Extended Linear Models with Gaussian Prior on the Parameters and Adaptive Expansion Vectors

Ignacio Barrio, Enrique Romero, and Lluís Belanche

Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya, Barcelona, Spain

Abstract. We present an approximate Bayesian method for regression and classification with models linear in the parameters. Similar to the Relevance Vector Machine (RVM), each parameter is associated with an expansion vector. Unlike the RVM, the number of expansion vectors is specified beforehand. We assume an overall Gaussian prior on the parameters and find, with a gradient based process, the expansion vectors that (locally) maximize the evidence. This approach has lower computational demands than the RVM, and has the advantage that the vectors do not necessarily belong to the training set. Therefore, in principle, better vectors can be found. Furthermore, other hyperparameters can be learned in the same smooth joint optimization. Experimental results show that the freedom of the expansion vectors to be located away from the training data causes overfitting problems. These problems are alleviated by including a hyperprior that penalizes expansion vectors located far away from the input data.

1 Introduction

In supervised learning, we are given a set of training data $\{x_i, t_i\}_{i=1}^N$ where $x_i \in \mathbf{R}^D$. We will consider regression tasks, where $t_i \in \mathbf{R}$, and binary classification tasks where $t_i \in \{+1, -1\}$. The objective is to infer a function $y(x)$ that underlies the training data and makes good predictions on unseen input vectors. It is common to express this function as an extended linear model with M fixed basis functions:

$$y(x; w) = w^T \phi(x), \tag{1}$$

where $\phi(x) = (\phi_1(x), \phi_2(x), .., \phi_M(x))^T$ are the outputs of the basis functions and $w = (\omega_1, \omega_2, .., \omega_M)^T$ are the model parameters. When each basis function is associated with a vector, we can write

$$\phi(x) = (k(x, \hat{x}_1), k(x, \hat{x}_2), .., k(x, \hat{x}_M))^T, \tag{2}$$

where $\hat{x}_i$ are the expansion vectors.

Within Bayesian learning, the Relevance Vector Machine (RVM) [1] considers a model where the expansion vectors correspond with all the training input vectors. An individual Gaussian prior distribution is assumed on each parameter, so that following approximate Bayesian inference leads to sparse solutions. The expansion vectors associated to non-zero parameters (known as relevance vectors) belong to the training set, and are found by marginal likelihood maximization. The η-RVM is an extension of the RVM that, besides selecting the relevance vectors, adapts the hyperparameters of the basis functions— *e.g.* the widths of Radial Basis Functions (RBF)— by marginal likelihood maximization. As Tipping stated, there is an important limitation in the η-RVM, since the learning stage requires an interleaved two-stage training procedure that leaves open the question of how to combine the optimization.

We present a method that finds a model with the form (1, 2) where the expansion vectors (the number of which is fixed beforehand) do not belong to the training set. We place an overall Gaussian prior on the parameters, and find the expansion vectors by marginal likelihood maximization with a gradient based process. We will refer to this method as *Linear Model with Adaptive expansion Vectors maximizing the Evidence* (LMAVE). One might argue that, since the expansion vectors are adaptive, the model is no longer linear. However, we use the term *linear model* because the expansion vectors are not parameters but hyperparameters. This work is inspired in previous works on sparsification of Support Vector Machines [2] and Gaussian processes [3]. These works do not require the expansion vectors to be part of the training data.

This manner of finding the expansion vectors has been suggested in [4,5], but it has not been studied in depth. However, it should include a number of enhancements with respect to the RVM, for example, the expansion vectors do not necessarily belong to the training data, so better estimates can be found. Furthermore, the expansion vectors and other hyperparameters are learned in one joint optimization. The computational demands are lower than for the RVM, which makes this method suitable for large data sets. However, experimental results show that, at least for RBFs, the freedom of the expansion vectors to be away from the training data can cause overfitting problems.

We propose a hyperprior that penalizes expansion vectors located far away from the input data and show that a substantial improvement is achieved. We refer to this method as *Linear Model with Adaptive expansion Vectors maximizing the Posterior* (LMAVP).

2 Linear Models with Overall Gaussian Prior

This section introduces inference with extended linear models where an overall Gaussian prior distribution is assumed on the parameters. For regression we follow previous work on Bayesian interpolation [6]. For classification we use a generalization of the linear model.

2.1 Regression

Consider the targets to be deviated from the real underlying function by *i.i.d.* additive zero-mean Gaussian noise with variance σ_ν^2. The likelihood of the parameters follows a Gaussian distribution: (to save notation, we omit the conditioning on the input vectors, x)

$$p(t|w, \sigma_\nu^2) \sim \mathcal{N}(\Phi w, \sigma_\nu^2), \tag{3}$$

where Φ is the $N \times M$ design matrix with elements $\Phi_{ij} = k(x_i, \hat{x}_j)$.

In order to control the smoothness of $y(x;w)$, we assume a zero-mean overall Gaussian prior distribution over w with variance σ_w^2:

$$p(w|\sigma_w^2) \sim \mathcal{N}(0, \sigma_w^2), \tag{4}$$

Following Bayes' rule, the posterior parameter distribution is written $p(w|t, \sigma_\nu^2, \sigma_w^2) = p(t|w, \sigma_\nu^2)p(w|\sigma_w^2)/p(t|\sigma_w^2, \sigma_\nu^2)$. This is a product of two Gaussians (the likelihood and the prior) divided by a normalizing constant (the marginal likelihood) and it can be rewritten as

$$p(w|t, \sigma_\nu^2, \sigma_w^2) \sim \mathcal{N}(\mu, \Sigma), \tag{5}$$

where

$$\Sigma = (\sigma_\nu^{-2}\Phi^T\Phi + \sigma_w^{-2}I)^{-1} \qquad \text{and} \qquad \mu = \sigma_\nu^{-2}\Sigma\Phi^T t. \tag{6}$$

The marginal likelihood (also known as the evidence) is given by $p(t|\sigma_w^2, \sigma_\nu^2) = \int p(t|w, \sigma_\nu^2)p(w|\sigma_w^2)dw$. This is a convolution of Gaussians, which is also a Gaussian:

$$p(t|\sigma_w^2, \sigma_\nu^2) = \mathcal{N}(0, C) = (2\pi)^{-N/2}|C|^{-1/2}\exp\Big(-\frac{1}{2}t^T C^{-1} t\Big), \tag{7}$$

where $C = \sigma_\nu^2 I + \sigma_w^2\Phi\Phi^T$. Since matrix C is sized $N \times N$, it is convenient to use established matrix identities and rewrite the logarithm of the marginal likelihood as

$$\begin{aligned}\log p(t|\sigma_w^2, \sigma_\nu^2) = -\frac{1}{2}\big[&N\log 2\pi - \log|\Sigma| + N\log\sigma_\nu^2 + \\ &+ M\log\sigma_w^2 + \sigma_\nu^{-2}\|t - \Phi\mu\|^2 + \sigma_w^{-2}\|\mu\|^2\big].\end{aligned}$$

It is common to include extra hyperparameters controlling some aspect of the basis functions (*e.g.* RBF widths). We will refer to all the hyperparameters (including also σ_w^2 and σ_ν^2) as Θ.

In order to estimate the most probable hyperparameters, Θ_{MP}, we can use Bayes' rule: $p(\Theta|t) \propto p(t|\Theta)p(\Theta)$. If a flat (improper) hyperprior $p(\Theta)$ is assumed for all the hyperparameters, then the marginal likelihood $p(t|\Theta)$ (7) should be maximized.

2.2 Binary Classification

In binary classification we model the probability of success $p(t_i = +1|w)$. We map $y(x;w)$ to the unit interval by applying the cumulative distribution function for a Gaussian (probit), $\Psi(z) = \int_{-\infty}^{z}\mathcal{N}(x|0,1)dx$. The likelihood is

$$p(t|w) = \prod_{i=1}^{N} p(t_i|w^T\phi(x_i)) = \prod_{i=1}^{N}\Psi(t_i w^T\phi(x_i)). \tag{8}$$

We use an overall Gaussian prior for the parameters (4). Since the likelihood is not Gaussian, unlike the regression case, we cannot arrive at analytical solutions for the posterior. Instead, we approximate the posterior distribution as[1]

$$p(w|t, \sigma_w^2) \propto p(w|\sigma_w^2)\prod_i p(t_i|w) \approx p(w|\sigma_w^2)\prod_i q(t_i|w). \tag{9}$$

[1] We use p to denote exact quantities and q to denote approximations.

We follow Expectation Propagation (EP) [7] and choose the approximation terms to be Gaussian, parameterized by (m_i, v_i, s_i): $q(t_i|w) = s_i \exp(-\frac{1}{2v_i}(t_i w^T \phi(x_i) - m_i)^2)$. Then the approximate posterior is also Gaussian: $q(w|t, \sigma_w^2) = \mathcal{N}(m_w, V_w)$.

EP chooses the approximation terms such that the posterior using the exact terms and the posterior using the approximation terms are close in Kullback-Leibler (KL) divergence. Qi *et al.* [8] presented an iterative algorithm to choose the approximation terms following EP in $O(INM^2)$ complexity (I is the number of iterations). The algorithm was designed for the RVM, but it is also applicable to linear models with an overall Gaussian prior. It is omitted here for brevity.

Once the approximation terms are found, the marginal likelihood can be approximated by:

$$p(t|\Theta) \approx q(t|\Theta) = \int \prod_i p(w|\sigma_w^2) q(t_i|w) dw = |V_w|^{1/2} \sigma_w^{-M} \exp(B/2) \prod_i s_i \quad (10)$$

where $B = (m_w)^T V_w^{-1} m_w - \sum_i (m_i^2/v_i)$.

3 Linear Models with Adaptive Expansion Vectors

The Linear Model with Adaptive expansion Vectors (LMAV) can be seen as a linear model with an overall Gaussian prior (section 2) where the expansion vectors are considered as hyperparameters. The number of expansion vectors is fixed beforehand. In order to optimize the expansion vectors and the rest of hyperparameters, the LMAV maximizing the evidence (LMAVE) uses a gradient based algorithm that maximizes the evidence (7, 10).

3.1 Derivatives with Respect to the Expansion Vectors

Regression Tasks. For regression tasks, we can decompose the computation of the derivatives of the marginal likelihood (7) with respect to the expansion vectors into two steps [1]. For convenience we compute the derivative of the logarithm $\mathcal{L} = \log p(t|\Theta)$:

$$\frac{\partial \mathcal{L}}{\partial \hat{x}_{ij}} = \sum_{n=1}^{N} \sum_{m=1}^{M} \frac{\partial \mathcal{L}}{\partial \phi_{nm}} \frac{\partial \phi_{nm}}{\partial \hat{x}_{ij}} = \sum_{n=1}^{N} \sum_{m=1}^{M} A_{nm} \frac{\partial \phi_{nm}}{\partial \hat{x}_{ij}},$$

where $\hat{x}_i = (\hat{x}_{i1}, \hat{x}_{i2}, .., \hat{x}_{iD})$, $A = \sigma_\nu^{-2}[(t - \Phi\mu)\mu^T - \Phi\Sigma]$ and $\phi_{nm} = k(x_n, \hat{x}_m)$.

Classification Tasks. For classification tasks, the approximation terms $q(t_i|w)$ have the same form as a likelihood term in a regression problem (3), that is, we convert the classification problem into a regression problem with targets m_i and input dependent (heteroscedastic) noise with variances v_i. Following this interpretation, the covariance matrix and the mean for the posterior $q(w|t, \sigma_w)$ can be rewritten as

$$V_w = (\sigma_w^{-2} I + H^T \Lambda^{-1} H)^{-1} \text{ and } m_w = V_w H^T \Lambda^{-1} m_o, \quad (11)$$

where $m_o = (m_1, .., m_N)^T$, $\Lambda = \text{diag}(v_1, .., v_N)^T$ and H is an $N \times M$ matrix with elements $H_{ij} = t_i k(x_i, \hat{x}_j)$.

We can use (11) to compute the derivatives of the marginal likelihood (10) with respect to the hyperparameters. For the gradient computation, the parameters m_i, v_i, s_i can be considered fixed [9] and we can write: (notice $\mathcal{L} = \log q(t|\Theta)$)

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \hat{x}_{ij}} = & -\frac{1}{2} tr(V_w E \frac{\partial H}{\partial \hat{x}_{ij}}) + \frac{1}{2} m_w^T E \frac{\partial H}{\partial \hat{x}_{ij}} m_w - \\ & - \frac{1}{2} m_w^T E \frac{\partial H}{\partial \hat{x}_{ij}} V_w E m_o + m_w^T \frac{\partial H^T}{\partial \hat{x}_{ij}} \Lambda^{-1} m_o, \end{aligned} \tag{12}$$

where $E = 2H^T \Lambda^{-1}$.

3.2 Computational Cost

In regression tasks, computing the gradient requires the matrix multiplication $\Phi\Sigma$ in the calculation of matrix A, which has complexity $O(M^2 \times N)$. In addition, the optimization of M expansion vectors has $M \times D$ dimensions. However, the matrix $\partial\Phi/\partial\hat{x}_{ij}$ only has one non-zero column, which lowers the cost of some computations. The cost of LMAVE for regression is $O(n \times (N \times M \times D + M^2 \times N))$, where n is the number of epochs of the gradient based algorithm.

In classification tasks, each epoch requires the EP approximation and the computation of the gradient. Computing the gradient requires the matrix multiplication $V_w E$ in (12), which has complexity $O(M^2 \times N)$. This complexity is lower than the EP approximation $O(I \times M^2 \times N)$, where I is the number of iterations of the EP algorithm (usually less than 10, so it can be considered as a constant). The optimization of M expansion vectors has $M \times D$ dimensions, but again the matrix $\partial H/\partial\hat{x}_{ij}$ only has one non-zero column. The cost of LMAVE for binary classification is $O(n \times (N \times M \times D + I \times M^2 \times N))$. We have not included the cost of optimizing additional hyperparameters.

3.3 Introducing a Non-flat Hyperprior

The plot on the left in Figure 1 shows an illustrative example of LMAVE with ten expansion vectors. The data are shown with dots. The initial and the final locations of the expansion vectors are marked with crosses (lower and upper row respectively). The function produced by the LMAVE is also shown. We trained the LMAVE with RBFs $k(x, \hat{x}_i) = \exp(-\gamma \sum_{j=1}^{D} (x_j - \hat{x}_{ij})^2)$ with adaptive inverse squared width, γ. There is an input range where no data are provided. We find that an expansion vector is located precisely in that range and the function values become large near that expansion vector. A similar effect happens with the expansion vector located on the right of the data.

Where the data are scarce, the prior assumptions should take high importance. The Gaussian prior assumption on the parameters favours small function values, but the freedom of the expansion vectors to be located anywhere can produce large function values. That is, the prior assumption on the parameters is not enough to control the smoothness.

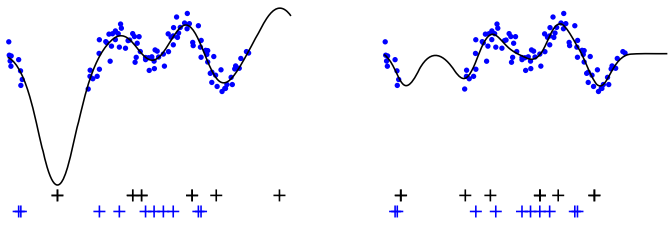
Fig. 1. Comparison between LMAVE (left) and LMAVP (right). See text.

The LMAVE assumes a flat hyperprior on the expansion vectors. To tackle situations like the one in Figure 1 we can include a hyperprior that penalizes expansion vectors located far away from the input data. Taking advantage of $k(x, \hat{x}_i) \in [0, 1]$ being bounded and localized at the expansion vectors, we propose the following hyperprior (note that $k(x, \hat{x}_i)$ depends on γ):

$$p(\hat{x}_i, \gamma) \propto \exp\Big(- \frac{1}{2} \frac{(1 - \max_{j=1}^{N}(k(x_j, \hat{x}_i)))^2}{s^2} \Big). \tag{13}$$

It is exclusively designed for the RBFs described above. Note that we have saved notation and this hyperprior should be written $p(\hat{x}_i, \gamma | \{x_j\}_{j=1}^{N}, k(\cdot, \cdot))$. This hyperprior satisfies a tradeoff[2] between flexibility on the expansion vectors to model the data and constraint to be located near the input data (the concept 'near' depends on the RBF width). Throughout this work we set $s = 0.1$. This value was chosen after some preliminary experiments.

In order to find maximum a posteriori estimates of the hyperparameters we maximize $\log p(t|\Theta) + \sum_{i=1}^{M} \log p(\hat{x}_i, \gamma)$. The complexity of computing the gradient of the hyperpriors is negligible when compared to the gradient of the evidence, therefore the cost is the same as LMAVE. We call this method *LMAV maximizing the posterior* (LMAVP). The plot on the right in Figure 1 shows the performance of LMAVP on the same data set. The expansion vectors are located near the training data and large function values are avoided.

4 Comparison with Related Methods

4.1 Relevance Vector Machines

The RVM [1] considers a linear model with one expansion vector coincident with each input vector. Unlike the LMAV, an individual Gaussian prior is assumed for each parameter, and following approximate Bayesian inference, most of the parameters tend to zero, so in practice the model can be expressed as an expansion of a subset of input vectors. Unlike the LMAV, the size of the model adapts to the problem at hand. This can be an advantage, but it can become a drawback for applications very restrictive in prediction time and memory, where the level of sparsity may be not enough.

[2] The term s^2 in the denominator controls this tradeoff.

Unlike the RVM, the expansion vectors in LMAV do not necessarily belong to the training data, so better estimates can be found —but if they are located far away, problems can appear (see Figure 1). Furthermore, the optimization of expansion vectors and other hyperparameters can be performed in a smooth joint optimization.

RVM implementations [10,11,12] either store the entire $N \times N$ design matrix or recompute all its columns at each iteration, while the LMAV recomputes an $N \times M$ design matrix at each iteration. Furthermore, the cost of an iteration of LMAV depends only linearly on the number of training data (it depends on other variables, see section 3.2), while this dependence is at least quadratic in RVM implementations. This makes the LMAV especially interesting for large data sets (and low number of expansion vectors).

Unlike the RVM, the LMAV requires the basis functions to be differentiable with respect to the expansion vectors. Furthermore, a bad initialization of the expansion vectors may lead to a bad local maximum.

4.2 Other Related Methods

Sparse large margin classifiers (SLMC) [2] are a sparse alternative to Support Vector Machines (SVMs) [13] for classification. A gradient based algorithm learns the locations of the expansion vectors (the number of which is set beforehand) so that the margin in some feature space is maximized. SLMC makes use of kernels, which allows to work in infinite feature spaces. Unlike SLMC, the LMAV does not use kernels, and it is limited to finite feature spaces. This limitation sometimes turns into an advantage, since the LMAV can use basis functions that do not satisfy Mercer's condition (required to be kernel). Furthermore, the Bayesian framework allows to tune other hyperparameters, such as RBF widths, that are not easily tuned for SLMC.

A Sparse Pseudo-input Gaussian process (SPGP) [3] is a Gaussian process (GP) with a particular covariance function parameterized by the pseudo-inputs (the counterpart to the expansion vectors). This covariance defines a prior over functions. Following approximate Bayesian inference, the pseudo-inputs are optimized by marginal likelihood maximization. The pseudo-inputs and the hyperparameters are found in a gradient based smooth joint optimization. The LMAV is closely related to SPGP, but it is based on a different prior over functions (a different covariance matrix C in (7)), with the advantage that the basis functions $k(x, \hat{x}_i)$ do not need to be covariance functions.

Much work in the literature has been devoted to the location of RBF centers. *Generalized radial basis functions* (GRBF) [14] find the centers with a gradient based algorithm minimizing some cost function based on regularization techniques. In [15] not only the centers are found with this method but also the RBF widths. The authors report overfitting problems. Similarly, the hidden parameters in a feed-forward neural network are commonly trained with gradient based algorithms [16]. LMAV can be considered as a sort of neural network. However, the main difference between both methods is that in neural networks all the parameters (output and hidden ones) are usually trained to minimize the same cost function, whereas in LMAV the expansion vectors (the counterpart to hidden layer parameters) are found by integrating out the (output) parameters. This should bring an improvement over maximum likelihood approaches.

Table 1. Results on regression benchmarks. The test MSE is shown for each method.

Data set		CPU	bank-32nh	pumadyn-32nh	kin-40k	kin-40k	
N		1024	1024	1024	2000	30000	
D		21	32	32	8	8	
η-RVM	$M/N(\%)$	1.8	40.7	2.5	12.6		-
	Error (MSE)	0.090	1.707	0.602	0.0043		-
$M/N = 2\%$	SPGP	0.080	1.262	0.586	0.0071	$M = 50$	0.0061
	LMAVE	0.110	2.570	0.704	0.0054		0.0043
	LMAVP	0.093	1.458	0.627	0.0054		0.0045
$M/N = 5\%$	SPGP	0.083	1.259	0.597	0.0054	$M = 100$	0.0039
	LMAVE	0.086	3.944	0.846	0.0043		0.0030
	LMAVP	0.077	1.861	0.642	0.0041		0.0029
$M/N = 10\%$	SPGP	0.079	1.264	0.616	0.0045	$M = 200$	0.0033
	LMAVE	0.088	3.032	0.987	0.0040		0.0015
	LMAVP	0.072	2.058	0.636	0.0031		0.0016

5 Experimental Study

In this section we compared the LMAVE and LMAVP to some related methods. Conjugate gradients was used to optimize all the hyperparameters. We used $n = 200$ epochs.

5.1 Regression Benchmarks

We used four benchmark data sets.[3] To check performance, for the *kin-40k* task we used 20 disjoint sets with 2000 cases each. Ten sets were used for training and ten for test. For each of the other tasks we used eight disjoint sets with 1024 cases each (four sets for training and four for test). To show the utility of LMAV on large data sets, we also split the *kin-40k* data set into 30000 training cases and 10000 test cases.

We used RBFs with multiple adaptive widths $k(x, \hat{x}_i) = \exp(-\sum_{j=1}^{D} \gamma_j (x_j - \hat{x}_{ij})^2)$. We compared the LMAVE and LMAVP to SPGP with covariance function $K(x_1, x_2) = \theta_0 k(x_1, x_2)$ and to η-RVM (RVM with adaptive RBF widths). The number of expansion vectors for LMAV and SPGP was 2%, 5% and 10% of the number of training data (except for the large data set). For SPGP we used the code provided by E. Snelson at www.gatsby.ucl.ac.uk/$\sim$snelson. For the η-RVM we followed the implementation proposed by Tipping [1].

Table 1 shows the results. For each task, the number of training data, N, and the input dimension, D, are shown. The mean squared error (MSE) is reported for each method. LMAVP was competitive with SPGP and the RVM, and it outperformed the LMAVE. The hyperprior alleviated overfitting problems in most cases but it was not strong enough for the *bank-32nh* problem. In the large data set, the RVM is hopeless, since it requires too much storage and computation, while the LMAV required from 16 minutes (with $M = 50$) to 110 minutes (with $M = 200$) on a Pentium IV 3.0 GHz.

[3] The *kin-40k* data set is available at http://ida.first.fraunhofer.de/$\sim$anton/data.html. The rest of data sets are available at http://www.cs.toronto.edu/$\sim$delve

Table 2. Results on classification tasks. The test error rates are shown for each method.

Data set		Banana	Breast	Titanic	Waveform	Image
N		400	200	150	400	1300
D		2	9	3	21	18
SVM	N_{SV}	86.7	112.8	70.6	158.9	172.1
	Error(%)	11.8	28.6	22.1	9.9	2.8
$M/N_{SV} = 5\%$	SLMC	16.5	27.9	26.4	9.9	5.2
	LMAVE	26.3	30.8	22.6	12.8	2.9
	LMAVP	26.1	29.9	22.7	12.5	3.5
$M/N_{SV} = 10\%$	SLMC	11.0	27.9	22.4	9.9	3.6
	LMAVE	11.5	29.5	22.7	12.5	2.6
	LMAVP	11.0	28.7	22.8	12.1	3.0
RVM	$M/N_{SV}(\%)$	13.2	5.6	92.5	9.2	20.1
	Error(%)	10.8	29.9	23.0	10.9	3.9

5.2 Classification Benchmarks

Five classification tasks available at http://ida.first.fraunhofer.de/projects/bench were used. These data sets were split in 100 training/test partitions. Similar to [1,2] the results reported in this work show averages over the first 10 partitions of each data set.

In this experiment we used RBFs with a single width $k(x, \hat{x}_i) = \exp(-\gamma \sum_{j=1}^{D} (x_j - \hat{x}_{ij})^2)$. We compared the LMAVE and LMAVP to the RVM, SLMC and SVM. The number of expansion vectors for SLMC, LMAVE and LMAVP is 5% and 10% of the number of support vectors found by the SVM.

Results are shown in Table 2. The results for RVM are taken from [1], where cross-validation was used to find the RBF width. The results for SVM and SLMC are taken from [2], where the regularization parameter and the RBF width were the ones provided by G. Rätsch at the aforementioned website. For LMAVE and LMAVP, the RBF width was jointly optimized with the expansion vectors (see section 3).

The LMAVE and LMAVP were competitive both with the RVM and SLMC. The difference in accuracy between LMAVP and LMAVE was not as notable as in regression tasks.

6 Conclusions

Experimental results show that the freedom of the expansion vectors to be located away from the training data can cause overfitting problems to the LMAVE. Since the prior on the parameters is not enough to tackle overfitting, we have included a particular hyperprior (designed for radial basis functions) penalizing expansion vectors located far away from the input data. This approach (LMAVP) results in a significant improvement over the LMAVE, especially in regression tasks, and is competitive with the RVM. The number of expansion vectors required to achieve good performance is usually very low. This fact makes this method very interesting for large data sets, since the memory and computational demands are lower than for the RVM. A study on different hyperpriors may be potentially interesting (also for non-radial basis functions).

In this work, the expansion vectors are initialized to random training input vectors, but other scenarios are possible, for example we could use a clustering algorithm such as K-means.

The (Gaussian) noise and (probit) 'slack' assumptions taken can be rather stringent, and the presence of outliers can decrease performance. Furthermore, the optimization of hyperparameters based on the evidence (LMAVE) and on maximum a posteriori estimates (LMAVP) are just approximations to full Bayesian inference, which consists on integrating out all the hyperparameters. As such, they are not inmune to overfitting.

Acknowledgments

The authors thank Joaquín Quiñonero Candela for providing the code for the η-RVM. This work was supported by the Consejo Interministerial de Ciencia y Tecnología (CICYT), under projects CGL2004-04702-C02-02 and TIN2006-08114.

References

1. Tipping, M.: Sparse Bayesian learning and the relevance vector machine. Journal of Machine Learning Research 1, 211–244 (2001)
2. Wu, M., Schölkopf, B., Bakir, G.: Building sparse large margin classifiers. In: 22nd International Conf. on Machine learning, pp. 996–1003. ACM Press, New York (2005)
3. Snelson, E., Ghahramani, Z.: Sparse Gaussian Processes using Pseudo-inputs. Advances in Neural Information Processing Systems 18, 1257–1264 (2006)
4. Minka, T.: Bayesian linear regression (1998), note obtainable from `http://research.microsoft.com/~minka/papers/`
5. Quiñonero Candela, J., Rasmussen, C.E.: A Unifying View of Sparse Approximate Gaussian Process Regression. Journal of Machine Learning Research 6, 1935–1959 (2005)
6. MacKay, D.J.C.: Bayesian Interpolation. Neural Computation 4, 415–447 (1992)
7. Minka, T.P.: Expectation Propagation for approximate Bayesian inference. In: 17th Conference in Uncertainty in Artificial Intelligence, pp. 362–369 (2001)
8. Qi, Y.A., Minka, T.P., Picard, R.W., Ghahramani, Z.: Predictive automatic relevance determination by expectation propagation. In: 21st International Conference on Machine Learning (2004)
9. Seeger, M.: Expectation propagation for exponential families (2005), note obtainable from `www.kyb.tuebingen.mpg.de/~seeger/`
10. Quiñonero Candela, J., Winther, O.: Incremental Gaussian processes. Advances in Neural Information Processing Systems 15, 1001–1008 (2003)
11. Tipping, M., Faul, A.: Fast Marginal Likelihood Maximisation for Sparse Bayesian Models. In: Ninth International Workshop on Artificial Intelligence and Statistics (2003)
12. D'Souza, A., Vijayakumar, S., Schaal, S.: The Bayesian backfitting relevance vector machine. In: 21st International Conference on Machine Learning (2004)
13. Vapnik, V.: The Nature of Statistical Learning Theory. Springer, Heidelberg (1995)
14. Poggio, T., Girosi, F.: A theory of networks for approximation and learning. Technical Report AI-1140, MIT Artificial Intelligence Laboratory, Cambridge, MA (1989)
15. Wettschereck, D., Dietterich, T.G.: Improving the performance of radial basis function networks by learning center locations. Advances in Neural Information Processing Systems 4, 1133–1140 (1992)
16. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press Inc., New York (1995)