}$, which reduces to $f = \eta - \xi$, by substituting $\alpha = K^\dagger(\gamma\mathbf{1} + \eta - \xi)$, $b_{offset} = \gamma\mathbf{1}$ and $K = reshape(\underline{K}\beta)$.

Example 11 (ν-Regression) *We derive the SDP for ν regression (Schölkopf et al., 2000), which automatically selects the ε insensitive tube for regression. As in the ν-SVM case in Example 8, the user defined parameter ν controls the fraction of errors and support vectors. Using the ε-insensitive loss, $l(x_i,y_i,f(x_i)) = \max(0,|y_i - f(x_i)| - \varepsilon)$, and the ν-parameterized quality functional, $Q_{\mathrm{emp}}(k,X,Y) = \min_{f\in\mathcal{H}} C\left(\nu\varepsilon + \frac{1}{m}\sum_{i=1}^m(\zeta_i+\zeta_i^*)\right)$ subject to $f(x_i) - y_i \leqslant \varepsilon - \zeta_i$, $y_i - f(x_i) \leqslant \varepsilon - \zeta_i^*$,*

$\zeta_i^{(*)} \geqslant 0$ *for all* $i = 1, \ldots, m$ *and* $\varepsilon \geqslant 0$, *the corresponding SDP is*

$$\begin{array}{ll} \underset{\beta,\gamma,\eta,\xi,\chi}{\text{minimize}} & \frac{1}{2}t_1 + \frac{\chi\nu}{\lambda} + \xi^\top \frac{1}{m\lambda} + \frac{\lambda_Q}{2\lambda}t_2 \\ \textit{subject to} & \chi \geqslant 0, \eta \geqslant 0, \xi \geqslant 0, \beta \geqslant 0 \\ & \|\underline{K}^{\frac{1}{2}}\beta\| \leqslant t_2, \mathbf{1}^\top\beta = \text{stddev}(Y_{train}) \\ & \begin{bmatrix} F(\beta) & z \\ z^\top & t_1 \end{bmatrix} \succeq 0 \end{array}, \tag{31}$$

where $z = \begin{bmatrix} -y \\ y \end{bmatrix} - \gamma \begin{bmatrix} \mathbf{1} \\ -\mathbf{1} \end{bmatrix} + \eta - \xi - \chi \begin{bmatrix} \mathbf{1} \\ \mathbf{1} \end{bmatrix}$ *and* $F(\beta) = \begin{bmatrix} K & -K \\ -K & K \end{bmatrix}$.

The Lagrange function is minimized for $\alpha = F(\beta)^\dagger z$, *and substituting into* $f = K\alpha - b_{offset}$, *we obtain the regression function* $f = \begin{bmatrix} -K & K \end{bmatrix} F(\beta)^\dagger z - \gamma$.

Example 12 (Kernel Target Alignment) *For the Kernel Target Alignment approach (Cristianini et al., 2002),* $Q_{\text{emp}} = tr(Kyy^\top) = y^\top Ky$, *we directly minimize the regularized quality functional, obtaining the following optimization problem (Lanckriet et al., 2002),*

$$\begin{array}{ll} \underset{\beta}{\text{minimize}} & \frac{1}{2}t_1 + \frac{\lambda_Q}{2}t_2 \\ \textit{subject to} & \beta \geqslant 0 \\ & \|\underline{K}^{\frac{1}{2}}\beta\| \leqslant t_2, \mathbf{1}^\top\beta = 1 \\ & \begin{bmatrix} K & y \\ y^\top & t_1 \end{bmatrix} \succeq 0. \end{array} \tag{32}$$

Note that for the case of Kernel Target Alignment, Q_{emp} *does not provide a direct formulation for the hypothesis function, but instead, it determines a kernel matrix K. This kernel matrix, K, can be utilized in a traditional SVM, to obtain a classification function.*

7. Experiments

In the following experiments, we use data from the UCI repository. Where the data attributes are numerical, we *did not perform any preprocessing* of the data. Boolean attributes are converted to $\{-1, 1\}$, and categorical attributes are arbitrarily assigned an order, and numbered $\{1, 2, \ldots\}$. The optimization problems in Section 6 were solved with an approximate hyperkernel matrix as described in Section 7.1. The SDPs were solved using SeDuMi (Sturm, 1999), and YALMIP (Löfberg, 2002) was used to convert the equations into standard form. We used the hyperkernel for automatic relevance determination defined by (14) for the hyperkernel optimization problems. The scaling freedom that (14) provides for each dimension means we do not have to normalize data to some arbitrary distribution.

For the classification and regression experiments, the datasets were split into 100 random permutations of 60% training data and 40% test data. We deliberately did not attempt to tune parameters and instead made the following choices uniformly for all datasets in classification, regression and novelty detection:

- The kernel width σ_i, for each dimension, was set to 50 times the 90% quantile of the value of $|x_i - x_j|$ over the training data. This ensures sufficient coverage without having too wide a kernel. This value was estimated from a 20% random sampling of the training data.

- λ was adjusted so that $\frac{1}{\lambda m} = 100$ (that is $C = 100$ in the Vapnik-style parameterization of SVMs). This has commonly been reported to yield good results.
- $\nu = 0.3$ for classification and regression. While this is clearly suboptimal for many datasets, we decided to choose it beforehand to avoid having to change *any* parameter. Clearly we could use previous reports on generalization performance to set ν to this value for better performance. For novelty detection, $\nu = 0.1$ (see Section 7.6 for details).
- λ_h for the Harmonic Hyperkernel was chosen to be 0.6, giving adequate coverage over various kernel widths in (12) (small λ_h emphasizes wide kernels almost exclusively, λ_h close to 1 will treat all widths equally).
- The hyperkernel regularization constant was set to $\lambda_Q = 1$.
- For the scale breaking constraint $\mathbf{1}^\top \beta = c$, c was set to 1 for classification as the hypothesis class only involves the sign of the trained function, and therefore is scale free. However, for regression, $c := \text{stddev}(Y_{\text{train}})$ (the standard deviation of the training labels) so that the hyperkernel coefficients are of the same scale as the output (the constant offset b_{offset} takes care of the mean).

In the following experiments, the hypothesis function is computed using the variables of the SDP. In certain cases, numerical problems in the SDP optimizer or in the pseudo-inverse may prevent this hypothesis from optimizing the regularized risk for the particular kernel matrix. In this case, one can use the kernel matrix K from the SDP and obtain the hypothesis function via a standard SVM.

7.1 Low Rank Approximation

Although the optimization of (17) has reduced the problem of optimizing over two possibly infinite dimensional Hilbert spaces to a finite problem, it is still formidable in practice as there are m^2 coefficients for β. For an explicit expansion of type (15) one can optimize in the expansion coefficients $k_i(\underline{x})k_i(\underline{x}')$ directly, which leads to a quality functional with an ℓ_2 penalty on the expansion coefficients. Such an approach is appropriate if there are few terms in (15).

In the general case (or if the explicit expansion has many terms), one can use a low-rank approximation, as described by Fine and Scheinberg (2001) and Zhang (2001). This entails picking from $\left\{\underline{k}((x_i, x_j), \cdot) | 1 \leq i, j \leq m^2\right\}$ a small fraction of terms, p (where $m^2 \gg p$), which approximate $\underline{k}$ on $X_{\text{train}} \times X_{\text{train}}$ sufficiently well. In particular, we choose an $m \times p$ truncated lower triangular matrix G such that $\|P\underline{K}P^\top - GG^\top\|_F \leqslant \delta$, where P is the permutation matrix which sorts the eigenvalues of $\underline{K}$ into decreasing order, and δ is the level of approximation needed. The norm, $\|\cdot\|_F$ is the Frobenius norm. In the following experiments, the hyperkernel matrix was approximated to $\delta = 10^{-6}$ using the incomplete Cholesky factorization method (Bach and Jordan, 2002).

7.2 Classification Experiments

Several binary classification datasets[1] from the UCI repository were used for the experiments. A set of synthetic data (labeled syndata in the results) sampled from two Gaussians was created to illustrate the scaling freedom between dimensions. The first dimension had a standard deviation of 1000 whereas the second dimension had a standard deviation of 1 (a sample result is shown in Figure 1). The results of the experiments are shown in Table 3.

1. We classified window vs. non-window for glass data, the other datasets are all binary.

From Table 3, we observe that our method achieves state of the art results for all the datasets, except the "heart" dataset. We also achieve results much better than previously reported for the "credit" dataset. Comparing the results for C-SVM and Tuned SVM, we observe that our method is always equally good, or better than a C-SVM tuned using 10-fold cross validation.

Data	C-SVM	ν-SVM	Lag-SVM	Best other	CV Tuned SVM (C)
syndata	2.8±2.4	**1.9±1.9**	2.4±2.2	NA	5.9±5.4 (10^8)
pima	**23.5±2.0**	27.7±2.1	23.6±1.9	23.5	24.1±2.1 (10^4)
ionosph	6.6±1.8	6.7±1.8	6.4±1.9	**5.8**	6.1±1.8 (10^3)
wdbc	3.3±1.2	3.8±1.2	**3.0±1.1**	3.2	5.2±1.4 (10^6)
heart	19.7±3.3	19.3±2.4	20.1±2.8	**16.0**	23.2±3.7 (10^4)
thyroid	7.2±3.2	10.1±4.0	6.2±3.1	**4.4**	5.2±2.2 (10^5)
sonar	14.8±3.7	15.3±3.7	**14.7±3.6**	15.4	15.3±4.1 (10^3)
credit	14.6±1.8	**13.7±1.5**	14.7±1.8	22.8	15.3±2.0 (10^8)
glass	6.0±2.4	8.9±2.6	**6.0±2.2**	NA	7.2±2.7 (10^3)

Table 3: Hyperkernel classification: Test error and standard deviation in percent. The second, third and fourth columns show the results of the hyperkernel optimizations of C-SVM (Example 7), ν-SVM (Example 8) and Lagrangian SVM (Example 9) respectively. The results in the fifth column shows the best results from (Freund and Schapire, 1996, Rätsch et al., 2001, Meyer et al., 2003). The rightmost column shows a C-SVM tuned in the traditional way. A Gaussian RBF kernel was tuned using 10-fold cross validation on the training data, with the best value of C shown in brackets. A grid search was performed on (C,σ). The values of C tested were $\{10^{-2}, 10^{-1}, \ldots, 10^9\}$. The values of the kernel width, σ, tested were between 10% and 90% quantile of the distance between a pair of sample of points in the data. These quantiles were estimated by a random sample of 20% of the training data.

7.3 Effect of λ_Q and λ_h on Classification Error

In order to investigate the effect of varying the hyperkernel regularization constant, λ_Q, and the Harmonic Hyperkernel parameter, λ_h, we performed experiments using the C-SVM hyperkernel optimization (Example 7). We performed two sets of experiments with each of our chosen datasets. The results shown in Table 4.

From Table 4, we observe that the variation in classification accuracy over the whole range of the hyperkernel regularization constant, λ_Q is less than the standard deviation of the classification accuracies of the various datasets (compare with Table 3). This demonstrates that our method is quite insensitive to the regularization parameter over the range of values tested for the various datasets.

The method shows a higher sensitivity to the harmonic hyperkernel parameter. Since this parameter effectively selects the scale of the problem, by selecting the "width" of the kernel, it is to be expected that each dataset would have a different ideal value of λ_h. It is to be noted that the generalization accuracy at $\lambda_h = 0.6$ is within one standard deviation (see Table 3 and 4) of the best accuracy achieved over the whole range tested.

	λ_h		λ_Q	
Data	Error	Deviation	Error	Deviation
syndata	3.0±1.1	2.2	2.8±0.0	2.2
pima	25.7±2.6	1.9	24.5±0.1	1.5
ionosph	6.6±1.0	1.7	7.2±0.1	1.9
wdbc	2.9±0.4	0.9	2.7±0.2	0.8
heart	19.7±2.0	3.0	19.4±0.9	2.8
thyroid	6.5±2.8	3.0	6.7±0.3	3.7
sonar	15.7±1.6	3.4	15.1±0.2	3.3
credit	16.0±1.8	1.6	14.7±0.4	1.6
glass	5.9±1.0	2.3	5.2±0.3	2.3

Table 4: Effect of varying λ_h and λ_Q on classification error. In the left experiment, we fixed $\lambda_Q = 1$, and λ_h was varied with the values $\lambda_h = \{0.1, 0.2, \ldots, 0.9, 0.92, 0.94, 0.96, 0.98\}$.In the right, we set $\lambda_h = 0.6$ and varied $\lambda_Q = \{10^{-4}, 10^{-3}, \ldots, 10^{5}\}$. The error columns (columns 2 and 4) report the average error on the test set and the standard deviation of the error over the different parameter settings. The deviation columns (columns 3 and 5) report the average standard deviation over 10 random 60%/40% splits.

7.4 Computational Time

One of the concerns of an SDP optimization problem is the computational complexity. Instead of performing worst case analysis of computational complexity, we perform an empirical test to investigate the scaling behaviour of the proposed method. The total computation time for the first 10 splits of the data was measured, and the average time taken for each split was computed and plotted on a log scale plot in Figure 4. The slope of the graph demonstrates that we have an approximately cubic scaling in computational time.

7.5 Regression Experiments

In order to demonstrate that we can solve problems other than binary classification using the same framework, we performed some experiments using regression and novelty detection datasets. The results of the regression experiments are shown in Table 5. We used *the same parameter settings* as in the previous section.

Comparing the second and fourth columns, we observe that the hyperkernel optimization problem performs better than a ε-SVR tuned using cross validation for all the datasets except the servo dataset. Meyer et al. (2003) used a 90%/10% split of the data for their experiments, while we used a 60%/40% split, which may account for the better performance in the cpu and servo datasets. The reason for the much better rate on the “auto imports” dataset remains a mystery.

7.6 Novelty Detection

We applied the single class support vector machine to detect outliers in the USPS data. The test set of the default split in the USPS database was used in the following experiments. The parameter ν was set to 0.1 for these experiments, hence selecting up to 10% of the data as outliers.

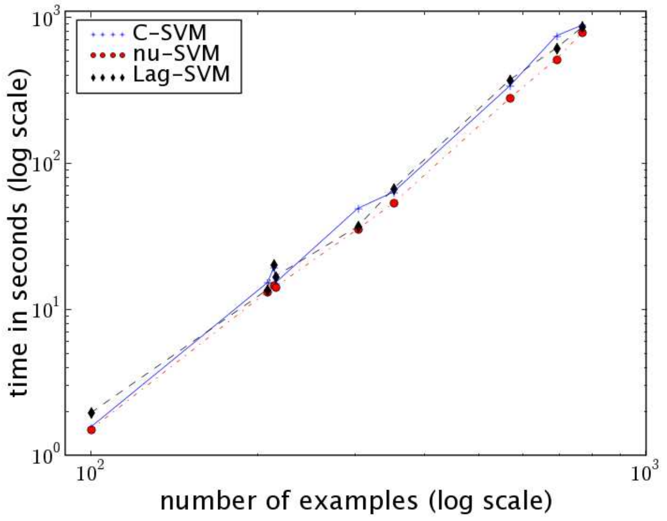


Figure 4: A log scale plot of computational time (in seconds), measured using MATLAB's cputime, against the number of examples in the respective datasets. The slope of the least squares fit through the points are 3.13, 3.05 and 3.03 for C-SVM (Example 7), ν-SVM (Example 8) and Lag-SVM (Example 9) respectively, demonstrating that the algorithms have approximately cubic scaling.

Data	ν-SVR	Best other	CV Tuned ε-SVR
auto-mpg	7.83±0.96	7.11	9.47±1.55
boston	12.96±3.38	9.60	15.78±4.30
auto imports($\times 10^6$)	5.91±2.41	0.25	7.51±5.33
cpu($\times 10^3$)	4.41±3.64	3.16	12.02±20.73
servo	0.74±0.26	0.25	0.62±0.25

Table 5: Hyperkernel regression: Mean Squared Error. The second column shows the results from the hyperkernel optimization of the ν-regression (Example (11)). The results in the third column shows the best results from (Meyer et al., 2003). The rightmost column shows a ε-SVR with a gaussian kernel tuned using 10-fold cross validation on the training data. Similar to the classification setting, grid search was performed on (C, σ). The values of C tested were $\{10^{-2}, 10^{-1}, \ldots, 10^9\}$. The values of the kernel width, σ, tested were between the 10% and 90% quantiles of the distance between a pair of sample of points in the data. These quantiles were estimated by a random 20% sample of the training data.

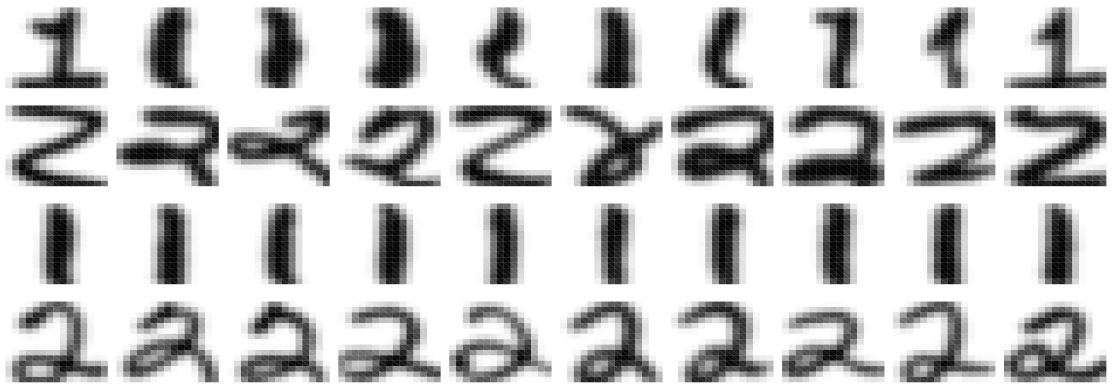


Figure 5: Top rows: Images of digits ‘1’ and ‘2’, considered novel by algorithm; Bottom: typical images of digits ‘1’ and ‘2’.

Since there is no quantitative method for measuring the performance of novelty detection, we cannot directly compare our results with the traditional single class SVM. We can only subjectively conclude, by visually inspecting a sample of the digits, that our approach works for novelty detection of USPS digits. Figure 5 shows a sample of the digits. We can see that the algorithm identifies ‘novel’ digits, such as in the top two rows of Figure 5. The bottom two rows shows a sample of digits which have been deemed to be ‘common’.

8. Summary and Outlook

The regularized quality functional allows the systematic solution of problems associated with the choice of a kernel. Quality criteria that can be used include Kernel Target Alignment, regularized risk and the log posterior. The regularization implicit in our approach allows the control of overfitting that occurs if one optimizes over a too large a choice of kernels.

We have shown that when the empirical quality functional is the regularized risk functional, the resulting optimization problem is convex, and in fact is a SDP. This SDP, which learns the best kernel given the data, has a Bayesian interpretation in terms of a hierarchical Gaussian process. We define more general kernels which may have many free parameters, and optimize over them without overfitting. The experimental results on classification demonstrate that it is possible to achieve state of the art performance using our approach with no manual tuning. Furthermore, the same framework and parameter settings work for various data sets as well as regression and novelty detection.

This approach makes support vector based estimation approaches more automated. Parameter adjustment is less critical compared to when the kernel is fixed, or hand tuned. Future work will focus on deriving improved statistical guarantees for estimates derived via hyperkernels which match the good empirical performance.

Acknowledgments

The authors would like to thank Stéphane Canu, Laurent El Ghaoui, Michael Jordan, John Lloyd, Daniela Pucci de Farias, Matthias Seeger, Grace Wahba and the referees for their helpful comments and suggestions. The authors also thank Alexandros Karatzoglou for his help with SVLAB. National ICT Australia is funded through the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian Research Council.

References

A. Albert. Conditions for positive and nonnegative definiteness in terms of pseudoinverses. *SIAM Journal on Applied Mathematics*, 17(2):434 – 440, 1969.

N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68:337 – 404, 1950.

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

K. P. Bennett and O. L. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1:23 – 34, 1992.

C. M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995.

O. Bousquet and D. Herrmann. On the complexity of learning the kernel matrix. In *Advances in Neural Information Processing Systems 15*, pages 399–406, 2002.

C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121 – 167, 1998.

O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1):131 – 159, 2002.

C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273 – 297, 1995.

D. Cox and F. O'Sullivan. Asymptotic analysis of penalized likelihood and related estimators. *Annals of Statistics*, 18:1676 – 1695, 1990.

K. Crammer, J. Keshet, and Y. Singer. Kernel design using boosting. In *Advances in Neural Information Processing Systems 15*, pages 537–544, 2002.

N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. Kandola. On kernel-target alignment. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 367 – 373, Cambridge, MA, 2002. MIT Press.

N. Cristianini, J. Kandola, A. Elisseeff, and J. Shawe-Taylor. On optimizing kernel alignment. Technical report, UC Davis Department of Statistics, 2003.

K. Duan, S.S. Keerthi, and A.N. Poo. Evaluation of simple performance measures for tuning svm hyperparameters. *Neurocomputing*, 51:41 – 59, 2003.

S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243 – 264, Dec 2001. http://www.jmlr.org.

Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the International Conference on Machine Learing*, pages 148 – 146. Morgan Kaufmann Publishers, 1996.

D. Haussler. Convolutional kernels on discrete structures. Technical Report UCSC-CRL-99 - 10, Computer Science Department, UC Santa Cruz, 1999.

R. Herbrich and R.C. Williamson. Algorithmic luckiness. *Journal of Machine Learning Research*, 3:175 – 212, 2002.

G. S. Kimeldorf and G. Wahba. Some results on Tchebycheffian spline functions. *J. Math. Anal. Applic.*, 33:82 – 95, 1971.

G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. Jordan. Learning the kernel matrix with semidefinite programming. In *Proceedings of the International Conference on Machine Learning*, pages 323–330. Morgan Kaufmann, 2002.

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.

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

J. Löfberg. YALMIP, yet another LMI parser, 2002. http://www.control.isy.liu.se/ ˜johanl/yalmip.html.

A. Luntz and V. Brailovsky. On estimation of characters obtained in statistical procedure of recognition (in Russian). *Technicheskaya Kibernetica*, 3, 1969.

D. J. C. MacKay. Bayesian non-linear modelling for the energy prediction competition. *ASHRAE Transcations*, 4:448 – 472, 1994.

O. L. Mangasarian and D. R. Musicant. Lagrangian support vector machines. *Journal of Machine Learning Research*, 1:161 – 177, 2001.

D. Meyer, F. Leisch, and K. Hornik. The support vector machine under test. *Neurocomputing*, 55 (1–2):169–186, 2003.

R. Neal. *Bayesian Learning in Neural Networks*. Springer, 1996.

C. S. Ong and A. J. Smola. Machine learning using hyperkernels. In *Proceedings of the International Conference on Machine Learning*, pages 568–575, 2003.

C. S. Ong, A. J. Smola, and R. C. Williamson. Hyperkernels. In *Neural Information Processing Systems*, volume 15, pages 495–502. MIT Press, 2002.

M. Opper and O. Winther. Gaussian processes and SVM: Mean field and leave-one-out. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 311 – 326, Cambridge, MA, 2000. MIT Press.

G. Rätsch, T. Onoda, and K. R. Müller. Soft margins for adaboost. *Machine Learning*, 42(3):287 – 320, 2001.

B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2002.

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

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.

M. Seeger. Bayesian methods for support vector machines and Gaussian processes. Master's thesis, University of Edinburgh, Division of Informatics, 1999.

A. J. Smola, B. Schölkopf, and K.-R. Müller. The connection between regularization operators and support vector kernels. *Neural Networks*, 11(5):637 – 649, 1998.

J. F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11/12(1 - 4):625 – 653, 1999.

K. Tsuda, S. Akaho, and K. Asai. The EM algorithm for kernel matrix completion with auxiliary data. *Journal of Machine Learning Research*, 4:67–81, 2003.

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

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

G. Wahba. *Spline Models for Observational Data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM, Philadelphia, 1990.

C. K. I. Williams and C. E. Rasmussen. Gaussian processes for regression. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 514 – 520, Cambridge, MA, 1996. MIT Press.

Christopher K. I. Williams and David Barber. Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI*, 20(12):1342 – 1351, 1998.

T. Zhang. Some sparse approximation bounds for regression problems. In *Proc. 18th International Conf. on Machine Learning*, pages 624 – 631. Morgan Kaufmann, San Francisco, CA, 2001.

Journal of Machine Learning Research 6 (2005) 1073–1097 Submitted 6/04; Revised 7/05; Published 7/05

A Generalization Error for Q-Learning

Susan A. Murphy SAMURPHY@UMICH.EDU
Department of Statistics
University of Michigan
Ann Arbor, MI 48109-1107, USA

Editor: Michael Littman

Abstract

Planning problems that involve learning a policy from a single training set of finite horizon trajectories arise in both social science and medical fields. We consider Q-learning with function approximation for this setting and derive an upper bound on the generalization error. This upper bound is in terms of quantities minimized by a Q-learning algorithm, the complexity of the approximation space and an approximation term due to the mismatch between Q-learning and the goal of learning a policy that maximizes the value function.

Keywords: multistage decisions, dynamic programming, reinforcement learning, batch data

1. Introduction

In many areas of the medical and social sciences the following planning problem arises. A training set or batch of n trajectories of $T+1$-decision epochs is available for estimating a policy. A decision epoch at time t, $t = 0, 1, \ldots, T$, is composed of information observed at time t, O_t, an action taken at time t, A_t and a reward, R_t. For example there are currently a number of ongoing large clinical trials for chronic disorders in which, each time an individual relapses, the individual is re-randomized to one of several further treatments (Schneider et al., 2001; Fava et al., 2003; Thall et al., 2000). These are finite horizon problems with T generally quite small, $T = 2-4$, with known exploration policy. Scientists want to estimate the best "strategies," i.e. policies, for managing the disorder. Alternately the training set of n trajectories may be historical; for example data in which clinicians and their patients are followed with i! nformation about disease process, treatment burden and treatment decisions recorded through time. Again the goal is to estimate the best policy for managing the disease. Alternately, consider either catalog merchandizing or charitable solicitation; information about the client, and whether or not a solicitation is made and/or the form of the solicitation is recorded through time (Simester, Peng and Tsitsiklis, 2003). The goal is to estimate the best policy for deciding which clients should receive a mailing and the form of the mailing. These latter planning problems can be viewed as infinite horizon problems but only T decision epochs per client are recorded. If T is large, the rewards are bounded and the dynamics are stationary Markovian then this finite horizon problem provides an approximation to the discounted infinite horizon problem (Kearns, Mansour and Ng, 2000).

These planning problems are characterized by unknown system dynamics and thus can also be viewed as learning problems as well. Note there is no access to a generative model nor an online simulation model nor the ability to conduct offline simulation. All that is available is the n trajectories of $T+1$ decision epochs. One approach to learning a policy in this setting is Q-learning

(Watkins, 1989) since the actions in the training set are chosen according to a (non-optimal) exploration policy; Q-learning is an off-policy method (Sutton and Barto, 1998). When the observables are vectors of continuous variables or are otherwise of high dimension, Q-learning must be combined with function approximation.

The contributions of this paper are as follows. First a version of Q-learning with function approximation, suitable for learning a policy with one training set of finite horizon trajectories and a large observation space, is introduced; this "batch"version of Q-learning processes the entire training set of trajectories prior to updating the approximations to the Q-functions. An incremental implementation of batch Q-learning results in one-step Q-learning with function approximation. Second performance guarantees for this version of Q-learning are provided. These performance guarantees do not assume assume that the system dynamics are Markovian. The performance guarantees are upper bounds on the average difference in value functions or more specifically the average generalization error. Here the generalization error for batch Q-learning is defined analogous to the generalization error in supervised learning (Schapire et al., 1998); it is the average diffe! rence in value when using the optimal policy as compared to using the greedy policy (from Q-learning) in generating a separate test set. The performance guarantees are analogous to performance guarantees available in supervised learning (Anthony and Bartlett, 1999).

The upper bounds on the average generalization error permit an additional contribution. These upper bounds illuminate the mismatch between Q-learning with function approximation and the goal of finding a policy maximizing the value function (see the remark following Lemma 2 and the third remark following Theorem 2). This mismatch occurs because the Q-learning algorithm with function approximation does not directly maximize the value function but rather this algorithm approximates the optimal Q-function within the constraints of the approximation space in a least squares sense; this point is discussed as some length in section 3 of Tsitsiklis and van Roy (1997).

In the process of providing an upper bound on the average generalization error, finite sample bounds on the difference in average values resulting from different policies are derived. There are three terms in the upper bounds. The first term is a function of the optimization criterion used in batch Q-learning, the second term is due to the complexity of the approximation space and the last term is an approximation error due to the above mentioned mismatch. The third term which is a function of the complexity of the approximation space is similar in form to generalization error bounds derived for supervised learning with neural networks as in Anthony and Bartlett (1999). From the work of Kearns, Mansour, and Ng (1999, 2000) and Peshkin and Shelton (2002), we expect and find as well here that the number of trajectories needed to guarantee a specified error level is exponential in the horizon time, T. The upper bound does not depend on the dimension of the observables O_t's. This is in contrast to the results of Fiechter (1994, 1997) in which the upper bound on the average generalization error depends on the number of possible values for the observables.

A further contribution is that the upper bound on the average generalization error provides a mechanism for generalizing ideas from supervised learning to reinforcement learning. For example if the optimal Q-function belongs to the approximation space, then the upper bounds imply that batch Q-learning is a PAC reinforcement learning algorithm as in Feichter (1994, 1997); see the first remark following Theorem 1. And second the upper bounds provide a starting point in using structural risk minimization for model selection (see the second remark after Theorem 1).

In Section 2, we review the definition of the value function and Q-function for a (possibly non-stationary, non-Markovian) finite horizon decision process. Next we review batch Q-learning with function approximation when the learning algorithm must use a training set of n trajectories. In Section 5 we provide the two main results, both of which provide the number of trajectories needed to achieve a given error level with a specified level of certainty.

2. Preliminaries

In the following we use upper case letters, such as O and A, to denote random variables and lower case letters, such as o and a, to denote instantiates or values of the random variables. Each of the n trajectories is composed of the sequence $\{O_0, A_0, O_1, \ldots, A_T, O_{T+1}\}$ where T is a finite constant. Define $\mathbf{O}_t = \{O_0, \ldots, O_t\}$ and similarly for $\mathbf{A}_t$. Each action A_t takes values in finite, discrete action space $\mathcal{A}$ and O_t takes values in the observation space $\mathcal{O}$. The observation space may be multidimensional and continuous. The arguments below will not require the Markovian assumption with the value of O_t equal to the state at time t. The rewards are $R_t = r_t(\mathbf{O}_t, \mathbf{A}_t, O_{t+1})$ for r_t a reward function and for each $0 \leq t \leq T$ (if the Markov assumption holds then replace $\mathbf{O}_t$ with O_t and $\mathbf{A}_t$ with A_t). We assume that the rewards are bounded, taking values in the interval $[0, 1]$.

We assume the trajectories are sampled at random according to a fixed distribution denoted by P. Thus the trajectories are generated by one fixed distribution. This distribution is composed of the unknown distribution of each O_t conditional on $(\mathbf{O}_{t-1}, \mathbf{A}_{t-1})$ (call these unknown conditional densities $\{f_0, \ldots f_T\}$) and an exploration policy for generating the actions. Denote the exploration policy by $\mathbf{p}_T = \{p_0, \ldots, p_T\}$ where the probability that action a is taken given history $\{\mathbf{O}_t, \mathbf{A}_{t-1}\}$ is $p_t(a|\mathbf{O}_t, \mathbf{A}_{t-1})$ (if the Markov assumption holds then, as before, replace $\mathbf{O}_t$ with O_t and $\mathbf{A}_{t-1}$ with A_{t-1}.) We assume that $p_t(a|\mathbf{o}_t, \mathbf{a}_{t-1}) > 0$ for each action $a \in \mathcal{A}$ and for each possible value $(\mathbf{o}_t, \mathbf{a}_{t-1})$; that is, at each time all actions are possible. Then the likelihood (under P) of the trajectory, $\{o_0, a_0, o_1, \ldots, a_T, o_{T+1}\}$ is

$$f_0(o_0)p_0(a_0|o_0)\prod_{t=1}^{T} f_t(o_t|\mathbf{o}_{t-1}, \mathbf{a}_{t-1})p_t(a_t|\mathbf{o}_t, \mathbf{a}_{t-1})f_{T+1}(o_{T+1}|\mathbf{o}_T, \mathbf{a}_T). \tag{1}$$

Denote expectations with respect to the distribution P by an E.

Define a deterministic, but possibly non-stationary and non-Markovian, policy, π, as a sequence of decision rules, $\{\pi_1, \ldots, \pi_T\}$, where the output of the time t decision rule, $\pi_t(\mathbf{o}_t, \mathbf{a}_{t-1})$, is an action. Let the distribution P_π denote the distribution of a trajectory whereby the policy π is used to generate the actions. Then the likelihood (under P_π) of the trajectory $\{o_0, a_0, o_1, \ldots, a_T, o_{T+1}\}$ is

$$f_0(o_0)1_{a_0=\pi_0(o_0)}\prod_{j=1}^{T} f_j(o_j|\mathbf{o}_{j-1}, \mathbf{a}_{j-1})1_{a_j=\pi_j(\mathbf{o}_j, \mathbf{a}_{j-1})}f_{T+1}(o_{T+1}|\mathbf{o}_T, \mathbf{a}_T) \tag{2}$$

where for a predicate W, 1_W is 1 if W is true and is 0 otherwise. Denote expectations with respect to the distribution P_π by an E_π.

Note that since (1) and (2) differ only in regard to the policy for generating actions, an expectation with respect to either P or P_π that does not involve integration over the policy results in the same quantity. For example, $E\left[R_t|\mathbf{O}_t, \mathbf{A}_t\right] = E_\pi\left[R_t|\mathbf{O}_t, \mathbf{A}_t\right]$, for any policy π.

Let Π be the collection of all policies. In a finite horizon planning problem (permitting non-stationary, non-Markovian policies) the goal is to estimate a policy that maximizes $E_\pi[\sum_{j=1}^{T} R_j|O_0 = o_0]$ over $\pi \in \Pi$. If the system dynamics are Markovian and each $r_j(\mathbf{o}_j, \mathbf{a}_j, o_{j+1}) = \gamma^j r(o_j, a_j, o_{j+1})$

for r a bounded reward function and $\gamma \in (0,1)$ a discount factor, then this finite horizon problem provides an approximation to the discounted infinite horizon problem (Kearns Mansour and Ng, 2000) for T large.

Given a policy, π, the value function for an observation, o_0, is

$$V_\pi(o_0) = E_\pi\left[\sum_{j=1}^{T} R_j \middle| O_0 = o_0\right].$$

The t-value function for policy π is the value of the rewards summed from time t on and is

$$V_{\pi,t}(\mathbf{o}_t, \mathbf{a}_{t-1}) = E_\pi\left[\sum_{j=t}^{T} R_j \middle| \mathbf{O}_t = \mathbf{o}_t, \mathbf{A}_{t-1} = \mathbf{a}_{t-1}\right].$$

If the Markovian assumption holds then $(\mathbf{o}_t, \mathbf{a}_{t-1})$ in the definition of $V_{\pi,t}$ is replaced by o_t. Note that the time 0 value function is simply the value function ($V_{\pi,0} = V_\pi$). For convenience, set $V_{\pi,T+1} = 0$. Then the value functions satisfy the following relationship:

$$V_{\pi,t}(\mathbf{o}_t, \mathbf{a}_{t-1}) = E_\pi[R_t + V_{\pi,t+1}(\mathbf{O}_{t+1}, \mathbf{A}_t) | \mathbf{O}_t = \mathbf{o}_t, \mathbf{A}_{t-1} = \mathbf{a}_{t-1}]$$

for $t = 0, \ldots, T$. The time t Q-function for policy π is

$$Q_{\pi,t}(\mathbf{o}_t, \mathbf{a}_t) = E[R_t + V_{\pi,t+1}(\mathbf{O}_{t+1}, \mathbf{A}_t) | \mathbf{O}_t = \mathbf{o}_t, \mathbf{A}_t = \mathbf{a}_t].$$

(The subscript, π, can be omitted as this expectation is with respect to the distribution of O_{t+1} given $(\mathbf{O}_t, \mathbf{A}_t)$, f_{t+1}; this conditional distribution does not depend on the policy.) In Section 4 we express the difference in value functions for policy $\tilde{\pi}$ and policy π in terms of the advantages (as defined in Baird, 1993). The time t advantage is

$$\mu_{\pi,t}(\mathbf{o}_t, \mathbf{a}_t) = Q_{\pi,t}(\mathbf{o}_t, \mathbf{a}_t) - V_{\pi,t}(\mathbf{o}_t, \mathbf{a}_{t-1}).$$

The advantage can be interpreted as the gain in performance obtained by following action a_t at time t and thereafter policy π as compared to following policy π from time t on.

The optimal value function $V^*(o)$ for an observation o is

$$V^*(o) = \max_{\pi \in \Pi} V_\pi(o)$$

and the optimal t-value function for history $(\mathbf{o}_t, \mathbf{a}_{t-1})$ is

$$V_t^*(\mathbf{o}_t, \mathbf{a}_{t-1}) = \max_{\pi \in \Pi} V_{\pi,t}(\mathbf{o}_t, \mathbf{a}_{t-1}).$$

As is well-known, the optimal value functions satisfy the Bellman equations (Bellman, 1957)

$$V_t^*(\mathbf{o}_t, \mathbf{a}_{t-1}) = \max_{a_t \in \mathcal{A}} E[R_t + V_{t+1}^*(\mathbf{O}_{t+1}, \mathbf{A}_t) | \mathbf{O}_t = \mathbf{o}_t, \mathbf{A}_t = \mathbf{a}_t].$$

Optimal, deterministic, time t decision rules must satisfy

$$\pi_t^*(\mathbf{o}_t, \mathbf{a}_{t-1}) \in \arg\max_{a_t \in \mathcal{A}} E[R_t + V_{t+1}^*(\mathbf{O}_{t+1}, \mathbf{A}_t) | \mathbf{O}_t = \mathbf{o}_t, \mathbf{A}_t = \mathbf{a}_t].$$

The optimal time t Q-function is

$$Q_t^*(\mathbf{o}_t, \mathbf{a}_t) = E[R_t + V_{t+1}^*(\mathbf{O}_{t+1}, \mathbf{A}_t) | \mathbf{O}_t = \mathbf{o}_t, \mathbf{A}_t = \mathbf{a}_t],$$

and thus the optimal time t advantage, which is given by

$$\mu_t^*(\mathbf{o}_t, \mathbf{a}_t) = Q_t^*(\mathbf{o}_t, \mathbf{a}_t) - V_t^*(\mathbf{o}_t, \mathbf{a}_{t-1}),$$

is always nonpositive and furthermore it is maximized in a_t at $a_t = \pi_t^*(\mathbf{o}_t, \mathbf{a}_{t-1})$.

3. Batch Q-Learning

We consider a version of Q-learning for use in learning a non-stationary, non-Markovian policy with one training set of finite horizon trajectories. The term "batch"Q-learning is used to emphasize that learning occurs only after the collection of the training set. The Q-functions are estimated using an approximator (i.e. neural networks, decision-trees etc.) (Bertsekas and Tsitsiklis, 1996; Tsitsiklis and van Roy, 1997) and then the estimated decision rules are the argmax of the estimated Q functions. Let $\mathcal{Q}_t$ be the approximation space for the tth Q-function, e.g. $\mathcal{Q}_t = \{Q_t(\mathbf{o}_t, \mathbf{a}_t; \theta) : \theta \in \Theta\}$; θ is a vector of parameters taking values in a parameter space Θ which is a subset of a Euclidean space. For convenience set Q_{T+1} equal to zero and write $\mathbb{E}_n f$ for the expectation of an arbitrary function, f, of a trajectory with respect to the probability obtained by choosing a trajectory uniformly from the training set of n trajectories (for example, $\mathbb{E}_n[f(O_t)] = 1/n \sum_{i=1}^n f(O_{it})$ for O_{it} the tth observation in the ith trajectory). In batch Q-learning using dynamic programming and function approximation solve the following backwards through time $t =!T, T-1, \ldots, 1$ to obtain

$$\theta_t \in \arg\min_\theta \mathbb{E}_n \left[R_t + \max_{a_{t+1}} Q_{t+1}(\mathbf{O}_{t+1}, \mathbf{A}_t, a_{t+1}; \theta_{t+1}) - Q_t(\mathbf{O}_t, \mathbf{A}_t; \theta) \right]^2 . \tag{3}$$

Suppose that Q-functions are approximated by linear combinations of p features ($\mathcal{Q}_t = \{\theta^T q_t(\mathbf{o}_t, \mathbf{a}_t) : \theta \in \mathbf{R}^p\}$) then to achieve (3) solve backwards through time, $t = T, T-1, \ldots, 0$,

$$0 = \mathbb{E}_n \left[\left(R_t + \max_{a_{t+1}} Q_{t+1}(\mathbf{O}_{t+1}, \mathbf{A}_t, a_{t+1}; \theta_{t+1}) - Q_t(\mathbf{O}_t, \mathbf{A}_t; \theta_t) \right) q_t(\mathbf{O}_t, \mathbf{A}_t)^T \right] \tag{4}$$

for θ_t.

An incremental implementation with updates between trajectories of (3) and (4) results in one-step Q-learning (Sutton and Barto, 1998, pg. 148, put $\gamma = 1$, assume the Markov property and no need for function approximation). This is not surprising as Q-learning can be viewed as approximating least squares value iteration (Tsitsiklis and van Roy, 1996). To see the connection consider the following generic derivation of an incremental update. Denote the ith example in a training set by X_i. Define $\widehat{\theta}^n$ to be a solution of $\sum_{i=1}^n f(X_i, \theta) = 0$ for f a given p dimensional vector of functions and each integer n. Using a Taylor series, expand $\sum_{i=1}^{n+1} f(X_i, \widehat{\theta}^{(n+1)})$ in $\widehat{\theta}^{(n+1)}$ about $\widehat{\theta}^{(n)}$ to obtain a between-example update to $\widehat{\theta}^{(n)}$:

$$\widehat{\theta}^{(n+1)} \leftarrow \widehat{\theta}^{(n)} + \frac{1}{n+1} \left(\mathbb{E}_{n+1} \left(-\frac{\partial f(X, \widehat{\theta}^n)}{\partial \widehat{\theta}^n} \right) \right)^{-1} f(X_{n+1}, \widehat{\theta}^n).$$

Replace $\frac{1}{n+1} \left(\mathbb{E}_{n+1} \left(-\frac{\partial f(X, \widehat{\theta}^n)}{\partial \widehat{\theta}^n} \right) \right)^{-1}$ by a step-size α_n ($\alpha_n \to 0$ as $n \to \infty$) to obtain a general formula for the incremental implementation. Now consider an incremental implementation of (4) for each $t = 0, \ldots, T$. Then for each t, $X = (\mathbf{O}_{t+1}, \mathbf{A}_t)$, $\theta = \theta_t$ and

$$f(X, \theta_t) = \left(R_t + \max_{a_{t+1}} Q_{t+1}(\mathbf{O}_{t+1}, \mathbf{A}_t, a_{t+1}; \widehat{\theta}_{t+1}^{(n+1)}) - Q_t(\mathbf{O}_t, \mathbf{A}_t; \theta_t) \right) q_t(\mathbf{O}_t, \mathbf{A}_t)^T$$

is a vector of dimension p. The incremental update is

$$\widehat{\theta}_t^{(n+1)} \leftarrow \widehat{\theta}_t^{(n)} + \alpha_n \left(R_t + \max_{a_{t+1}} Q_{t+1}(\mathbf{O}_{t+1}, \mathbf{A}_t, a_{t+1}; \widehat{\theta}_{t+1}^{(n+1)}) - Q_t(\mathbf{O}_t, \mathbf{A}_t; \widehat{\theta}_t^{(n)}) \right) q_t(\mathbf{O}_t, \mathbf{A}_t)^T)$$

for $t = 0, \ldots, T$. This is the one-step update of Sutton and Barto (1998, pg. 148) with $\gamma = 1$ and generalized to permit function approximation and nonstationary Q-functions and is analogous to the TD(0) update of Tsitsiklis and van Roy (1997) permitting non-Markovian, nonstationary value functions.

Denote the estimator of the optimal Q-functions based on the training data by $\widehat{Q}_t$ for $t = 0, \ldots, T$ (for simplicity, θ is omitted). The estimated policy, $\widehat{\pi}$, satisfies $\widehat{\pi}_t(\mathbf{o}_t, \mathbf{a}_{t-1}) \in \arg\max_{a_t} \widehat{Q}_t(\mathbf{o}_t, \mathbf{a}_t)$ for each t. Note that members of the approximation space $\mathcal{Q}_t$ need not be "Q-functions" for any policy. For example the Q-functions corresponding to the use of a policy π ($Q_{\pi,t},\ t = 0, \ldots, T$) must satisfy

$$E[R_t + V_{\pi,t+1}(\mathbf{O}_{t+1}, \mathbf{A}_t) | \mathbf{O}_t, \mathbf{A}_t] = Q_{\pi,t}(\mathbf{O}_t, \mathbf{A}_t)$$

where $V_{\pi,t+1}(\mathbf{O}_{t+1}, \mathbf{A}_t) = Q_{\pi,t+1}(\mathbf{O}_{t+1}, \mathbf{A}_t, a_{t+1})$ with a_{t+1} set equal to $\pi_{t+1}(\mathbf{O}_{t+1}, \mathbf{A}_t)$. Q-learning does not impose this restriction on $\{\widehat{Q}_t, t = 0, \ldots, T\}$; indeed it may be that no member of the approximation space can satisfy this restriction. None-the-less we refer to the $\widehat{Q}_t$'s as estimated Q-functions. Note also that the approximation for the Q-functions combined with the definition of the estimated decision rules as the argmax of the estimated Q functions places implicit restrictions on the set of policies that will be considered. In effect the space of interesting polices is no longer Π but rather $\Pi_{\mathcal{Q}} = \{\pi_\theta, \theta \in \Theta\}$ where $\pi_\theta = \{\pi_{1,\theta}, \ldots, \pi_{T,\theta}\}$ and where each $\pi_{t,\theta}(\mathbf{o}_t, \mathbf{a}_{t-1}) \in \arg\max_{a_t} Q_t(\mathbf{o}_t, \mathbf{a}_t; \theta)$ for some $Q_t \in \mathcal{Q}_t$.

4. Generalization Error

Define the generalization error of a policy π at an observation o_0 as the average difference between the optimal value function and the value function resulting from the use of policy π in generating a separate test set. The generalization error of policy π at observation o_0 can be written as

$$V^*(o_0) - V_\pi(o_0) = -E_\pi\left[\sum_{t=0}^{T} \mu_t^*(\mathbf{O}_t, \mathbf{A}_t) \middle| O_0 = o_0\right] \tag{5}$$

where E_π denotes the expectation using the likelihood (2). So the generalization error can be expressed in terms of the optimal advantages evaluated at actions determined by policy π; that is when each $A_t = \pi_t(\mathbf{O}_t, \mathbf{A}_{t-1})$. Thus the closer each optimal advantage, $\mu_t^*(\mathbf{O}_t, \mathbf{A}_t)$ for $A_t = \pi_t(\mathbf{O}_t, \mathbf{A}_{t-1})$ is to zero, the smaller the generalization error. Recall that the optimal advantage, $\mu_t^*(\mathbf{O}_t, \mathbf{A}_t)$, is zero when $A_t = \pi_t^*(\mathbf{O}_t, \mathbf{A}_{t-1})$. The display in (5) follows from Kakade's (ch. 5, 2003) expression for the difference between the value functions for two policies.

Lemma 1

Given policies $\tilde{\pi}$ and π,

$$V_{\tilde{\pi}}(o_0) - V_\pi(o_0) = -E_\pi\left[\sum_{t=0}^{T} \mu_{\tilde{\pi},t}(\mathbf{O}_t, \mathbf{A}_t) \middle| O_0 = o_0\right].$$

Set $\tilde{\pi} = \pi^*$ to obtain (5). An alternate to Kakade's (2003) proof is as follows.

Proof. First note

$$V_\pi(o_0) = E_\pi\left[\sum_{t=0}^{T} R_t \middle| O_0 = o_0\right] = E_\pi\left[E_\pi\left[\sum_{t=0}^{T} R_t \middle| \mathbf{O}_T, \mathbf{A}_T\right] \middle| O_0 = o_0\right]. \tag{6}$$

And $E_\pi\left[\sum_{t=0}^{T} R_t \middle| \mathbf{O}_T, \mathbf{A}_T\right]$ is the expectation with respect to the distribution of O_{T+1} given the history $(\mathbf{O}_T, \mathbf{A}_T)$; this is the density f_{T+1} from Section 2 and f_{T+1} is independent of the policy used to choose the actions. Thus we may subscript E by either π or $\tilde{\pi}$ without changing the expectation; $E_\pi\left[\sum_{t=0}^{T} R_t \middle| \mathbf{O}_T, \mathbf{A}_T\right] = E_{\tilde{\pi}}\left[\sum_{t=0}^{T} R_t \middle| \mathbf{O}_T, \mathbf{A}_T\right] = \sum_{t=0}^{T-1} R_t + Q_{\tilde{\pi},T}(\mathbf{O}_T, \mathbf{A}_T)$. The conditional expectation can be written in a telescoping sum as

$$\begin{aligned} E_\pi\left[\sum_{t=0}^{T} R_t \middle| \mathbf{O}_T, \mathbf{A}_T\right] = & \sum_{t=0}^{T} Q_{\tilde{\pi},t}(\mathbf{O}_t, \mathbf{A}_t) - V_{\tilde{\pi},t}(\mathbf{O}_t, \mathbf{A}_{t-1}) \\ & + \sum_{t=1}^{T} R_{t-1} + V_{\tilde{\pi},t}(\mathbf{O}_t, \mathbf{A}_{t-1}) - Q_{\tilde{\pi},t-1}(\mathbf{O}_{t-1}, \mathbf{A}_{t-1}) \\ & + V_{\tilde{\pi},0}(O_0) \end{aligned}$$

The first sum is the sum of the advantages. The second sum is a sum of temporal-difference errors; integrating the temporal-difference error with respect to the conditional distribution of O_t given $(\mathbf{O}_{t-1}, \mathbf{A}_{t-1})$, denoted by f_t in Section 2, we obtain zero,

$$E\left[R_{t-1} + V_{\tilde{\pi},t}(\mathbf{O}_t, \mathbf{A}_{t-1}) | \mathbf{O}_{t-1}, \mathbf{A}_{t-1}\right] = Q_{\tilde{\pi},t-1}(\mathbf{O}_{t-1}, \mathbf{A}_{t-1})$$

(as before E denotes expectation with respect to (1); recall that expectations that do not integrate over the policy can be written either with an E or an E_π). Substitute the telescoping sum into (6) and note that $V_{\tilde{\pi},0}(o_0) = V_{\tilde{\pi}}(o_0)$ to obtain the result. ■

In the following Lemma the difference between value functions corresponding to two policies, $\tilde{\pi}$ and π, is expressed in terms of both the L_1 and L_2 distances between the optimal Q-functions and *any* functions $\{Q_0, Q_1, \ldots, Q_T\}$ satisfying $\pi_t(\mathbf{o}_t, \mathbf{a}_{t-1}) \in \arg\max_{a_t} Q_t(\mathbf{o}_t, \mathbf{a}_t)$, $t = 0, \ldots, T$ and *any* functions $\{\tilde{Q}_0, \tilde{Q}_1, \ldots, \tilde{Q}_T\}$ satisfying $\tilde{\pi}_t(\mathbf{o}_t, \mathbf{a}_{t-1}) \in \arg\max_{a_t} \tilde{Q}_t(\mathbf{o}_t, \mathbf{a}_t)$, $t = 0, \ldots, T$. We assume that there exists a positive constant, L for which $p_t(a_t|\mathbf{o}_t, \mathbf{a}_{t-1}) \geq L^{-1}$ for each t and all pairs $(\mathbf{o}_t, \mathbf{a}_{t-1})$; if the stochastic decision rule, p_t, were uniform then L would be the size of the action space.

Lemma 2

For all functions, Q_t satisfying $\pi_t(\mathbf{o}_t, \mathbf{a}_{t-1}) \in \arg\max_{a_t} Q_t(\mathbf{o}_t, \mathbf{a}_t)$, $t = 0, \ldots, T$, and all functions $\tilde{Q}_t$ satisfying $\tilde{\pi}_t(\mathbf{o}_t, \mathbf{a}_{t-1}) \in \arg\max_{a_t} \tilde{Q}_t(\mathbf{o}_t, \mathbf{a}_t)$, $t = 0, \ldots, T$ we have,

$$\begin{aligned} |V_{\tilde{\pi}}(o_0) - V_\pi(o_0)| \leq & \sum_{t=0}^{T} 2L^{t+1} E\left[|Q_t(\mathbf{O}_t, \mathbf{A}_t) - \tilde{Q}_t(\mathbf{O}_t, \mathbf{A}_t)| \middle| O_0 = o_0\right] \\ & + \sum_{t=0}^{T} 2L^{t+1} E\left[|\tilde{Q}_t(\mathbf{O}_t, \mathbf{A}_t) - Q_{\tilde{\pi},t}(\mathbf{O}_t, \mathbf{A}_t)| \middle| O_0 = o\right] \end{aligned}$$

and

$$\begin{aligned} |V_{\tilde{\pi}}(o_0) - V_\pi(o_0)| \leq & \sum_{t=0}^{T} 2L^{(t+1)/2} \sqrt{E\left[\left(Q_t(\mathbf{O}_t, \mathbf{A}_t) - \tilde{Q}_t(\mathbf{O}_t, \mathbf{A}_t)\right)^2 \middle| O_0 = o_0\right]} \\ & + \sum_{t=0}^{T} 2L^{(t+1)/2} \sqrt{E\left[\left(\tilde{Q}_t(\mathbf{O}_t, \mathbf{A}_t) - Q_{\tilde{\pi},t}(\mathbf{O}_t, \mathbf{A}_t)\right)^2 \middle| O_0 = o\right]}, \end{aligned}$$

where E denotes expectation with respect to the distribution generating the training sample (1).

Remark:

1. Note that in general $\arg\max_{a_t} Q_{\tilde{\pi},t}(\mathbf{o}_t, \mathbf{a}_t)$ may not be $\tilde{\pi}_t$ thus we can not choose $\tilde{Q}_t = Q_{\tilde{\pi},t}$. However if $\tilde{\pi} = \pi^*$ then we can choose $\tilde{Q}_t = Q_t^*$ ($= Q_{\pi^*,t}$ by definition) and the second term in both upper bounds is equal to zero.

2. This result can be used to emphasize one aspect of the mismatch between estimating the optimal Q function and the goal of learning a policy that maximizes the value function. Suppose $\tilde{Q}_t = Q_t^*$, $\tilde{\pi} = \pi^*$. The generalization error is

$$V^*(o_0) - V_\pi(o_0) \leq \sum_{t=0}^{T} 2L^{(t+1)/2} \sqrt{E\left[(Q_t(\mathbf{O}_t, \mathbf{A}_t) - Q_t^*(\mathbf{O}_t, \mathbf{A}_t))^2 \Big| O_0 = o_0\right]}$$

   for Q_t any function satisfying $\pi_t(\mathbf{o}_t, \mathbf{a}_{t-1}) \in \arg\max_{a_t} Q_t(\mathbf{o}_t, \mathbf{a}_t)$. Absent restrictions on the Q_ts, this inequality cannot be improved in the sense that choosing each $Q_t = Q_t^*$ and $\pi_t = \pi_t^*$ yields 0 on both sides of the inequality. However an inequality in the opposite direction is not possible, since as was seen in Lemma 1, $V^*(o_0) - V_\pi(o_0)$ involves the Q functions only through the advantages (see also (7) below with $\tilde{\pi} = \pi^*$). Thus for the difference in value functions to be small, the average difference between $Q_t(\mathbf{o}_t, \mathbf{a}_t) - \max_{a_t} Q_t(\mathbf{o}_t, \mathbf{a}_t)$ and $Q_t^*(\mathbf{o}_t, \mathbf{a}_t) - \max_{a_t} Q_t^*(\mathbf{o}_t, \mathbf{a}_t)$ must be small; this does not require that the average difference between $Q_t(\mathbf{o}_t, \mathbf{a}_t)$ and $Q_t^*(\mathbf{o}_t, \mathbf{a}_t)$ is small. The mismatch is not unexpected. For example, Baxter and Bartlett (2001) provide an example in which the approximation space for the value function includes a value function for which the greedy policy is optimal, yet the greedy policy found by temporal difference learning (TD(1!)) function performs very poorly.

Proof. Define $\mu_t(\mathbf{o}_t, \mathbf{a}_t) = Q_t(\mathbf{o}_t, \mathbf{a}_t) - \max_{a_t} Q_t(\mathbf{o}_t, \mathbf{a}_t)$ for each t; note that $\mu_t(\mathbf{o}_t, \mathbf{a}_t)$ evaluated at $a_t = \pi_t(\mathbf{o}_t, \mathbf{a}_{t-1})$ is zero. Start with the result of Lemma 1. Then note the difference between the value functions can be expressed as

$$V_{\tilde{\pi}}(o_0) - V_\pi(o_0) = \sum_{t=0}^{T} E_\pi\left[\mu_t(\mathbf{O}_t, \mathbf{A}_t) - \mu_{\tilde{\pi},t}(\mathbf{O}_t, \mathbf{A}_t) \Big| O_0 = o_0\right]. \tag{7}$$

since P_π puts $a_t = \pi_t(\mathbf{o}_t, \mathbf{a}_{t-1})$ and $\mu_t(\mathbf{o}_t, \mathbf{a}_t) = 0$ for this value of a_t. When it is clear from the context μ_t ($\mu_{\tilde{\pi},t}$) is used as abbreviation for $\mu_t(\mathbf{O}_t, \mathbf{A}_t)$ ($\mu_{\tilde{\pi},t}(\mathbf{O}_t, \mathbf{A}_t)$) in the following. Also $Q_{\tilde{\pi},t}(\mathbf{O}_t, \mathbf{A}_{t-1}, a_t)$ with a_t replaced by $\tilde{\pi}_t(\mathbf{O}_t, \mathbf{A}_{t-1})$ is written as $Q_{\tilde{\pi},t}(\mathbf{O}_t, \mathbf{A}_{t-1}, \tilde{\pi}_t)$. Consider the absolute value of the tth integrand in (7):

$$\begin{aligned}
&|\mu_t - \mu_{\tilde{\pi},t}| \\
&\quad = |Q_t(\mathbf{O}_t, \mathbf{A}_t) - \max_{a_t} Q_t(\mathbf{O}_t, \mathbf{A}_{t-1}, a_t) - Q_{\tilde{\pi},t}(\mathbf{O}_t, \mathbf{A}_t) + Q_{\tilde{\pi},t}(\mathbf{O}_t, \mathbf{A}_{t-1}, \tilde{\pi}_t)| \\
&\quad \leq |Q_t(\mathbf{O}_t, \mathbf{A}_t) - Q_{\tilde{\pi},t}(\mathbf{O}_t, \mathbf{A}_t)| + |\max_{a_t} Q_t(\mathbf{O}_t, \mathbf{A}_{t-1}, a_t) - Q_{\tilde{\pi},t}(\mathbf{O}_t, \mathbf{A}_{t-1}, \tilde{\pi}_t)|.
\end{aligned}$$

Since $\max_{a_t} \tilde{Q}_t(\mathbf{O}_t, \mathbf{A}_{t-1}, a_t) = \tilde{Q}_t(\mathbf{O}_t, \mathbf{A}_{t-1}, \tilde{\pi}_t)$ and for any functions, h and h', $|\max_{a_t} h(a_t) - \max_{a_t} h'(a_t)| \leq \max_{a_t} |h(a_t) - h'(a_t)|$,

$$\begin{aligned}
&\left|\max_{a_t} Q_t(\mathbf{O}_t, \mathbf{A}_{t-1}, a_t) - Q_{\tilde{\pi},t}(\mathbf{O}_t, \mathbf{A}_{t-1}, \tilde{\pi}_t)\right| \\
&\qquad \leq \max_{a_t} \left|Q_t(\mathbf{O}_t, \mathbf{A}_{t-1}, a_t) - \tilde{Q}_t(\mathbf{O}_t, \mathbf{A}_{t-1}, a_t)\right| \\
&\qquad\quad + \left|\tilde{Q}_t(\mathbf{O}_t, \mathbf{A}_{t-1}, \tilde{\pi}_t) - Q_{\tilde{\pi},t}(\mathbf{O}_t, \mathbf{A}_{t-1}, \tilde{\pi}_t)\right|.
\end{aligned}$$

We obtain $|\mu_t - \mu_{\tilde{\pi},t}|$

$$\begin{aligned}
\leq\ & 2\max_{a_t}|Q_t(\mathbf{O}_t,\mathbf{A}_{t-1},a_t) - \tilde{Q}_t(\mathbf{O}_t,\mathbf{A}_{t-1},a_t)| \\
& \qquad + 2\max_{a_t}|\tilde{Q}_t(\mathbf{O}_t,\mathbf{A}_{t-1},a_t) - Q_{\tilde{\pi},t}(\mathbf{O}_t,\mathbf{A}_{t-1},a_t)| \qquad (8)\\
\leq\ & 2L\sum_{a_t}|Q_t(\mathbf{O}_t,\mathbf{A}_{t-1},a_t) - \tilde{Q}_t(\mathbf{O}_t,\mathbf{A}_{t-1},a_t)|p_t(a_t|\mathbf{O}_t,\mathbf{A}_{t-1}) \\
& \qquad + 2L\sum_{a_t}|\tilde{Q}_t(\mathbf{O}_t,\mathbf{A}_{t-1},a_t) - Q_{\tilde{\pi},t}(\mathbf{O}_t,\mathbf{A}_{t-1},a_t)|p_t(a_t|\mathbf{O}_t,\mathbf{A}_{t-1}).
\end{aligned}$$

Insert the above into (7) and use Lemma A1 to obtain $|V_{\tilde{\pi}}(o_0) - V_\pi(o_0)|$

$$\begin{aligned}
\leq\ & 2L\sum_{t=0}^{T}E_\pi\left[\sum_{a_t}|Q_t(\mathbf{O}_t,\mathbf{A}_{t-1},a_t) - \tilde{Q}_t(\mathbf{O}_t,\mathbf{A}_{t-1},a_t)|p_t(a_t|\mathbf{O}_t,\mathbf{A}_{t-1})\Big|O_0=o_0\right] \\
& \qquad + E_\pi\left[\sum_{a_t}|\tilde{Q}_t(\mathbf{O}_t,\mathbf{A}_{t-1},a_t) - Q_{\tilde{\pi},t}(\mathbf{O}_t,\mathbf{A}_{t-1},a_t)|p_t(a_t|\mathbf{O}_t,\mathbf{A}_{t-1})\Big|O_0=o_0\right] \\
=\ & 2L\sum_{t=0}^{T}E\left[\left(\prod_{\ell=0}^{t-1}\frac{1_{A_\ell=\pi_\ell}}{p_\ell(A_\ell|\mathbf{O}_\ell,\mathbf{A}_{\ell-1})}\right)|Q_t-\tilde{Q}_t|\Big|O_0=o_0\right] \\
& \qquad + E\left[\left(\prod_{\ell=0}^{t-1}\frac{1_{A_\ell=\pi_\ell}}{p_\ell(A_\ell|\mathbf{O}_\ell,\mathbf{A}_{\ell-1})}\right)|\tilde{Q}_t-Q_{\tilde{\pi},t}|\Big|O_0=o_0\right] \\
\leq\ & 2\sum_{t=0}^{T}L^{t+1}E\left[|Q_t-\tilde{Q}_t|\Big|O_0=o_0\right] + 2\sum_{t=0}^{T}L^{t+1}E\left[|\tilde{Q}_t-Q_{\tilde{\pi},t}|\Big|O_0=o_0\right]
\end{aligned}$$

(Q_t, $Q_{\tilde{\pi},t}$ is used as abbreviation for $Q_t(\mathbf{O}_t,\mathbf{A}_t)$, respectively $Q_{\tilde{\pi},t}(\mathbf{O}_t,\mathbf{A}_t)$). This completes the proof of the first result.

Start from (8) and use Hölder's inequality to obtain, $|V_{\tilde{\pi}}(o_0) - V_\pi(o_0)|$

$$\begin{aligned}
\leq\ & 2\sum_{t=0}^{T}E_\pi\left[\max_{a_t}|Q_t(\mathbf{O}_t,\mathbf{A}_{t-1},a_t) - \tilde{Q}_t(\mathbf{O}_t,\mathbf{A}_{t-1},a_t)|\Big|O_0=o_0\right] \\
& \qquad + E_\pi\left[\max_{a_t}|\tilde{Q}_t(\mathbf{O}_t,\mathbf{A}_{t-1},a_t) - Q_{\tilde{\pi},t}(\mathbf{O}_t,\mathbf{A}_{t-1},a_t)|\Big|O_0=o_0\right] \\
\leq\ & 2\sum_{t=0}^{T}\sqrt{E_\pi\left[\max_{a_t}|Q_t(\mathbf{O}_t,\mathbf{A}_{t-1},a_t) - \tilde{Q}_t(\mathbf{O}_t,\mathbf{A}_{t-1},a_t)|^2\Big|O_0=o_0\right]} \\
& \qquad + \sqrt{E_\pi\left[\max_{a_t}|\tilde{Q}_t(\mathbf{O}_t,\mathbf{A}_{t-1},a_t) - Q_{\tilde{\pi},t}(\mathbf{O}_t,\mathbf{A}_{t-1},a_t)|^2\Big|O_0=o_0\right]} \\
\leq\ & 2\sum_{t=0}^{T}\sqrt{LE_\pi\left[\sum_{a_t}|Q_t(\mathbf{O}_t,\mathbf{A}_{t-1},a_t) - \tilde{Q}_t(\mathbf{O}_t,\mathbf{A}_{t-1},a_t)|^2 p_t(a_t|\mathbf{O}_t,\mathbf{A}_{t-1})\Big|O_0=o_0\right]} \\
& +2\sum_{t=0}^{T}\sqrt{LE_\pi\left[\sum_{a_t}|\tilde{Q}_t(\mathbf{O}_t,\mathbf{A}_{t-1},a_t) - Q_{\tilde{\pi},t}(\mathbf{O}_t,\mathbf{A}_{t-1},a_t)|^2 p_t(a_t|\mathbf{O}_t,\mathbf{A}_{t-1})\Big|O_0=o_0\right]}.
\end{aligned}$$

Now use Lemma A1 and the lower bound on the p_t's to obtain the result,

$$\begin{aligned}
|V_{\tilde{\pi}}(o_0) - V_{\pi}(o_0)| &\leq 2\sum_{t=0}^{T} \sqrt{LE\left[\prod_{\ell=0}^{t-1} \frac{1_{A_\ell=\pi_\ell}}{p_\ell(A_\ell|\mathbf{O}_\ell,\mathbf{A}_{\ell-1})}\left(Q_t - \tilde{Q}_t\right)^2 \Big| O_0 = o_0\right]} \\
&\quad + \sqrt{LE\left[\prod_{\ell=0}^{t-1} \frac{1_{A_\ell=\pi_\ell}}{p_\ell(A_\ell|\mathbf{O}_\ell,\mathbf{A}_{\ell-1})}\left(\tilde{Q}_t - Q_{\tilde{\pi},t}\right)^2 \Big| O_0 = o_0\right]} \\
&\leq 2L^{(t+1)/2}\sum_{t=0}^{T} \sqrt{E\left[\left(Q_t - \tilde{Q}_t\right)^2 \Big| O_0 = o_0\right]} \\
&\quad + \sqrt{E\left[\left(\tilde{Q}_t - Q_{\tilde{\pi},t}\right)^2 \Big| O_0 = o_0\right]}.
\end{aligned}$$

■

5. Finite Sample Upper Bounds on the Average Generalization Error

Traditionally the performance of a policy π is evaluated in terms of maximum generalization error: $\max_o[V^*(o) - V_\pi(o)]$ (Bertsekas and Tsitsiklis, 1996). However here we consider an average generalization error as in Kakade (2003) (see also Fiechter, 1997; Kearns, Mansour and Ng, 2000; Peshkin and Shelton, 2002); that is $\int_o [V^*(o) - V_\pi(o)]dF(o)$ for a specified distribution F on the observation space. The choice of F with density $f = f_0$ (f_0 is the density of O_0 in likelihoods (1) and (2)) is particularly appealing in the development of a policy in many medical and social science applications. In these cases, f_0 represents the distribution of initial observations corresponding to a particular population of subjects. The goal is to produce a good policy for this population of subjects. In general as in Kakade (2003) F may be chosen to incorporate domain knowledge concerning the steady state dis! tribution of a good policy. If only a training set of trajectories is available for learning and we are unwilling to assume that the system dynamics are Markovian, then the choice of F is constrained by the following consideration. If the distribution of O_0 in the training set (f_0) assigns mass zero to an observation o', then the training data will not be able to tell us anything about $V_\pi(o')$. Similarly if f_0 assigns a very small positive mass to o' then only an exceptionally large training set will permit an accurate estimate of $V_\pi(o')$. Of course this will not be a problem for the average generalization error, as long as F also assigns very low mass to o'. Consequently in our construction of the finite sample error bounds for the *average* generalization error, we will only consider distributions F for which the density of F, say f, satisfies $\sup_o |\frac{f(o)}{f_0(o)}| \leq M$ for some finite constant M. In this case the average generalization error is bounded above by

$$\begin{aligned}
\int V^*(o) - V_\pi(o)\, dF(o) &\leq ME\left[V^*(O_0) - V_\pi(O_0)\right] \\
&= -ME_\pi\left[\sum_{t=0}^{T} \mu_t^*(\mathbf{O}_t, \mathbf{A}_t)\right].
\end{aligned}$$

The second line is a consequence of (5) and the fact that the distribution of O_0 is the same under likelihoods (1) and (2).

In the following theorem a non-asymptotic upper bound on the average generalization error is provided; this upper bound depends on the number of trajectories in the training set (n), the performance of the approximation on the training set, the complexity of the approximation space and of course on the confidence (δ) and accuracy (ε) demanded. The batch Q-learning algorithm minimizes quadratic forms (see (3)); thus we represent the performance of functions $\{Q_0, Q_1, \ldots, Q_T\}$ on the training set by these quadratic forms,

$$Err_{n,Q_{t+1}}(Q_t) = \mathbb{E}_n \left[R_t + \max_{a_{t+1}} Q_{t+1}(\mathbf{O}_{t+1}, \mathbf{A}_t, a_{t+1}) - Q_t(\mathbf{O}_t, \mathbf{A}_t) \right]^2$$

for each j (recall Q_{T+1} is set to zero and $\mathbb{E}_n$ represents the expectation with respect to the probability obtained by choosing a trajectory uniformly from the training set).

The complexity of each $\mathcal{Q}_t$ space can be represented by it's covering number (Anthony and Bartlett, 1999, pg 148). Suppose $\mathcal{F}$ is a class of functions from a space, X, to $\mathbb{R}$. For a sequence $x = (x_1, \ldots, x_n) \in X^n$, define $\mathcal{F}_{|x}$ to be a subset of $\mathbb{R}^n$ given by $\mathcal{F}_{|x} = \{(f(x_1), \ldots, f(x_n)) : f \in \mathcal{F}\}$. Define the metric d_p on $\mathbb{R}^n$ by $d_p(z,y) = (1/n \sum_{i=1}^n |z_i - y_i|^p)^{1/p}$ for p a positive integer (for $p = \infty$, define $d_\infty(z,y) = \max_{i=1}^n |z_i - y_i|$). Then $\mathcal{N}(\varepsilon, \mathcal{F}_{|x}, d_p)$ is defined as the minimum cardinality of an ε-covering of $\mathcal{F}_{|x}$ with respect to the metric d_p. Next given $\varepsilon > 0$, positive integer n, metric d_p and function class, $\mathcal{F}$, the covering number for $\mathcal{F}$ is defined as

$$\mathcal{N}_p(\varepsilon, \mathcal{F}, n) = \max\{\mathcal{N}(\varepsilon, \mathcal{F}_{|x}, d_p) : x \in X^n\}.$$

In the following theorem, $\mathcal{F} = \{\max_{a_{t+1}} Q_{t+1}(\mathbf{o}_{t+1}, \mathbf{a}_t) - Q_t(\mathbf{o}_t, \mathbf{a}_t) : Q_t \in \mathcal{Q}_t, t = 0, \ldots, T\}$ and $(x)^+$ is x if $x > 0$ and zero otherwise.

Theorem 1

Assume that the functions in $\mathcal{Q}_t$, $t \in 0, \ldots, T$ are uniformly bounded. Suppose that there exists a positive constant, say L, for which $p_t(a_t|\mathbf{o}_t, \mathbf{a}_t) \geq L^{-1}$ for all $(\mathbf{o}_t, \mathbf{a}_t)$ pairs, $0 \leq t \leq T$. Then for $\varepsilon > 0$ and with probability at least $1 - \delta$, over the random choice of the training set, every choice of functions, $Q_j \in \mathcal{Q}_j,\ j = 0, \ldots, T$ with associated policy π defined by $\pi_j(\mathbf{o}_j, \mathbf{a}_{j-1}) = \arg\max_{a_j} Q_j(\mathbf{o}_j, \mathbf{a}_j)$ and every choice of functions $\tilde{Q}_j \in \mathcal{Q}_j,\ j = 0, \ldots, T$ with associated policy $\tilde{\pi}$ defined by $\tilde{\pi}_j(\mathbf{o}_j, \mathbf{a}_{j-1}) = \arg\max_{a_j} \tilde{Q}_j(\mathbf{o}_j, \mathbf{a}_j)$ the following bound is satisfied,

$$\begin{aligned}
&\int |V_{\tilde{\pi}}(o) - V_\pi(o)|\, dF(o) \\
&\leq \quad 6ML^{1/2} \sum_{t=0}^{T} \left[\sum_{i=t}^{T} (16)^{i-t} L^i \left(Err_{n,Q_{i+1}}(Q_i) - Err_{n,Q_{i+1}}(\tilde{Q}_i) \right)^+ \right]^{1/2} \\
&\qquad + 12ML^{1/2}\varepsilon \\
&\qquad + 6ML^{1/2} \sum_{t=0}^{T} \sum_{i=t}^{T} (16)^{(i-t)/2} L^{i/2} \sqrt{E[\tilde{Q}_i(\mathbf{O}_i, \mathbf{A}_i) - Q_{\tilde{\pi},i}(\mathbf{O}_i, \mathbf{A}_i)]^2}.
\end{aligned}$$

for n satisfying

$$4(T+1)\mathcal{N}_1\left(\frac{\varepsilon^2}{32M'(16L)^{(T+2)}}, \mathcal{F}, 2n \right) \exp\left\{ -\frac{\varepsilon^4 n}{32(M')^2(16L)^{2(T+2)}} \right\} \leq \delta \tag{9}$$

and where M' is a uniform upper bound on the absolute value of $f \in \mathcal{F}$ and E represents the expectation with respect to the distribution (1) generating the training set.

Remarks:

1. Suppose that $Q_t^* \in \mathcal{Q}_t$ for each t. Select $\tilde{Q}_t = Q_t^*$ and $\widehat{Q}_t = \arg\min_{Q_t \in \mathcal{Q}_t} Err_{n,\widehat{Q}_{t+1}}(Q_t)$, $t = T, T-1, \ldots, 0$ (recall Q_{T+1}, $\widehat{Q}_{T+1}$ are identically zero). Then with probability greater than $1-\delta$, we obtain,

$$\int V^*(o) - V_{\widehat{\pi}}(o)\, dF(o) \leq 12ML^{1/2}\varepsilon \tag{10}$$

   for all n satisfying (9). Thus, as long as the covering numbers for each $\mathcal{Q}_t$ and thus for $\mathcal{F}$ do not grow too fast, estimating each Q_t by minimizing $Err_{n,\widehat{Q}_{t+1}}(Q_t)$ yields a policy that consistently achieves the optimal value. Suppose the approximation spaces $\mathcal{Q}_t$, $t = 0, \ldots, T$ are feed-forward neural networks as in remark 4 below. In this case the training set size n sufficient for (10) to hold need only be polynomial in $(1/\delta, 1/\varepsilon)$ and batch Q-learning is a probably approximate correct (PAC) reinforcement learning algorithm as defined by Fiechter (1997). As shown by Fiechter (1997) this algorithm can be converted to an efficient on-line reinforcement learning algorithm (here the word on-line implies updating the policy between trajectories).

2. Even when Q_t^* does not belong to $\mathcal{Q}_t$ we can add the optimal Q function at each time, t, to the approximation space, $\mathcal{Q}_t$ with a cost of no more than an increase of 1 to the covering number $\mathcal{N}_1\left(\frac{\varepsilon^2}{32M'(16L)^{(T+2)}}, \mathcal{F}, 2n\right)$. If we do this the result continues to hold when we set $\tilde{\pi}$ to an optimal policy π^* and set $\tilde{Q}_t = Q_t^*$ for each t; the generalization error is

$$\begin{aligned}\int V^*(o) - V_{\pi}(o)\, dF(o) \quad &\leq \quad 6ML^{1/2}\sum_{t=0}^{T}\left[\sum_{i=t}^{T}(16)^{i-t}L^i Err_{n,Q_{i+1}}(Q_i)\right]^{1/2} \\ &\quad + 12ML^{1/2}\varepsilon\end{aligned}$$

   for all n satisfying (9). This upper bound is consistent with the practice of using a policy $\widehat{\pi}$ for which $\widehat{\pi}_t(\mathbf{o}_t, \mathbf{a}_{t-1}) \in \arg\max_{a_t} \widehat{Q}_t(\mathbf{o}_t, \mathbf{a}_t)$ and $\widehat{Q}_t \in \arg\min_{Q_t \in \mathcal{Q}_t} Err_{n,\widehat{Q}_{t+1}}(Q_t)$. Given that the covering numbers for the approximation space can be expressed in a sufficiently simple form (as in remark 4 below), this upper bound can be used to carry out model selection using structural risk minimization (Vapnik, 1982). That is, one might consider a variety of approximation spaces and use structural risk minimization to use the training data to choose which approximation space is best. The resulting upper bound on the average generalization error can be found by using the above result and Lemma 15.5 of Anthony and Bartlett (1999).

3. The restriction on n in (9) is due to the complexity associated with the approximation space (e.g. the $\mathcal{Q}_t$'s). The restriction is crude; to see this, note that if there were only a finite number of functions in $\mathcal{F}$ then n need only satisfy

$$2(T+1)|\mathcal{F}|\exp\left\{-\frac{2\varepsilon^4 n}{(3M')^2\,(16L)^{2(T+2)}}\right\} = \delta$$

   (use Hoeffding's inequality; see Anthony and Bartlett, pg 361, 1999) and thus for a given (ε, δ) we may set the number of trajectories in the training set n equal to

$$\frac{(3M')^2(16L)^{2(T+2)}}{2\varepsilon^4}\ln\left(\frac{2(T+1)|\mathcal{F}|}{\delta}\right).$$

This complexity term appears similar to that achieved by learning algorithms (e.g. see Anthony and Bartlett, 1999, pg. 21) or in reinforcement learning (e.g. Peshkin and Shelton, 2002) however note that n is of the order ε^{-4} rather than the usual ε^{-2}. The ε^{-4} term (instead of ε^{-2}) is attributable to the fact that $Err_{Q_{t+1}}(Q_t)$ is not only a function of Q_t but also of Q_{t+1}. However further assumptions on the approximation space permit an improved result. See Theorem 2 below for one possible refinement. Note the needed training set size n depends exponentially on the horizon time T but not on the dimension of the observation space. Thi! s is not unexpected as the upper bounds on the generalization error of both Kearns, Mansour and Ng (2000) and Peshkin and Shelton's (2002) policy search methods (the latter using a training set and importance sampling weights) also depend exponentially on the horizon time.

4. When $\mathcal{F}$ is infinite, we use covering numbers for the approximation space $\mathcal{Q}_t$ and then appeal to Lemma A2 in the appendix to derive a covering number for $\mathcal{F}$; this results in

$$\mathcal{N}_1(\varepsilon, \mathcal{F}, n) \leq (T+1) \max_{t=0,\ldots,T} \mathcal{N}_1\left(\frac{\varepsilon}{2|\mathcal{A}|}, \mathcal{Q}_t, |\mathcal{A}|n\right)^2.$$

One possible approximation space is based on feed-forward neural networks. From Anthony and Bartlett (1999) we have that if each $\mathcal{Q}_j$ is the class of functions computed by a feed-forward network with W weights and k computation units arranged in L layers and each computation unit has a fixed piecewise-polynomial activation function with q pieces and degree no more than ℓ, then $\mathcal{N}_1(\varepsilon, \mathcal{Q}_t, n) \leq e(d+1)\left(\frac{2eM'}{\varepsilon}\right)^d$ where $d = 2(W+1)(L+1)\log_2(4(W+1)(L+1)q(k+1)/\ln 2) + 2(W+1)(L+1)^2\log_2(\ell+1) + 2(L+1)$. To see this combine Anthony and Bartlett's Theorems 8.8, 14.1 and 18.4. They provide covering numbers for functions computed by other types of neural networks as well. A particularly simple neural network is an affine combination of a given set of p input features; i.e. $f(x) = \omega_0 + \sum_{i=1}^{p-1}\omega_i x_i$ for $(1,x)$ a vector of p real valued features and each $\omega_i \in \mathbb{R}$. Suppose each $\mathcal{Q}_t$ is a class of functions computed by this network. Then Theorems 11.6 and 18.4 of Anthony and Bartlett imply that $\mathcal{N}_1(\varepsilon, \mathcal{Q}_t, n) \leq e(p+1)\left(\frac{2eM'}{\varepsilon}\right)^p$. In this case

$$n \geq \frac{32(M')^4(16L)^{2(T+2)}}{\varepsilon^4}\log\left(\frac{4(T+1)^2e^2(p+1)^2\left(128e|\mathcal{A}|(M')^2(16L)^{T+2}\right)^{2p}}{\delta\varepsilon^{4p}}\right).$$

This number will be large for any reasonable accuracy, ε and confidence, δ.

Proof of Theorem 1. An upper bound on the average difference in value functions can be obtained from Lemma 2 by using Jensen's inequality and the assumption that the the density of F (f) satisfies $\sup_o |\frac{f(o)}{f_0(o)}| \leq M$ for some finite constant M:

$$\begin{aligned}\int |V_{\tilde{\pi}}(o) - V_\pi(o)|\,dF(o) \quad &\leq \quad M\sum_{t=0}^{T} 2L^{(t+1)/2}\sqrt{E\left[Q_t(\mathbf{O}_i,\mathbf{A}_i) - \tilde{Q}_t(\mathbf{O}_i,\mathbf{A}_i)\right]^2} \\ &\quad +M\sum_{t=0}^{T} 2L^{(t+1)/2}\sqrt{E\left[\tilde{Q}_t - Q_{\tilde{\pi},t}\right]^2}\end{aligned} \qquad (11)$$

where $\tilde{Q}_t$, $Q_{\tilde{\pi},t}$ is used as abbreviation for $\tilde{Q}_t(\mathbf{O}_t, \mathbf{A}_t)$, respectively $Q_{\tilde{\pi},t}(\mathbf{O}_t, \mathbf{A}_t)$. In the following an upper bound on each $E\left[Q_t - \tilde{Q}_t\right]^2$ is constructed.

The performance of the approximation on an infinite training set can be represented by

$$Err_{Q_{t+1}}(Q_t) = E\left[R_t + \max_{a_{t+1}} Q_{t+1}(\mathbf{O}_{t+1}, \mathbf{A}_t, a_{t+1}) - Q_t\right]^2$$

for each t (recall $Q_{T+1} = 0$, also we abbreviate $Q_t(\mathbf{O}_t, \mathbf{A}_t)$ by Q_t whenever no confusion may arise). The errors, Err's, can be used to provide an upper bound on the L_2 norms on the Q-functions by the following argument. Consider $Err_{Q_{t+1}}(Q_t) - Err_{Q_{t+1}}(\tilde{Q}_t)$ for each t. Within each of these quadratic forms add and subtract

$$Q_{\tilde{\pi},t+1}(\mathbf{O}_{t+1}, \mathbf{A}_t, \tilde{\pi}_{t+1}) - Q_{\tilde{\pi},t} - E\left[\max_{a_{t+1}} Q_{t+1}(\mathbf{O}_{t+1}, \mathbf{A}_t, a_{t+1}) - Q_{\tilde{\pi},t+1}(\mathbf{O}_{t+1}, \mathbf{A}_t, \tilde{\pi}_{t+1}) | \mathbf{O}_t, \mathbf{A}_t\right].$$

In the above $Q_{\tilde{\pi},t+1}(\mathbf{O}_{t+1}, \mathbf{A}_t, \tilde{\pi}_{t+1})$ is defined as $Q_{\tilde{\pi},t+1}(\mathbf{O}_{t+1}, \mathbf{A}_t, a_{t+1})$ with a_{t+1} replaced by $\tilde{\pi}_{t+1}(\mathbf{O}_{t+1}, \mathbf{A}_t)$. Expand each quadratic form and use the fact that $E\left[R_t + Q_{\tilde{\pi},t+1}(\mathbf{O}_{t+1}, \mathbf{A}_t, \tilde{\pi}_{t+1}) | \mathbf{O}_t, \mathbf{A}_t\right] = Q_{\tilde{\pi},t}$. Cancelling common terms yields

$$\begin{aligned} E&\left[Q_{\tilde{\pi},t} - Q_t + E\left[\max_{a_{t+1}} Q_{t+1}(\mathbf{O}_{t+1}, \mathbf{A}_t, a_{t+1}) - Q_{\tilde{\pi},t+1}(\mathbf{O}_{t+1}, \mathbf{A}_t, \tilde{\pi}_{t+1}) | \mathbf{O}_t, \mathbf{A}_t\right]\right]^2 \\ &\quad - E\left[Q_{\tilde{\pi},t} - \tilde{Q}_t + E\left[\max_{a_{t+1}} Q_{t+1}(\mathbf{O}_{t+1}, \mathbf{A}_t, a_{t+1}) - Q_{\tilde{\pi},t+1}(\mathbf{O}_{t+1}, \mathbf{A}_t, \tilde{\pi}_{t+1}) | \mathbf{O}_t, \mathbf{A}_t\right]\right]^2. \end{aligned}$$

Add and subtract $\tilde{Q}_t$ in the first quadratic form and expand. This yields

$$\begin{aligned} &Err_{Q_{t+1}}(Q_t) - Err_{Q_{t+1}}(\tilde{Q}_t) = \\ &\qquad E\left[\tilde{Q}_t - Q_t\right]^2 + 2E\left[\tilde{Q}_t - Q_t\right]\left[\tilde{Q}_t - Q_{\tilde{\pi},t}\right] \\ &\qquad + 2E\left[(\tilde{Q}_t - Q_t)\left(\max_{a_{t+1}} Q_{t+1}(\mathbf{O}_{t+1}, \mathbf{A}_t, a_{t+1}) - \max_{a_{t+1}} \tilde{Q}_{t+1}(\mathbf{O}_{t+1}, \mathbf{A}_t, a_{t+1})\right)\right] \\ &\qquad + 2E\left[(\tilde{Q}_t - Q_t)\left(\max_{a_{t+1}} \tilde{Q}_{t+1}(\mathbf{O}_{t+1}, \mathbf{A}_t, a_{t+1}) - Q_{\tilde{\pi},t+1}(\mathbf{O}_{t+1}, \mathbf{A}_t, \tilde{\pi}_{t+1})\right)\right]. \end{aligned} \tag{12}$$

Using the arguments similar to those used around Equation (8) and using the fact that $(x+y)^2 \le 2x^2 + 2y^2$ we obtain,

$$\begin{aligned} Err_{Q_{t+1}}(Q_t) - Err_{Q_{t+1}}(\tilde{Q}_t) &\ge E\left[Q_t - \tilde{Q}_t\right]^2 \\ &- 4\left(E[Q_t - \tilde{Q}_t]^2\left(E\left[\tilde{Q}_t - Q_{\tilde{\pi},t}\right]^2 + LE\left[Q_{t+1} - \tilde{Q}_{t+1}\right]^2 + LE\left[\tilde{Q}_{t+1} - Q_{\tilde{\pi},t+1}\right]^2\right)\right)^{1/2}. \end{aligned}$$

Using this inequality we can now derive an upper bound on each $E\left[Q_t - \tilde{Q}_t\right]^2$ in terms of the Err's and the $E\left[\tilde{Q}_{t+1} - Q_{\tilde{\pi},t+1}\right]$'s. Define

$$m_t = L^{-(T-t)} E\left[\tilde{Q}_t - Q_{\tilde{\pi},t}\right]^2 \text{ and } b_t = L^{-(T-t)} E\left[Q_t - \tilde{Q}_t\right]^2$$

and

$$e_t = L^{-(T-t)}\left(Err_{Q_{t+1}}(Q_t) - Err_{Q_{t+1}}(\tilde{Q}_t)\right)$$

for $t \leq T$ and $b_{T+1} = m_{T+1} = e_{T+1} = 0$. We obtain

$$e_t \geq b_t - 4\sqrt{b_t(m_t + b_{t+1} + m_{t+1})}.$$

Completing the square, reordering terms, squaring once again and using the inequality $(x+y)^2 \leq 2x^2 + 2y^2$ yields $b_t \leq 16(b_{t+1} + m_t + m_{t+1}) + 2e_t$ for $t \leq T$. We obtain

$$b_{T-t} \quad \leq \quad 2\sum_{i=0}^{t}(16)^i e_{T-t+i} + \sum_{i=1}^{t}(16)^i(16+1)m_{T-t+i} + 16m_{T-t}.$$

Inserting the definitions of b_{T-t}, e_{T-t+i} and reordering, yields

$$\begin{aligned} E\left[Q_t - \tilde{Q}_t\right]^2 \quad \leq \quad & 2\sum_{i=t}^{T}(16L)^{i-t}\left(Err_{Q_{i+1}}(Q_i) - Err_{Q_{i+1}}(\tilde{Q}_i)\right) \\ & + \sum_{i=t+1}^{T}(16)^{i-t}(16+1)L^{T-t}m_i + L^{T-t}m_t. \end{aligned} \tag{13}$$

As an aside we can start from (12) and derive the upper bound,

$$\begin{aligned} & Err_{Q_{t+1}}(Q_t) - Err_{Q_{t+1}}(\tilde{Q}_t) \leq E\left[Q_t - \tilde{Q}_t\right]^2 \\ & \qquad + 4L^{(T-t)}\sqrt{L^{-(T-t)}E\left[Q_t - \tilde{Q}_t\right]^2\left(m_t + L^{-(T-t-1)}E\left[Q_{t+1} - \tilde{Q}_{t+1}\right]^2 + m_{t+1}\right)}. \end{aligned}$$

This combined with (13) implies that minimizing each $Err_{Q_{t+1}}(Q_t) - Err_{Q_{t+1}}(\tilde{Q}_t)$ in Q_t is equivalent to minimizing each $E\left[Q_t - \tilde{Q}_t\right]^2$ in Q_t modulo the approximation terms m_t for $t = 0, \ldots, T$.

Returning to the proof next note that

$$\begin{aligned} Err_{Q_{t+1}}(Q_t) - Err_{Q_{t+1}}(\tilde{Q}_t) \quad \leq \quad & \left|Err_{Q_{t+1}}(Q_t) - Err_{n,Q_{t+1}}(Q_t)\right| \\ & + \left|Err_{Q_{t+1}}(\tilde{Q}_t) - Err_{n,Q_{t+1}}(\tilde{Q}_t)\right| \\ & + \left(Err_{n,Q_{t+1}}(Q_t) - Err_{n,Q_{t+1}}(\tilde{Q}_t)\right)^+ \end{aligned}$$

where $(x)^+$ is equal to x if $x \geq 0$ and is equal to 0 otherwise. Note that if each Q_t minimizes $Err_{n,Q_{t+1}}$ as in (3) then the third term is zero. Substituting into (13), we obtain

$$\begin{aligned} E\left[Q_t - \tilde{Q}_t\right]^2 \quad \leq \quad & 2\sum_{i=t}^{T}(16L)^{i-t}\Big(\left|Err_{Q_{i+1}}(Q_i) - Err_{n,Q_{i+1}}(Q_i)\right| \\ & \qquad + \left|Err_{Q_{i+1}}(\tilde{Q}_i) - Err_{n,Q_{i+1}}(\tilde{Q}_i)\right| \\ & \qquad + \left(Err_{n,Q_{i+1}}(Q_i) - Err_{n,Q_{i+1}}(\tilde{Q}_i)\right)^+\Big) \\ & + \sum_{i=t+1}^{T}(16)^{i-t}(16+1)L^{i-t}E[\tilde{Q}_i - Q_{\tilde{\pi},i}]^2 + E[\tilde{Q}_t - Q_{\tilde{\pi},t}]^2. \end{aligned}$$

Combine this inequality with (11); simplify the sums and use the fact that for x, y both nonnegative $\sqrt{x+y} \leq \sqrt{x} + \sqrt{y}$ to obtain $\int |V_{\tilde{\pi}}(o) - V_{\pi}(o)|\, dF(o)$

$$\leq \quad 6ML^{1/2}\sum_{t=0}^{T}\left[\sum_{i=t}^{T}(16)^{i-t}L^i\left(Err_{n,Q_{i+1}}(Q_i) - Err_{n,Q_{i+1}}(\tilde{Q}_i)\right)^+\right]^{1/2}$$

$$+\ 12ML^{1/2}(16L)^{(T+2)/2}\sqrt{\max_t \sup_{Q_t,Q_{t+1}} \left|Err_{Q_{t+1}}(Q_t) - Err_{n,Q_{t+1}}(Q_t)\right|}$$
$$+\ 6ML^{1/2}\sum_{t=0}^{T}\sum_{i=t}^{T}(16)^{(i-t)/2}L^{i/2}\sqrt{E[\tilde{Q}_i - Q_{\tilde{\pi},i}]^2}.$$

All that remains is to provide an upper bound on

$$P\left[\bigcup_{i=0}^{T}\left\{\text{for some}_{Q_t\in \mathcal{Q}_t, t=0,\ldots,T}\left|Err_{Q_{t+1}}(Q_i) - Err_{n,Q_{i+1}}(Q_i))\right| > \varepsilon'\right\}\right].$$

This probability is in turn bounded above by

$$\sum_{i=0}^{T} P\left[\text{ for some}_{Q_t\in \mathcal{Q}_t, t=0,\ldots,T}\left|Err_{Q_{i+1}}(Q_i) - Err_{n,Q_{i+1}}(Q_i)\right| > \varepsilon'\right].$$

Anthony and Bartlett (1999, pg. 241) use Hoeffding's inequality along with the classical techniques of symmetrization and permutation to provide the upper bound (see also van der Vaart and Wellner, 1996),

$$P\left[\text{ for some}_{Q_t\in \mathcal{Q}_t, t=0,\ldots,T}\left|Err_{Q_{i+1}}(Q_i) - Err_{n,Q_{i+1}}(Q_i)\right| > \varepsilon'\right]$$
$$\leq 4\mathcal{N}_1\left(\frac{\varepsilon'}{32M'}, \mathcal{F}, 2n\right)\exp\left\{-\frac{(\varepsilon')^2 n}{32(M')^2}\right\}.$$

Put $\varepsilon = (16L)^{(T+2)/2}\sqrt{\varepsilon'}$ to obtain the results of the theorem. ■

Suppose the Q functions are approximated by linear combinations of p features; for each $t = 0,\ldots,T$, denote the feature vector by $q_t(\mathbf{o}_t,\mathbf{a}_t)$. The approximation space is then,

$$\mathcal{Q}_t = \{Q_t(\mathbf{o}_t,\mathbf{a}_t) = \theta^T q_t(\mathbf{o}_t,\mathbf{a}_t) : \theta \in \Theta\}$$

where Θ is a subset of $\mathbf{R}^p$. In this case, the batch Q-learning algorithm may be based on (4); we represent the performance of the functions $\{Q_0,\ldots,Q_t\}$ on the training set by

$$\widetilde{Err}_{n,Q_{t+1}}(Q_t) = \mathbb{E}_n\left[\left(R_t + \max_{a_{t+1}} Q_{t+1}(\mathbf{O}_{t+1},\mathbf{A}_t,a_{t+1}) - Q_t(\mathbf{O}_t,\mathbf{A}_t)\right)q_t(\mathbf{O}_t,\mathbf{A}_t)\right]$$

for $t = 0,\ldots,T$ (recall $\mathbb{E}_n$ represents the expectation with respect to the probability obtained by choosing a trajectory uniformly from the training set). In this theorem

$$\mathcal{F}' = \bigcup_{i=1}^{p}\bigcup_{t=1}^{T}\left\{\left(r_t + \max_{a_{t+1}} Q_{t+1}(\mathbf{o}_{t+1},\mathbf{a}_{t+1};\theta_{t+1}) - Q_t(\mathbf{o}_t,\mathbf{a}_t;\theta_t)\right)q_{ti}(\mathbf{o}_t,\mathbf{a}_t) : \theta_t,\theta_{t+1}\in\Theta\right\}.$$

Define the functions $\{\overline{Q}_0,\ldots,\overline{Q}_T\}$, and the policy, $\overline{\pi}$, as follows. First define $\overline{Q}_T(\mathbf{O}_T,\mathbf{A}_T)$ to be the projection of $E\left[R_T|\mathbf{O}_T,\mathbf{A}_T\right]$ on the space spanned by q_T. Then set $\overline{\pi}_T(\mathbf{o}_T,\mathbf{a}_{T-1}) \in \arg\max_{a_T}\overline{Q}_T(\mathbf{o}_T,\mathbf{a}_T)$. Next for $t = T-1,\ldots,0$, set $\overline{Q}_t(\mathbf{O}_t,\mathbf{A}_t)$ as the projection of $E\left[R_t + \overline{Q}_{t+1}(\mathbf{O}_{t+1},\mathbf{A}_t,\overline{\pi}_{t+1})|\mathbf{O}_t,\mathbf{A}_t\right]$ on the space spanned by q_t (recall $\overline{Q}_{t+1}(\mathbf{O}_{t+1},\mathbf{A}_t,\overline{\pi}_{t+1})$ is defined as $\overline{Q}_{t+1}(\mathbf{O}_{t+1},\mathbf{A}_t,a_{t+1})$ with a_{t+1} replaced by $\overline{\pi}_{t+1}(\mathbf{O}_{t+1},\mathbf{A}_t)$). And set $\overline{\pi}_t(\mathbf{o}_t,\mathbf{a}_{t-1}) \in \arg\max_{a_t}\overline{Q}_t(\mathbf{o}_t,\mathbf{a}_t)$. These projections are with respect to P, the distribution which generated the trajectories in the training set (the likelihood is in (1)).

Theorem 2

Suppose that there exists a positive constant, say L, for which $p_t(a_t|\mathbf{o}_t,\mathbf{a}_{t-1}) \geq L^{-1}$ for all $(\mathbf{o}_t,\mathbf{a}_{t-1})$, $0 \leq t \leq T$. Suppose that for each t, $x \in \mathbf{R}^p$, $x^T E q_t q_t^T x > \eta ||x||^2$ where $\eta > 0$ ($||\cdot||$ is the Euclidean norm). Also assume that Θ is a closed subset of $\{x \in \mathbf{R}^p : ||x|| \leq M_\Theta\}$ and for all (t,i), the ith component in the vector q_t is pointwise bounded; $|q_{ti}| \leq M_Q$ for M_Q a constant. Then for $\varepsilon > 0$, with probability at least $1-\delta$, over the random choice of the training set, every choice of functions, $Q_t \in \mathcal{Q}_t$ and functions $\tilde{Q}_t$, $t = 0,\ldots,T$ with associated policies defined by π with $\pi_t(\mathbf{o}_t,\mathbf{a}_{t-1} \in \arg\max_{a_t} Q_t(\mathbf{o}_t,\mathbf{a}_t)$ and $\tilde{\pi}$ with $\tilde{\pi}_t(\mathbf{o}_t,\mathbf{a}_{t-1}) \in \arg\max_{a_t} \tilde{Q}_t(\mathbf{o}_t,\mathbf{a}_t)$ respectively, the following bounds are satisfied,

$$\sum_{t=0}^{T} L^{(t+1)} E|\bar{Q}_t(\mathbf{O}_t,\mathbf{A}_t) - Q_t(\mathbf{O}_t,\mathbf{A}_t)| \leq \sqrt{p} M_Q/\eta \sum_{t=0}^{T} L^{(t+1)} \sum_{j=t}^{T} \left(LpM_Q^2/\eta\right)^{j-t} ||\widetilde{Err}_{n,Q_{j+1}}(Q_j)|| + 4\varepsilon.$$

for $t = 0,\ldots,T$, where E represents the expectation with respect to the distribution (1) generating the training set and

$$\begin{aligned}\int |V_{\tilde{\pi}}(o) - V_\pi(o)|\, dF(o) \leq\ & 2M\sqrt{p} M_Q/\eta \sum_{t=0}^{T} L^{(t+1)} \sum_{j=t}^{T} \left(LpM_Q^2/\eta\right)^{j-t} ||\widetilde{Err}_{n,Q_{j+1}}(Q_j)|| \\ & + 8M\varepsilon \\ & +2M \sum_{t=0}^{T} L^{(t+1)} E \left|\bar{Q}_t(\mathbf{O}_t,\mathbf{A}_t) - \tilde{Q}_t(\mathbf{O}_t,\mathbf{A}_t)\right| \\ & +2M \sum_{t=0}^{T} L^{(t+1)} E \left|\tilde{Q}_t(\mathbf{O}_t,\mathbf{A}_t) - Q_{\tilde{\pi},t}(\mathbf{O}_t,\mathbf{A}_t)\right|\end{aligned}$$

for n larger than

$$\left(\frac{C}{\varepsilon}\right)^2 \log\left(\frac{B}{\delta}\right) \tag{14}$$

where $C = 4\sqrt{2} M' p^{T+1/2} M_Q^{2T+1} \eta^{-(T+1)} L^{T+1}$, M' is a uniform upper bound on the absolute value on all $f \in \mathcal{F}'$ and $B = \varepsilon^{-2p} 4^{6p+3} p^{2Tp+p+3} (T+1)^2 e^{2p+2} (M')^{4p} |\mathcal{A}|^p M_Q^{(2T+1)2p} \eta^{-2p(T+1)}\ L^{2p(T+1)}$

Remarks:

1. Define $\widehat{Q}_t$ as a zero of $\widetilde{Err}_{n,\widehat{Q}_{t+1}}(Q_t)$, $t = T, T-1, \ldots, 0$ (recall that $\widehat{Q}_{T+1}$ is identically zero). Suppose that $Q_t^* \in \mathcal{Q}_t$ for each t; in this case $\bar{Q}_t = Q_t^*$ for all t (we ignore sets of measure zero in this discussion). Then with probability greater than $1-\delta$ and $\tilde{\pi} = \pi^*$, $\tilde{Q}_t = Q_t^*$ we obtain

   $$\int V^*(o) - V_{\widehat{\pi}}(o)\, dF(o) \leq 8M\varepsilon$$

   for all n satisfying (14). Thus estimating each Q_t by solving $\widetilde{Err}_{n,Q_{t+1}}(Q_t) = 0$, $t = T,\ldots,0$, yields a policy that consistently achieves the optimal value.

2. Again define $\widehat{Q}_t$ as a zero of $\widetilde{Err}_{n,\widehat{Q}_{t+1}}(Q_t)$, $t = T, T-1, \ldots, 0$. Given two $T+1$ vectors of functions $Q' = \{Q'_0, \ldots, Q'_T\}$ and $Q = \{Q_0, \ldots, Q_T\}$ define
$$\ell(Q', Q) = \sum_{t=0}^{T} L^{t+1} E \left| Q'_t(\mathbf{O}_t, \mathbf{A}_t) - Q_t(\mathbf{O}_t, \mathbf{A}_t) \right|.$$
Then the first result of Theorem 2 implies that $\ell(\overline{Q}, \widehat{Q})$ converges in probability to zero. From Lemma 2 we have that $\int |V_{\tilde{\pi}}(o) - V_{\pi}(o)|\, dF(o) \le 2M\ell(Q, \tilde{Q}) + 2M\ell(Q_{\tilde{\pi}}, \tilde{Q})$ and thus $\int |V_{\tilde{\pi}}(o) - V_{\widehat{\pi}}(o)|\, dF(o)$ is with high probability bounded above by $2M\ell(\overline{Q}, \tilde{Q}) + 2M\ell(Q_{\tilde{\pi}}, \tilde{Q})$. Consequently the presence of the third and fourth terms in Theorem 2 is not surprising. It is unclear whether the "go-between"$\tilde{Q}_t$ is necessary.

3. Recall the space of policies implied by the approximation spaces for the Q-functions is given by $\Pi_{\mathcal{Q}} = \{\pi_\theta, \theta \in \Theta\}$ where $\pi_\theta = \{\pi_{1,\theta}, \ldots, \pi_{T,\theta}\}$ and where each $\pi_{t,\theta}(\mathbf{o}_t, \mathbf{a}_{t-1}) \in \arg\max_{a_t} Q_t(\mathbf{o}_t, \mathbf{a}_t; \theta)$ for some $Q_t \in \mathcal{Q}_t$. Suppose that $\max_{\pi \in \Pi_{\mathcal{Q}}} \int V_\pi(o)\, dF(o)$ is achieved by some member of $\Pi_{\mathcal{Q}}$ and $\tilde{\pi} \in \arg\max_{\pi \in \Pi_{\mathcal{Q}}} \int V_\pi(o)\, dF(o)$. Ideally Q-learning would provide a policy that achieves the highest value as compared to other policies in $\Pi_{\mathcal{Q}}$ (as is the case with $\tilde{\pi}$). This is not necessarily the case. As discussed in the above remark batch Q-learning yields estimated Q-functions for which $\ell(\overline{Q}, \widehat{Q})$ converges to zero. The policy $\overline{\pi}$ may not produce a maximal value; that is $\int V_{\tilde{\pi}}(o) - V_{\overline{\pi}}(o)\, dF(o)$ need not be zero (see also the remark following Lemma 2). Recall from Lemma 2 that $2M\ell(\overline{Q}, \tilde{Q}) + 2M\ell(\tilde{Q}, Q_{\tilde{\pi}})$ is an upper bound on this difference. It is not hard to see that $\ell(\tilde{Q}, Q_{\tilde{\pi}})$ is zero if and only if $\tilde{\pi}$ is the optimal policy; indeed the optimal Q-function would belong to the approximation space. The Q-learning algorithm does not directly maximize the value function. As remarked in Tsitsiklis and van Roy (1997) the goal of the Q-learning algorithm is to construct an approximation to the optimal Q-function within the constraints imposed by the app! roximation space; this approximation is a projection when the approximation space is linear. Approximating the Q-function yields an optimal policy if the approximating class is sufficiently rich. Ormoneit and Sen (2002) consider a sequence of approximation spaces (kernel based spaces indexed by a bandwidth) and make assumptions on the optimal value function which guarantee that this sequence of approximations spaces is sufficient rich (as the bandwidth decreases with increasing training set size) so as to approximate the optimal value function to any desired degree.

4. Again define $\widehat{Q}_t$ as a zero of $\widetilde{Err}_{n,\widehat{Q}_{t+1}}(Q_t)$, $t = T, T-1, \ldots, 0$. Since $\ell(\overline{Q}, \widehat{Q})$ converges in probability to zero, one might think that $\int |V_{\overline{\pi}}(o) - V_{\widehat{\pi}}(o)|\, dF(o)$ should be small as well. Referring to Lemma 1, we have that the difference in value functions $\int |V_{\overline{\pi}}(o) - V_{\widehat{\pi}}(o)|\, dF(o)$ can be expressed as the sum over t of the expectation of $Q_{\overline{\pi},t}(\mathbf{O}_t, \mathbf{A}_{t-1}, \widehat{\pi}_t) - Q_{\overline{\pi},t}(\mathbf{O}_t, \mathbf{A}_{t-1}, \overline{\pi}_t)$. However $\ell(\overline{Q}, \widehat{Q})$ small does not imply that $\widehat{\pi}$ and $\overline{\pi}$ will be close nor does it imply that $Q_{\overline{\pi},t}(\mathbf{O}_t, \mathbf{A}_{t-1}, \widehat{\pi}_t) - Q_{\overline{\pi},t}(\mathbf{O}_t, \mathbf{A}_{t-1}, \overline{\pi}_t)$ will be small. To see the former consider an action space with 10 actions, $1, \ldots, 10$ and $\widehat{Q}_t(o_t, a_t) = 1$ for $a = 1, \ldots, 9$, $\widehat{Q}_t(o_t, 10) = 1 + 1/2\varepsilon$ and $\overline{Q}_t(o_t, a_t) = 1 - 1/2\varepsilon$ for $a = 2, \ldots, 10$, $\overline{Q}_t(o_t, 1) = 1$. So $\overline{Q}_t$ and $\widehat{Q}_t$ are uniformly less than ε apart yet the argument of their maxima are 1 and 10.

Proof of Theorem 2. Fix $Q_t = \theta_t^T q_t$, $\theta \in \Theta$ for $t = 0, \ldots, T$. Define an infinite training sample version of $\widetilde{Err}_n$ as
$$\widetilde{Err}_{Q_{t+1}}(Q_t) \quad = \quad E\left[\left(R_t + \max_{a_{t+1}} Q_{t+1}(\mathbf{O}_{t+1}, \mathbf{A}_t, a_{t+1}) - Q_t\right) q_t\right]$$

$$= \quad E\left[\left(\bar{Q}_t + \max_{a_{t+1}} Q_{t+1}(\mathbf{O}_{t+1}, \mathbf{A}_t, a_{t+1}) - \bar{Q}_{t+1}(\mathbf{O}_{t+1}, \mathbf{A}_t, \bar{\pi}_{t+1}) - Q_t\right) q_t\right]$$

where Q_t is an abbreviation for $Q_t(\mathbf{O}_t, \mathbf{A}_t)$. To derive the last equality recall that $\bar{Q}_t(\mathbf{O}_t, \mathbf{A}_t)$ is the projection of
$E\left[R_t + \bar{Q}_{t+1}(\mathbf{O}_{t+1}, \mathbf{A}_t, \bar{\pi}_{t+1}) | \mathbf{O}_t, \mathbf{A}_t\right]$ on the space spanned by q_t. Since $\bar{Q}_t$ is a projection we can write $\bar{Q}_t = \theta_{\bar{\pi},t}^T q_t$ for some $\theta_{\bar{\pi},t} \in \Theta$. Also we can write $Q_t = \theta_t^T q_t$ for some $\theta_t \in \Theta$. The $\widetilde{Err}$'s provide a pointwise upper bound on the differences, $|\bar{Q}_t - Q_t|$, as follows. Rearrange the terms in $\widetilde{Err}_{Q_{t+1}}$ using the fact that $E q_t q_t^T$ is invertible to obtain

$$\begin{aligned}(\theta_{\bar{\pi},t} - \theta_t) \quad = \quad & \left(E q_t q_t^T\right)^{-1} \widetilde{Err}_{Q_{t+1}}(Q_t) \\ & - \left(E q_t q_t^T\right)^{-1} E\left[\left(\max_{a_{t+1}} Q_{t+1}(\mathbf{O}_{t+1}, \mathbf{A}_t, a_{t+1}) - \bar{Q}_{t+1}(\mathbf{O}_{t+1}, \mathbf{A}_t, \bar{\pi}_{t+1})\right) q_t\right].\end{aligned}$$

Denote the Euclidean norm of a p dimensional vector x by $||x||$. Then

$$\begin{aligned}\left|(\theta_{\bar{\pi},t} - \theta_t)^T q_t\right| \quad \leq \quad & (1/\eta)\left|\left|\widetilde{Err}_{Q_{t+1}}(Q_t)\right|\right| \, ||q_t|| + \\ & (1/\eta) E\left[\left|\max_{a_{t+1}} Q_{t+1}(\mathbf{O}_{t+1}, \mathbf{A}_t, a_{t+1}) - \bar{Q}_{t+1}(\mathbf{O}_{t+1}, \mathbf{A}_t, \bar{\pi}_{t+1})\right| ||q_t||\right] \, ||q_t|| \\ \leq \quad & (1/\eta)\left|\left|\widetilde{Err}_{Q_{t+1}}(Q_t)\right|\right| \, ||q_t|| + (1/\eta) L E\left[\left|Q_{t+1} - \bar{Q}_{t+1}\right| \, ||q_t||\right] \, ||q_t|| \\ \leq \quad & (1/\eta)\sqrt{p} M_Q \left|\left|\widetilde{Err}_{Q_{t+1}}(Q_t)\right|\right| + (1/\eta) L p M_Q^2 E\left[\left|Q_{t+1} - \bar{Q}_{t+1}\right|\right]\end{aligned}$$

for $t \leq T$. To summarize

$$E\left|\bar{Q}_t - Q_t\right| \quad \leq \quad (1/\eta)\sqrt{p} M_Q \left|\left|\widetilde{Err}_{Q_{t+1}}(Q_t)\right|\right| + (1/\eta) L p M_Q^2 E\left[\left|Q_{t+1} - \bar{Q}_{t+1}\right|\right]$$

where Q_t, $\bar{Q}_t$ is an abbreviation for $Q_t(\mathbf{O}_t, \mathbf{A}_t)$, respectively $\bar{Q}_t(\mathbf{O}_t, \mathbf{A}_t)$, for each t.

As in the proof of Theorem 1, these inequalities can be solved for each $\mathrm{E}\left|\bar{Q}_t - Q_t\right|$ to yield

$$\begin{aligned}E\left|\bar{Q}_t - Q_t\right| \quad \leq \quad & (\sqrt{p} M_Q/\eta) \sum_{j=t}^{T} (L p M_Q^2/\eta)^{j-t} \left|\left|\widetilde{Err}_{Q_{j+1}}(Q_j)\right|\right| \\ \leq \quad & (\sqrt{p} M_Q/\eta) \sum_{j=t}^{T} (L p M_Q^2/\eta)^{j-t} \left|\left|\widetilde{Err}_{n,Q_{j+1}}(Q_j) - \widetilde{Err}_{Q_{j+1}}(Q_j)\right|\right| \\ & + (\sqrt{p} M_Q/\eta) \sum_{j=t}^{T} (L p M_Q^2/\eta)^{j-t} \left|\left|\widetilde{Err}_{n,Q_{j+1}}(Q_j)\right|\right|.\end{aligned}$$

Simplifying terms we obtain

$$\begin{aligned}\sum_{t=0}^{T} L^{(t+1)} E|\bar{Q}_t - Q_t| \quad \leq \quad & \sqrt{p} M_Q/\eta \sum_{t=0}^{T} L^{(t+1)} \sum_{j=t}^{T} \left(L p M_Q^2/\eta\right)^{j-t} ||\widetilde{Err}_{n,Q_{j+1}}(Q_j)|| \\ & + 4 p^{T+1/2} M_Q^{2T+1} \eta^{-(T+1)} L^{T+1} \max_t \left|\left|\widetilde{Err}_{n,Q_{t+1}}(Q_t) - \widetilde{Err}_{Q_{t+1}}(Q_t)\right|\right|.\end{aligned} \tag{15}$$

Consider each component of each of the $T+1$, p dimensional vectors, $\widetilde{Err}_{n,Q_{i+1}}(Q_i)) - \widetilde{Err}_{Q_{i+1}}(Q_i)$ for an $\varepsilon' > 0$:

$$P\left[\bigcup_{i=0}^{T}\bigcup_{j=1}^{p}\left\{\text{for some}_{\theta_i,\theta_{i+1}\in\Theta, q_i\in\mathcal{Q}_i, q_{i+1}\in\mathcal{Q}_{i+1}}\left|\widetilde{Err}_{n,Q_{i+1}}(Q_i)_j - \widetilde{Err}_{Q_{i+1}}(Q_i))_j\right| > \varepsilon'\right\}\right].$$

This probability is in turn bounded above by

$$\sum_{i=0}^{T}\sum_{j=1}^{p} P\left[\text{for some}_{\theta_i,\theta_{i+1}\in\Theta, q_i\in\mathcal{Q}_i, q_{i+1}\in\mathcal{Q}_{i+1}}\left|\widetilde{Err}_{n,Q_{i+1}}(Q_i)_j - \widetilde{Err}_{Q_{i+1}}(Q_i))_j\right| > \varepsilon'\right].$$

In Lemmas 17.2, 17.3, 17.5, Anthony and Bartlett (1999) provide an upper bound on the probability

$$P\left[\text{ for some } f\in\mathcal{F} \text{ has } |E_n(\ell_f) - E(\ell_f)| \geq \varepsilon'\right]$$

where $\ell_f(x,y) = (y - f(x))^2$. These same lemmas (based on the classical arguments of symmetrization, permutation and reduction to a finite set) can be used for $f \in \mathcal{F}'$ since the functions in $\mathcal{F}'$ are uniformly bounded. Hence for each $j = 1,\ldots,p$ and $t = 0,\ldots,T$

$$P\left[\text{ for some } \theta_t,\theta_{t+1}\in\Theta, q_t\in\mathcal{Q}_t, q_{t+1}\in\mathcal{Q}_{t+1} \text{ has } \left|\widetilde{Err}_{n,Q_{t+1}}(Q_t)_j - \widetilde{Err}_{Q_{t+1}}(Q_t)_j\right| > \varepsilon'\right] \leq 4\mathcal{N}_1\left(\frac{\varepsilon'}{16M'},\mathcal{F}',2n\right)\exp\left\{-\frac{(\varepsilon')^2 n}{32(M')^2}\right\}.$$

Set $\varepsilon = p^{T+1/2}M_{\mathcal{Q}}^{2T+1}\eta^{-(T+1)}L^{T+1}\varepsilon'$. Thus for n satisfying

$$4p(T+1)\mathcal{N}_1\left(\frac{\varepsilon}{16M'p^{T+1/2}M_{\mathcal{Q}}^{2T+1}\eta^{-(T+1)}L^{T+1}},\mathcal{F}',2n\right) \exp\left\{-\frac{\varepsilon^2 n}{32(M')^2\left(p^{T+1/2}M_{\mathcal{Q}}^{2T+1}\eta^{-(T+1)}L^{T+1}\right)^2}\right\} \leq \delta, \qquad (16)$$

the first result of the theorem holds.

To simplify the constraint on n, we derive a covering number for $\mathcal{F}'$ from covering numbers for the $\mathcal{Q}_t$'s. Apply Lemma A2 part 1, to obtain

$$\mathcal{N}_1(\varepsilon,\mathcal{V}_{t+1},n) \leq \mathcal{N}_1\left(\frac{\varepsilon}{|\mathcal{A}|},\mathcal{Q}_{t+1},|\mathcal{A}|n\right)$$

for $\mathcal{V}_{t+1} = \left\{\max_{a_{t+1}} Q_{t+1}(\mathbf{o}_{t+1},\mathbf{a}_{t+1}) : Q_{t+1}\in\mathcal{Q}_{t+1}\right\}$. Next apply Lemma A2, parts 2 and 3, to obtain

$$\mathcal{N}_1(\varepsilon,\mathcal{F}',n) \leq \sum_{t=0}^{T-1}\mathcal{N}_1\left(\frac{\varepsilon}{2|\mathcal{A}|M'},\mathcal{Q}_{t+1},|\mathcal{A}|n\right)\mathcal{N}_1\left(\frac{\varepsilon}{2M'},\mathcal{Q}_t,n\right) + N_1\left(\frac{\varepsilon}{M'},\mathcal{Q}_T,n\right).$$

Theorems 11.6 and 18.4 of Anthony and Bartlett imply that $\mathcal{N}_1(\varepsilon, \mathcal{Q}_t, n) \leq e(p+1)\left(\frac{2e}{\varepsilon}\right)^p$ for each t. Combining this upper bound with (16) and simplifying the algebra yields (14).

Next Lemma 2 implies:

$$\begin{aligned}\int |V_{\tilde{\pi}}(o) - V_{\pi}(o)|\, dF(o) \quad &\leq \quad M\sum_{t=0}^{T} 2L^{(t+1)} E\left|Q_t - \bar{Q}_t\right| \\ &\quad +M\sum_{t=0}^{T} 2L^{(t+1)} E\left|\bar{Q}_t - \tilde{Q}_t\right| +M\sum_{t=0}^{T} 2L^{(t+1)} E\left|\tilde{Q}_t - Q_{\tilde{\pi},t}\right|.\end{aligned}$$

This combined with the first result of the theorem implies the second result. ■

6. Discussion

Planning problems involving a single training set of trajectories are not unusual and can be expected to increase due to the widespread use of policies in the social and behavioral/medical sciences (see, for example, Rush et al., 2003; Altfeld and Walker, 2001; Brooner, and Kidorf, 2002); at this time these policies are formulated using expert opinion, clinical experience and/or theoretical models. However there is growing interest in formulating these policies using empirical studies (training sets). These training sets are collected under fixed exploration policies and thus while they allow exploration they do not allow exploitation, that is, online choice of the actions. If subjects are recruited into the study at a much slower rate than the calendar duration of the horizon, then it is possible to permit some exploitation; some of this occurs in the field of cancer research (Thall, Sung and Estey, 2002).

This paper considers the use of Q-learning with dynamic programming and function approximation for this planning purpose. However the mismatch between Q-learning and the goal of learning a policy that maximizes the value function has serious consequences and emphasizes the need to use all available science in choosing the approximation space. Often the available behaviorial or psychosocial theories provide qualitative information concerning the importance of different observations. In addition these theories are often represented graphically via directed acyclic graphs. However information at the level of the form of the conditional distributions connecting the nodes in the graphs is mostly unavailable. Also due to the complexity of the problems there are often *unknown* missing common causes of different nodes in the graphs. See http://neuromancer.eecs.umich.edu/dtr for more information and references. Methods that can use this qualitative information to minimize t! he mismatch are needed.

Acknowledgments

We would like to acknowledge the help of the reviewers and of Min Qian in improving this paper. Support for this project was provided by the National Institutes of Health (NIDA grants K02 DA15674 and P50 DA10075 to the Methodology Center).

Appendix A.

Recall that the distributions, P and P_π differ only with regards to the policy (see (1) and (2)). Thus the following result is unsurprising. Let $f(\mathbf{O}_{T+1}, \mathbf{A}_T)$ be a (measurable) nonnegative function; then

$E_\pi f$ can be expressed in terms of an expectation with respect to the distribution P if we assume that $p_t(a_t|\mathbf{o}_t,\mathbf{a}_{t-1}) > 0$ for each $(\mathbf{o}_t,\mathbf{a}_t)$ pair and each t. The presence of the p_js in denominator below represent the price we pay because we only have access to training trajectories with distribution P; we do not have access to trajectories from distribution P_π.

Lemma A1 Assume that $P_\pi[p_0(A_0|S_0) > 0] = 1$ and $P_\pi[p_t(A_t|\mathbf{O}_t,\mathbf{A}_{t-1}) > 0] = 1$ for $t = 1,\ldots,T$. For any (measurable) nonnegative function of $g(\mathbf{O}_t,\mathbf{A}_t)$, the P-probability that

$$E_\pi\left[g(\mathbf{O}_t,\mathbf{A}_t)|S_0\right] = E\left[\left(\prod_{\ell=0}^{t}\frac{1_{A_\ell=\pi_\ell}}{p_\ell(A_\ell|\mathbf{O}_\ell,\mathbf{A}_{\ell-1})}\right)g(\mathbf{O}_t,\mathbf{A}_t)\bigg|S_0\right]$$

is one for $t = 0,\ldots,T$.

Proof: We need only prove that

$$E\left[h(S_0)E_\pi\left[g(\mathbf{O}_t,\mathbf{A}_t)|S_0\right]\right] = E\left[h(S_0)E\left[\left(\prod_{\ell=0}^{t}\frac{1_{A_\ell=\pi_\ell}}{p_\ell(A_\ell|\mathbf{O}_\ell,\mathbf{A}_{\ell-1})}\right)g(\mathbf{O}_t,\mathbf{A}_t)\bigg|S_0\right]\right]$$

for any (measurable) nonnegative function, h. Consider the two likelihoods ((1) and (2)) for a trajectory up to time t. Denote the dominating measure for the two likelihoods for the trajectory up to time t as λ_t. By assumption,

$$\begin{aligned}\int h(s_0)g(\mathbf{o}_t,\mathbf{a}_t)&\left(\prod_{\ell=0}^{T}\frac{1_{A_\ell=\pi_\ell}}{p_\ell(a_\ell|\mathbf{o}_\ell,\mathbf{a}_{\ell-1})}\right)f_0(s_0)p_0(a_0|s_0)\\ &\qquad\qquad\prod_{j=1}^{t}f_j(s_j|\mathbf{o}_{j-1},\mathbf{a}_{j-1})p_j(a_j|\mathbf{o}_j,\mathbf{a}_{j-1})\,d\lambda_t(\mathbf{o}_t,\mathbf{a}_t)\\ = \int h(s_0)g(\mathbf{o}_t,\mathbf{a}_t)f_0(s_0)&1_{a_0=\pi_0(s_0)}\prod_{j=1}^{t}f_j(s_j|\mathbf{o}_{j-1},\mathbf{a}_{j-1})1_{a_j=\pi_j(\mathbf{o}_j,\mathbf{a}_{j-1})}\,d\lambda_t(\mathbf{o}_t,\mathbf{a}_t).\end{aligned}$$

By definition the left hand side is $E\left[h(S_0)g(\mathbf{O}_t,\mathbf{A}_t)\left(\prod_{\ell=0}^{j}\frac{1_{A_\ell=\pi_\ell}}{p_\ell(A_\ell|\mathbf{O}_\ell,\mathbf{A}_{\ell-1})}\right)\right]$ and the right hand side is $E_\pi[h(S_0)g(\mathbf{O}_t,\mathbf{A}_t)]$. Expressing both sides as the expectation of a conditional expectation, we obtain,

$$E_\pi[h(S_0)E_\pi[g(\mathbf{O}_t,\mathbf{A}_t)|S_0]] = E\left[h(S_0)E\left[g(\mathbf{O}_t,\mathbf{A}_t)\left(\prod_{\ell=0}^{t}\frac{1_{A_\ell=\pi_\ell}}{p_\ell(A_\ell|\mathbf{O}_\ell,\mathbf{A}_{\ell-1})}\right)\bigg|S_0\right]\right].$$

Note that the distribution of S_0 is the same regardless of how the actions are chosen, that is the distribution of S_0 is the same under both P and P_π. Thus

$$E[h(S_0)E_\pi[g(\mathbf{O}_t,\mathbf{A}_t)|S_0]] = E\left[h(S_0)E\left[g(\mathbf{O}_t,\mathbf{A}_t)\left(\prod_{\ell=0}^{t}\frac{1_{A_\ell=\pi_\ell}}{p_\ell(A_\ell|\mathbf{A}_\ell,\mathbf{A}_{\ell-1})}\right)\bigg|S_0\right]\right].$$

■

Lemma A2 For p, q, r, s, N positive integers and $M_\mathcal{F}$, $M_\mathcal{G}$, M_Θ positive reals, define the following classes of real valued functions,

$$\begin{aligned}\mathcal{H} &\subseteq \{h(x,a) : x \in \mathbb{R}^p,\ a \in \{1,\ldots,N\}\}\\ \mathcal{F} &\subseteq \left\{f(x) : x \in \mathbb{R}^q,\ \sup_x|f(x)| \le M_\mathcal{F}\right\}\\ \mathcal{G} &\subseteq \left\{g(x,y) : x \in \mathbb{R}^q,\ y \in \mathbb{R}^r,\ \sup_{x,y}|g(x,y)| \le M_\mathcal{G}\right\}\end{aligned}$$

and

$$\Theta \subseteq \left\{ \theta \in \mathbb{R}^s : \max_{i=1,\dots,s} |\theta_i| \le M_\Theta \right\}.$$

The following hold.

1. If $\mathcal{V} = \left\{ \max_a h(x,a) : h \in \mathcal{H} \right\}$ then $\mathcal{N}_1(\varepsilon, \mathcal{V}, n) \le \mathcal{N}_1(\varepsilon/N, \mathcal{H}, Nn)$.

2. For $|a| \le 1$, $|b| \le 1$, if $\mathcal{V} = \{af(x) + bg(x,y) : f \in \mathcal{F},\ g \in \mathcal{G}\}$ then $\mathcal{N}_1(\varepsilon, V, n) \le \mathcal{N}_1(\varepsilon/2, \mathcal{F}, n)\mathcal{N}_1(\varepsilon/2, \mathcal{G}, n)$.

3. If $\mathcal{V} = \mathcal{F} \cup \mathcal{G}$ then $\mathcal{N}_1(\varepsilon, \mathcal{V}, n) \le \mathcal{N}_1(\varepsilon, \mathcal{F}, n) + \mathcal{N}_1(\varepsilon, \mathcal{G}, n)$.

4. If $\mathcal{V} = \{\theta_1 f_1(x) + \ldots + \theta_s f_s(x) : f_i \in \mathcal{F},\ (\theta_1, \dots, \theta_s) \in \Theta\}$ then $\mathcal{N}_1(\varepsilon, \mathcal{V}, n) \le e(s+1)\left(\frac{4esM_\Theta M_\mathcal{F}}{\varepsilon}\right)^s \mathcal{N}_\infty\left(\frac{\varepsilon}{4sM_\Theta}, \mathcal{F}, n\right)^s$.

Proof. We prove 1. and 4.; the proofs of 2. and 3. are straightforward and are omitted. Consider 1. Given $(x_1, \dots, x_n)$, the ε-covering number for the class of points in $\mathbb{R}^{Nn}$, $\left\{ (h(x_i,a) : i = 1, \dots, n, a = 1, \dots N)\, ; h \in \mathcal{H} \right\}$ is bounded above by $\mathcal{N}_1(\varepsilon, \mathcal{H}, Nn)$. Note that for $(z_{ia}, i = 1, \dots, n,\ a = 1, \dots, N)$,

$$\begin{aligned} 1/n \sum_{i=1}^n \left| \max_{a=1,\dots,N} h(x_i,a) - \max_{a=1,\dots,N} z_{ia} \right| &\le 1/n \sum_{i=1}^n \max_{a=1,\dots,N} |h(x_i,a) - z_{ia}| \\ &\le 1/n \sum_{i=1}^n \sum_{a=1}^N |h(x_i,a) - z_{ia}|. \end{aligned}$$

Thus the ε-covering number for the class of points in $\mathbb{R}^n$, $\{ \left(\max_{a=1}^N h(x_i,a) : i = 1, \dots, n\right); h \in \mathcal{H}\}$ is bounded above by $\mathcal{N}_1(\varepsilon, \mathcal{H}, Nn)$. Using the definition of covering numbers for classes of functions we obtain $\mathcal{N}_1(\varepsilon, \mathcal{V}, n) \le \mathcal{N}_1\left(\frac{\varepsilon}{N}, \mathcal{H}, Nn\right)$.

Next consider 4. Put $x = (x_1, \dots, x_n)$ (each $x_i \in \mathbb{R}^q$) and $f(x_i) = (f_1(x_i), \dots, f_s(x_i))^T$. Then there exists $\{z_1, \dots, z_\mathcal{N}\}$, ($\mathcal{N} = \mathcal{N}_\infty(\varepsilon/(4sM_\Theta), \mathcal{F}, n)$; $z_j \in \mathbb{R}^n$) that form the centers of an $\varepsilon/(4sM_\Theta)$-cover for $\mathcal{F}$. To each z_j we can associate an $f \in \mathcal{F}$, say f_j^* so that $\{f_1^*, \dots, f_\mathcal{N}^*\}$ form the centers of an $\varepsilon/(2sM_\Theta)$-cover for $\mathcal{F}$. Then given $\{f_1, \dots, f_s\} \in \mathcal{F}$ there exists $j^* \in \{1, \dots, \mathcal{N}\}$ for $j = 1, \dots, s$, so that $\max_{1 \le j \le s} \max_{1 \le i \le n} |f_j(x_i) - f_{j^*}^*(x_i)| \le \varepsilon/(2sM_\Theta)$. Then

$$(1/n)\sum_{i=1}^n \left| \sum_{j=1}^s \theta_j f_j(x_i) - \theta_j f_{j^*}^*(x_i) \right| \le \varepsilon/2.$$

Define $\mathcal{F}' = \left\{ \sum_{j=1}^s \theta_j f_{j^*}^* : \theta_j \in \Theta \right\}$. Theorems 11.6 and 18.4 of Anthony and Bartlett (1996) imply that $\mathcal{N}_1(\varepsilon/2, \mathcal{F}', n) \le e(s+1)\left(\frac{4esM_\Theta M_\mathcal{F}}{\varepsilon}\right)^s$ These two combine to yield the result. ■

References

M. Altfeld and B. D. Walker. Less is more? STI in acute and chronic HIV-1 infection. *Nature Medicine* 7:881–884, 2001.

M. Anthony and P. L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge, UK: Cambridge University Press, 1999.

L. Baird. Advantage updating. Technical Report. WL-TR-93-1146, Wright-Patterson Air Force Base, 1993.

J. Baxter and P. L. Bartlett. Infinite-horizon policy-gradient estimation. *J. Artificial Intelligence Research* 15:319–350, 2001.

R. E. Bellman *Dynamic Programming*. Princeton: Princeton University Press, 1957.

D. P. Bertsekas and J. N. Tsitsiklis. Neuro-Dynamic Programming. Belmont, MA.: Athena Scientific. 1996.

R. K. Brooner and M. Kidorf. Using behavioral reinforcement to improve methadone treatment participation. *Science and Practice Perspectives* 1:38–48, 2002.

M. Fava, A. J. Rush, M. H. Trivedi, A. A. Nierenberg, M. E. Thase, H. A. Sackeim, F. M. Quitkin, S. Wisniewski, P. W. Lavori, J. F. Rosenbaum, D. J. Kupfer. Background and rationale for the sequenced treatment alternative to relieve depression (STAR*D) study. *Psychiatric Clinics of North America* 26(3):457–494, 2003.

C. N. Fiechter Efficient Reinforcement Learning. In *Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory (COLT 1994)*, pages 88–97, New Brunswick, NJ, 1994.

C. N. Fiechter. Expected Mistake Bound Model for On-Line Reinforcement Learning. In *Proceedings of the Fourteenth International Conference on Machine Learning*, Douglas H. Fisher (Ed.), Nashville, Tennessee, pages 116–124, 1997.

S. M. Kakade. On the Sample Complexity of Reinforcement Learning. Ph.D. thesis, University College, London, 2003.

M. Kearns, Y. Mansour, and A. Y. Ng. A Sparse sampling algorithm for near-optimal planning in large Markov decision processes. *Machine Learning*, 49(2-3): 193–208, 1999.

M. Kearns, Y. Mansour and A. Y. Ng. Approximate planning in large POMDPs via reusable trajectories. In *Advances in Neural Information Processing Systems*, 12, MIT Press, 2000.

D. Ormoneit and S. Sen: Kernel-Based Reinforcement Learning. *Machine Learning* 49(2-3):161–178 2002.

L. Peshkin and C. R. Shelton. Learning from scarce experience. In *Proceedings of the Nineteenth International Conference on Machine Learning (ICML 2002)* Claude Sammut, Achim G. Hoffmann (Eds.) pages 498–505, Sydney, Australia, 2002.

A. J. Rush, M. L. Crismon, T. M. Kashner, M. G. Toprac, T. J. Carmody, M. H. Trivedi, T. Suppes, A. L. Miller, M. M. Biggs, K. Shores-Wilson, B. P. Witte, S. P. Shon, W. V. Rago, K. Z. Altshuler, TMAP Research Group. Texas medication algorithm project, phase 3 (TMAP-3): Rationale and study design. *Journal of Clinical Psychiatry*, 64(4):357–69, 2003.

L. S. Schneider, P. N. Tariot, C. G. Lyketsos, K. S. Dagerman, K. L. Davis, S. Davis, J. K. Hsiao, D. V. Jeste, I. R. Katz, J. T. Olin, B. G. Pollock, P. V. Rabins, R. A. Rosenheck, G. W. Small, B. Lebowitz, J. A. Lieberman. National Institute of Mental Health clinical antipsychotic trials of intervention effectiveness (CATIE). *American Journal of Geriatric Psychiatry*, 9(4):346–360, 2001.

R. E. Schapire, P. Bartlett, Y. Freund and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, 1998.

D. I. Simester, P. Sun, and J. N. Tsitsiklis. Dynamic catalog mailing policies. *unpublished manuscript*, Available electronically at http://web.mit.edu/jnt/www/Papers/P-03-sun-catalog-rev2.pdf, 2003.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, Mass, 1998.

P. F. Thall, R. E. Millikan and H. G. Sung. Evaluating multiple treatment courses in clinical trials. *Statistics and Medicine* 19:1011-1028, 2000.

P. F. Thall, H. G. Sung and E. H. Estey. Selecting therapeutic strategies based on efficacy and death in multicourse clinical trials. *Journal of the American Statistical Association*, 97:29-39, 2002.

J. N. Tsitsiklis and B. Van Roy. Feature-based methods for large scale dynamic programming, *Machine Learning*, 22:59-94, 1996.

J. N. Tsitsiklis and B. Van Roy. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5):674-690, 1997.

A. W. van der Vaart and J. A. Wellner. *Weak Convergence and Empirical Processes*. Springer, New York, 1996.

C. J. C. H. Watkins. Learning from Delayed Rewards. Ph.D. thesis, Cambridge University, 1989.