Adaptive Sparseness for Supervised Learning

Mário A.T. Figueiredo, *Senior Member*, *IEEE*

Abstract—The goal of supervised learning is to infer a functional mapping based on a set of training examples. To achieve good generalization, it is necessary to control the "complexity" of the learned function. In Bayesian approaches, this is done by adopting a prior for the parameters of the function being learned. We propose a Bayesian approach to supervised learning, which leads to sparse solutions; that is, in which irrelevant parameters are automatically set exactly to zero. Other ways to obtain sparse classifiers (such as Laplacian priors, support vector machines) involve (hyper)parameters which control the degree of sparseness of the resulting classifiers; these parameters have to be somehow adjusted/estimated from the training data. In contrast, our approach does not involve any (hyper)parameters to be adjusted or estimated. This is achieved by a hierarchical-Bayes interpretation of the Laplacian prior, which is then modified by the adoption of a Jeffreys' noninformative hyperprior. Implementation is carried out by an expectation-maximization (EM) algorithm. Experiments with several benchmark data sets show that the proposed approach yields state-of-the-art performance. In particular, our method outperforms SVMs and performs competitively with the best alternative techniques, although it involves no tuning or adjustment of sparseness-controlling hyperparameters.

Index Terms—Supervised learning, classification, regression, sparseness, feature selection, kernel methods, expectation-maximization algorithm.

1 Introduction: The Learning Problem

Supervised learning can be formalized as the problem of inferring a function $y = f(\mathbf{x})$, based on a training set $\mathcal{D} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$. Usually, the *inputs* are d-dimensional vectors, $\mathbf{x}_i = [x_{i,1}, \dots, x_{i,d}]^T \in \mathbb{R}^d$. When y is continuous (e.g., $y \in \mathbb{R}$), we are in the context of *regression*, whereas in *classification* problems, y is of categorical nature (e.g., binary, $y \in \{-1, 1\}$). The obtained function is evaluated by how well it generalizes, i.e., how accurately it performs on new data assumed to follow the same distribution as the training data. Usually, the function $f(\cdot)$ is assumed to have a fixed structure and to depend on a set of parameters β (say, the coefficients in a linear classifier, or the weights in a feed-forward neural network); in this case, we write $y = f(\mathbf{x}, \beta)$, and the goal becomes to estimate β from the training data.

It is known that, to achieve good generalization, 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 (what is usually called *overfitting*). Conversely, an overly simple function may not be rich enough to capture the the true underlying relationship (*underfitting*). This is a well-known problem which has been addressed with a variety of formal tools (see, e.g., [3], [4], [5], [6], [7], and references therein).

2 Overview of Methods and the Proposed Approach

2.1 Discriminative versus Generative Learning

Supervised learning can be formulated using either a *generative* approach or a *discriminative* approach. Basically, the generative approach involves estimating the joint probability density function $p(\mathbf{x}, y)$ from $\mathcal{D}$. In classification, this is usually done by learning the so-called *class-conditional* densities, $p(\mathbf{x}|y)$, and the probability of each class, $p(y)$, [6]. In regression, this can be done, for example, by representing the joint density using a kernel method or a Gaussian mixture (see [8] and references therein). From this joint probability function estimate, optimal Bayesian decision rules can be derived by the standard Bayesian decision theory machinery [6].

In the discriminative approach, the focus is on learning the function $f(\mathbf{x}, \beta)$ directly from the training set. Well known discriminative approaches include linear and generalized linear models, k-nearest neighbor classifiers, tree classifiers [9], feed-forward neural networks [6], support vector machines (SVM), and other kernel-based methods [4], [7], [10]. Although, usually computationally more demanding, discriminative approaches tend to perform better, especially with small training data sets (see [7]). The approach described in this paper falls in the *discriminative* category.

2.2 Bayesian Discriminative Learning with Gaussian Priors

A Bayesian approach to complexity control in discriminative learning consists in using a prior $p(\beta|\alpha)$ favoring *simplicity*, or *smoothness*, in some sense, of the function to be learned (where α is a vector of *hyperparameters*) [5]. The usual choice, namely for analytical and computational tractability, is a zero-mean Gaussian prior, which appears under different guises, like *ridge regression* [11], or *weight decay* [5], [6]. Gaussian priors are also at the heart of the nonparametric *Gaussian processes* approach [5], [10], [12], which has roots in earlier spline models [13] and regularized radial basis functions [14].

The main disadvantage of Gaussian priors is that they do not control the *structural* complexity of the learned function. That is, if one of the components of β (say, a given coefficient of a linear classifier) happens to be irrelevant, it will not be set

• *The author is with the Institute of Telecommunications and the Department of Electrical and Computer Engineering, Instituto Superior Técnico, 1049-001 Lisboa, Portugal. E-mail: mtf@lx.it.pt.*

Manuscript received 13 Mar. 2002; revised 26 Sept. 2002; accepted 20 Mar. 2003.
Recommended for acceptance by A. Del Bimbo.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 116078.

exactly to zero. Any structural simplification will have to be based on additional tests.

2.3 Sparseness

A *sparse estimate* of β is defined as one in which irrelevant or redundant components are exactly zero. Sparseness is desirable in supervised learning for several reasons, namely:

- Sparseness leads to a structural simplification of the estimated function.
- Obtaining a sparse estimate corresponds to performing feature/variable selection.
- In kernel-based methods, the generalization ability improves with the degree of sparseness (a key idea behind SVM [4], [7]).

One of the possible ways to achieve sparse estimates is to use a zero-mean Laplacian (rather than Gaussian) prior,

$$p(\beta|\alpha) \propto \prod_i \exp\{-\alpha\,|\beta_i|\} = \exp\{-\alpha\,\|\beta\|_1\}, \qquad (1)$$

where $\|\beta\|_1 = \sum_i |\beta_i|$ denotes the l_1 norm, and α is the parameter of this density. The sparseness-inducing nature of the Laplacian prior (or, equivalently, of the l_1 penalty from a regularization point of view) is well-known and has been exploited in several areas [15], [16], [17].

SVMs are another approach to supervised learning leading to sparse structures. Both in approaches based on Laplacian priors and in SVMs, there are hyperparameters (e.g., α in (1)) controlling the degree of sparseness of the obtained estimates. These parameters are commonly adjusted by crossvalidation methods which may be very time consuming.

2.4 Proposed Approach

We propose a Bayesian approach to sparse regression and classification whose main advantage is the absence of parameters controlling the degree of sparseness. This is achieved with the following building blocks:

1. a hierarchical-Bayes interpretation of the Laplacian prior as a *normal/independent* distribution (as proposed for robust regression in [18]);
2. a Jeffreys' noninformative second-level hyperprior (in the same spirit as [19]), which expresses scale-invariance and, more importantly, is parameter-free [20];
3. an *expectation-maximization* (EM) algorithm which yields a *maximum a posteriori* (MAP) estimate of β (and of the observation noise variance, in the case of regression).

Experimental evaluation of the proposed method (Sections 5.1 and 5.2), both with synthetic and real data, show that our method performs competitively with (often better than) the state-of-the-art methods (such as SVM).

2.5 Related Approaches

Our method is formally and conceptually related to the *automatic relevance determination* (ARD) concept [21], [5], which underlies the recently proposed *relevance vector machine* (RVM) [22], [23]. The RVM exhibits state-of-the-art performance: It beats SVMs, both in terms of accuracy and sparseness [22], [23]. However, our approach does not rely on a *type-II maximum likelihood* approximation [20] (as in ARD and RVM); rather, our modeling assumptions lead to a marginal a posteriori probability function on β whose mode is located by a very simple EM algorithm. A more detailed explanation of the difference between our approach and the RVM will be presented in Section 3.5.

2.6 Organization of the Paper

In Section 3, after reviewing linear regression and Laplacian priors, we introduce our approach. Section 4 extends the formulation to classification problems by using a (*probit*) generalized linear model. Experimental results are reported in Section 5. Finally, Section 6 concludes the paper by presenting some concluding remarks, including a brief discussion of the limitations of the approach and possible directions of future research.

3 Regression

3.1 Introduction: Linear Regression with Gaussian Prior

In this paper, we consider regression functions that are linear with respect to the parameter vector β (whose dimensionality we will denote by k), i.e.,

$$f(\mathbf{x},\beta) = \sum_{i=1}^{k} \beta_i h_i(\mathbf{x}) = \beta^T \mathbf{h}(\mathbf{x}), \qquad (2)$$

where $\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x}),\ldots,h_k(\mathbf{x})]^T$ is a vector of k fixed functions of the input, often called *features.* Functions of this form include the following well-known formulations:

1. Linear regression, in which $\mathbf{h}(\mathbf{x}) = [1, x_1, \ldots, x_d]^T$; in this case, $k = d + 1$.
2. Nonlinear regression via a set of k fixed basis functions, where $\mathbf{h}(\mathbf{x}) = [\phi_1(\mathbf{x}),\ldots,\phi_k(\mathbf{x})]^T$; usually, $\phi_1(\mathbf{x}) = 1$.
3. Kernel regression, $\mathbf{h}(\mathbf{x}) = [1, K(\mathbf{x},\mathbf{x}_1),\ldots,K(\mathbf{x},\mathbf{x}_n)]^T$, where $K(\mathbf{x},\mathbf{x}_i)$ is some (symmetric) kernel function [4] (as in SVM and RVM regression); in this case $k = n + 1$.

We follow the standard assumption that the output variables in the training set were contaminated by additive white Gaussian noise, $y_i = f(\mathbf{x}_i,\beta) + w_i$, for $i = 1,\ldots,n$, where $[w_1,\ldots,w_n]$ is a set of independent zero-mean Gaussian samples with variance σ^2. With $\mathbf{y} = [y_1,\ldots,y_n]^T$, the likelihood function is then

$$p(\mathbf{y}|\beta) = \mathcal{N}(\mathbf{y}|\mathbf{H}\beta, \sigma^2\mathbf{I}),$$

where $\mathcal{N}(\mathbf{v}|\mu,\mathbf{C})$ stands for a Gaussian density of mean μ and covariance $\mathbf{C}$, evaluated at $\mathbf{v}$, $\mathbf{H}$ is the so-called *design matrix*, and $\mathbf{I}$ is an identity matrix. The element (i,j) of $\mathbf{H}$, denoted H_{ij}, is given by $H_{ij} = h_j(\mathbf{x}_i)$.

With a zero-mean Gaussian prior for β, with covariance $\mathbf{A}$, $p(\beta|\mathbf{A}) = \mathcal{N}(\beta|0,\mathbf{A})$, the posterior $p(\beta|\mathbf{y})$ is still Gaussian with the mode given by the well-known expression

$$\widehat{\beta} = (\sigma^2\mathbf{A}^{-1} + \mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{y}.$$

This is, of course, the *maximum a posteriori* (MAP) estimate of β under this Gaussian prior. When $\mathbf{A}$ is proportional to identity, say $\mathbf{A} = \mu^2\mathbf{I}$, this is called *ridge regression* [11]. When $\mu \to \infty$, we obtain $\widehat{\beta} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{y}$, the well-known *ordinary least squares* estimate, which may have undesirably large variance when $\mathbf{H}^T\mathbf{H}$ is an ill-conditioned matrix.

3.2 Regression with a Laplacian Prior

To favor a sparse estimate, a Laplacian prior can be adopted for β; that is

$$p(\beta|\alpha) = \prod_{i=1}^{k} \frac{\alpha}{2}\exp\{-\alpha\,|\beta_i|\} = \left(\frac{\alpha}{2}\right)^k \exp\{-\alpha\,\|\beta\|_1\}.$$

In this case, the posterior probability density function $p(\beta|\mathbf{y})$ is no longer Gaussian. The MAP estimate of β is no longer a linear function of $\mathbf{y}$; it is given by

$$\widehat{\beta} = \arg\min_{\beta}\{\|\mathbf{H}\beta - \mathbf{y}\|_2^2 + 2\,\sigma^2\alpha\,\|\beta\|_1\}, \tag{3}$$

where $\|\mathbf{v}\|_2^2 = \sum_i v_i^2$ denotes the squared Euclidean (l_2) norm. In regression, this criterion was named the LASSO (*least absolute shrinkage and selection operator*) in [16]. To understand why the l_1 norm induces sparseness, notice that moving a vector away from the coordinate axes increases the l_1 norm more than the l_2 norm. For example, $\|[1,\,0]^T\|_2 = \|[1/\sqrt{2},\, 1/\sqrt{2}]^T\|_2 = 1$, while $\|[1,\,0]^T\|_1 = 1 < \|[1/\sqrt{2},\, 1/\sqrt{2}]^T\|_1 = \sqrt{2}$.

The particular case where $\mathbf{H}$ is an orthogonal matrix gives us some further insight on the l_1 penalty. In this case, since $\mathbf{H}^T\mathbf{H} = I$, (3) can be solved separately for each β_i,

$$\begin{aligned}\widehat{\beta}_i &= \arg\min_{\beta_i}\{\beta_i^2 - 2\beta_i(\mathbf{H}^T\mathbf{y})_i + 2\,\sigma^2\,\alpha\,|\beta_i|\}\\ &= \operatorname{sgn}((\mathbf{H}^T\mathbf{y})_i)\left(|(\mathbf{H}^T\mathbf{y})_i| - \sigma^2\,\alpha\right)_+,\end{aligned} \tag{4}$$

where $(\mathbf{H}^T\mathbf{y})_i$ denotes the ith component of $\mathbf{H}^T\mathbf{y}$, $(\cdot)_+$ is the *positive part operator* (defined as $(a)_+ = a$, if $a \geq 0$, and $(a)_+ = 0$, if $a < 0$), and $\operatorname{sgn}(\cdot)$ is the sign function. Observe that, when the absolute value of $(\mathbf{H}^T\mathbf{y})_i$ is below a threshold, the estimate $\widehat{\beta}_i$ is exactly zero; otherwise, the estimate is obtained by subtracting a constant (equal to the threshold) from the absolute value of $(\mathbf{H}^T\mathbf{y})_i$. This rule is called the *soft threshold* (see Fig. 1), and is widely used in wavelet-based signal/image estimation [24], [25].

3.3 A Hierarchical-Bayes View of the Laplacian Prior

Let us now consider that each β_i has a zero-mean Gaussian prior $p(\beta_i|\tau_i) = \mathcal{N}(\beta_i|0,\tau_i)$, with its own variance τ_i, and that each τ_i has an exponential (hyper)prior

$$p(\tau_i|\gamma) = \frac{\gamma}{2}\exp\left\{-\frac{\gamma}{2}\,\tau_i\right\}, \quad \text{for } \tau_i \geq 0. \tag{5}$$

It is now possible to integrate out τ_i,

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

obtaining a Laplacian density. This shows that the Laplacian prior is equivalent to a two-level hierarchical-Bayes model: zero-mean Gaussian priors with independent, exponentially distributed variances. This equivalence has been previously exploited to derive EM algorithms for robust regression under Laplacian noise models [18]. A related equivalence was also considered in [26].

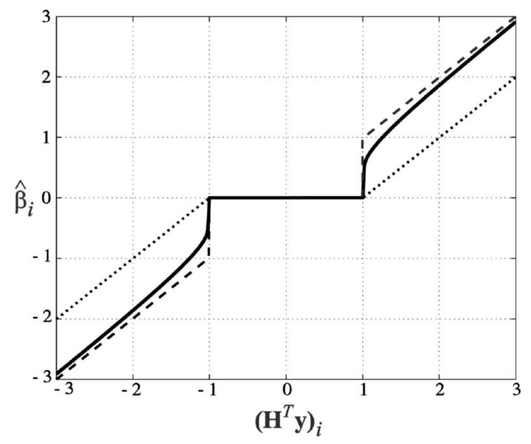

Fig. 1. Solid line: Estimation rule produced by the EM Algorithm with the Jeffrey's hyperprior, in the orthogonal $\mathbf{H}$ case. Dashed line: Hard-threshold rule. Dotted line: Soft-threshold rule (obtained with a Laplacian prior).

3.4 Sparse Regression via EM

The hierarchical decomposition of the Laplacian prior allows using the expectation-maximization (EM) algorithm to implement the LASSO criterion in (3). This is done simply by regarding $\tau = [\tau_1, \ldots, \tau_k]$ as *hidden/missing data*. If we could observe τ, the complete log-posterior $\log p(\beta, \sigma^2|\mathbf{y}, \tau)$ could be easily obtained; in fact,

$$p(\beta, \sigma^2|\mathbf{y}, \tau) \propto p(\mathbf{y}|\beta, \sigma^2)\,p(\beta|\tau)\,p(\sigma^2), \tag{6}$$

where both $p(\mathbf{y}|\beta, \sigma^2)$ and $p(\beta|\tau)$ are Gaussian densities (thus, the product can be carried out analytically), and we set $p(\sigma^2) =$ "constant" (of course, we could also adopt a conjugate inverse-Gamma prior for σ^2, but, for large n, the influence of the prior on the estimate of σ^2 is very small). In this case, it would be easy to obtain the MAP (or any other) estimate of β and σ^2. Since we do not observe τ, we may resort to the EM algorithm, which produces a sequence of estimates, $\widehat{\beta}_{(t)}$ and $\widehat{\sigma^2}_{(t)}$, for $t = 1, 2, 3, \ldots$, by applying two alternating steps:

E-step. Computes the expected value (with respect to the missing variables τ) of the complete log-posterior, given the current parameter estimates and the observed data; this is usually called the Q-function

$$Q(\beta,\,\sigma^2|\,\widehat{\beta}_{(t)},\,\widehat{\sigma^2}_{(t)}) = \int \log p(\beta, \sigma^2|\,\mathbf{y}, \tau)\,p(\tau|\,\widehat{\beta}_{(t)},\,\widehat{\sigma^2}_{(t)},\,\mathbf{y})\,d\tau.$$

M-step. Updates the parameter estimates by maximizing the Q-function with respect to parameters:

$$(\widehat{\beta}_{(t+1)}, \widehat{\sigma^2}_{(t+1)}) = \arg\max_{\beta,\sigma^2} Q(\beta,\,\sigma^2|\,\widehat{\beta}_{(t)},\,\widehat{\sigma^2}_{(t)}).$$

The EM algorithm converges to a local maximum of the a posteriori probability density function $p(\beta, \sigma^2|\mathbf{y}) \propto p(\mathbf{y}|\beta, \sigma^2)\,p(\beta|\gamma)$, without explicitly using the marginal prior

$p(\beta|\gamma)$, which is not Gaussian, but simply using the conditional Gaussian prior $p(\beta|\tau)$ in each iteration.

Let us now find the particular expressions for the Q-function and for the parameter updates, under the modelling assumptions above expressed, which we now summarize:

$$\begin{aligned} p(\mathbf{y}|\beta,\sigma^2) &= \mathcal{N}(\mathbf{y}|\mathbf{H}\beta,\sigma^2\,\mathbf{I})\\ p(\sigma^2) &\propto \text{"constant"},\\ p(\beta|\tau) &= \prod_{i=1}^{k}\mathcal{N}(\beta_i|0,\tau_i) = \mathcal{N}\Big(\beta|0,(\Upsilon(\tau))^{-1}\Big),\\ p(\tau|\gamma) &= \Big(\frac{\gamma}{2}\Big)^k\prod_{i=1}^{k}\exp\Big\{-\frac{\gamma}{2}\,\tau_j\Big\}, \quad \text{for } \tau_j \geq 0, \end{aligned}$$

where $\Upsilon(\tau) \equiv \mathrm{diag}(\tau_1^{-1},\dots,\tau_k^{-1})$ is a diagonal matrix with the inverse variances of all the β_i's. Moreover, $\mathbf{y}$ is the observed data, and τ is missing data.

- First, applying logarithms to (6), since $p(\sigma^2)$ is flat, we have

$$\begin{aligned} \log p(\beta,\sigma^2|\mathbf{y},\tau) &\propto \log p(\mathbf{y}|\,\beta,\sigma^2) + \log p(\beta|\tau)\\ &\propto\ -n\,\log\sigma^2 - \frac{\|y-\mathbf{H}\beta\|_2^2}{\sigma^2} - \beta^T\Upsilon(\tau)\beta. \end{aligned} \tag{7}$$

- Second, since the complete log-posterior is linear with respect to $\Upsilon(\tau)$ (see (7)), and its other two terms do not depend on τ, the E-step reduces to computing the conditional expectation of $\Upsilon(\tau)$, given $\mathbf{y}$ and the current estimates $\widehat{\sigma^2}_{(t)}$ and $\widehat{\beta}_{(t)}$, which we denote as

$$\begin{aligned} \mathbf{V}_{(t)} &= E[\Upsilon(\tau)|\,\mathbf{y},\widehat{\sigma^2}_{(t)},\widehat{\beta}_{(t)}]\\ &= \mathrm{diag}\Big\{E[\tau_1^{-1}|\,\mathbf{y},\widehat{\sigma^2}_{(t)},\widehat{\beta}_{(t)}],\dots,E[\tau_k^{-1}|\,\mathbf{y},\widehat{\sigma^2}_{(t)},\widehat{\beta}_{(t)}]\Big\}. \end{aligned} \tag{8}$$

- Now, observe that $p(\tau_i|\,\mathbf{y},\beta,\sigma^2) = p(\tau_i|\,\beta_i)$ because, given β_i, τ_i does not depend on $\mathbf{y}$, σ^2, or β_j, for $j\neq i$. So, $p(\tau_i|\,\mathbf{y},\widehat{\sigma^2}_{(t)},\widehat{\beta}_{(t)}) \propto p(\widehat{\beta}_{i,(t)}|\,\tau_i)\,p(\tau_i)$.
- Since $p(\widehat{\beta}_{i,(t)}|\,\tau_i) = \mathcal{N}(\widehat{\beta}_{i,(t)}|\,0,\tau_i)$ and $p(\tau_i)$ is the exponential hyperprior given by (5), elementary integration yields

$$\begin{aligned} E[\tau_i^{-1}|\,&\mathbf{y},\widehat{\sigma^2}_{(t)},\widehat{\beta}_{(t)}]\\ &= \frac{\int_0^\infty \frac{1}{\tau_i}\,\mathcal{N}(\widehat{\beta}_{i,(t)}|\,0,\tau_i)\,\frac{\gamma}{2}\exp\{-\frac{\gamma}{2}\,\tau_i\}\,d\tau_i}{\int_0^\infty \mathcal{N}(\widehat{\beta}_{i,(t)}|\,0,\tau_i)\,\frac{\gamma}{2}\exp\{-\frac{\gamma}{2}\,\tau_i\}\,d\tau_i}\\ &= \frac{\gamma}{|\widehat{\beta}_{i,(t)}|}, \end{aligned} \tag{9}$$

thus,

$$\mathbf{V}_{(t)} = \gamma\,\mathrm{diag}\Big\{|\widehat{\beta}_{1,(t)}|^{-1},\dots,|\widehat{\beta}_{k,(t)}|^{-1}\Big\}. \tag{10}$$

- The Q-function is obtained by plugging $\mathbf{V}_{(t)}$ in the place of $\Upsilon(\tau)$ in (7),

$$\begin{aligned} Q(\beta,\,\sigma^2|\,&\widehat{\beta}_{(t)},\,\widehat{\sigma^2}_{(t)}) =\\ &-n\,\log\sigma^2 - \frac{\|\mathbf{y}-\mathbf{H}\beta\|_2^2}{\sigma^2} - \beta^T\mathbf{V}_{(t)}\beta. \end{aligned} \tag{11}$$

- Finally, the M-step (estimate update equations) consists in maximizing $Q(\beta,\,\sigma^2|\,\widehat{\beta}_{(t)},\,\widehat{\sigma^2}_{(t)})$ with respect to σ^2 and β, yielding

$$\begin{aligned} \widehat{\sigma^2}_{(t+1)} &= \arg\max_{\sigma^2}\left(-n\,\log\sigma^2 - \frac{\|y-\mathbf{H}\beta\|_2^2}{\sigma^2}\right)\\ &= \frac{\|\mathbf{y}-\mathbf{H}\widehat{\beta}_{(t)}\|_2^2}{n} \end{aligned} \tag{12}$$

and

$$\begin{aligned} \widehat{\beta}_{(t+1)} &= \arg\max_{\beta}\left(-\frac{\|\mathbf{y}-\mathbf{H}\beta\|_2^2}{\sigma^2} - \beta^T\mathbf{V}_{(t)}\beta\right)\\ &= \Big(\widehat{\sigma^2}_{(t+1)}\mathbf{V}_{(t)} + \mathbf{H}^T\mathbf{H}\Big)^{-1}\mathbf{H}^T\mathbf{y}. \end{aligned} \tag{13}$$

Summarizing, the E-step is implemented by (10), while (12) and (13) constitute the M-step. This EM algorithm is not the most computationally efficient way to solve (3); see, e.g., the methods proposed in [27], [16]. However, it is very simple to implement and serves our main goal which is to open the way to the adoption of different hyperpriors.

3.5 Comparison with the RVM

We are now in position to explain the difference between this EM approach and the *relevance vector machine* (RVM) [23]; for simplicity, we will consider that σ^2 is known and omit it from the notation. In the RVM, the modeling assumptions are similar: each β_i has a zero-mean Gaussian prior $p(\beta_i|\tau_i) = \mathcal{N}(\beta_i|0,\tau_i)$, with its own variance τ_i. However, rather than integrating out the (hyper)parameters $\tau = [\tau_1,\dots,\tau_k]$, explicitly, or implicitly via the EM algorithm, a maximum likelihood estimate $\widehat{\tau}$ is obtained from the marginal likelihood $p(\mathbf{y}|\tau)$, and then plugged into the a posteriori density $p(\beta|\widehat{\tau},\mathbf{y})$. In the Bayesian literature, this is known as an *empirical Bayes* or *type-II maximum likelihood* approach [20].

3.6 Getting Rid of the Hyperparameter: The Jeffreys' Prior

One question remains: How to adjust γ, which controls the degree of sparseness of $\widehat{\beta}$? Our proposal is to remove γ from the model, by replacing the exponential hyperprior on each τ_i (here, generically denoted as τ) by a noninformative Jeffreys hyperprior:

$$p(\tau) \propto \frac{1}{\tau}. \tag{14}$$

This prior expresses ignorance with respect to scale (see [20], [19]) and, most importantly, it is parameter-free. To see why this prior is scale-invariant, suppose we change the measurement units (scale) in which τ is expressed; this defines a new variable $\tau' = K\tau$, where K is the constant expressing the change of units/scale. Then, by applying the rule for the change of variable in a probability density function to

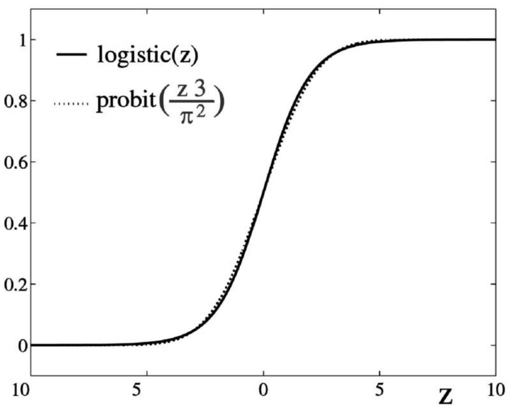

Fig. 2. The logistic and (rescaled) probit link functions.

$p(\tau) = 1/\tau$, we retain the same prior, $p(\tau') \propto 1/\tau'$. Notice that this prior is not normalizable (its integral is not finite); it is a so-called *improper* prior [20]. Of course, this prior no longer leads to a Laplacian prior on β, but to some other prior (see [19] for details). As will be shown experimentally, this prior strongly induces sparseness and leads to a very good performance.

Computationally, this choice leads to a minor modification of the EM algorithm described above: Matrix $\mathbf{V}_{(t)}$ is now given by

$$\mathbf{V}_{(t)} = \mathrm{diag}(|\widehat{\beta}_{1,(t)}|^{-2}, \ldots, |\widehat{\beta}_{k,(t)}|^{-2}); \tag{15}$$

notice the absence of any free parameter in this E-step.

Finally, to get some insight into the behavior of this scheme, we consider the case in which $\mathbf{H}$ is an orthogonal matrix. In this case, all the equations in the EM algorithm decouple, and the final estimate of each coefficient $\widehat{\beta}_i$ is only a function of the corresponding $(\mathbf{H}^T\mathbf{y})_i$. In Fig. 1, we plot $\widehat{\beta}_i$ as a function of $(\mathbf{H}^T\mathbf{y})_i$, obtained by the EM algorithm initialized with $\widehat{\beta}_{i,(0)} = (\mathbf{H}^T\mathbf{y})_i$. For comparison purposes, we also plot the well-known hard and soft threshold estimation rules [25]. Observe how the resulting estimation rule is between those two rules; for large values of $(\mathbf{H}^T\mathbf{y})_i$, the estimate is very close to $(\mathbf{H}^T\mathbf{y})_i$ (equal in the limit), approaching the behavior of the hard rule; as $(\mathbf{H}^T\mathbf{y})_i$ gets smaller, the estimate becomes progressively shrunken, approaching the behavior of the soft rule; when $(\mathbf{H}^T\mathbf{y})_i$ is below a threshold, the estimate is exactly zero (as in the hard and soft rules), this being the reason why sparseness is achieved.

3.7 An Important Implementation Detail

Since several elements of $\widehat{\beta}$ are expected to go to zero, it is not convenient to deal with $\mathbf{V}_{(t)}$, as given by (15), because that would imply handling arbitrarily large numbers. However, it turns out that we can rewrite the M-step ((13)) as

$$\widehat{\beta}_{(t+1)} = \mathbf{U}_{(t)}(\widehat{\sigma^2}_{(t+1)}\mathbf{I} + \mathbf{U}_{(t)}\mathbf{H}^T\mathbf{H}\mathbf{U}_{(t)})^{-1}\mathbf{U}_{(t)}\mathbf{H}^T\mathbf{y},$$

where

$$\mathbf{U}_{(t)} \equiv \mathrm{diag}(|\widehat{\beta}_{1,(t)}|, \ldots, |\widehat{\beta}_{k,(t)}|), \tag{16}$$

thus avoiding the inversion of the elements of $\widehat{\beta}_{(t)}$. Moreover, it is not necessary to obtain the inverse matrix, but simply to solve the corresponding linear system, whose dimension is only the number of nonzero elements in $\mathbf{U}_{(t)}$.

4 Classification

4.1 Introduction: Generalized Linear Models

In classification problems, the formulation is somewhat more complicated due to the categorical nature of the output variable. A standard approach is to consider a *generalized linear model* (GLM) [28]. For a two-class problem ($y \in \{-1, 1\}$), a GLM models the probability that an observation $\mathbf{x}$ belongs to, say, class 1, as given by a nonlinearity applied to the output of a linear regression function. Formally, let this nonlinearity (called the *link function*) be denoted as $\psi : \mathbb{R} \to [0, 1]$; then, the GLM is defined as

$$P(y = 1|\mathbf{x}) = \psi(\beta^T\mathbf{h}(\mathbf{x})),$$

where $\mathbf{h}(\mathbf{x})$ was defined in Section 3.1. In *logistic* regression [28], the link function is

$$\psi(z) = (1 + \exp(-z))^{-1}, \tag{17}$$

shown in Fig. 2. An important feature of the GLM approach is that it yields class probabilities, rather than a hard classifier; it can thus be used to obtain optimal classifiers under different cost functions. For example, if the cost function is simply the misclassification error, then the classifier is obtained by thresholding $\psi(\beta^T\mathbf{h}(\mathbf{x}))$ at $1/2$.

4.2 The Probit

Although the most common link is the *logistic* function, here we adopt the probit model $\psi(z) = \Phi(z)$, where $\Phi(z)$ is the standard Gaussian *cumulative distribution function* (cdf)

$$\Phi(z) \equiv \int_{-\infty}^{z} \mathcal{N}(x|0, 1)\, dx. \tag{18}$$

The rescaled probit $\Phi(3\,z\,\pi^{-2})$ is plotted in Fig. 2, together with the logistic function; notice that they are almost indistinguishable [29]. Of course, both the logistic and probit functions can be rescaled (horizontally), but this scale is implicitly absorbed by β.

4.3 Learning a Probit Classifier via EM

The fundamental reason behind our choice of the probit model is its simple interpretation in terms of hidden variables [30]. Consider a hidden variable $z = \beta^T\mathbf{h}(\mathbf{x}) + w$, where w is a zero-mean unit-variance Gaussian noise sample $p(w) = \mathcal{N}(w|0, 1)$. Then, if the classification rule is $y = 1$ if $z \geq 0$, and $y = -1$ if $z < 0$, we obtain the probit model:

$$P(y = 1|\mathbf{x}) = P(\beta^T\mathbf{h}(\mathbf{x}) + w \geq 0) = \Phi(\beta^T\mathbf{h}(\mathbf{x})).$$

Given training data $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, consider the corresponding vector of hidden/missing variables $\mathbf{z} = [z_1, \ldots, z_n]^T$, where

$$z_i = \beta^T\mathbf{h}(\mathbf{x}_i) + w_i;$$

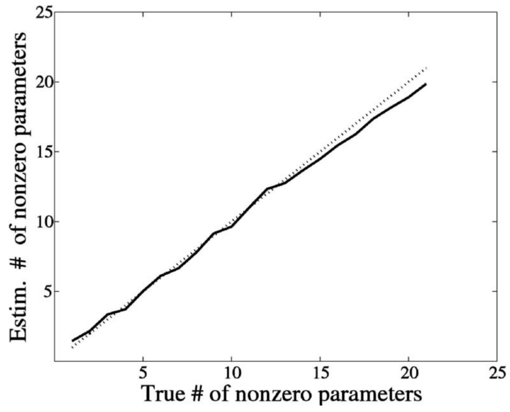

Fig. 3. Mean number of nonzero components in $\widehat{\beta}$ versus the number of nonzero components in β (the dotted line is the identity).

of course, z_i is not observed because we also do not know w_i. However, if we had $\mathbf{z}$, we would have a simple linear regression likelihood with unit noise variance: $p(\mathbf{z}|\beta) = \mathcal{N}(\mathbf{z}|\mathbf{H}\beta, \mathbf{I})$. This observation suggests the use of the EM algorithm to estimate β, by treating $\mathbf{z}$ as missing data.

To promote sparseness, we will adopt the same hierarchical prior on β that we have used for regression: $p(\beta_i|\tau_i) = \mathcal{N}(\beta_i|0, \tau_i)$ and $p(\tau_i) \propto 1/\tau_i$ (the Jeffreys prior). The complete log posterior (with the hidden vectors τ and $\mathbf{z}$) is

$$\begin{aligned}\log p(\beta|\,\mathbf{y}, \tau, \mathbf{z}) &\propto \log p(\beta, \mathbf{y}, \tau, \mathbf{z}) \\ &\propto\; p(\mathbf{z}|\beta)\, p(\beta|\,\tau)\, p(\tau)\, p(\mathbf{y}|\,\mathbf{z}) \\ &\propto -\beta^T\mathbf{H}^T\mathbf{H}\beta - 2\beta^T\mathbf{H}^T z - \beta^T\Upsilon(\tau)\beta,\end{aligned} \tag{19}$$

after dropping all terms that do not depend on β. Notice that this is similar to (7), with two differences: the noise variance is 1; the missing $\mathbf{z}$ plays the role of observations in the regression equations. The expected value of $\Upsilon(\tau)$ is, of course, similar to the regression case, since it only depends on β; accordingly, we consider the same diagonal matrix defined in (16). In addition, we also need $E[\mathbf{z}|\widehat{\beta}_{(t)}, \mathbf{y}]$ (notice that the complete log-posterior is linear with respect to $\mathbf{z}$), which can be expressed in closed form, for each z_i, as

$$s_{i,(t)} \equiv E[z_i|\widehat{\beta}_{(t)}, \mathbf{y}]$$

$$= \begin{cases} \widehat{\beta}_{(t)}^T\mathbf{h}(\mathbf{x}_i) + \frac{\mathcal{N}(\widehat{\beta}_{(t)}^T\mathbf{h}(\mathbf{x}_i)|0,1)}{1-\Phi(-\widehat{\beta}_{(t)}^T\mathbf{h}(x_i))} & \text{if } y_i = 1 \\ \widehat{\beta}_{(t)}^T\mathbf{h}(\mathbf{x}_i) - \frac{\mathcal{N}(\widehat{\beta}_{(t)}^T\mathbf{h}(\mathbf{x}_i)|0,1)}{\Phi(-\widehat{\beta}_{(t)}^T\mathbf{h}(\mathbf{x}_i))} & \text{if } y_i = -1. \end{cases} \tag{20}$$

These expressions are easily derived after noticing that z_i is (conditionally) Gaussian with mean $\widehat{\beta}_{(t)}^T\mathbf{h}(\mathbf{x}_i)$, but left-truncated at zero if $y_i = 1$, and right-truncated at zero if $y_i = -1$. With $s_{(t)} \equiv [s_{1,(t)}, \ldots, s_{n,(t)}]^T$, the M-step is similar to the regression case:

$$\widehat{\beta}_{(t+1)} = \mathbf{U}_{(t)}(\mathbf{I} + \mathbf{U}_{(t)}\mathbf{H}^T\mathbf{H}\mathbf{U}_{(t)})^{-1}\mathbf{U}_{(t)}\mathbf{H}^T\mathbf{s}_{(t)}, \tag{21}$$

with $s_{(t)}$ playing the role of observed data.

Summarizing, for the classification case, the E-step corresponds to (16) and (20), while the M-step is carried out by (21).

TABLE 1
Relative (%) Improvement in Modeling Error of Several Methods

Method	$\boldsymbol{\beta}_a$	$\boldsymbol{\beta}_b$
Proposed method	28%	74%
LASSO (CV)	13%	69%
LASSO (GCV)	30%	65%
Subset selection	13%	77%

5 Experimental Results

5.1 Regression

5.1.1 *Variable Selection in Linear Regression*

Our first example illustrates the use of the proposed method for variable selection in standard linear regression; that is, when the regression equation is $y = \beta^T\mathbf{x}$.

We consider a sequence of 20 β vectors, all 20-dimensional, with an increasing number (from 1 to 20) of nonzero components. Specifically, $\beta^{(1)} = [3, 0, 0, \ldots, 0]$, $\beta^{(2)} = [3, 3, 0, \ldots, 0]$, ..., $\beta^{(20)} = [3, 3, 3, \ldots, 3]$. For each $\beta^{(i)}$, $i = 1, \ldots, 20$, we obtain 100 random (50×20) design matrices, following the procedure in [16], and for each of these, we obtain 50 data-points with unit noise variance. Finally, for each $\beta^{(i)}$, we obtain the 100 estimates of $\beta^{(i)}$ (one for each design matrix). Fig. 3 shows the mean number of estimated nonzero components, as a function of the true number. Our method exhibits a very good ability to find the correct number of nonzero components in β, in an adaptive manner.

We next consider two of the experimental conditions that were studied in [16]: $\beta^{(a)} = [3, 1.5, 0, 0, 2, 0, 0, 0]$, with $\sigma = 3$, and $\beta^{(b)} = [5, 0, 0, 0, 0, 0, 0, 0]$, with $\sigma = 2$. In both cases, the (20×8) design matrices are generated as described in [16]. We have compared the relative modeling error (defined as $ME = E[\|\mathbf{H}\widehat{\beta} - \mathbf{H}\beta\|^2]$) improvement (with respect to ordinary least squares) of our solution against several methods studied in [16]: LASSO with α adjusted by *cross-validation* (CV) and *generalized crossvalidation* (GCV), and subset selection (also based on crossvalidation; see [16] for details). As shown in Table 1, our method performs comparably with the best method for each case, although it involves no tuning or adjustment of parameters, and is computationally simpler and faster.

5.1.2 *Kernel Regression*

We now study the performance of our method in kernel regression; that is, with

$$y = f(\mathbf{x}, \beta) = \beta_0 + \sum_{i=1}^{n}\beta_i\; K(\mathbf{x}, \mathbf{x}_i),$$

where $K(\mathbf{x}, \mathbf{x}_i)$ is a kernel function. This type of representation is used in *support vector machine* (SVM) regression [4] and in *relevance vector machine* (RVM) regression [22], [23]. In all the examples, we use Gaussian kernels, i.e.,

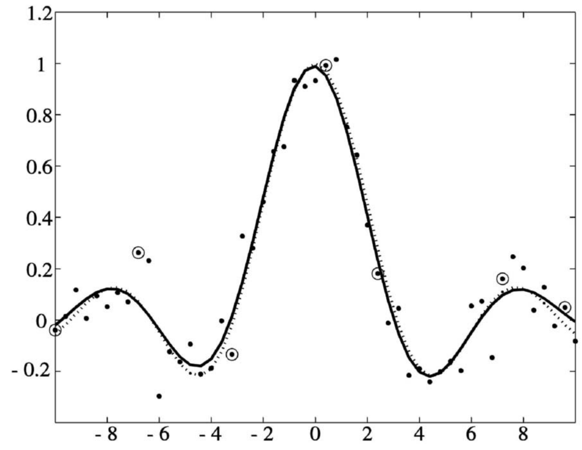

Fig. 4. Kernel regression example; dotted line true function $y = \sin(x)/x$. Dots: 50 noisy observations ($\sigma = 0.1$). Solid line: the estimated function. Circles: data points corresponding to the kernels with nonzero weight, the "support points."

$$K(\mathbf{x}, \mathbf{x}_i) = \exp\left\{ -\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{2\,\delta^2} \right\}, \tag{22}$$

where δ is a parameter which controls the kernel width.

We begin by considering the synthetic example studied in [22] and [23], where the true function is $y = \sin(x)/x$ (see Fig. 4). To compare our results to the RVM and the variational RVM (VRVM), we ran the algorithm on 25 generations of the noisy data. The results are summarized in Table 2 (which also includes the SVM results reported in [22]). Finally, we have also applied our method to the well-known *Boston housing* data set (using 20 random partitions of the full data set into 481 training samples and 25 test samples); Table 2 shows the results, again, versus SVM, RVM, and VRVM regression (as reported in [22]). In

TABLE 2
Mean Root Squared Errors and Mean Number of Kernels for the "$\sin(x)/x$" Function and the Boston Housing Examples

"$\sin(x)/x$" function

Method	MSE	No. kernels
New method	0.0455	7.0
SVM	0.0519	28.0
RVM	0.0494	6.9
VRVM	0.0494	7.4

Boston housing

Method	MSE	No. kernels
New method	9.98	45.2
SVM	10.29	235.2
RVM	10.17	41.1
VRVM	10.36	40.9

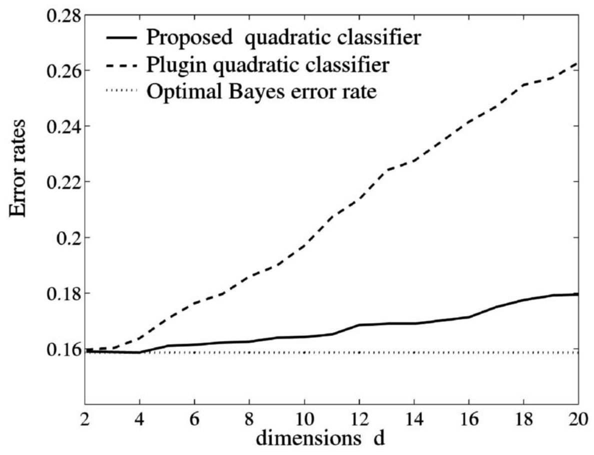

Fig. 5. Average (90 repetitions) test error rates of quadratic classifiers versus dimensionality.

these tests, our method performs better than RVM, VRVM, and SVM regression, although it does not require any adjustment of a sparseness-related parameter; of course, the results depend on the choice of kernel width δ (as do the RVM and SVM results).

5.2 Classification

5.2.1 Feature Selection for Linear and Quadratic Classifiers

For linear, 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, both with covariance matrices equal to identity, and means

$$\mu_1 = [\,\underbrace{1/\sqrt{2},\, 1/\sqrt{2},\, \overbrace{0, 0, \ldots, 0}^{d-2 \text{ zeros}}}_{d \text{ dimensions}}\,]^T$$

and $\mu_2 = -\mu_1$. The optimal Bayes error rate is given by $\Phi(-1|0,1) \simeq 0.1587$, independently of d. 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 3.1) include all the d components, their squares, and all the pair-wise products, in 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 1,000 (500 per class). Fig. 5 shows the resulting average (over 90 repetitions) test set error rate as a function of d. Fig. 6 reports a similar experiment, now with 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, when compared with the common plug-in classifier.

5.2.2 Kernel Classifiers

The final experiments address kernel-based classifiers. As in the regression case, we use Gaussian kernels (see (22)). Our

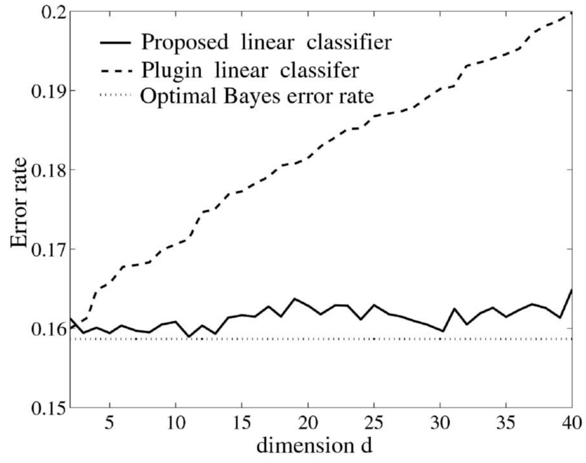

Fig. 6. Average (90 repetitions) test error rates of linear classifiers versus dimensionality.

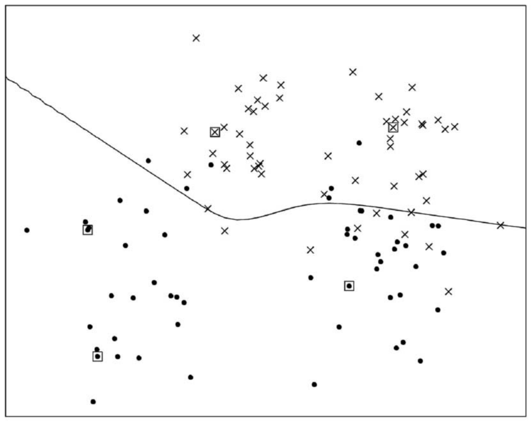

Fig. 7. Classification boundary for Ripley's data. The data points corresponding to the selected kernels are marked by squares.

first experiment uses Ripley's synthetic data set,[1] in which each class is a bimodal mixture of two Gaussians; the optimal error rate for this problem is 0.08 [6]. Fig. 7 shows 100 points from the training set and the resulting classification boundary learned by our algorithm. The five training samples needed to support the selected kernels (like *support vectors* in SVM) are marked by small squares. Table 3 shows the average test set error (and the final number 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 [22] 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 $\delta = 0.5$, which is the value used in [22].

For additional tests, we used three well-known benchmark real-data problems: the *Pima indians diabetes*,[2] where the goal is to decide whether a subject has diabetes or not, based on seven measured variables; the *Leptograpsus crabs*[3] where the problem consists in determinng the sex of crabs, based on five geometric measurements; and the *Wisconsin breast cancer*[4] (*WBC*), where the tast 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 [31]). 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 $\delta = 4$, for the *Pima* and *crabs* problems, and to $\delta = 12$ for the WBC. Table 4 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 support vectors (more than half the training set) selected by the SVM (reported in [22]) on the *Pima* data set.

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 binary classifiers to multiclass problems, but we will not focus on this issue here. To test the performance of our

TABLE 3
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

TABLE 4
Numbers of Test Errors on Three Data Sets Studied

Method	Pima	Crabs	WBC
Proposed method	62	0	8.5
SVM [31]	64	4	9
RVM [22]	65	N/A	N/A
VRVM [22]	65	N/A	N/A
Neural network [12]	75	3	N/A
Logistic regression [12]	66	4	N/A
Linear discriminant [31]	67	3	19
Gaussian process [31] [12]	67, 68	3	8

1. Available at www.stats.ox.ac.uk/pub/PRNN/.
2. Available at www.stats.ox.ac.uk/pub/PRNN/.
3. Available at www.stats.ox.ac.uk/pub/PRNN/.
4. Available at www.ics.uci.edu/~mlearn/MLSummary.html.

TABLE 5
Test Error on the "Forensinc Glass" Data Estimated by 10-Fold Crossvalidation

Method	% error
Proposed method	21.5
Neural network [12]	23.8
Gaussian mixture [12]	30.8
Gaussian process [12]	23.3

method on a multiclass problem, we used the *forensic glass* data set,[5] which is a 6-class problem with nine features. Following [12], the classification error rate was estimated using 10-fold cross-validation. The results in Table 5 show that our method outperforms the best method referred in [12], a Gaussian process (GP) classifier implemented by MCMC. The GP-MCMC classifier requires about 24 hours of computer time [12], while ours is learned in a few seconds.

6 Concluding Remarks

We have introduced a new sparseness-inducing prior related to the Laplacian prior. Its key feature is the absence of hyperparameters to be adjusted or estimated, achieved through the use of a (parameter-free) Jeffreys' prior. Experiments with several publicly available benchmark data sets, both for regression and classification, have shown state-of-the-art performance. In particular, on the data sets considered, our approach outperforms support vector machines and Gaussian process classifiers, although it involves no tuning or adjusting of sparseness-controlling hyperparameters.

In its current form, the kernel version of our approach is not applicable to large-scale problems (that is, with large training sets, like in handwritten digit classification), which were thus not considered in this paper. This fact is due to the need (in the M-step) to solve a linear system of dimension $n+1$, (where n is the number of training points), whose computational requirements scale with the third power of n. This computational issue is a topic of current interest to researchers in kernel-based methods (e.g., [32]), and we also intend to focus on it. Another well-known limitation of kernel methods is the need to adjust the kernel parameter(s) (e.g., the Gaussian kernel width δ in (22)). Of course, the results of our method strongly depend on an adequate choice of these parameters, and our formulation does not contribute to the solution of this problem.

Acknowledgments

Earlier versions of this work were presented in [1] and [2]. This work was supported by the Foundation for Science and Technology, Portuguese Ministry of Science and Technology, under project POSI/33143/SRI/2000.

5. Available at www.stats.ox.ac.uk/pub/PRNN/.

References

[1] M. Figueiredo, "Adaptive Sparseness Using Jeffreys Prior," *Proc. Advances in Neural Information Processing Systems 14,* T. Dietterich, S. Becker, and Z. Ghahramani, eds. Cambridge, Mass.: MIT Press, pp. 697-704, 2002.

[2] M. Figueiredo and A.K. Jain, "Bayesian Learning of Sparse Classifiers," *Proc. IEEE Computer Soc. Conf. Computer Vision and Pattern Recognition,* vol. 1, pp. 35-41, 2001.

[3] Z. Chen and S. Haykin, "On Different Facets of Regularization Theory," *Neural Computation,* vol. 14, no. 12, pp. 2791-2846, 2002.

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

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

[6] B. Ripley, *Pattern Recognition and Neural Networks.* Cambridge, U.K.: Cambridge Univ. Press, 1996.

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

[8] M. Figueiredo, "On Gaussian Radial Basis Function Approximations: Interpretation, Extensions, and Learning Strategies," *Proc. Int'l Conf. Pattern Recognition,* vol. 2, pp. 618-621, 2000.

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

[10] C. Williams, "Prediction with Gaussian Processes: From Linear Regression to Linear Prediction and Beyond," *Learning in Graphical Models,* M.I. Jordan, ed., The Netherlands: Kluwer, pp. 599-621, 1998.

[11] A. Hoerl and R. Kennard, "Ridge Regression: Biased Estimation for Nonorthogonal Problems," *Technometrics,* vol. 12, pp. 55-67, 1970.

[12] C. Williams and D. Barber, "Bayesian Classification with Gaussian Priors," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 20, no. 12, pp. 1342-1351, Dec. 1998.

[13] G. Wahba, *Spline Models for Observational Data.* Soc. for Industrial and Applied Math. (SIAM), Philadelphia, 1990.

[14] T. Poggio and F. Girosi, "Networks for Approximation and Learning," *Proc. IEEE,* vol. 78, pp. 1481-1497, 1990.

[15] S. Chen, D. Donoho, and M. Saunders, "Atomic Decomposition by Basis Pursuit," *SIAM J. Scientific Computation,* vol. 20, no. 1, pp. 33-61, 1998.

[16] R. Tibshirani, "Regression Shrinkage and Selection via the Lasso," *J. Royal Statistical Soc. (B),* vol. 58, pp. 267-288, 1996.

[17] P. Williams, "Bayesian Regularization and Pruning Using a Laplace Prior," *Neural Computation,* vol. 7, pp. 117-143, 1995.

[18] K. Lange and J. Sinsheimer, "Normal/Independent Distributions and Their Applications in Robust Regression," *J. Computational and Graphical Statistics,* vol. 2, pp. 175-198, 1993.

[19] M. Figueiredo and R. Nowak, "Wavelet-Based Image Estimation: An Empirical Bayes Approach Using Jeffreys' Noninformative Prior," *IEEE Trans. Image Processing,* vol. 10, no. 9, pp. 1322-1331, 2001.

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

[21] D. MacKay, "Bayesian Non-Linear Modelling for the 1993 Energy Prediction Competition," *Maximum Entropy and Bayesian Methods,* G. Heidbreder, ed. Kluwer, pp. 221-234, 1996.

[22] C. Bishop and M. Tipping, "Variational Relevance Vector Machines," *Proc. 16th Conf. Uncertainty in Artificial Intelligence,* pp. 46-53, 2000.

[23] M. Tipping, "The Relevance Vector Machine," *Proc. Advances in Neural Information Processing Systems 12,* S. Solla, T. Leen, and K.-R. Müller, eds. Cambridge, Mass.: MIT Press, pp. 652-658, 2000.

[24] D. Donoho and I. Johnstone, "Ideal Spatial Adaptation via Wavelet Shrinkage," *Biometrika,* vol. 81, pp. 425-455, 1994.

[25] P. Moulin and J. Liu, "Analysis of Multiresolution Image Denoising Schemes Using Generalized-Gaussian and Complexity Priors," *IEEE Trans. Information Theory,* vol. 45, pp. 909-919, 1999.

[26] Y. Grandvalet, "Least Absolute Shrinkage is Equivalent to Quadratic Penalization," *Perspectives in Neural Computing,* L. Niklasson, M. Boden, and T. Ziemske, eds. Springer Verlag, pp. 201-206, 1998.

[27] M. Osborne, B. Presnell, and B. Turlach, "A New Approach to Variable Selection in Least Squares Problems," *IMA J. Numerical Analysis,* vol. 20, pp. 389-404, 2000.

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

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

[30] J. Albert and S. Chib, "Bayesian Analysis of Binary and Polychotomous Response Data," *J. Am. Statistical Assoc.*, vol. 88, pp. 669-679, 1993.

[31] M. Seeger, "Bayesian Model Selection for Support Vector Machines, Gaussian Processes, and Other Kernel Classifiers," *Proc. Advances in Neural Information Processing Systems 12,* S. Solla, T. Leen, and K.-R. Müller, eds. Cambridge, Mass.: MIT press, pp. 603-609, 2000.

[32] C. Williams and M. Seeger, "Using the Nystrom Method to Speedup Kernel Machines," *Proc. Advances in Neural Information Processing Systems 13,* T. Leen, T. Dietterich, and V. Tresp, eds. Cambridge, Mass.: MIT Press, pp. 682-688, 2001.

Mário A.T. Figueiredo (S'87-M'95-SM'2000) received the EE, MSc, and PhD degrees in electrical and computer engineering, all from the Instituto Superior Técnico (I.S.T., the engineering school of the Technical University of Lisbon), Lisbon, Portugal, in 1985, 1990, and 1994, respectively. Since 1994, he has been an assistant professor with the Department of Electrical and Computer Engineering, I.S.T. He is also with the Communication Theory and Pattern Recognition Group, Institute of Telecommunications, Lisbon. In 1998, he held a visiting position with the Department of Computer Science and Engineering at Michigan State University, East Lansing. His scientific interests include image processing and analysis, computer vision, statistical pattern recognition, and statistical learning. Dr. Figueiredo received the Portuguese IBM Scientific Prize in 1995. He is an associate editor of the journals *Pattern Recognition Letters*, *IEEE Transactions on Image Processing*, and *IEEE Transactions on Mobile Computing*, and guest coeditor of forthcoming special issues of the journals *IEEE Transactions on Pattern Analysis and Machine Intelligence* and *IEEE Transactions on Signal Processing*. He is a senior member of the IEEE and a member of the IEEE Computer Society.