Local Feature Selection for the Relevance Vector Machine Using Adaptive Kernel Learning

Dimitris Tzikas[1], Aristidis Likas[1], and Nikolaos Galatsanos[2]

[1] Department of Computer Science, University of Ioannina, Ioannina, Greece, 45110
{tzikas,arly}@cs.uoi.gr
[2] Department of Electrical Engineering, University of Patras, Rio, Greece, 26500
ngalatsanos@upatras.gr

Abstract. A Bayesian learning algorithm is presented that is based on a sparse Bayesian linear model (the Relevance Vector Machine (RVM)) and learns the parameters of the kernels during model training. The novel characteristic of the method is that it enables the introduction of parameters called 'scaling factors' that measure the significance of each feature. Using the Bayesian framework, a sparsity promoting prior is then imposed on the scaling factors in order to eliminate irrelevant features. Feature selection is local, because different values are estimated for the scaling factors of each kernel, therefore different features are considered significant at different regions of the input space. We present experimental results on artificial data to demonstrate the advantages of the proposed model and then we evaluate our method on several commonly used regression and classification datasets.

1 Introduction

In supervised learning we are given a training set $\{\boldsymbol{x}_n, t_n\}_{n=1}^N$, so that t_n is a noisy measurement of the output of a function y when its input is $\boldsymbol{x}_n$. Then, we wish to predict the output $y(\boldsymbol{x})$ of the function at any arbitrary test point $\boldsymbol{x}$. In regression the outputs t_n are continuous and they usually contain additive noise ϵ_n:

$$t_n = y(\boldsymbol{x}_n) + \epsilon_n, \tag{1}$$

while in classification the outputs t_n are discrete and assuming K classes they can be coded so that $t_{nk} = 1$ if x_n belongs to class k, otherwise $t_{nk} = 0$.

In order to make predictions a specific parametric form may be assumed for the unknown function y,such as a linear model:

$$y(\boldsymbol{x}|\boldsymbol{w}) = \sum_{i=1}^{M} w_i \phi_i(\boldsymbol{x}), \tag{2}$$

where $\boldsymbol{w} = (w_1, \ldots, w_M)^T$ are the weights of the linear model and $\{\phi_i(\boldsymbol{x})\}_{i=1}^M$ is the set of basis functions, which are assumed fixed and must be selected a priori. The sparse Bayesian linear model [2] is a Bayesian treatment of the linear model, which assumes a Gaussian prior with separate precision variable α_i for each weight w_i

$$p(\boldsymbol{w}|\boldsymbol{\alpha}) = \prod_{i=1}^{M} \mathrm{N}(w_i|0, \alpha_i^{-1}), \tag{3}$$

where $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_M)^T$. Using this prior, many of the available basis functions are pruned during learning. Because of this, we can design linear models that initially assume a large number of basis functions and the final estimation will use only a small number of the available basis functions. For example, the relevance vector machine (RVM) is a specific instance of the sparse Bayesian linear model, which assumes that the basis functions are kernels and one kernel is placed at each training example.

An adaptive kernel learning methodology [1] has been recently proposed that automatically learns parameters of the basis functions of a sparse linear model. More specifically, it assumes that *each basis function has different parameters*, and in principle it can even have different parametric form, therefore it is very flexible. In order to avoid overfitting a sparsity enforcing prior has been used for the weight precisions $\boldsymbol{\alpha}$ that directly *controls the number of effective parameters* of the model [3]. Learning is then achieved using an algorithm that is similar to the incremental RVM algorithm[4]. It starts with an empty model and at each iteration it adds to the model an appropriate basis function, in order to maximize the marginal likelihood of the model. In the typical incremental RVM [4] selecting a basis function is achieved using discrete optimization over the location of the basis functions; all candidate basis functions are tested for addition to the model. In contrast, the adaptive kernel learning methodology [1] uses *continuous optimization* with respect to the parameters (such as location and scale) of the basis functions. This methodology has been applied to learn the center (mean) and width (variance) parameters of Gaussian kernel basis functions.

In supervised learning problems, feature selection is typically performed as a preprocessing step, which is performed before building a classification or regression model. The general idea is to eliminate irrelevant features in the training set, in order to improve the generalization performance of the model and simultaneously reduce the computational cost of its training. However, it is possible to design supervised learning models that incorporate feature selection mechanisms, in order to perform feature selection simultaneously with estimation of model parameters. These models need to consider all the available features for training and, for this reason, they have relatively high computational cost. However, they can achieve better peformance in feature selection, because they can exploit information that the trained model provides. For example, in [5] the JCFO classification method is suggested that jointly selects relevant features and estimates parameters of the classifier. The classifier that they use is based on a linear model and feature selection is achieved by estimating parameters of the kernel function. More specifically, a *scaling factor* h_i is estimated for each feature x_i, which measures the significance of that feature. For example, in the case of *anisotropic* Gaussian kernels, the scaling factors correspond to the *inverse variance* of each feature. Then a Laplacian sparsity prior is enforced on the scaling factors $\boldsymbol{h} = (h_1, \ldots, h_d)^T$ in order to eliminate irrelevant features.

In this work we propose a Bayesian method to incorporate a feature selection mechanism in the adaptive kernel learning approach for the RVM (called aRVM) proposed in [1]. This method is similar in spirit to JCFO in that they both estimate parameters of kernels that are called scaling factors in order to measure the significance of each feature. However, the proposed approach (that is based on aRVM) learns *separate scaling factors for each kernel*, therefore feature selection is *local*, since it is performed for each kernel separately. This might be useful for example when different features are significant for discriminating examples of each class, as demonstrated in the example of Fig. 1.

The rest of this paper is organized as follows. In Section 2 we present an overview of sparse Bayesian linear models for regression and classification problems. In Section 3 we propose a method to incorporate local feature selection to the sparse Bayesian linear model, by adapting the kernel learning algorithm of [1]. In Section 4 we first present an artificial example to demonstrate the advantages of the proposed method and then we evaluate its performance on regression and classification datasets.

2 Sparse Bayesian Linear Models

2.1 Sparse Bayesian Regression

Since RVM is a Bayesian linear model, in this section we will review sparse Bayesian learning of linear models given by (2) [2]. We assume Gaussian distributed noise with separate precision β_n for each data point $p(\epsilon_n|\beta_n) = N(\epsilon_n|0, \beta_n^{-1})$, and a Gaussian prior with separate variance for the weights given by (3).

Defining the fixed 'design' matrix $\boldsymbol{\Phi} = (\boldsymbol{\phi}(\boldsymbol{x}_1), \ldots, \boldsymbol{\phi}(\boldsymbol{x}_M))^T$, with $\boldsymbol{\phi}(\boldsymbol{x}) = (\phi_1(\boldsymbol{x}), \ldots, \phi_M(\boldsymbol{x}))^T$, the likelihood can be written as:

$$p(\boldsymbol{t}|\boldsymbol{w}, \boldsymbol{\beta}) = \mathrm{N}(\boldsymbol{t}|\boldsymbol{\Phi}\boldsymbol{w}, \boldsymbol{B}^{-1}), \tag{4}$$

where $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_N)^T$ and $\boldsymbol{B} = \mathrm{diag}(\boldsymbol{\beta})$.

The posterior distribution of the weights can be computed using Bayes's law:

$$p(\boldsymbol{w}|\boldsymbol{t}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{p(\boldsymbol{t}|\boldsymbol{w}, \boldsymbol{\beta})p(\boldsymbol{w}|\boldsymbol{\alpha})}{p(\boldsymbol{t}|\boldsymbol{\alpha}, \boldsymbol{\beta})}, \tag{5}$$

where $p(\boldsymbol{w}|\boldsymbol{a})$ is given by (3). It can be shown that the weight posterior distribution is given by [2]:

$$p(\boldsymbol{w}|\boldsymbol{t}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = N(\boldsymbol{w}|\boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{6}$$

where

$$\boldsymbol{\mu} = \boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{B}\boldsymbol{t}, \tag{7}$$

$$\boldsymbol{\Sigma} = (\boldsymbol{\Phi}^T\boldsymbol{B}\boldsymbol{\Phi} + \boldsymbol{A})^{-1}, \tag{8}$$

and $\boldsymbol{A} = \mathrm{diag}(\boldsymbol{\alpha})$.

In order to control the amount of sparsity, we use a prior on $\boldsymbol{\alpha}$ that directly penalizes models with large number of effective parameters [3]. The output of the model at the training points $\boldsymbol{y} = (y(\boldsymbol{x}_1), \ldots, y(\boldsymbol{x}_N)))^T$ can be evaluated as $\boldsymbol{y} = \boldsymbol{S}\boldsymbol{t}$, where $\boldsymbol{S} = \boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{B}$ is the so called smoothing matrix. The 'degrees of freedom' of $\boldsymbol{S}$, given by $DF = \mathrm{trace}(\boldsymbol{S})$, measure the effective number of parameters of the model. This motivates the following sparsity prior [3]:

$$p(\boldsymbol{\alpha}|\boldsymbol{\beta}) \propto \exp(-c\,\mathrm{trace}(\boldsymbol{S})), \tag{9}$$

where the parameter c provides a mechanism to control the amount of desired sparsity. When using specific values of the sparsity parameter c, some known model selection criteria are obtained [6]:

$$c = \begin{cases} 0 & \text{None (typical RVM)}, \\ 1 & \text{AIC (Akaike information criterion)}, \\ \log(N)/2 & \text{BIC (Bayesian information criterion)}, \\ \log(N) & \text{RIC (Risk inflation criterion)}. \end{cases} \tag{10}$$

Update formulas for the weight precisions $\boldsymbol{\alpha}$ can be obtained by maximizing the posterior distribution $p(\boldsymbol{\alpha}|\boldsymbol{t}, \boldsymbol{\beta}) \propto p(\boldsymbol{t}|\boldsymbol{\alpha}, \boldsymbol{\beta})p(\boldsymbol{\alpha})$ [1]:

$$\alpha_i = \frac{\gamma_i}{\mu_i^2 - 2c\gamma_i\Sigma_{ii}}, \tag{11}$$

where $\gamma_i = 1 - \alpha_i\Sigma_{ii}$.

Furthermore, since the noise is assumed i.i.d., then $\boldsymbol{B} = \beta\boldsymbol{I}$ with $\beta = \beta_1 = \cdots = \beta_N$ and we can also update the noise precision β. Assuming an uninformative prior $p(\beta)$ we only need to maximize the marginal likelihood $p(\boldsymbol{t}|\boldsymbol{\alpha}, \boldsymbol{\beta})$. Setting its derivative to zero, gives the equation

$$\frac{1}{2}\left[\frac{N}{\beta} - \|\boldsymbol{t} - \boldsymbol{\Phi}\boldsymbol{\mu}\|^2 - \mathrm{trace}(\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{\Phi})\right] - \beta c\,\mathrm{trace}(\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{\Phi}) = 0, \tag{12}$$

which can be easily solved numerically, in order to update the noise precision β.

2.2 Sparse Bayesian Classification

For simplicity we only consider binary classification and assume that the outputs are coded so that $t_n \in \{0, 1\}$[1]. Then, the likelihood is given by:

$$p(\boldsymbol{t}|\boldsymbol{w}) = \prod_{n=1}^{N} y_n^{t_n}(1 - y_n)^{1-t_n}, \tag{13}$$

where $y_n = \sigma(y(\boldsymbol{x}_n|\boldsymbol{w}))$. Using the Laplacian approximation, the classification problem can be mapped to a regression problem [2] with heteroscedastic noise $p(\epsilon_n) = \mathrm{N}(\epsilon_n|0, \beta_n)$. The noise precision is given by:

$$\beta_n = y_n(1 - y_n), \tag{14}$$

[1] Multiclass problems can be solved using the one-vs-all approach, which builds only two-class models.

and the regression targets $\hat{\boldsymbol{t}} = (\hat{t}_1, \ldots, \hat{t}_N)^T$ are:

$$\hat{\boldsymbol{t}} = \boldsymbol{\Phi} \boldsymbol{w} + \boldsymbol{B}^{-1}(\boldsymbol{t} - \boldsymbol{y}), \tag{15}$$

where $\boldsymbol{y} = (y_1, \ldots, y_N)^T$ and $\boldsymbol{B} = \mathrm{diag}(\beta_1, \ldots, \beta_N)$.

3 A Bayesian Model for Local Feature Selection

Consider now the more general form of the sparse Bayesian linear model proposed in [1]:

$$y(\boldsymbol{x}) = \sum_{n=1}^{M} w_n \phi(\boldsymbol{x}; \boldsymbol{\theta}_n), \tag{16}$$

which assumes that all the basis functions have the same parametric form $\phi(\boldsymbol{x}; \boldsymbol{\theta}_n)$, but assumes separate parameter vector $\boldsymbol{\theta}_n$ for each basis function. This model is called adaptive RVM (aRVM). For example, the basis functions may be RBF kernels and the parameters $\boldsymbol{\theta}_n$ may include their location (center) and scale (width). Moreover, we assume a zero mean Gaussian prior for the weights $\boldsymbol{w}$, using a separate parameter α_n for the precision of each weight w_n, given by (3). Because we estimate different values for the parameters of each basis function, this model is very flexible and in order to avoid overfitting we use the sparsity enforcing prior of (9).

Unlike the typical linear model, which assumes that the basis functions are fixed and known in advance, we use the algorithm proposed in [1] to estimate the parameters $\boldsymbol{\theta}_n$ of the basis functions during model learning.

3.1 Adaptive Kernel Learning

The adaptive kernel learning algorithm [1] has been proposed to estimate the parameters of aRVM, which is based on an incremental learning algorithm for the typical RVM [4]. The incremental algorithm initially assumes an empty model, by setting $\alpha_i = \infty$, for all $i = 1, \ldots, M$. Then, at each iteration one basis function may be either added to the model or re-estimated or removed from the current model.

Since we add only one basis function at each iteration, we maximize with respect to a *single parameter* α_i each time. Using the sparsity prior $p(\boldsymbol{\alpha})$ of (9) and keeping only the terms of the objective function that depend on α_i, the following function should be maximized with respect to α_i [1,4]

$$l_i^s = \frac{1}{2}\left(\log \alpha_i - \log(\alpha_i + s_i) + \frac{q_i^2 + 2c\alpha_i}{\alpha_i + s_i} \right), \tag{17}$$

where

$$s_i = \boldsymbol{\phi}_i^T \boldsymbol{C}_{-i}^{-1} \boldsymbol{\phi}_i, \qquad q_i = \boldsymbol{\phi}_i^T \boldsymbol{C}_{-i}^{-1} \hat{\boldsymbol{t}}, \tag{18}$$

$\boldsymbol{\phi}_i = (\phi(\boldsymbol{x}_1;\boldsymbol{\theta}_i),\ldots,\phi(\boldsymbol{x}_N;\boldsymbol{\theta}_i))^T$ and $\boldsymbol{C}_{-i} = \boldsymbol{B} + \sum_{j\neq i}\alpha_j\boldsymbol{\phi}_j\boldsymbol{\phi}_j^T$. In regression we have $\hat{\boldsymbol{t}} = \boldsymbol{t}$ and usually $\boldsymbol{B} = \beta\boldsymbol{I}$, while in classification $\boldsymbol{B}$ and $\hat{\boldsymbol{t}}$ are given by (14) and (15) respectively. It has been shown in [1] that maximization of l_i^s with respect to α_i has a single maximum at:

$$\begin{aligned} \alpha_i &= \frac{s_i^2}{q_i^2 - (2c+1)s_i} && \text{if } q_i^2 > (2c+1)s_i, \\ \alpha_i &= \infty && \text{if } q_i^2 \leq (2c+1)s_i. \end{aligned} \tag{19}$$

Appropriate values for the basis function parameters θ_{ik} are determined using the quasi-Newton BFGS optimization method to maximize (17). The required derivatives can be analytically computed as:

$$\frac{\partial l_i^s}{\partial \theta_{ik}} = -\left(\frac{1}{\alpha_i + s_i} + \frac{q_i^2 + c\alpha_i}{(\alpha_i + s_i)^2}\right) r_i + \frac{q_i}{\alpha_i + s_i} w_i, \tag{20}$$

where

$$r_i \equiv \frac{1}{2}\frac{\partial s_i}{\partial \theta_{ik}} = \boldsymbol{\phi}_i^T \boldsymbol{C}_{-i}^{-1}\frac{\partial \boldsymbol{\phi}_i}{\partial \theta_{ik}}, \qquad w_i \equiv \frac{\partial q_i}{\partial \theta_{ik}} = \boldsymbol{t}^T \boldsymbol{C}_{-i}^{-1}\frac{\partial \boldsymbol{\phi}_i}{\partial \theta_{ik}}, \tag{21}$$

and $\boldsymbol{\phi}_i = (\phi(\boldsymbol{x}_1;\boldsymbol{\theta}_i),\ldots,\phi(\boldsymbol{x}_N;\boldsymbol{\theta}_i))^T$.

The adaptive kernel learning algorithm proceeds iteratively, selecting at each iteration the most appropriate basis function to add to the model.

3.2 Local Feature Selection

In (16) we assume that all basis function have the same parametric form, but different values $\boldsymbol{\theta}_n$ for the parameters. In order to facilitate feature selection we need to parameterize the kernel function so that it incorporates scaling factors for each dimension. Here, we consider anisotropic Gaussian kernel functions, which have a separate precision parameter h_{ni} for each feature i:

$$\phi(\boldsymbol{x};\boldsymbol{m}_n,\boldsymbol{h}_n) = \exp\left[-\sum_{i=1}^{d}(h_{ni})^2(x_i - m_{ni})^2\right], \tag{22}$$

where $\boldsymbol{h}_n = (h_{n1},\ldots,h_{nd})^T$ and $\boldsymbol{m}_n = (m_{n1},\ldots,m_{nd})^T$. We call feature selection *local*, because we have assigned separate scaling factors for the features of each kernel. This approach is much more flexible to the typical feature selection approach, since different features are considered to be significant at different regions of the input space.

Notice that if we assign a very small value to a scaling factor h_{ni} of the n-th kernel, the corresponding feature x_i does not contribute to that kernel. Under the Bayesian framework, elimination of irrelevant features can be motivated by imposing a prior distribution on the scaling factors $\boldsymbol{h} = (\boldsymbol{h}_1^T,\ldots,\boldsymbol{h}_M^T)^T$ that enforces sparsity. In analogy to the prior on the weights w_n, the prior distribution that we use on $\boldsymbol{h}_n$ is the Student's t distribution, which is known to give sparse solutions for few degrees of freedom [7]. Since a Student's t distributed random variable is equivalent to a Gaussian distributed random variable whose precision parameter is assumed Gamma distributed, we can write:

$$p(\boldsymbol{h}|\boldsymbol{\delta}) = \mathrm{N}(\boldsymbol{h}|\mathbf{0}, \boldsymbol{\Delta}^{-1}), \tag{23}$$

with $\boldsymbol{\delta}_n = (\delta_{n1}, \ldots, \delta_{nd})^T$, $\boldsymbol{\Delta} = \mathrm{diag}\{\boldsymbol{\delta}_1, \ldots, \boldsymbol{\delta}_M\}$ and

$$p(\boldsymbol{\delta}) = \prod_{n=1}^{M} \prod_{i=1}^{d} \mathrm{Gamma}(\delta_{ni}|a, b), \tag{24}$$

In the above definition we set $a = b = 0$ to define an uninformative Gamma distribution.

Since we assume different parameter values h_{ni} for each kernel, the above model selects different features to contribute at each kernel. Therefore, this model is much more flexible to the typical feature selection approach that selects a set of significant features that is *common* for all the kernels. An example that demonstrates the advantages of the proposed model is presented in Section 4.

3.3 Parameter Learning

The learning method is similar to the adaptive kernel learning algorithm of [1]. It incrementally adds basis functions to an initially empty model and at the same time it assigns appropriate values to their parameters. Estimation of the parameters $\boldsymbol{m}_n$ can be performed using the quasi-Newton BFGS method with derivatives given from (20).

Estimation of the scaling factors $\boldsymbol{h}$ can also be performed using the adaptive kernel learning algorithm. However, we must take into account the existence of the prior distribution of (23) that we have assigned to the scaling factors. Therefore, we now want to jointly maximize the posterior $p(\boldsymbol{\alpha}, \boldsymbol{h}|\boldsymbol{t})$ with respect to the weight precisions $\boldsymbol{\alpha}$ and the scaling factors $\boldsymbol{h}$, which is given by

$$\log p(\boldsymbol{\alpha}, \boldsymbol{h}|\boldsymbol{t}) = \log p(\boldsymbol{t}|\boldsymbol{\alpha}, \boldsymbol{h}) + \log p(\boldsymbol{h}) + \log p(\boldsymbol{\alpha}) = l_i^s - \frac{1}{2}\boldsymbol{h}^T \boldsymbol{\Delta} \boldsymbol{h}. \tag{25}$$

This optimization is performed using (19) to update the weight precisions α_i and using a quasi-Newton optimization method to update the scaling factors h_{ni}. The required derivatives are obtained from (20) by adding the term corresponding to the derivative of the prior $p(\boldsymbol{h})$:

$$\frac{\partial \log p(\boldsymbol{h}|\boldsymbol{t})}{\partial h_{ni}} = \frac{\partial \log p(\boldsymbol{t}|\boldsymbol{\theta})}{\partial h_{ni}} - \delta_{ni} h_{ni}. \tag{26}$$

Furthermore, we also have to specify the update equations for the parameters $\boldsymbol{\delta}$ that define the precision of $\boldsymbol{h}$. By setting $a = b = 0$ in (24), we assume an

Algorithm 1. Feature Selection Using Adaptive Kernel Learning

```
1. Select an inactive basis function to add to the model (convert to active) as follows:
    (a) Consider an initial set of inactive candidate basis functions by sampling their
        parameters at random.
    (b) Optimize separately the parameters of each candidate basis function to
        maximize the marginal likelihood.
    (c) Add to the model the candidate basis function that increases the marginal
        likelihood the most.
2. Optimize the parameters $\boldsymbol{\theta}$ of all currently active basis functions.
3. Update hyperparameters $\boldsymbol{\alpha}$ and noise precision $\beta$, using (19) and (12).
4. Update hyperparameters $\boldsymbol{\delta}$ using (27).
5. Remove from the model any unnecessary active basis functions.
6. Repeat steps 1 to 5 until convergence.
```

uninformative prior distribution for them. Then, maximization of the likelihood with respect to $\boldsymbol{\delta}$ gives

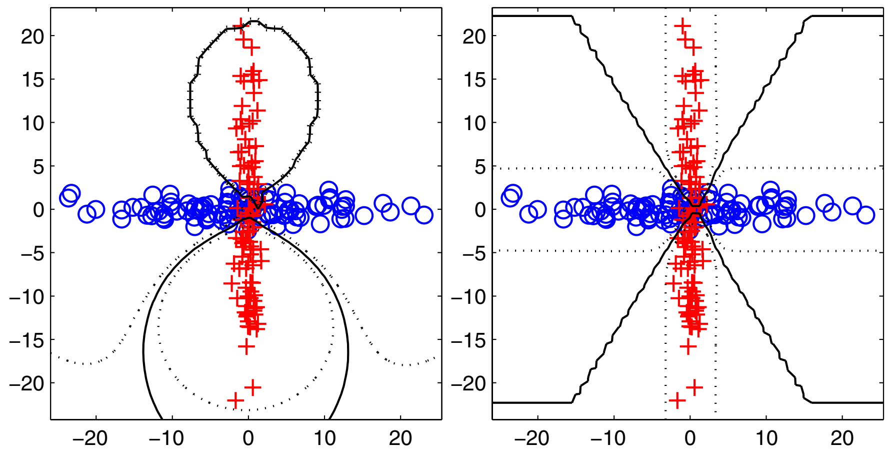
$$\delta_{ni} = \frac{1}{h_{ni}^2}. \tag{27}$$

In summary the proposed learning method, which is descibed in Algorithm 1, is based on the incremental adaptive kernel learning algorithm of [1], but it also takes into account the sparsity prior for the scaling factors $\boldsymbol{h}$ and updates the parameters $\boldsymbol{\delta}$ that have been introduced to facilitate feature selection.

4 Numerical Experiments

The purpose of the first experiment is to demonstrate the feature selection capabilities of the proposed method. For this reason, we have generated samples from two two-dimensional zero-mean Gaussian distributions, each corresponding to one of the classes. More specifically, we selected the variance of the Gaussian distributions to be $s_1 = (1, 10)^T$ and $s_2 = (10, 1)^T$, so that only one feature is significant for discriminating each class. In Fig. 1 we show the estimated models using i) the RVM with adaptive kernel learning algorithm of [1] and ii) the proposed modification to incorporate feature selection. Notice, that the model obtained using the proposed approach contains only one basis function for each class, with scaling factors $\theta_1 = (0.6, 0.0)^T$ and $\theta_2 = (0.0, 0.4)^T$, therefore it successfully identifies the relevant features for each basis function. For this reason, it achieves better classification performance (measured by the percentage of misclassified examples), which is also shown in Fig. 1.

Furthermore, in order to evaluate the method we have performed experiments with several regression and classification datasets from the UCI Machine Learning Repository that were also used in [1]. More specifically, we estimate the generalization error of each method by performing ten-fold cross validation on each dataset. In regression, the error is the *mean square error*,



(a) aRVM, error=10.7% (b) aRVM^d, error=7.7%

Fig. 1. Artificial classification example. Solution provided by aRVM classifiers: (a) without feature selection (b) with feature selection. Solid lines show the estimated decision boundary and dotted lines correspond to the points where the estimated probability of misclassification is 0.25.

$MSE = \sum_n (t_n - \hat{y}_n)^2/N$, where t_n is the value given by the test set, $\hat{y}_n$ the predicted value and N the number of test examples. In classification the error is the percentage of misclassified examples in the test set. We evaluate three methods; i) the typical RVM with Gaussian kernel (denoted as RVM), ii) adaptive RVM with learning of Gaussian kernel parameters [1] (denoted as aRVM) and iii) the proposed adaptive RVM with simultaneous feature selection by learning the parameters of anisotropic Gaussian kernels (denoted as aRVM^d). The regression and classification results are shown in Table 1 and Table 2 respectively, where for each dataset the error and number of relevance vectors(RV) are presented. It can be observed that in most cases the proposed approach, which incorporates feature selection, provides improved performance compared to both the typical RVM model and the aRVM method [1].

Table 1. Experimental results on regression datasets

			RVM		aRVM		aRVM^d	
Dataset	patterns (N)	features (d)	error	RVs	error	RVs	error	RVs
computer	209	6	30004	140.5	22379	5.0	4089	13.6
Boston	506	13	12.48	69.5	11.53	13.27	13.66	17.5
concrete	1030	8	44.204	140.2	34.515	9.10	28.868	42.3

Table 2. Experimental results on classification datasets

			RVM		aRVM		aRVM^d	
Dataset	patterns (N)	features (d)	error	RVs	error	RVs	error	RVs
banana	5300	2	0.1092	12.1	0.1126	6.3	0.0994	4.4
titanic	2200	3	0.2292	31.0	0.2270	2.0	0.2254	4.0
image	2310	18	0.0390	34.6	0.0387	6.9	0.0342	21.3
breast-cancer	277	9	0.2818	9.6	0.2844	4.4	0.2629	3.0
pima	768	8	0.243	27.9	0.2303	5.6	0.2276	5.1

5 Conclusions

In this work we have presented an approach to incorporate feature selection to the sparse Bayesian linear model by adapting the kernel learning approach of [1]. In contrast to typical feature selection approaches, the significance of each feature is assessed separately for each kernel. Therefore, for each kernel a different set of significant features is selected. This approach might be useful, for example in a classification problem when different features are significant for discriminating the examples of each class. Experimental results on several regression and classification datasets demonstrate the merits of the method. As a future work, we aim to test the effectiveness of this approach to bioinformatics problems (eg. gene microarray classification), where the number of features is very high and feature selection is a very important issue.

References

1. Tzikas, D., Likas, A., Galatsanos, N.: Sparse bayesian modeling with adaptive kernel learning. IEEE Transactions on Neural Networks (to appear)
2. Tipping, M.E.: Sparse Bayesian learning and the relevance vector machine. Journal of Machine Learning Research 1, 211–244 (2001)
3. Schmolck, A., Everson, R.: Smooth relevance vector machine: a smoothness prior extension of the RVM. Machine Learning 68(2), 107–135 (2007)
4. Tipping, M.E., Faul, A.: Fast marginal likelihood maximisation for sparse Bayesian models. In: Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics (2003)
5. Krishnapuram, B., Hartemink, A.J., Figueiredo, M.A.T.: A Bayesian approach to joint feature selection and classifier design. IEEE Transactions on Pattern Analysis and Machine Intelligence 26(9), 1105–1111 (2004)
6. Holmes, C.C., Denison, D.G.T.: Bayesian wavelet analysis with a model complexity prior. In: Bernardo, J.M., Berger, J.O., Dawid, A.P., Smith, A.F.M. (eds.) Bayesian Statistics 6: Proceedings of the Sixth Valencia International Meeting. Oxford University Press, Oxford (1999)
7. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, Heidelberg (2006)