Bayesian Learning of Sparse Classifiers

Mário A. T. Figueiredo
Instituto de Telecomunicações,
Instituto Superior Técnico
1049-001 Lisboa, Portugal
mtf@lx.it.pt

Anil K. Jain
Dept. of Computer Science and Eng.
Michigan State University,
East Lansing, MI 48824, U.S.A.
jain@cse.msu.edu

Abstract

Bayesian approaches to supervised learning use priors on the classifier parameters. However, few priors aim at achieving "sparse" classifiers, where irrelevant/redundant parameters are automatically set to zero. Two well-known ways of obtaining sparse classifiers are: use a zero-mean Laplacian prior on the parameters, and the "support vector machine" (SVM). Whether one uses a Laplacian prior or an SVM, one still needs to specify/estimate the parameters that control the degree of sparseness of the resulting classifiers. We propose a Bayesian approach to learning sparse classifiers which does not involve any parameters controlling the degree of sparseness. This is achieved by a hierarchical-Bayes interpretation of the Laplacian prior, followed by the adoption of a Jeffreys' non-informative hyper-prior. Implementation is carried out by an EM algorithm. Experimental evaluation of the proposed method shows that it performs competitively with (often better than) the best classification techniques available.

1. Introduction

1.1. Supervised Learning

Supervised learning aims at inferring a functional relation $y = f(\mathbf{x})$, from a set of training examples $\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), ..., (\mathbf{x}^{(n)}, y^{(n)})\}$. Usually, each $\mathbf{x}^{(i)} = [x_1^{(i)}, ..., x_d^{(i)}]^T \in I\!R^d$ is called an input, or feature vector. This paper deals with *classification* problems, in which each $y^{(i)}$ is of categorical nature (*e.g.*, $y^{(i)} \in \{0, 1\}$, in the two-class case) and is called the class label of $\mathbf{x}^{(i)}$. If the $y^{(i)}$ were continuous, we would be in the context of *regression*. Usually, the structure of the functional relation is assumed fixed and the objective is to estimate a vector of parameters $\boldsymbol{\beta}$ defining it; accordingly we write $y = f(\mathbf{x}, \boldsymbol{\beta})$. For example, in a two-class linear discriminant, $y = h(\beta_0 + \sum_i \beta_i x_i)$, where $h(\cdot)$ denotes the Heavyside function ($h(v) = 1$, if $v \geq 0$, $h(v) = 0$, if $v < 0$).

It is often desirable, not simply to classify $\mathbf{x}$ into one of the classes, but to know the degree of confidence of that classification. In that case we are interested in learning a function $g(\mathbf{x}, \boldsymbol{\beta})$ taking values in $[0, 1]$ (rather than just $\{0, 1\}$) which can be interpreted as the probability that $\mathbf{x}$ belongs to, say, class 1. In *logistic (linear) regression* [18], $P(y = 1|\mathbf{x}) = g(\mathbf{x}, \boldsymbol{\beta}) = \psi(\beta_0 + \sum_i \beta_i x_i)$, where

$$\psi(z) = (1 + \exp(-z))^{-1} \tag{1}$$

is called the *logistic* function (see Fig. 1). The function $\psi(\cdot)$ yielding the class probabilities is known as the *link*. An advantage of a function giving class probabilities, over a hard classifier, is that it can be used to obtain optimal classifiers under different cost functions. For example, if the cost function is simply the misclassification error, then a classifier is obtained by thresholding $g(\mathbf{x}, \boldsymbol{\beta})$ at $1/2$.

1.2. Discriminative vs. generative learning

Supervised learning of classifiers can be formulated either using a *generative* (*informative*) or a *discriminative* approach [9]. In the *generative* approach, each of the class-conditional probability functions is learned separately from the training data; then a Bayes classifier is obtained by inserting (*plugging in*) these class-conditional probability functions and the *a priori* class probabilities into the Bayes decision rule [13]. In *discriminative* learning, the class-conditional densities are not explicit modelled; the classifier is directly learned from the data. Well known discriminative techniques include linear and logistic discrimination, k-nearest neighbor classifiers, tree classifiers [5], feedforward neural networks [3, 19, 21], support vector machines (SVM) and other kernel-based methods [8, 25, 27, 28]. This paper focuses on *discriminative* learning.

M. Figueiredo's research was supported by the Foundation for Science and Technology, Portugal, project POSI/33143/SRI/2000.

A. K. Jain's research was supported by the Office of Naval Research, U.S.A., grant number N00014-01-1-0266.

1.3. Over-fitting and under-fitting

A main concern in supervised learning is to avoid *over-fitting* the training data. In other words, to achieve good *generalization* (*i.e.*, to perform well on yet unseen data) it is necessary to control the *complexity* of the learned function. If it is too complex, it may follow irrelevant properties of the particular data set on which it is trained (*over-fitting*), thus performing poorly on future data. An overly simple function, on the other hand, may not be able to capture the main behavior of the underlying relationship (*under-fitting*). This well-known trade-off has been addressed with a variety of formal tools (see, *e.g.*, [3, 7, 8, 19, 21, 25]).

1.4. Bayesian discriminative learning

The Bayesian approach to controlling the complexity in discriminative supervised learning is to place a prior on the function to be learned (*i.e.*, on $\boldsymbol{\beta}$) favoring *simplicity*, or *smoothness*, in some sense. Let this prior be denoted as $p(\boldsymbol{\beta}|\boldsymbol{\alpha})$, where $\boldsymbol{\alpha}$ is a vector of *hyper-parameters*. The most common choice, namely for analytical and computational tractability, is a zero-mean Gaussian; in the neural network literature this is known as *weight decay* [3, 19, 21]. Gaussian priors are also used in non-parametric contexts, like the *Gaussian processes* (GP) approach [8, 19, 27, 28], which has roots in earlier work on spline models [14, 26] and regularized radial basis function (RBF) approximations [20].

The main disadvantage of Gaussian priors is that they do not explicitly control the structural complexity of the classifiers. That is, if one of the components of $\boldsymbol{\beta}$ (say, the weight of a given feature in a linear classifier) happens to be close to zero, under a Gaussian prior it will not be set exactly to zero (thus eliminating, or pruning, that parameter) but to some small value. Any structural simplification of the functional form will have to be based on additional tests. In the case of a linear discriminant, setting some parameters to zero, corresponds to ignoring some of the input features, *i.e.*, to performing feature selection.

1.5. Learning sparse classifiers

Let us define a *sparse estimate* of $\boldsymbol{\beta}$ as one in which irrelevant or redundant components are exactly zero. Sparseness is a desirable feature in classifier learning for several reasons, namely:

- Sparseness leads to a structural simplification of the estimated function.
- In kernel classifiers, the generalization performance increases with the degree of sparseness of $\boldsymbol{\beta}$ [8, 25]; this is a key idea behind SVM. Moreover, in a sparse kernel classifier, only a subset of the training data has to be kept (unlike in a standard kernel classifier [21]).

One of the possible ways to achieve sparse estimates consists in adopting a zero-mean Laplacian (rather than Gaussian) prior on $\boldsymbol{\beta}$, with parameter α,

$$p(\boldsymbol{\beta}|\alpha) \propto \exp\left\{-\alpha \sum_i |\beta_i|\right\} = \exp\left\{-\alpha \|\boldsymbol{\beta}\|_1\right\},$$

where $\|\cdot\|_1$ denotes the l_1 norm. The sparseness-inducing nature of the Laplacian prior (or equivalently, of the l_1 penalty) is well known and has been exploited in several research areas [6, 16, 23, 30].

When using a Laplacian prior on $\boldsymbol{\beta}$, the question remains of how to adjust or estimate the parameter α which ultimately controls the degree of sparseness of the obtained estimates. Concerning the SVM, in addition to its disadvantage of outputting "hard" classifications rather than class probabilities, it also involves parameters which control the degree of sparseness of the obtained classifier. Estimating/adjusting these parameters commonly involves cross-validation methods which do not optimally utilize the available learning data, and are time consuming.

1.6. Proposed approach

We propose a Bayesian approach to learning sparse classifiers whose main advantage is that it does not involve any parameter controlling the degree of sparseness. This is achieved using the following building blocks:

1. A *probit* model, in which the *link* function is the Gaussian cumulative distribution function (cdf) [18].
2. A hierarchical-Bayes interpretation of the Laplacian prior as a *normal/independent* distribution (as has been used in robust regression [15]). More specifically, a Laplacian prior can be decomposed into a continuous mixture of zero mean Gaussian priors with an exponential hyper-prior for the variance.
3. Replacement of the exponential hyper-prior by the Jeffreys' prior which expresses scale-invariance and, more importantly, is parameter-free [2].
4. An *expectation-maximization* (EM) algorithm which yields a *maximum a posteriori* estimate of $\boldsymbol{\beta}$.

Experimental evaluation of the proposed method shows that it performs competitively with (often better than) the best classification techniques available.

Our method is related to the *automatic relevance determination* (ARD) idea [19, 17], which underlies the recently proposed *relevance vector machine* (RVM) [4, 24]. The RVM exhibits state-of-the-art performance, beating SVM both in terms of accuracy and sparseness [4, 24]. However, rather than using a *type-II maximum likelihood* approximation [2] (as in ARD and RVM), our modelling assumptions

lead to a marginal *a posteriori* probability function on $\boldsymbol{\beta}$ whose mode can be located by a very simple EM algorithm.

2. Probit Regression

In the *generalized linear regression* [18] approach to two-class problems, the most commonly used link is the logistic function (Eq. (1)). In this paper, we adopt the probit link, defined as the standard Gaussian cdf,

$$\psi(z) = \Phi(z|0,1) \equiv \int_{-\infty}^{z} \mathcal{N}(x|0,1)\, dx, \qquad (2)$$

where $\mathcal{N}(\mathbf{v}|\mathbf{m},\mathbf{C})$ denotes a Gaussian density with mean $\mathbf{m}$ and (co)variance $\mathbf{C}$. The re-scaled probit $\Phi(3\,z\,\pi^{-2}|0,1)$ is plotted in Fig. 1, together with the logistic function, showing that (apart from a scale factor) they are almost indistinguishable [11]. Of course, both the logistic and probit functions can be re-scaled (horizontally), but this scale is implicitly absorbed by $\boldsymbol{\beta}$.

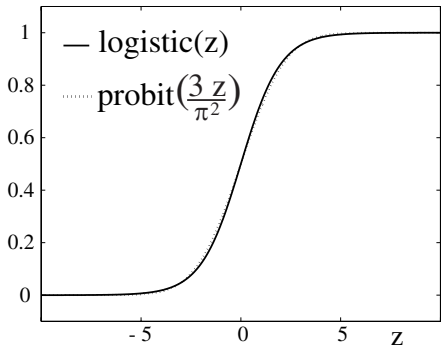

Figure 1. The logistic and (re-scaled) probit link functions.

To extend the probit (or the logistic) model to include non-linear transformations of the input features, the link can be applied to a non-linear function of $\mathbf{x}$: $P(y=1|\mathbf{x}) = \psi(u(\mathbf{x},\boldsymbol{\beta}))$. Here, we will only consider classifiers where this non-linear function is of the form $u(\mathbf{x},\boldsymbol{\beta}) = \boldsymbol{\beta}^T\mathbf{h}(\mathbf{x})$, *i.e.*, linear with respect to $\boldsymbol{\beta}$. This includes:

- Linear classifiers; $\mathbf{h}(\mathbf{x}) = [1, x_1, ..., x_d]$, in which case $\boldsymbol{\beta}$ is a $(d+1)$-dimensional vector.
- Non-linear classifiers; $\mathbf{h}(\mathbf{x}) = [1, \phi_1(\mathbf{x}), ..., \phi_k(\mathbf{x})]^T$, where the $\phi_i(\cdot)$ are nonlinear functions. Here, the dimensionality of $\boldsymbol{\beta}$ is $k+1$.
- Kernel classifiers; $\mathbf{h}(\mathbf{x}) = [1, K(\mathbf{x},\mathbf{x}^{(1)}), ..., K(\mathbf{x},\mathbf{x}^{(n)})]^T$, where $K(\cdot,\cdot)$ is some (symmetric) kernel function [8]. Here, $\boldsymbol{\beta}$ is $(n+1)$-dimensional. This is used in SVM and RVM approaches.

The important characteristic of the probit link that we exploit is that it has a simple interpretation in terms of hidden (or latent) variables [1]. Let $z(\mathbf{x},\boldsymbol{\beta}) = u(\mathbf{x},\boldsymbol{\beta}) + w$, where w is a zero-mean unit-variance Gaussian variable. If the classifier is defined as $y=1$, if $z(\mathbf{x},\boldsymbol{\beta}) \geq 0$, and $y=0$, if $z(\mathbf{x},\boldsymbol{\beta}) < 0$, then we recover the probit model, because

$$P(y=1|\mathbf{x}) = P(u(\mathbf{x},\boldsymbol{\beta}) + w \geq 0) = \Phi(u(\mathbf{x},\boldsymbol{\beta})|0,1).$$

Given training data $\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), ..., (\mathbf{x}^{(n)}, y^{(n)})\}$, consider the corresponding vector of hidden/missing variables $\mathbf{z} = [z^{(1)}, ..., z^{(n)}]^T$. If we had $\mathbf{z}$, we would simply have a linear regression problem,

$$\mathbf{z} = \mathbf{H}\boldsymbol{\beta} + \mathbf{w}, \qquad (3)$$

where $\mathbf{H}$ is the *design matrix*

$$\mathbf{H} = \left[\mathbf{h}^T(\mathbf{x}^{(1)}), ..., \mathbf{h}^T(\mathbf{x}^{(n)})\right]^T, \qquad (4)$$

and $\mathbf{w}$ a vector of i.i.d. zero-mean unit-variance Gaussian samples. This suggests using the EM algorithm [10] to find a *maximum a posteriori* (MAP) estimate of $\boldsymbol{\beta}$.

The EM algorithm produces a sequence of estimates $\widehat{\boldsymbol{\beta}}^{(t)}$, for $t = 0, 1, 2, ...$, by alternating between two steps:

E-step: Compute the expected value of the complete log-posterior, given the current estimate $\widehat{\boldsymbol{\beta}}^{(t)}$ and the observations, usually denoted as the Q-function,

$$Q(\boldsymbol{\beta}|\widehat{\boldsymbol{\beta}}^{(t)}) = \int p(\mathbf{z}|\mathbf{y},\widehat{\boldsymbol{\beta}}^{(t)}) \log p(\boldsymbol{\beta}|\mathbf{z},\mathbf{y})\, d\mathbf{z}.$$

M-step: Update the estimate according to

$$\widehat{\boldsymbol{\beta}}^{(t+1)} = \arg\max_{\boldsymbol{\beta}} Q(\boldsymbol{\beta}|\widehat{\boldsymbol{\beta}}^{(t)}).$$

Let $\boldsymbol{\beta}$ be assigned a zero-mean Gaussian prior with covariance $\mathbf{A}$, $p(\boldsymbol{\beta}) = \mathcal{N}(\boldsymbol{\beta}|0,\mathbf{A})$. Its complete log-posterior $p(\boldsymbol{\beta}|\mathbf{y},\mathbf{z})$, is given by

$$\begin{aligned} \log p(\boldsymbol{\beta}|\mathbf{y},\mathbf{z}) &\propto \log p(\mathbf{z}|\boldsymbol{\beta}) + \log p(\boldsymbol{\beta}) \\ &\propto -\|\mathbf{H}\boldsymbol{\beta} - \mathbf{z}\|^2 - \boldsymbol{\beta}^T\mathbf{A}^{-1}\boldsymbol{\beta} \\ &\propto -\boldsymbol{\beta}^T\mathbf{H}^T(\mathbf{H}\boldsymbol{\beta} - 2\,\mathbf{z}) - \boldsymbol{\beta}^T\mathbf{A}^{-1}\boldsymbol{\beta}, \end{aligned} \qquad (5)$$

which is linear with respect to the missing $\mathbf{z}$. Accordingly, in the E-step of EM, all that is needed is $E[\mathbf{z}|\widehat{\boldsymbol{\beta}}^{(t)},\mathbf{y}]$. This expectation can be expressed in closed form as

$$v_i \equiv E[z_i|\widehat{\boldsymbol{\beta}}^{(t)},\mathbf{y}] = \begin{cases} \boldsymbol{\beta}^T\mathbf{h}(\mathbf{x}^{(i)}) + \dfrac{\mathcal{N}(\boldsymbol{\beta}^T\mathbf{h}(\mathbf{x}^{(i)})|0,1)}{1-\Phi(-\boldsymbol{\beta}^T\mathbf{h}(\mathbf{x}^{(i)})|0,1)} & \text{if } y^{(i)}=1 \\ \boldsymbol{\beta}^T\mathbf{h}(\mathbf{x}^{(i)}) - \dfrac{\mathcal{N}(\boldsymbol{\beta}^T\mathbf{h}(\mathbf{x}^{(i)})|0,1)}{\Phi(-\boldsymbol{\beta}^T\mathbf{h}(\mathbf{x}^{(i)})|0,1)} & \text{if } y^{(i)}=0, \end{cases} \qquad (6)$$

since z_i is Gaussian distributed with mean $\boldsymbol{\beta}^T \mathbf{h}(\mathbf{x}^{(i)})$, but left-truncated at zero if $y^{(i)} = 1$, and right-truncated at zero if $y^{(i)} = 0$ (see [1]). Denoting $\mathbf{v} \equiv [v_1, ..., v_n]^T$, the M-step that results from maximizing, with respect to $\boldsymbol{\beta}$, the complete log-posterior in (5) with $\mathbf{v}$ replacing the missing $\mathbf{z}$, is simply a ridge-regression-type equation

$$\widehat{\boldsymbol{\beta}}^{(t+1)} = (\mathbf{A}^{-1} + \mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{v}, \tag{7}$$

with $\mathbf{v}$ playing the role of observed data.

3. Laplacian Prior

To favor sparseness, we adopt an independent Laplacian prior for $\boldsymbol{\beta}$,

$$p(\boldsymbol{\beta}|\alpha) = \prod_{i=1}^{m} \frac{\alpha}{2} \exp\{-\alpha|\beta_i|\} = \left(\frac{\alpha}{2}\right)^m \exp\{-\alpha\|\boldsymbol{\beta}\|_1\} \tag{8}$$

where m is the dimensionality of $\boldsymbol{\beta}$. The main feature of this criterion is that due to the presence of the l_1 penalty, some of the components of $\widehat{\boldsymbol{\beta}}$ may be exactly zero [23]; in other words, the Laplacian prior promotes sparseness. In general, the presence of the (non-differentiable) l_1 norm requires special purpose computational methods.

3.1. A hierachical-Bayes interpretation

Let us consider an alternative model, where each β_i is given a zero-mean Gaussian prior with its own variance τ_i,

$$p(\beta_i|\tau_i) = \mathcal{N}(\beta_i|0, \tau_i). \tag{9}$$

As a hyper-prior for the variances τ_i, we adopt an exponential distribution

$$p(\tau_i|\gamma) = \frac{\gamma}{2} \exp\{-\frac{\gamma\,\tau_i}{2}\}, \quad \text{for } \tau_i \geq 0. \tag{10}$$

Integrating with respect to τ_i, we obtain

$$p(\beta_i|\gamma) = \int_0^\infty p(\beta_i|\tau_i)p(\tau_i|\gamma)\, d\tau_i = \frac{\sqrt{\gamma}}{2} \exp\{-\sqrt{\gamma}\,|\beta_i|\},$$

showing that the Laplacian prior on $\boldsymbol{\beta}$ is equivalent to a 2-level hierachical-Bayes model: zero-mean Gaussian priors with independent variances and exponential hyper-priors for these variances. This decomposition has been exploited in robust *least absolute deviation* (LAD) regression, where the noise model, rather than the prior, is Laplacian [15].

The hierarchical decomposition of the Laplacian prior allows using the EM algorithm to estimate $\boldsymbol{\beta}$, by seeing $\boldsymbol{\tau} \equiv [\tau_1, ..., \tau_m]^T$ (in addition to $\mathbf{z}$) as missing data. The complete log-posterior (*i.e.*, including $\boldsymbol{\tau}$ and $\mathbf{z}$) is

$$\begin{aligned} \log p(\boldsymbol{\beta}|\mathbf{y}, \boldsymbol{\tau}, \mathbf{z}) &\propto \log p(\mathbf{z}|\boldsymbol{\beta}) + \log p(\boldsymbol{\beta}|\boldsymbol{\tau}) \\ &\propto -\|\mathbf{H}\boldsymbol{\beta} - \mathbf{z}\|^2 - \boldsymbol{\beta}^T\boldsymbol{\Upsilon}\boldsymbol{\beta}. \end{aligned} \tag{11}$$

where $\boldsymbol{\Upsilon} = \text{diag}(\tau_1^{-1}, ..., \tau_m^{-1})$. In addition to the expected values of the z_i's, now the E-step also requires computing $E[\tau_i^{-1}|\widehat{\boldsymbol{\beta}}^{(t)}, \mathbf{y}, \gamma]$. This can be computed, by observing that

$$\begin{aligned} p(\tau_i|\widehat{\boldsymbol{\beta}}^{(t)}, \mathbf{y}, \gamma) &\propto p(\mathbf{y}|\widehat{\boldsymbol{\beta}}^{(t)}, \tau_i)\, p(\widehat{\boldsymbol{\beta}}^{(t)}|\tau_i)\, p(\tau_i|\gamma) \\ &\propto p(\widehat{\boldsymbol{\beta}}^{(t)}|\tau_i)\, p(\tau_i|\gamma), \end{aligned}$$

because $p(\mathbf{y}|\widehat{\boldsymbol{\beta}}^{(t)}, \tau_i) = p(\mathbf{y}|\widehat{\boldsymbol{\beta}}^{(t)})$. Finally,

$$\begin{aligned} \omega_i \;\equiv\; E[\tau_i^{-1}|\widehat{\boldsymbol{\beta}}^{(t)}, \mathbf{y}, \gamma] \;&=\; \frac{\int_0^\infty \frac{1}{\tau_i}\, p(\tau_i|\gamma) p(\widehat{\boldsymbol{\beta}}^{(t)}|\tau_i)\, d\tau_i}{\int_0^\infty p(\tau_i|\gamma) p(\widehat{\boldsymbol{\beta}}^{(t)}|\tau_i)\, d\tau_i} \\ &=\; \gamma\, |\widehat{\beta}_i^{(t)}|^{-1}, \end{aligned} \tag{12}$$

where both integrations can be performed analytically, and $p(\widehat{\boldsymbol{\beta}}^{(t)}|\tau_i) = \mathcal{N}(\widehat{\boldsymbol{\beta}}^{(t)}|0, \tau_i)$.

We can now replace $\boldsymbol{\Upsilon}$ by its conditional expectation $\boldsymbol{\Omega} = \text{diag}(\omega_1, ..., \omega_m)$, and $\mathbf{z}$ by $\mathbf{v}$ (as seen in Section 2) in the complete log-posterior (Eq. (11)). Maximization with respect to $\boldsymbol{\beta}$ leads to the M-step

$$\widehat{\boldsymbol{\beta}}^{(t+1)} = (\boldsymbol{\Omega} + \mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{v}. \tag{13}$$

3.2. A non-informative hyper-prior

One question remains: how to adjust γ, which is the main parameter controlling the degree of sparseness of the estimates? We could consider cross-validation methods, but these are computationally demanding and inefficient in terms of data usage, and fall outside of the Bayesian paradigm. We consider a radically different alternative: remove γ from the model. To do so, we replace the exponential hyper-prior in Eq. (10) by a non-informative Jeffreys hyper-prior: $p(\tau_i) = 1/\tau_i$. The Jeffreys prior expresses the notion of ignorance/invariance, in this case with respect to changes in measurement scale (see [2, 12]). Of course, we no longer have the Laplacian prior on $\boldsymbol{\beta}$, but some other prior resulting from the adoption of the Jeffreys hyper-prior.

It turns out that this new hyper-prior leads to a minor modification of the EM algorithm described above, where only the computation of $\boldsymbol{\Omega}$ is affected. The integrations in Eq. (12) can still be performed analytically leading to

$$\boldsymbol{\Omega} = \text{diag}\left(|\widehat{\beta}_1^{(t)}|^{-2}, ..., |\widehat{\beta}_m^{(t)}|^{-2}\right). \tag{14}$$

3.3. The complete algorithm

Since we expect several components of $\boldsymbol{\beta}$ to become zero, it may be tricky to deal with $\boldsymbol{\Omega}$ as defined in Eq. (14). To avoid this problem, we define a new (diagonal) matrix

$$\boldsymbol{\Theta} = \text{diag}\left(|\widehat{\beta}_1^{(t)}|, ..., |\widehat{\beta}_m^{(t)}|\right), \tag{15}$$

based on which Eq. (13) can be re-written as

$$\widehat{\boldsymbol{\beta}}^{(t+1)} = \boldsymbol{\Theta}(\mathbf{I} + \boldsymbol{\Theta}\mathbf{H}^T\mathbf{H}\boldsymbol{\Theta})^{-1}\boldsymbol{\Theta}\mathbf{H}^T\mathbf{v}, \qquad (16)$$

thus avoiding the inversion of the elements of $\boldsymbol{\beta}$.

Summarizing, the complete learning algorithm is:

Step 1: Given the training data set, compute matrix $\mathbf{H}$ according to the type of classifier adopted.

Step 2: Compute an initial estimate $\widehat{\boldsymbol{\beta}}^{(0)}$. In all the experiments reported ahead, we compute a weakly penalized ridge-type estimate using the labels as data $\widehat{\boldsymbol{\beta}}^{(0)} = (\epsilon\mathbf{I} + \mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}\mathbf{y}$ (with, *e.g.*, $\epsilon = 10^{-6}$).

Step 3: (E-step) Given the current estimate $\widehat{\boldsymbol{\beta}}^{(t)}$, compute the diagonal matrix $\boldsymbol{\Theta}$ according to Eq. (15), and vector $\mathbf{v}$ (with elements given by Eq. (6)).

Step 4: (M-step) Obtain a new estimate $\widehat{\boldsymbol{\beta}}^{(t+1)}$ (Eq. (16)). If $\|\widehat{\boldsymbol{\beta}}^{(t+1)} - \widehat{\boldsymbol{\beta}}^{(t)}\|/\|\widehat{\boldsymbol{\beta}}^{(t)}\| < \delta$, stop; else, go back to Step 3. In the examples reported below, $\delta = 10^{-3}$.

4. Experiments

4.1. Linear and quadratic classifiers

When used with linear and quadratic (or higher order) classifiers, our method may be seen as a feature selection criterion embedded into the learning algorithm. To illustrate this, consider two Gaussian classes with means

$$\boldsymbol{\mu}_1 = [\underbrace{1/\sqrt{2},\ 1/\sqrt{2},\ \overbrace{0, 0, ..., 0}^{d-2 \text{ zeros}}}_{d \text{ dimensions}}]^T$$

and $\boldsymbol{\mu}_2 = -\boldsymbol{\mu}_1$, and both the covariances are identity matrices. The optimal Bayes error rate, regardless of d, is equal to $\Phi(2|0,1) \simeq 0.1587$. Of course, the optimal classifier for this data is linear and only uses the first two dimensions of the data. We first trained our classifier, using both linear and quadratic functions; *i.e.*, the functions $\phi_i(\mathbf{x})$ (see Section 2) include all the d components, their squares, and all the pairwise products, giving a total number of $d + d(d+1)/2$ features. We also trained a standard Bayesian plug-in classifier obtained by estimating the mean and covariance of each class. Both classifiers were trained on 100 samples per class, and then tested on independent test sets of size 1000 (500+500). Fig. 2 shows the resulting (averaged over 30 repetitions) test set error rate as a function of d. Fig. 3 reports a similar experiment, now involving linear classifiers learned from training sets with 50 samples per class. These results show that the proposed method exhibits a much smaller performance degradation as more irrelevant features are included, compared with the common plug-in classifier.

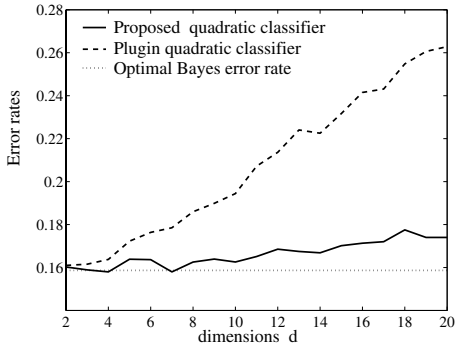

Figure 2. Average (30 repetitions) test error rates of quadratic classifiers vs. dimensionality.

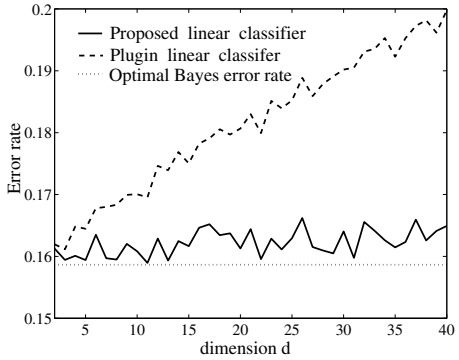

Figure 3. Average (30 repetitions) test error rates of linear classifiers vs. dimensionality.

4.2. Kernel classifiers

We will now consider experiments involving kernel classifiers, where $\mathbf{h}(\mathbf{x}) = [1, K(\mathbf{x}, \mathbf{x}^{(1)}), ..., K(\mathbf{x}, \mathbf{x}_{(n)})]^T$, and matrix $\mathbf{H}$ is as shown in Eq. (4). All the results were obtained with Gaussian kernels, *i.e.*,

$$K(\mathbf{x}, \mathbf{x}^{(i)}) = \exp\left\{-\frac{\|\mathbf{x} - \mathbf{x}^{(i)}\|^2}{2\,h^2}\right\},$$

where h is a parameter that controls the kernel width.

Our first experiment uses Ripley's synthetic data set[1], in which each class is a bi-modal mixture of two Gaussians; the optimal error rate for this problems is 8% [21]. Fig. 4 shows 100 points from the training set and the resulting classification boundary learned by our algorithm. The 5 training samples needed to support the selected kernels (like *support vectors* in SVM) are marked by small squares. Table 1 shows the average test set error (and the final number

[1]Available at `http://www.stats.ox.ac.uk/pub/PRNN/`

of kernels) of 20 classifiers learned from 20 random subsets of size 100 from the original 250 training samples. For comparison, the table also includes results reported in [4] for RVM, VRVM (variational RVM), and SVM classifiers. Our method is comparable to RVM and VRVM and clearly better than SVM (specially in terms of sparseness) on this data set. To allow the comparisons, we chose $h = 0.5$, which is the value used in [4].

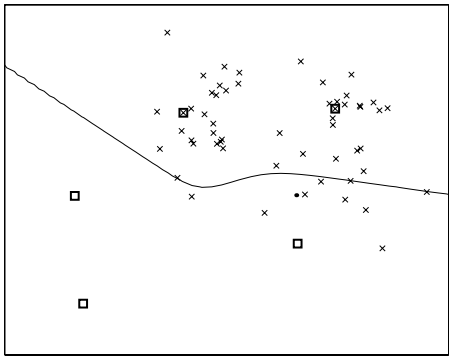

Figure 4. Classification boundary for Ripley's data. The data points corresponding to the selected kernels are marked by squares.

Table 1. Mean test error and number of kernels on Ripley's synthetic data set.

Method	Mean error rate	No. kernels
Proposed method	0.095	4.8
SVM	0.106	38
RVM	0.093	4
VRVM	0.092	4

For additional tests we used three well-known benchmark real-data problems: the *Pima indians diabetes*[1], where the goal is to decide wether a subject has diabetes or not, based on 8 measured variables; the *Leptograpsus crabs*[1] where the problem consists in determining the sex of crabs, based on 5 geometric measurements; and the *Wisconsin breast cancer*,[2] (*WBC*) where the task is to produce a benign/malignant diagnosis from a set of 30 numerical features. In the "Pima" case, we have 200 predefined training samples and 332 test samples. For the "crabs" problem, there are 80 (also predefined) training samples and 120 test samples. For the WBC problem, there is a total of 569 samples; the results reported were obtained by averaging over 30 random partitions with 300 training samples and 269 test samples (as in [22]). Prior to applying our algorithm, all the inputs are normalized to zero mean and unit variance, as is customary in kernel-based methods. The kernel width was set to $h = 4$, for the *Pima* and *crabs* problems, and to $h = 12$ for the WBC. Table 2 reports the numbers of errors achieved by the proposed method and by several other state-of-the-art techniques. On the *Pima* and *crabs* data sets, our algorithm outperforms all the other techniques. On the WBC data set, our method performs nearly as well as the best available alternative. The running time of our learning algorithm (implemented in MATLAB) is less than 1 second for the *crabs* data set, and about 2 seconds for the *Pima* and WBC problems. Finally, we note that the kernel classifiers obtained with our algorithm use only 6, 5, and 5 kernels (they are very sparse), for the *Pima*, *crabs*, and WBC data sets, respectively. Compare this with the 110 kernels selected by the SVM (reported in [4]) on the "Pima" data set.

[2] Available at the *Machine Learning Repository*: `http://www.ics.uci.edu/~mlearn/MLSummary.html`

Table 2. Numbers of test errors on three data sets studied.

Method	Pima	Crabs	WBC
Proposed method	61	0	8.5
SVM [22]	64	4	9
RVM [4]	65	N/A	N/A
VRVM [4]	65	N/A	N/A
Neural network [28]	75	3	N/A
Logistic regression [28]	66	4	N/A
Linear discriminant [22]	67	3	19
Gaussian process [22, 28]	67, 68	3	8

Like the SVM, our learning algorithm can be extended to k-class problems (with $k > 2$) by learning k binary classifiers (each class versus the others). There are other ways of extending the probit model to multi-class problems (see [1]) but we will not address them here. To test the performance of our method on a multiclass problem, we used the *forensic glass* data set[1], which is a 6-class problem with 9 features. As above, we set $h = 4$. Following [28], the classification error rate was estimated using 10-fold cross-validation. The results in Table 3 show that our method outperforms the best method referred in [28], which was a Gaussian process (GP) classifier implemented by MCMC. The GP-MCMC classifier requires about 24 hours of computer time, while ours is learned in a few seconds.

5. Concluding Remarks

We have introduced a Bayesian approach to supervised learning of sparse classifiers which (unlike other ap-

Table 3. Test error on the "forensinc glass" data estimated by 10-fold cross-validadtion.

Method	% error
Proposed method	21.5
Neural network [28]	23.8
Gaussian mixture [28]	30.8
Gaussian process [28]	23.3

proaches) does not involve any parameter explicitly controlling the degree of sparseness. Experimental evaluation of the proposed method on several publicly available standard benchmark data sets has shown that it is comparable to the state-of-the-art classification techniques.

Future research includes experiments on larger problems, like handwritten digit classification. One of the problems of our approach (for kernel-based classification) is the matrix inversion in the M-step, Eq. (16), meaning that the computational requirements scale as $O(n^3)$, making the method impractical for very large data sets. This issue is of current interest to researchers in kernel-based methods (see, *e.g.*, [29]), and we also intend to focus on it.

References

[1] J. Albert and S. Chib. Bayesian analysis of binary and polychotomous response data. *Jou. of the Amer. Stat. Assoc.*, 88:669–679, 1993.

[2] J. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer-Verlag, New York, 1980.

[3] C. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, New York, 1995.

[4] C. Bishop and M. Tipping. Variational relevance vector machines. In *Proc. of the 16th Conf. in Uncert. in Artif. Intell.*, pp. 46–53. Morgan Kaufmann Publishers, 2000.

[5] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. CRC Press, 1983.

[6] S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. *SIAM Jour. of Scient. Comput.*, 20(1):33–61, 1998.

[7] V. Cherkassky and F. Mulier. *Learning from Data: Concepts, Theory, and Methods*. Wiley, New York, 1998.

[8] N. Cristianini and J. Shawe-Taylor. *Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, Cambridge, U.K., 2000.

[9] Y. Dan Rubinstein and T. Hastie. Discriminative versus informative learning. In *Proc. of the The Third Int. Conf. on Knowledge Discovery and Data Mining*, pp. 49-53. AAAI Press, 1997.

[10] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood estimation from incomplete data via the EM algorithm. *Jour. of the Royal Stat. Soc. B*, 39:1–38, 1977.

[11] L. Fahrmeir and G. Tutz. *Multivariate Statistical Modelling Based on Generalized Linear Models*. Springer Verlag, New York, 1994.

[12] M. Figueiredo and R. Nowak. Wavelet-based image estimation: an empirical Bayes approach using Jeffreys' noninformative prior. *IEEE Trans. Image Proc.*, 10:1322–1331, 2001.

[13] A. K. Jain, R. Duin, and J. Mao. Statistical pattern recognition: A review. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 22(1):4–38, 2000.

[14] G. Kimeldorf and G. Wahba. A correspondence between Bayesian estimation of stochastic processes and smoothing by splines. *Annals of Math. Stat.*, 41:495–502, 1990.

[15] K. Lange and J. Sinsheimer. Normal/independent distributions and their applications in robust regression. *Jour. of Comp. and Graph. Stat.*, 2:175–198, 1993.

[16] M. Lewicki and T. Sejnowski. Learning overcomplete representations. *Neural Computation*, 12:337–365, 2000.

[17] D. MacKay. Bayesian non-linear modelling for the 1993 energy prediction competition. In G. Heidbreder, editor, *Max. Entropy and Bayesian Meth.*, pp. 221–234. Kluwer, Dordrecht, 1996.

[18] P. McCullagh and J. Nelder. *Generalized Linear Models*. Chapman and Hall, London, 1989.

[19] R. Neal. *Bayesian Learning for Neural Networks*. Springer Verlag, New York, 1996.

[20] T. Poggio and F. Girosi. Networks for approximation and learning. *Proc. of the IEEE*, 78:1481–1497, 1990.

[21] B. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge, U.K., 1996.

[22] M. Seeger. Bayesian model selection for support vector machines, Gaussian processes and other kernel classifiers. In S. Solla, T. Leen, and K.-R. Müller, editors, *Adv. in Neural Info. Proc. Systems 12*, pp. 603–609. MIT Press, 2000.

[23] R. Tibshirani. Regression shrinkage and selection via the lasso. *Jour. of the Royal Stat. Soc. (B)*, 58, 1996.

[24] M. Tipping. The relevance vector machine. In S. Solla, T. Leen, and K.-R. Müller, editors, *Adv. in Neural Info. Proc. Systems 12*, pp. 652–658. MIT Press, 2000.

[25] V. Vapnik. *Statistical Learning Theory*. John Wiley, New York, 1998.

[26] G. Wahba. *Spline Models for Observational Data*. SIAM, Philadelphia, 1990.

[27] C. Williams. Prediction with Gaussian processes: from linear regression to linear prediction and beyond. In *Learning and Inference in Graphical Models*. Kluwer, 1998.

[28] C. Williams and D. Barber. Bayesian classification with Gaussian priors. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 20(12):1342–1351, 1998.

[29] C. Williams and M. Seeger. Using the Nystrom method to speedup kernel machines. In *NIPS 13*. MIT Press, 2001.

[30] P. Williams. Bayesian regularization and pruning using a Laplace prior. *Neural Computation*, 7:117–143, 1995.