Sparse Bayesian Modeling With Adaptive Kernel Learning

Dimitris G. Tzikas, Aristidis C. Likas, *Senior Member, IEEE*, and Nikolaos P. Galatsanos, *Senior Member, IEEE*

***Abstract*—Sparse kernel methods are very efficient in solving regression and classification problems. The sparsity and performance of these methods depend on selecting an appropriate kernel function, which is typically achieved using a cross-validation procedure. In this paper, we propose an incremental method for supervised learning, which is similar to the relevance vector machine (RVM) but also learns the parameters of the kernels during model training. Specifically, we learn different parameter values for each kernel, resulting in a very flexible model. In order to avoid overfitting, we use a sparsity enforcing prior that controls the effective number of parameters of the model. We present experimental results on artificial data to demonstrate the advantages of the proposed method and we provide a comparison with the typical RVM on several commonly used regression and classification data sets.**

***Index Terms*—Classification, kernel learning, regression, relevance vector machine (RVM), sparse Bayesian learning.**

I. 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

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

Predictions are typically made by assuming a parametric form for the function y, for example, 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 basis function set. Furthermore, the noise ϵ_n is typically assumed to be independent, identically distributed (i.i.d.), following a Gaussian distribution with zero mean and variance β^{-1}

$$p(\epsilon_n|\beta) = \mathrm{N}(\epsilon_n|0, \beta^{-1}). \tag{3}$$

From (1) and (3), we get the likelihood of this model, which is

$$p(\boldsymbol{t}|\boldsymbol{w}, \beta) = \prod_{n=1}^{N} \mathrm{N}(t_n|y(\boldsymbol{x}_n|\boldsymbol{w}), \beta^{-1}) \tag{4}$$

where $\boldsymbol{t} = (t_1, \ldots, t_N)^T$ and $\mathrm{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = (2\pi)^{-(M)/(2)}|\boldsymbol{\Sigma}|^{-(1)/(2)} \exp[-(1)/(2)(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})]$ denotes the multivariate M-dimensional Gaussian distribution over $\boldsymbol{x}$ with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$.

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 $t_{nk} = 0$. Predictions can be made by assuming that the outputs t_n follow a multinomial distribution, whose parameters are given by applying a sigmoid function to a linear model with K outputs

$$p(\boldsymbol{t}|\boldsymbol{w}) = \prod_{n=1}^{N}\prod_{k=1}^{K} \sigma(y_k(\boldsymbol{x}_n|\boldsymbol{w}))^{t_{nk}}. \tag{5}$$

The linear model of (2) is a very powerful model that can approximate any function provided that appropriate (both in number and in shape) basis functions have been selected. For this reason, it is important to select an appropriate basis function set. The relevance vector machine (RVM) [1], [2] assumes that the basis function set consists of kernel functions "centered" at each training point

$$y(\boldsymbol{x}|\boldsymbol{w}) = \sum_{n=1}^{N} w_n K(\boldsymbol{x}, \boldsymbol{x}_n) \tag{6}$$

where $K(\boldsymbol{x}_1, \boldsymbol{x}_2)$ is some kernel function. Although this model has as many parameters as training points, overfitting can be avoided by using some regularization technique, for example, assuming a prior distribution for the weights w. The simplest choice is to use a Gaussian distribution on the weights $p(\boldsymbol{w}) = \prod_{i=1}^{M} \mathrm{N}(w_i|0, \alpha^{-1})$, which provides good regularization and allows inference at small computational cost. However, such prior cannot capture the local properties of the data. Furthermore, since it assumes identical variance α^{-1} for all weights, it cannot yield sparse estimations, i.e., estimations in which only a few basis functions are used. Sparsity is a desirable property that has attracted a lot of attention lately since it is an efficient way to control the complexity of the model and avoid overfitting. Furthermore, making predictions with sparse models is very computationally efficient. In order to obtain sparse estimations, the

Manuscript received April 23, 2008; revised September 17, 2008 and December 02, 2008; accepted January 02, 2009. First published May 05, 2009; current version published June 03, 2009.

D. G. Tzikas and A. C. Likas are with the Department of Computer Science, University of Ioannina, Ioannina 45110, Greece (e-mail: tzikas@cs.uoi.gr; arly@cs.uoi.gr).

N. P. Galatsanos is with the Department of Electrical and Computer Engineering, University of Patras, Rio 26500, Greece (e-mail: ngalatsanos@upatras.gr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2009.2014060

RVM assumes a Gaussian prior with *different* precision α_i for each weight w_i

$$p(\boldsymbol{w}|\boldsymbol{\alpha}) = \prod_{i=1}^{M} \mathrm{N}\left(w_i|0, \alpha_i^{-1}\right) \tag{7}$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_M)^T$. Under the Bayesian framework, the parameters α_i can be estimated by maximizing the marginal likelihood $p(\boldsymbol{t}|\boldsymbol{\alpha}, \beta)$, obtained by integrating out the weights $\boldsymbol{w}$. Sparsity is achieved because most of the parameters α_i are estimated to very large values, thus pruning the corresponding basis functions by forcing their weights to become zero. Recently, RVM has been used successfully in many applications, for example, in recognition of hand motions [3], recovery of 3-D human pose from silhouettes [4], detection of clustered microclassifications for mammography [5], classification of gene expression data [6], [7], detection of activations for neuroimaging [8], real-time tracking [9], and object detection in scenes [10]. In spite of this, in order to obtain good generalization performance, *it is important to select an appropriate kernel function*.

Although typically the kernel is selected using a cross-validation technique, there has been work on learning the kernel function simultaneously with model parameters. It has been proposed in [11] that the width parameter of Gaussian kernels can be learned by maximizing the marginal likelihood of the model. Also, in [12]–[14], the kernel has been modeled as a linear combination of other basis functions. In [15], feature selection has been achieved by using a kernel function with separate scaling factor for each feature and applying a sparsity prior to the scaling factors. Finally, in [16], an alternative to the Gaussian process (GP) model has been proposed that learns a set of pseudoinputs, which are similar to the relevance vectors (RVs), but do not necessarily coincide with points of the training set. All these methods attempt to learn parameters of kernels that are centered at many different locations, however, they assume that all these kernels have the same parameters. This might be a significant limitation if the data that we attempt to model have different characteristics at different locations, such as a signal with varying frequency. In the context of GPs, such cases can be treated using nonstationary covariance functions [17].

In this paper, we propose a new methodology to automatically learn the basis functions of a sparse linear model. Unlike the existing literature, the proposed methodology 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, we use a sparsity enforcing prior that directly *controls the number of effective parameters* of the model. This prior has previously been used for orthogonal wavelet basis function sets [18], but here we extend it for arbitrary basis function sets. Learning in the proposed model is achieved using an algorithm that is similar to the incremental RVM algorithm [19]. 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 incremental RVM, 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 proposed methodology uses *continuous optimization* with respect to the parameters (such as location and scale) of the basis functions. We then employ this methodology to learn the center (mean) and width (variance) parameters of Gaussian kernel basis functions.

There are several advantages of the proposed methodology as compared to traditional RVM [1].

- There is no need to select the parameters of the kernel via cross validation, since they are selected automatically.
- Because each kernel may have different parameter values, the model is very flexible and it can accurately solve a wide variety of problems.
- The obtained models are typically much sparser compared to the typical RVM.

The rest of this paper is organized as follows. In Section II, we review sparse linear models and the RVM and we generalize the sparsity prior of [18] for nonorthogonal basis function sets. In Section III, we present an algorithm for learning the basis function set. In Section IV, we provide experiments on artificial data sets that demonstrate the advantages of the proposed method and we compare the proposed algorithm with the typical RVM algorithm on benchmark data sets. In Section V, we discuss the computational cost of the method and provide a probabilistic interpretation of the kernel function, and finally, in Section VI, we provide some conclusions.

II. Sparse Bayesian Linear Models

A. Sparse Bayesian Regression

In this section, we will review sparse Bayesian learning of linear models [1] under the assumptions of a Gaussian prior with separate variance for the weights, given by (7), and Gaussian noise with separate precision β_n for each data point $p(\epsilon_n|\beta_n) = N(\epsilon_n|0, \beta_n^{-1})$. Assigning separate precision for each data point is important to derive the classification algorithm, and furthermore, it allows designing robust regression models by selecting an appropriate noise precision prior $p(\boldsymbol{\beta})$. For example, assuming a Gamma probability density function (pdf) for the noise precisions, $p(\boldsymbol{\beta})$ corresponds to a student's t pdf for the noise $p(\boldsymbol{\epsilon}) = \int p(\boldsymbol{\epsilon}|\boldsymbol{\beta})p(\boldsymbol{\beta})\,\mathrm{d}\boldsymbol{\beta}$, which achieves robustness because it has heavy tails [20]. The likelihood of this model is a generalization of (4) and by defining the fixed "design" matrix $\boldsymbol{\Phi} = (\boldsymbol{\phi}(\boldsymbol{x}_1), \ldots, \boldsymbol{\phi}(\boldsymbol{x}_M))^T$, with $\boldsymbol{\phi}(\boldsymbol{x}) = (\phi_1(\boldsymbol{x}), \ldots, \phi_M(\boldsymbol{x}))^T$, it can be written as

$$p(\boldsymbol{t}|\boldsymbol{w}, \boldsymbol{\beta}) = \mathrm{N}(\boldsymbol{t}|\boldsymbol{\Phi}\boldsymbol{w}, \boldsymbol{B}^{-1}) \tag{8}$$

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' law

$$p(\boldsymbol{w}|\boldsymbol{t}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{p(\boldsymbol{t}|\boldsymbol{w}, \boldsymbol{\beta})p(\boldsymbol{w}|\boldsymbol{\alpha})}{p(\boldsymbol{t}|\boldsymbol{\alpha}, \boldsymbol{\beta})} \tag{9}$$

where $p(\boldsymbol{w}|\boldsymbol{a})$ is given by (7). It can be shown that the weight posterior distribution is given by [1]

$$p(\boldsymbol{w}|\boldsymbol{t}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = N(\boldsymbol{w}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{10}$$

where

$$\boldsymbol{\mu} = \boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{B}\boldsymbol{t} \tag{11}$$

$$\boldsymbol{\Sigma} = (\boldsymbol{\Phi}^T\boldsymbol{B}\boldsymbol{\Phi} + \boldsymbol{A})^{-1} \tag{12}$$

and $\boldsymbol{A} = \mathrm{diag}(\boldsymbol{\alpha})$. Here, although we introduced a prior distribution on $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, we do not attempt to compute the joint posterior $p(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}|\boldsymbol{t})$ of the hidden variables, but we only compute the weight posterior $p(\boldsymbol{w}|\boldsymbol{t}, \boldsymbol{\alpha}, \boldsymbol{\beta})$. Then, we find maximum *a posteriori* (MAP) estimates for $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ by finding the mode $(\boldsymbol{\alpha}_{MP}, \boldsymbol{\beta}_{MP})$ of its posterior $p(\boldsymbol{\alpha}, \boldsymbol{\beta}|\boldsymbol{t}) \propto p(\boldsymbol{t}|\boldsymbol{\alpha}, \boldsymbol{\beta})p(\boldsymbol{\alpha}, \boldsymbol{\beta})$. Assuming uninformative priors for $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, update formulas can be obtained by maximizing the logarithm of the marginal likelihood $p(\boldsymbol{t}|\boldsymbol{\alpha}, \boldsymbol{\beta}) = \int p(\boldsymbol{t}|\boldsymbol{w}, \boldsymbol{\beta})p(\boldsymbol{w}|\boldsymbol{\alpha})\,\mathrm{d}\,\boldsymbol{w}$ [1]

$$L = \log p(\boldsymbol{t}|\boldsymbol{\alpha}, \boldsymbol{\beta}) = -\frac{1}{2}(N\log 2\pi + |\boldsymbol{C}| + \boldsymbol{t}^T\boldsymbol{C}^{-1}\boldsymbol{t}) \tag{13}$$

where $\boldsymbol{C} = \boldsymbol{B}^{-1} + \boldsymbol{\Phi}\boldsymbol{A}^{-1}\boldsymbol{\Phi}^T$. The derivative of the marginal likelihood with respect to $\log\alpha_i$ is

$$\frac{\partial L}{\partial \log\alpha_i} = \frac{1}{2}\left(1 - \alpha_i\Sigma_{ii} - \alpha_i\mu_i^2\right) \tag{14}$$

and by setting it to zero, we obtain the following update formula for α_i [1]:

$$\alpha_i = \frac{\gamma_i}{\mu_i^2} \tag{15}$$

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. The derivative of the marginal likelihood with respect to $\log\beta$ is given by

$$\frac{\partial L}{\partial \log\beta} = \frac{1}{2}\left[\frac{N}{\beta} - \|\boldsymbol{t} - \boldsymbol{\Phi}\boldsymbol{\mu}\|^2 - \mathrm{trace}(\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{\Phi})\right] \tag{16}$$

and by setting it to zero, we obtain the following update formula for β [1]:

$$\beta = \frac{N - \sum_{i=1}^{N}\gamma_i}{\|\boldsymbol{t} - \boldsymbol{\Phi}\boldsymbol{\mu}\|^2}. \tag{17}$$

B. 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 multinomial likelihood in (5) simplifies to a Bernoulli likelihood

$$p(\boldsymbol{t}|\boldsymbol{w}) = \prod_{n=1}^{N} y_n^{t_n}(1 - y_n)^{1-t_n} \tag{18}$$

where $y_n = \sigma(y(\boldsymbol{x}_n|\boldsymbol{w}))$. Using the Laplacian approximation, the classification problem can be mapped to a regression problem with heteroscedastic noise $p(\epsilon_n) = \mathrm{N}(\epsilon_n|0, \beta_n)$ [1]. The noise precision is given by

$$\beta_n = y_n(1 - y_n) \tag{19}$$

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

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

$$\hat{\boldsymbol{t}} = \boldsymbol{\Phi}\boldsymbol{w} + \boldsymbol{B}^{-1}(\boldsymbol{t} - \boldsymbol{y}) \tag{20}$$

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

C. Incremental Optimization for Sparse Bayesian Learning

Notice that the computational cost of the sparse Bayesian learning algorithm is high for large data sets, because the computation of $\boldsymbol{\Sigma}$ in (12) requires $O(N^3)$ operations. A more computationally efficient incremental algorithm has been proposed in [19]. It initially assumes that $\alpha_i = \infty$, for all $i = 1, \ldots, M$, which corresponds to assuming that all basis functions have been pruned because of the sparsity constraint. Then, at each iteration, one basis function may be either added to the model or reestimated or removed from the current model. When adding a basis function to the model, the corresponding parameter α_i is set to the value that maximizes the marginal likelihood.

More specifically, the terms of the marginal likelihood (13) that depend on a single parameter α_i are [19]

$$l_i = \frac{1}{2}\left(\log\alpha_i - \log(\alpha_i + s_i) + \frac{q_i^2}{\alpha_i + s_i}\right) \tag{21}$$

where

$$s_i = \boldsymbol{\phi}_i^T\boldsymbol{C}_{-i}^{-1}\boldsymbol{\phi}_i \tag{22}$$

$$q_i = \boldsymbol{\phi}_i^T\boldsymbol{C}_{-i}^{-1}\hat{\boldsymbol{t}} \tag{23}$$

$\boldsymbol{\phi}_i = (\phi_i(\boldsymbol{x}_1), \ldots, \phi_i(\boldsymbol{x}_N))^T$ 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 (19) and (20), respectively.

In order to simplify computations, one can define

$$S_i = \boldsymbol{\phi}_i^T\boldsymbol{C}^{-1}\boldsymbol{\phi}_i \tag{24}$$

$$Q_i = \boldsymbol{\phi}_i^T\boldsymbol{C}^{-1}\hat{\boldsymbol{t}} \tag{25}$$

and compute s_i and q_i from

$$s_i = \frac{\alpha_i S_i}{\alpha_i - S_i} \tag{26}$$

$$q_i = \frac{\alpha_i Q_i}{\alpha_i - S_i}. \tag{27}$$

Also the inversion of $\boldsymbol{C}$ can be avoided by using the Woodbury identity to write

$$S_i = \boldsymbol{\phi}_i^T\boldsymbol{B}\boldsymbol{\phi}_i - \boldsymbol{\phi}_i^T\boldsymbol{B}\boldsymbol{\Phi}\Sigma\Phi^T\boldsymbol{B}\boldsymbol{\phi}_i \tag{28}$$

$$Q_i = \boldsymbol{\phi}_i^T\boldsymbol{B}\hat{\boldsymbol{t}} - \boldsymbol{\phi}_i^T\boldsymbol{B}\boldsymbol{\Phi}\Sigma\Phi^T\boldsymbol{B}\hat{\boldsymbol{t}}. \tag{29}$$

It has been shown in [21] that l has a single maximum at

$$\alpha_i = \frac{s_i^2}{q_i^2 - s_i}, \qquad \text{if } q_i^2 > s_i \tag{30}$$

$$\alpha_i = \infty, \qquad \text{if } q_i^2 \leq s_i. \tag{31}$$

Based on this result, the incremental algorithm proceeds iteratively, adding each time a basis function ϕ_i if $q_i^2 > s_i$ and removing it otherwise.

An important question that arises in the incremental RVM algorithm is which basis function to update at each iteration. There are several possibilities, for example, we could choose a basis function at random, or with some additional computational

cost, we could test several and select the one whose addition will cause the largest increase to the marginal likelihood.

D. Adjusting Sparsity

In Bayesian modeling, the characteristics of the estimation depend on the assumed prior distribution $p(\boldsymbol{w})$. Thus, the sparsity of the weights $\boldsymbol{w}$ of a sparse linear model is motivated by their prior distribution $p(\boldsymbol{w}) = \int p(\boldsymbol{w}|\boldsymbol{\alpha})p(\boldsymbol{\alpha})\,\mathrm{d}\boldsymbol{\alpha}$. Since $p(\boldsymbol{w}|\boldsymbol{\alpha})$ is given by (7), sparsity depends on selecting an appropriate distribution $p(\boldsymbol{\alpha})$. The typical RVM [1] proposed to use independent Gamma distributions $p(\boldsymbol{\alpha}|a,b) \propto \prod_{i=1}^{M} \alpha_i^{a-1} e^{-b\alpha_i}$. Then, the weight prior $p(\boldsymbol{w})$ is a student's t distribution, which supports sparse models because of its heavy tails. Because it is difficult to select appropriate values for the parameters a and b of the Gamma distribution, they are typically set to $a = b = 0$. These values define an improper uninformative distribution for α_i and correspond to $p(\boldsymbol{w}) = \prod_{i=1}^{M} 1/|w_i|$, which again has heavy tails and supports sparse estimations.

Another approach to control the amount of sparsity is to define a prior on $\boldsymbol{\alpha}$ that directly penalizes models with large number of effective parameters [18]. Notice that 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 the trace of the smoothing matrix $\mathrm{trace}(\boldsymbol{S})$, measure the effective number of parameters of the model. This motivates the following sparsity prior [18]:

$$p(\boldsymbol{\alpha}|\boldsymbol{\beta}) \propto \exp(-c\,\mathrm{trace}(\boldsymbol{S})) \tag{32}$$

where the sparsity 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 [22]

$$c = \begin{cases} 0, & \text{none (typical RVM)} \\ 1, & \text{AIC (Akaike information criterion)} \\ \log(N)/2, & \text{BIC (Bayesian information criterion)} \\ \log(N), & \text{RIC (risk inflation criterion).} \end{cases} \tag{33}$$

Learning using this prior is achieved by maximizing the posterior $p(\boldsymbol{\alpha},\boldsymbol{\beta}|\boldsymbol{t}) \propto p(\boldsymbol{t}|\boldsymbol{\alpha},\boldsymbol{\beta})p(\boldsymbol{\alpha}|\boldsymbol{\beta})p(\boldsymbol{\beta})$. If the basis function set is orthogonal ($\boldsymbol{\Phi}^T\boldsymbol{\Phi} = \boldsymbol{I}$) and the noise precision β is the same for each data point ($\boldsymbol{B} = \beta\boldsymbol{I}$), this prior reduces to

$$p(\alpha_i|\beta) \propto \exp\left(-\frac{c}{1+\alpha_i/\beta}\right). \tag{34}$$

Assuming an uninformative prior for the noise ($p(\beta) = \mathrm{const}$), the use of the sparsity prior of (34) leads to the addition of a normalization term to the marginal log-likelihood of (13)

$$L^o = L - \sum_{i=1}^{M} \frac{c}{1+\alpha_i/\beta}. \tag{35}$$

Keeping only the terms that depend on a single parameter α_i, we can write

$$l_i^o = l_i - \frac{c}{1+\alpha_i/\beta}. \tag{36}$$

Based on this decomposition, an incremental algorithm that maximizes the marginal likelihood has been proposed in [18], which is similar to the typical incremental RVM algorithm [19]. However, because of the sparsity prior, setting the derivative of (36) to zero does not provide analytical updates [such as (30)] for the weight precisions α_i, but instead a numerical solution is used to update them.

In this work, we consider the general case of nonorthogonal basis functions and heteroscedastic noise with different noise precision β_n at each data point. Since $\mathrm{trace}(\boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{B}) = M - \sum_{i=1}^{M} \alpha_i\Sigma_{ii}$, we can write the proposed sparsity prior as

$$p(\boldsymbol{\alpha}|\boldsymbol{\beta}) \propto \exp\left(-c\left(M - \sum_{i=1}^{M} \alpha_i\Sigma_{ii}\right)\right). \tag{37}$$

Learning is again performed by maximizing the posterior $p(\boldsymbol{\alpha},\boldsymbol{\beta}|\boldsymbol{t}) \propto p(\boldsymbol{t}|\boldsymbol{\alpha},\boldsymbol{\beta})p(\boldsymbol{\alpha}|\boldsymbol{\beta})p(\boldsymbol{\beta})$, which leads to adding to the marginal log-likelihood of (13) an additional term that is obtained from (37)

$$L^s = L - c\left(M - \sum_{i=1}^{M} \alpha_i\Sigma_{ii}\right). \tag{38}$$

Setting the derivative of L^s with respect to $\log\alpha_i$ to zero

$$\frac{\partial L^s}{\partial \log\alpha_i} = \frac{1}{2}\left(1 - \alpha_i\Sigma_{ii} - \alpha_i\mu_i^2\right) + c(1 - \alpha_i\Sigma_{ii})\alpha_i\Sigma_{ii} = 0. \tag{39}$$

We obtain the following update formula for α_i:

$$\alpha_i = \frac{\gamma_i}{\mu_i^2 - 2c\gamma_i\Sigma_{ii}}. \tag{40}$$

In the regression case, assuming that $\boldsymbol{B} = \beta\boldsymbol{I}$, we can also update β by setting the derivative of L^s with respect to $\log\beta$ to zero

$$\begin{aligned} \frac{\partial L^s}{\partial \log\beta} &= \frac{1}{2}\left[\frac{N}{\beta} - \|\boldsymbol{t} - \boldsymbol{\Phi}\boldsymbol{\mu}\|^2 - \mathrm{trace}(\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{\Phi})\right] \\ &\quad - \beta c\,\mathrm{trace}(\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{\Phi}) \\ &= 0. \end{aligned} \tag{41}$$

Because of the sparsity prior, we cannot solve this equation analytically. However, we can easily obtain a numerical solution that we use to update β.

Regarding the incremental algorithm, keeping only the terms of L that depend on a single parameter α_i and because $\Sigma_{ii} = 1/(\alpha_i + s_i)$ [19], we obtain

$$l_i^s = l_i - c\left(1 - \frac{\alpha_i}{\alpha_i + s_i}\right) \tag{42}$$

whose gradient is given by

$$\frac{\partial l^s(\alpha_i)}{\partial\alpha_i} = \frac{1}{2}\left[\frac{1}{\alpha_i} - \frac{1}{\alpha_i + s_i} - \frac{q_i^2 - 2cs_i}{(\alpha_i + s_i)^2}\right]. \tag{43}$$

Setting this gradient to zero, we find that $l^s(\alpha_i)$ is maximized at

$$\begin{aligned} &\alpha_i = \frac{s_i^2}{q_i^2 - (2c+1)s_i}, \qquad \text{if } q_i^2 > (2c+1)s_i \\ &\alpha_i = \infty, \qquad \text{if } q_i^2 \le (2c+1)s_i. \end{aligned} \tag{44}$$

III. Kernel Learning

A. Sparse Infinite Linear Models

Consider a linear model of the form (2). Applying a sparsity prior on the weights of this model allows us to use very flexible models, for example, the RVM assumes one kernel function for each training point. We can even consider linear models with infinite number of basis functions

$$y[\boldsymbol{x}|w(\boldsymbol{\xi})] = \int w(\boldsymbol{\xi})\phi(\boldsymbol{x};\boldsymbol{\xi})\,\mathrm{d}\,\boldsymbol{\xi} \tag{45}$$

which are defined by using a family of basis functions $\phi(\boldsymbol{x};\boldsymbol{\xi})$ with parameters $\boldsymbol{\xi}$. Then, $w(\boldsymbol{\xi})$ is a function whose output is the weight for the basis function with parameters $\boldsymbol{\xi}$. In this context, sparsity implies that there will be only a finite number of nonzero weights

$$w(\boldsymbol{\xi}) = \sum_{i=1}^{M} w_i \delta(\boldsymbol{\xi}, \boldsymbol{\theta}_i) \tag{46}$$

where $\delta(\boldsymbol{\xi}, \boldsymbol{\theta}_i) = 1$ if $\boldsymbol{\xi} = \boldsymbol{\theta}_i$, otherwise $\delta(\boldsymbol{\xi}, \boldsymbol{\theta}_i) = 0$. Thus, under the assumption of (46), the sparse infinite linear model is equivalent to a finite linear model with weights $\boldsymbol{w} = (w_1, \ldots, w_M)^T$ and kernel parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_M)^T$

$$y(\boldsymbol{x}|\boldsymbol{w}) = \sum_{i=1}^{M} w_i \phi(\boldsymbol{x};\boldsymbol{\theta}_i). \tag{47}$$

However, learning this model requires not only computing the posterior distribution of the weights $\boldsymbol{w}$ and estimating the weight precisions α_i, but also estimating the basis function parameters $\boldsymbol{\theta}$. This can be achieved by modifying the RVM algorithm in order to optimize the kernel parameters $\boldsymbol{\theta}$ at each iteration.

B. Learning Algorithm

In this section, we propose an algorithm for learning the model of (47). Notice that the typical RVM algorithm cannot be applied here, since it is based on the assumption that $\boldsymbol{\theta}_i$ are fixed in advance. Instead, the proposed algorithm is based on the incremental RVM algorithm, and therefore, it works with only a subset of the basis functions, which are named *active basis functions*. In order to explore the basis function space, there are mechanisms to convert inactive basis functions to active and *vice versa*.

Specifically, at each iteration, we select the most appropriate basis function to add to the model as measured by the increment of the marginal likelihood. Therefore, in order to select a basis function for addition to the model, we perform an optimization of the marginal likelihood with respect to the parameters of the basis function. In a typical RVM, where the basis functions are kernels, this optimization is performed with respect to the locations of the kernels. Furthermore, because the kernels are assumed to be located at the training points, this optimization is discrete. In contrast, an infinite linear model assumes continuous parameters for the basis functions, and therefore, continuous optimization must be employed, which uses the derivatives of the marginal likelihood with respect to the parameters of the basis functions. Furthermore, in contrast to the incremental RVM algorithm, which at each iteration selects a single basis function and it either adds it to the model or reestimates its parameters or removes it from the model, the proposed algorithm performs at each iteration all these three operations; it first attempts to add a basis function to the model, then updates all parameters of active basis functions, and finally, removes any active basis functions that no longer contribute to the model. The additional operations speed up convergence without introducing significant computational cost, since there are only few active basis functions.

Algorithm 1: Sparse Infinite Linear Model 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) Optimize hyperparameters $\boldsymbol{\alpha}$ and noise precision β.
4) Remove from the model any unnecessary active basis functions.
5) Repeat steps 1)–4) until convergence.

The steps of the proposed learning method are summarized in Algorithm 1 and we next discuss them in detail.

1) Select an Inactive Basis Function to Add to the Model: In order to add a basis function to the model, we first need to select a basis function $\phi_i(\boldsymbol{x})$ from the set of inactive basis functions, and then to assign a value to the precision parameter α_i. Both these tasks are performed by maximizing the marginal likelihood

$$(\alpha_i, \phi_i(\boldsymbol{x})) = \arg \max_{(\alpha_i, \phi_i(\boldsymbol{x}))} l_i^s. \tag{48}$$

In the typical incremental RVM method, there is one basis function for each training point, thus the set of candidate basis functions is discrete and finite $\phi_i(\boldsymbol{x}) \in \{\phi_i(\boldsymbol{x})\}_{i=1}^N$. Therefore, selecting a basis function to add to the model requires discrete optimization

$$(\alpha_i, i) = \arg\max\nolimits_{(\alpha_i, i)} l_i^s. \tag{49}$$

This maximization is performed by simply evaluating the marginal likelihood for each candidate basis function $\phi_i(\boldsymbol{x})$, $i = 1, \ldots, N$. In contrast, in the proposed method, we assume that all basis functions have the same parametric form $\phi_i(\boldsymbol{x}) \in \{\phi(\boldsymbol{x};\boldsymbol{\theta}_i) : \boldsymbol{\theta}_i \in \Re^d\}$, therefore, this optimization is now continuous, in the joint space defined by the hyperparameters α_i and the basis function parameters $\boldsymbol{\theta}_i$

$$(\alpha_i, \boldsymbol{\theta}_i) = \arg\max\nolimits_{(\alpha_i, \boldsymbol{\theta}_i)} l_i^s. \tag{50}$$

Notice that we can analytically maximize the marginal likelihood with respect to α_i [see (44)] and we maximize numerically only the basis function parameters $\boldsymbol{\theta}_i$. Specifically, we perform this continuous maximization using the quasi-Newton BFGS method, which requires the derivative for θ_{ik}, given by

$$\frac{\partial l_i^s}{\partial \theta_{ik}} = -\left(\frac{1}{\alpha_i + s_i} + \frac{q_i^2 + c\alpha_i}{(\alpha_i + s_i)^2}\right) r_i + \frac{q_i}{\alpha_i + s_i} w_i \quad (51)$$

where

$$r_i \equiv \frac{1}{2}\frac{\partial s_i}{\partial \theta_{ik}} = \boldsymbol{\phi}_i^T \boldsymbol{C}_{-i}^{-1} \frac{\partial \boldsymbol{\phi}_i}{\partial \theta_{ik}} \quad (52)$$

$$w_i \equiv \frac{\partial q_i}{\partial \theta_{ik}} = \boldsymbol{t}^T \boldsymbol{C}_{-i}^{-1} \frac{\partial \boldsymbol{\phi}_i}{\partial \theta_{ik}} \quad (53)$$

with $\boldsymbol{\phi}_i = (\phi(\boldsymbol{x}_1;\boldsymbol{\theta}_i), \ldots, \phi(\boldsymbol{x}_N;\boldsymbol{\theta}_i))^T$. These derivatives can be efficiently computed in a similar manner as in (28)

$$r_i = \frac{\alpha_i R_i}{\alpha_i - R_i} \quad (54)$$

$$w_i = \frac{\alpha_i W_i}{\alpha_i - W_i} \quad (55)$$

where

$$R_i = \boldsymbol{\phi}_i^T \boldsymbol{C}^{-1} \frac{\partial \boldsymbol{\phi}_i}{\partial \theta_{ik}} \quad (56)$$

$$W_i = \boldsymbol{\phi}_i^T \boldsymbol{C}^{-1} \frac{\partial \boldsymbol{\phi}_i}{\partial \theta_{ik}} \quad (57)$$

which gives

$$R_i = \boldsymbol{\phi}_i^T \boldsymbol{B} \frac{\partial \boldsymbol{\phi}_i}{\partial \theta_{ik}} - \boldsymbol{\phi}_i^T \boldsymbol{B} \boldsymbol{\Phi} \boldsymbol{\Sigma} \boldsymbol{\Phi}^T \boldsymbol{B} \frac{\partial \boldsymbol{\phi}_i}{\partial \theta_{ik}} \quad (58)$$

$$W_i = \boldsymbol{\phi}_i^T \boldsymbol{B} \frac{\partial \boldsymbol{\phi}_i}{\partial \theta_{ik}} - \boldsymbol{\phi}_i^T \boldsymbol{B} \boldsymbol{\Phi} \boldsymbol{\Sigma} \boldsymbol{\Phi}^T \boldsymbol{B} \frac{\partial \boldsymbol{\phi}_i}{\partial \theta_{ik}}. \quad (59)$$

Notice that since we use a local optimization method, we can only attain a local maximum of the marginal likelihood, which depends on the initialization. For this reason, we perform this maximization several times, each time with different initialization, and then we use the parameters that correspond to the best solution. The initialization is randomly performed by sampling from an uninformative (uniform) distribution $p(\theta)$. In order to speed up convergence, we can initially place a basis function with high probability at regions where the model does not fit the data well. For example, if the basis functions are Gaussian kernels, we can initialize the mean $\boldsymbol{m}$ of a Gaussian kernel at a training point $\boldsymbol{x}_n$ selected with probability proportional to the square of the error of the model at that point ϵ_n^2

$$p(\boldsymbol{m} = \boldsymbol{x}_n) = \frac{\epsilon_n^2}{\sum_{n=1}^N \epsilon_n^2}. \quad (60)$$

2) Optimize Active Basis Functions: Although we optimize the parameters of each basis function at the time that we add it to the model, it is possible that the optimal values for the already existing parameters will change, because of the addition of the new basis function. For this reason, after the addition of a basis function, we optimize the parameters α_i and θ_i of all the active basis functions of the current model. Specifically, the weight precision parameters α_i are updated using (30), while the basis function parameters θ_i are updated using an optimization algorithm. Instead of computing separately the derivative for each θ_i from (51), we use the following formula [1]:

$$\frac{\partial L^s}{\partial \theta_{ik}} = \sum_{n=1}^N \frac{\partial L^s}{\partial \phi(\boldsymbol{x}_n;\boldsymbol{\theta}_i)} \frac{\partial \phi(\boldsymbol{x}_n;\boldsymbol{\theta}_i)}{\partial \theta_{ik}} \quad (61)$$

where $(\partial L^s)/(\partial \phi(\boldsymbol{x}_n;\boldsymbol{\theta}_i)) \equiv D_{ni}$ is given by

$$\begin{aligned} \boldsymbol{D} &= (\boldsymbol{C}^{-1}\boldsymbol{t}\boldsymbol{t}^T\boldsymbol{C}^{-1} - \boldsymbol{C}^{-1})\boldsymbol{\Phi}\boldsymbol{A}^{-1} + 2c\boldsymbol{B}\boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{A}\boldsymbol{\Sigma} \\ &= \boldsymbol{B}[(\boldsymbol{t} - \boldsymbol{\Phi}\boldsymbol{\mu})\boldsymbol{\mu}^T - \boldsymbol{\Phi}\boldsymbol{\Sigma}] + 2c\boldsymbol{B}\boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{A}\boldsymbol{\Sigma}. \end{aligned} \quad (62)$$

3) Optimize Hyperparameters and Noise Precision: The hyperparameters $\boldsymbol{\alpha}$ of the active basis functions are updated at each iteration using (40). Similarly, in regression, the noise precision β is updated by numerically solving (41).

4) Remove Basis Functions: After updating the hyperparameters $\boldsymbol{\alpha}$ of the model, it is possible that some of the active basis functions will no longer have any contribution to the model. This happens because of the sparsity property, which allows only few of the basis functions to be used in the estimated model. For this reason, we remove from the model those basis functions that no longer contribute to the estimate, specifically those with $\alpha_i > 10^{12}$. Removing these basis functions is important, not only because we avoid the additional computational cost of updating their parameters, but also because we avoid possible singularities of the covariance matrices due to numerical errors in the updates.

5) Repeat Until Convergence: We assume that the algorithm has converged when the increment of the marginal likelihood is negligible ($\Delta L^s < 10^{-6}$). Because at each iteration we consider only a subset of the basis functions for addition to the model, we assume that convergence has occurred only when the above criterion is met for ten successive iterations.

IV. Numerical Experiments

In this section, we present results from the application of the proposed method (denoted with aRVM) to various artificial and real regression and classification problems. We compare our approach with 1) the typical RVM with Gaussian kernel [1], and 2) the RVM with a smoothness prior and orthogonal wavelet basis functions (denoted with sRVM) [18]. Notice that the sRVM approach is based on wavelet analysis requiring that the training data points are equally spaced. Therefore, it cannot be used for arbitrary multidimensional regression and classification problems and we test it only on 1-D artificial regression example.

More specifically, we consider Gaussian kernel functions of the form

$$\phi(\boldsymbol{x};\boldsymbol{m}_i,h_i) = \exp\left[-h_i^{-2}||\boldsymbol{x} - \boldsymbol{m}_i||^2\right] \quad (63)$$

whose derivatives with respect to the mean and variance parameters are

$$\frac{\partial \phi(\boldsymbol{x}_n;\boldsymbol{m}_i,h_i)}{\partial \boldsymbol{m}_i} = 2h_i^{-2}\phi_i(\boldsymbol{x}_n;\boldsymbol{m}_i,h_i)(\boldsymbol{x}_n - \boldsymbol{m}_i) \quad (64)$$

$$\frac{\partial \phi(\boldsymbol{x}_n;\boldsymbol{m}_i,h_i)}{\partial h_i} = 2h_i^{-3}\phi_i(\boldsymbol{x}_n;\boldsymbol{m}_i,h_i)||\boldsymbol{x}_n - \boldsymbol{m}_i||^2. \quad (65)$$

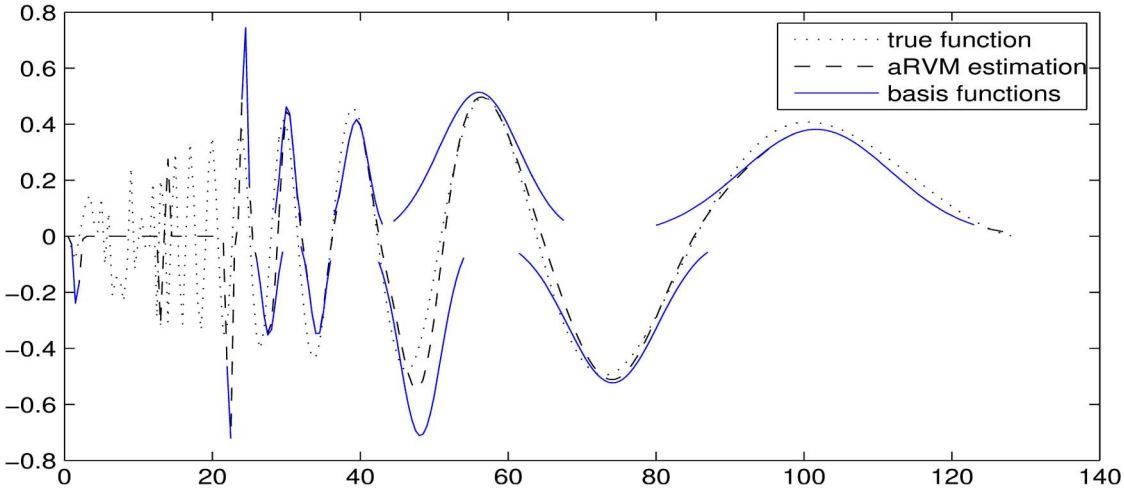

Fig. 1. Basis functions that aRVM uses for estimation of the Doppler signal. The dotted line shows the true signal, the dashed line shows the estimation of aRVM, and the solid lines show the basis functions that aRVM uses (multiplied by their corresponding weight).

Of course, we can use any other type of kernel functions, as long as we can compute the derivatives with respect to the parameters we want to optimize. We can even examine many types of basis functions simultaneously.

In our implementation, we use the quasi-Newton BFGS method to perform the necessary optimizations. Specifically, in order to select a basis function to add to the model, we perform 100 runs of the BFGS, each time starting from a different initialization, and each of these runs lasts no more than ten BFGS iterations. Then, we only keep the best solution and consider adding the corresponding basis function to the current model. When updating the parameters of all the active basis functions, we stop after 100 BFGS iterations.

A. Experiments on Artificial Data

1) Regression: In the first experiment, we generated $N = 128$ points from the well-known "Doppler" function [18]

$$g(x) = \sqrt{x(1-x)}\sin\frac{2\pi(1+\delta)}{x+\delta} \tag{66}$$

with $\delta = 0.01$ and added white Gaussian noise of variance $\sigma^2 = 0.1$.

First we applied the proposed aRVM method setting $c = \log(N)/2$, in order to demonstrate how the proposed method learns the kernels. Fig. 1 presents plots of the true and estimated signal and also the kernels that contribute to this estimation. Notice that kernels with large width are used for the right part of the estimation, where the Doppler function is relatively smooth, while kernels with small width are used for the left part of the estimation, where the Doppler function varies rapidly.

We also applied the three compared methods and evaluated the estimated model on the 128 training points. In order to measure the quality of the estimates, we compute the mean square error $\mathrm{MSE} = \sum_n (g(x_n) - \hat{y}_n)^2/N$, where $\hat{y}_n$ is the estimated value of the function at input x_n and N is the number of data points. For aRVM and sRVM, we set the sparsity parameter to $c = 1$, $c = \log(N)/2$, and $c = \log(N)$ and for the kernel width of RVM, we test several values and select to illustrate the cases $h = 1.5$, $h = 2$, and $h = 4$. The second of these cases $(h = 2)$ is the value that produced the smallest MSE among all tested values of h. The results are shown in Fig. 2. Notice that as the smoothness parameter c increases, the estimated aRVM model contains less basis functions, thus it exhibits robustness to noise. The same happens with the sRVM and also when increasing the width of the kernel in the typical RVM. Also notice that when using the typical RVM with a small kernel size [shown in Fig. 2(b)], noisy estimates are obtained, while when using a large kernel [shown in Fig. 2(h)], large fluctuations of the function (high frequencies) cannot be adequately estimated. Instead, the adaptive RVM and the sRVM [shown in Fig. 2(d) and (f)] can successfully estimate functions that exhibit smoothness in some regions and large fluctuations in other regions. However, the sRVM gives worse solutions in terms of MSE than aRVM, because Gaussian basis functions appear to be more appropriate than wavelets for modeling the "Doppler" signal.

In the next experiment, we compare the performance of aRVM, RVM, and sRVM for several noise levels, using again the "Doppler" function of (66). For aRVM and sRVM, we set the sparsity parameter to $c = \log(N)/2$, and for RVM, we selected to illustrate the cases $h = 1.5$, $h = 2$, and $h = 4$ for the width of the kernel. Notice that $h = 2$ is the optimal value for the width of the kernel when SNR $= 10$. In Fig. 3, we provide the MSE of the estimate of each method for various signal-to-noise ratios (SNRs). Here, we observe that the RVM model with a specific kernel width provides good performance only for a small SNR range. Instead, aRVM and sRVM provide effective models for any SNR value, but aRVM provides consistently better performance than sRVM.

2) Classification: In this section, we compare the typical RVM and adaptive RVM (aRVM) methods on classification problems (sRVM has been proposed only for regression problems). We generated two-class, 2-D, artificial data by obtaining 50 samples from each of the following Gaussian mixture distributions:

$$p(\boldsymbol{x}|C_1) = 0.25\mathrm{N}\left(\boldsymbol{\mu}_1, \sigma_1^2\boldsymbol{I}\right) + 0.75\mathrm{N}\left(\boldsymbol{\mu}_2, \sigma_2^2\boldsymbol{I}\right) \tag{67}$$

$$p(\boldsymbol{x}|C_2) = 0.25\mathrm{N}\left(\boldsymbol{\mu}_2, \sigma_1^2\boldsymbol{I}\right) + 0.75\mathrm{N}\left(\boldsymbol{\mu}_1, \sigma_2^2\boldsymbol{I}\right) \tag{68}$$

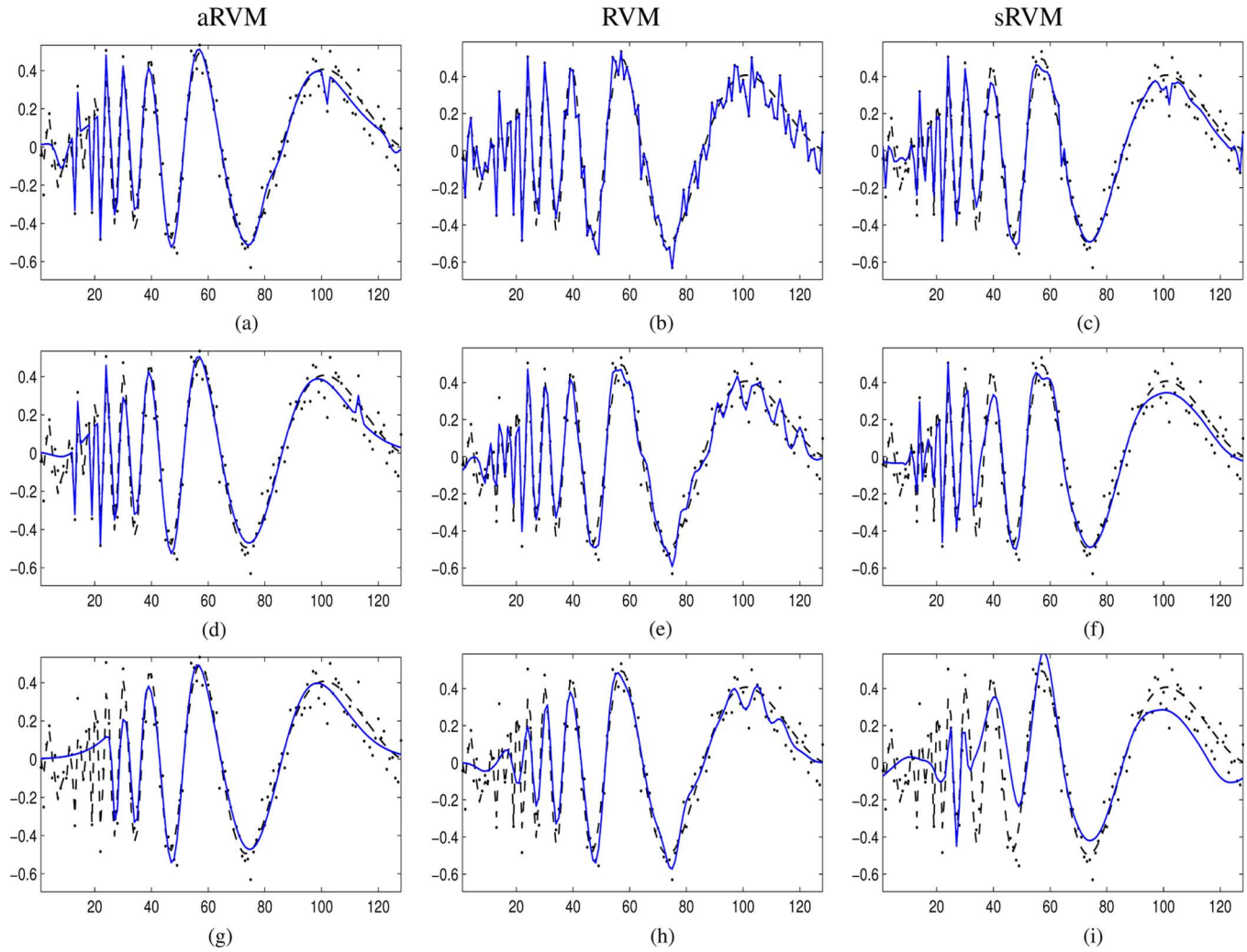

Fig. 2. Regression example with Doppler signal. Estimates obtained (a), (d), and (g) with aRVM; (b), (e), and (h) with RVM; and (c), (f), and (i) with sRVM. The dashed line shows the true signal, the dots are the noisy observations, and the solid line shows the estimate. Under each figure, the values of the kernel width h or sparsity parameter c, the test mean square error (MSE) of the model, and the number of RVs are shown. (a) $c = 1$, MSE $= 0.0037$, and RV $= 16$; (b) $h = 1.5$, MSE $= 0.0056$, and RV $= 57$; (c) $c = 1$, MSE $= 0.0047$, and RV $= 36$; (d) $c = (\log(N))/(2)$, MSE $= 0.0037$, and RV $= 12$; (e) $h = 2$, MSE $= 0.0050$, and RV $= 41$; (f) $c = (\log(N))/(2)$, MSE $= 0.0061$, and RV $= 19$; (g) $c = \log(N)$, MSE $= 0.0092$, and RV $= 5$; (h) $h = 4$, MSE $= 0.0090$, and RV $= 20$; and (i) $c = \log(N)$, MSE $= 0.0254$, and RV$= 4$.

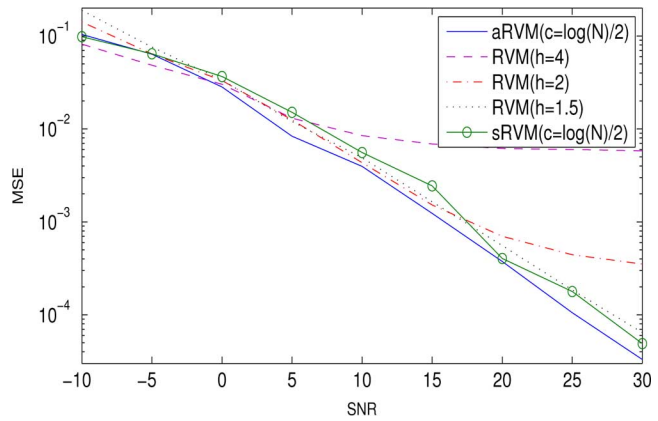

Fig. 3. Comparison of the performance of aRVM, typical RVM, and sRVM for several noise values.

with $\boldsymbol{\mu}_1 = (0.5, 0.5)^T$, $\boldsymbol{\mu}_2 = (-0.5, -0.5)^T$, $\sigma_1^2 = 0.5$, and $\sigma_2^2 = 0.05$. It can be observed that each class consists of two Gaussian clusters, one with large variance and another with small variance. We then trained RVM and aRVM classifiers and evaluated them by computing the percentage of misclassified examples over $N_t = 10\,000$ test points drawn from the mixture distributions of (67) and (68). For aRVM, we set the sparsity parameter to $c = 1$, $c = \log(N)/2$, and $c = \log(N)$, and for RVM, we test several values for the kernel width and select the values $h = 0.2$, $h = 0.4$, and $h = 0.8$, the second of which $(h = 0.4)$ is the value that minimizes the misclassified test examples. Notice in Fig. 4 that when using the typical RVM with a small kernel there is severe noise in the estimation of the decision boundary between the clusters with large variance. Instead, when using a large kernel, the model fails to estimate the decision boundary near the clusters with small variance. On the other hand, when using aRVM, both clusters can be estimated well, because kernels of different width are used. Although the ability to use very small kernels may lead to overfitting, this is avoided by selecting appropriate parameter value for the sparsity controlling prior [Fig. 4(c)].

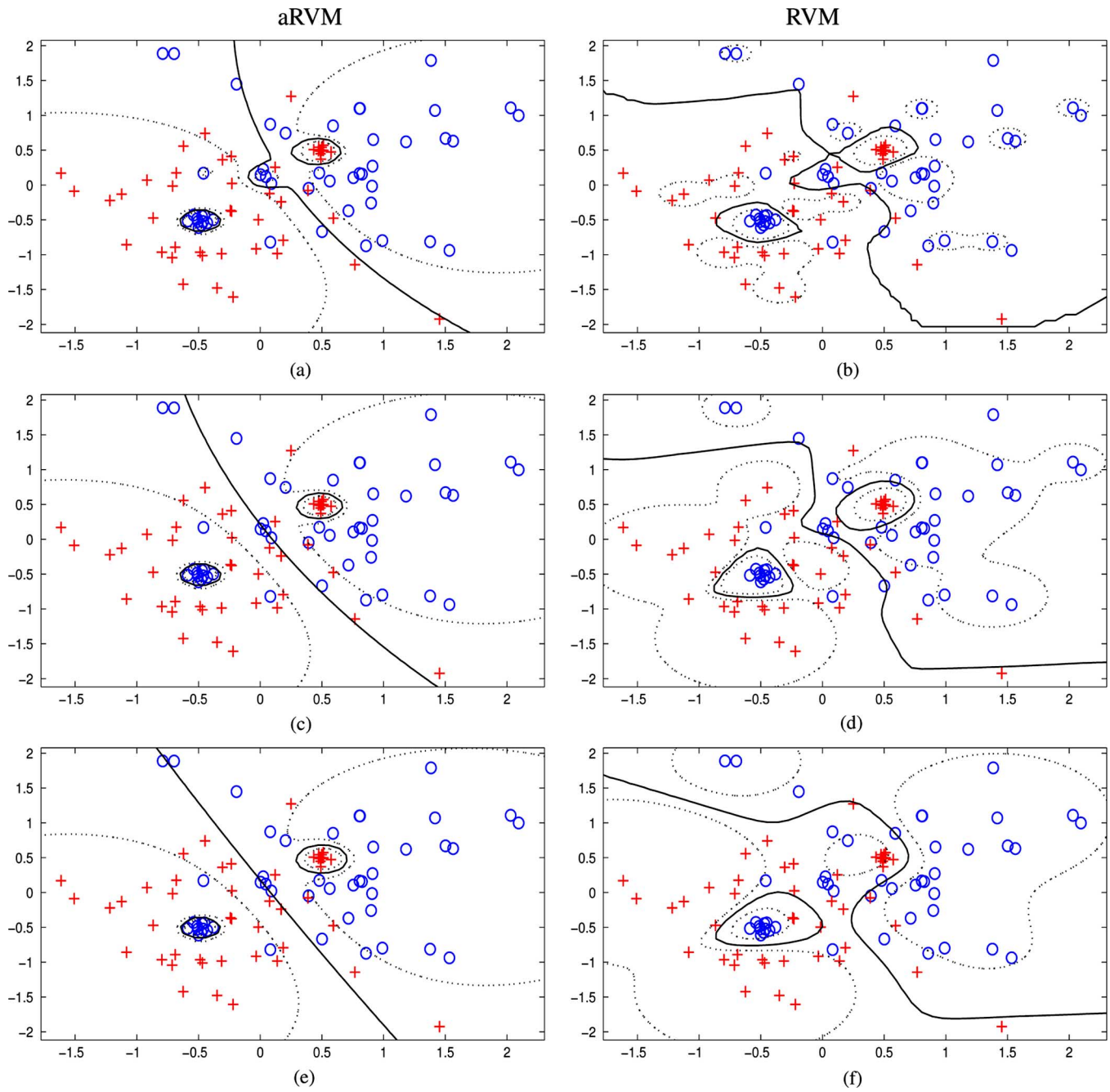

Fig. 4. Classification example on artificial Gaussian clusters. Estimates obtained with (a), (c), and (e) aRVM; and (b), (d), and (f) RVM. Circles and crosses correspond to the data points of the two classes, the solid line shows the decision boundary, and the dotted line shows the curves where the decision probability is 0.75. Under each figure, the values of the kernel width h (for RVM) or sparsity parameter c (for aRVM), the misclassification error (E), and the number of RVs are shown. (a) $c = 1$, $E = 16.71\%$, and RV$= 7$; (b) $h = 0.2$, $E = 19.24\%$, and RV $= 30$; (c) $c = \log(N)/2$, $E = 15.10\%$, and RV $= 4$; (d) $h = 0.4$, $E = 19.91\%$, and RV $= 16$; (e) $c = \log(N)$, $E = 15.24\%$, and RV $= 4$; and (f) $h = 0.6$, $E = 23.74\%$, and RV $= 10$.

B. Experiments on Real Data Sets

In this section, we compare the performance of the proposed method (aRVM) with the typical RVM method on several regression and classification data sets.[2] In what follows, we describe the experimental setup that we followed. For each data set, in order to estimate the generalization error of each method, we perform tenfold cross validation, i.e., we perform ten experiments using each time one fold as a *test set* and the remaining nine folds for training. In each experiment, we test several values for the parameters of the models, specifically, $h = 0.5, 1, 1.5, \ldots, 10$ for the width of RVM and $c = 1$, $c = \log(N)/2$, and $c = \log(N)$ for the sparsity parameter of aRVM. For each parameter value, we train nine models, using one of the nine folds as *validation set* and the remaining eight

[2]Computer Hardware, Concrete, and Pima data sets were obtained from the University of California at Irvine (UCI) Machine Learning Repository at http://archive.ics.uci.edu/ml/, the Boston Housing data set was obtained from http://lib.stat.cmu.edu/datasets/boston, and the Banana, Titanic, Image, and Breast Cancer data sets were obtained from http://ida.first.fraunhofer.de/projects/bench/.

TABLE I
COMPARISON OF aRVM AND RVM ON REGRESSION

			aRVM		RVM	
Dataset	patterns	features	error	RVs	error	RVs
computer hardware	209	6	22379	5.0	30004	140.5
Boston housing	506	13	11.53	13.27	12.48	69.5
concrete	1030	8	34.515	9.10	44.204	140.2

TABLE II
COMPARISON OF aRVM AND RVM ON CLASSIFICATION

			aRVM		RVM	
Dataset	patterns	features	error	RVs	error	RVs
banana	5300	2	0.1126	6.3	0.1092	12.1
titanic	2200	3	0.2270	2	0.2292	31
image	2310	18	0.0387	6.9	0.0390	34.6
breast-cancer	277	9	0.2844	4.4	0.2818	9.6
pima	768	8	0.2303	5.6	0.243	27.9

folds as *training set*. The parameter value providing the best average performance over the nine runs is selected and the corresponding model is subsequently evaluated by measuring the error on the test fold. In regression, the error is the mean square error MSE $= \sum_n (t_n - \hat{y}_n)^2/N$, where t_n is the value given by the test set, $\hat{y}_n$ is the predicted value, and N is the number of test examples. In classification, the error is the percentage of misclassified examples in the test set.

The results in Tables I and II show the cross-validation error and the number of RVs (averaged over ten folds) that were obtained by applying the RVM and aRVM methods on several regression and classification data sets. We can observe that in both regression and classification problems, the solutions obtained with aRVM use much less RVs than the solutions obtained with the typical RVM. Furthermore, in regression, the aRVM method provides more accurate estimates compared to the typical RVM. In the classification data sets, the accuracy of the two methods is generally comparable but the aRVM solution is considerably sparser.

V. DISCUSSION

A. Computational Cost

The computational cost of each iteration of the typical RVM algorithm is dominated by the inversion of the $N \times N$ matrix of (12), which is $O(N^3)$, where N is the number of training points, assuming that we use one basis function at each training point. In the incremental RVM algorithm, the size of the matrix $\boldsymbol{\Sigma}$ is $M \times M$, where M is the number of basis functions that are used in the estimated model and which is much smaller because the model is sparse. The computational cost of the incremental algorithm is dominated by the cost of selecting which basis function to add at each iteration, which is $O(N^2 M)$.

In the proposed aRVM algorithm selection of which basis function to add is achieved using a quasi-Newton optimization method, which is in general more computational expensive as compared to the incremental RVM basis function selection mechanism. However, generally, aRVM requires significantly less iterations, because it adds less basis functions than the incremental RVM. Furthermore, aRVM does not require the additional computational cost of performing cross validation to select the kernel width. The smoothness parameter c can be set to $c = \log(N)/2$, which corresponds to the BIC model selection criterion and which has been observed to give very good results in most problems (this value was also suggested in [18]). Even if we choose to use cross validation to select the smoothness parameter c, we typically need to consider only few values, in contrast to the RVM where selecting the width of the kernel is a much more tedious task.

B. Probabilistic Kernel Interpretation

A GP [17] models an unknown function $y(\boldsymbol{x})$ by assuming that the joint distribution $p(y(\boldsymbol{x}_1), \ldots, y(\boldsymbol{x}_N))$ of any subset of N values of this function $y(\boldsymbol{x})$ is Gaussian. Usually the mean of this Gaussian distribution is assumed zero and the GP is defined by the covariance function $K(\boldsymbol{x}_1, \boldsymbol{x}_2)$, which computes the covariance of the outputs of the function $y(\boldsymbol{x})$ at two arbitrary points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$.

As noted in [1], the marginal distribution of the observations in a sparse linear model is a Gaussian distribution given by $p(\boldsymbol{t}|\alpha, \beta) = N(\boldsymbol{t}|0, C)$ [see (13)], therefore the sparse linear model is a special case of GP, with covariance function given by

$$K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sum_{i=1}^{M} \alpha_i^{-1} \phi_i(\boldsymbol{x}_1) \phi_i(\boldsymbol{x}_2). \tag{69}$$

This covariance function depends directly on the basis functions $\phi_i(\boldsymbol{x})$. Furthermore, assume that the generative model $p(\boldsymbol{x})$ of the inputs $\boldsymbol{x}$ is a mixture model

$$p(\boldsymbol{x}) = \sum_{i=1}^{M} p(i) p(\boldsymbol{x}|i) \tag{70}$$

with the generative distributions $p(\boldsymbol{x}|i)$ proportional to the kernel functions $\phi_i(\boldsymbol{x})$

$$p(\boldsymbol{x}|i) \propto \phi_i(\boldsymbol{x}) \tag{71}$$

and $p(i) \propto \alpha_i^{-1}$. Then, the covariance function $K(\boldsymbol{x}_1, \boldsymbol{x}_2)$ of (69) is proportional to the probability to generate two inputs $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ from the same component $i_1 = i_2$ of the mixture model

$$\begin{aligned} K(\boldsymbol{x}_1, \boldsymbol{x}_2) &\propto p(\boldsymbol{x}_1, \boldsymbol{x}_2 | i_1 = i_2) \\ &= \sum_{i=1}^{M} p(i) p(\boldsymbol{x}_1|i) p(\boldsymbol{x}_2|i). \end{aligned} \tag{72}$$

Such probabilistic interpretation of the kernel function has been used to construct kernels in [23]. Here, it provides useful intuition on the advantages of learning the basis functions. Typically, in order to fit a mixture model to some training set, we learn the mixing coefficients and also parameters of the mixing distributions. However, the typical RVM learns only the mixing coefficients. For this reason, it heavily depends on a good choice of the mixing distributions—they are usually Gaussian kernels

but their variance is unknown. Furthermore, due to computational costs, we cannot consider very large numbers of basis functions, and therefore, typically all the basis functions have common width parameters. In contrast, aRVM which learns parameters of the basis functions can approximate $p(\boldsymbol{x})$ more accurately, because it is a much more flexible model. Therefore, it is important to use the sparsity prior [18] in order to avoid overfitting.

VI. Conclusion

We have presented a learning methodology according to which the parameters of the basis functions of sparse linear models can be determined automatically. More specifically, we assume that the basis functions of this model are kernels and, unlike most kernel methods, for each kernel we learn distinct values for a set of parameters (i.e., location, scale). Because many parameters are adjusted, the proposed model is very flexible. Therefore, to avoid overfitting, we use a sparsity prior that controls the effective number of parameters of the model, in order to encourage very sparse solutions.

The proposed approach has several advantages. First, it automatically learns the parameters of the kernel, therefore there is no need to select them using cross validation. Also, because each kernel may have different parameter values, the model is very flexible and it can solve difficult problems more efficiently than the typical RVM. This was demonstrated in Section IV where we considered regression of a function with varying frequencies and classification of data drawn from a mixture of distributions with very different characteristics. Furthermore, because of the sparsity prior that we use, the obtained models are typically much sparser than the models obtained using the typical RVM.

In this paper, we have assumed that the basis functions are isotropic Gaussian kernels and we learned the location and their width parameters. However, the proposed methodology can be also used for selecting other types of bases. For example, in some problems, it might be advantageous to use anisotropic kernel functions. This approach would assign a separate variance parameter for each feature dimension and each kernel, providing a mechanism to estimate the significance of each feature separately for each kernel. Furthermore, it is possible to simultaneously use several types of kernel functions. Then, when adding a basis function to the model, we would have to consider all types of kernel functions and maximize separately the marginal likelihood for each kernel type. Then, we could add to the model the kernel that provides the largest increase to the marginal likelihood.

References

[1] M. E. Tipping, "Sparse Bayesian learning and the relevance vector machine," *J. Mach. Learn. Res.*, vol. 1, pp. 211–244, 2001.

[2] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York: Springer-Verlag, Aug. 2006.

[3] S.-F. Wong and R. Cipolla, "Real-time adaptive hand motion recognition using a sparse Bayesian classifier," in *Proc. Int. Conf. Comput. Vis./Workshop Human Comput. Interaction*, 2005, pp. 170–179.

[4] A. Agarwal and B. Triggs, "3d human pose from silhouettes by relevance vector regression," in *Proc. Comput. Vis. Pattern Recognit.*, 2004, vol. 2, pp. 882–888.

[5] L. Wei, Y. Yang, R. Nishikawa, M. Wernick, and A. Edwards, "Relevance vector machine for automatic detection of clustered microclassifications," *IEEE Trans. Med. Imag.*, vol. 24, no. 10, pp. 1278–1285, Oct. 2005.

[6] Y. Li, C. Campbell, and M. Tipping, "Bayesian automatic relevance determination algorithms for classifying gene expression data," *Bioinformatics*, vol. 18, no. 10, pp. 1332–1339, 2002.

[7] D. Yang, S. Zakharkin, G. Page, J. Brand, J. Edwards, A. Bartolucci, and D. Allison, "Applications of Bayesian statistical methods in microarray data analysis," *Amer. J. PharmacoGenomics*, vol. 4, no. 1, pp. 53–62, 2004.

[8] A. Lukic, M. Wernickand, D. Tzikas, X. Chen, A. Likas, N. Galatsanos, Y. Yang, F. Zhao, and S. Strother, "Bayesian kernel methods for analysis of functional neuroimages," *IEEE Trans. Med. Imag.*, vol. 26, no. 12, pp. 1613–1624, Dec. 2007.

[9] O. Williams, A. Blake, and R. Cipolla, "Sparse Bayesian learning for efficient visual tracking," *Pattern Anal. Mach. Intell.*, vol. 27, pp. 1292–1304, 2005.

[10] D. Tzikas, A. Likas, and N. Galatsanos, "Large scale multikernel relevance vector machine for object detection," *Int. J. Artif. Intell. Tools*, vol. 16, no. 6, pp. 967–979, 2007.

[11] J. Quiñonero-Candela and L. K. Hansen, "Time series prediction based on the relevance vector machine with adaptive kernels," in *Proc. Int. Conf. Acoust. Speech Signal Process.*, 2002, vol. 1, pp. 985–988.

[12] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semidefinite programming," *J. Mach. Learn. Res.*, vol. 5, pp. 27–72, 2004.

[13] M. Girolami and S. Rogers, "Hierarchic Bayesian models for kernel learning," in *Proc. 22nd Int. Conf. Mach. Learn.*, New York City, NY, 2005, pp. 241–248.

[14] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf, "Large scale multiple kernel learning," *J. Mach. Learn. Res.*, vol. 7, pp. 1531–1565, 2006.

[15] B. Krishnapuram, A. J. Hartemink, and M. A. T. Figueiredo, "A Bayesian approach to joint feature selection and classifier design," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 9, pp. 1105–1111, Sep. 2004.

[16] E. Snelson and Z. Ghahramani, "Sparse Gaussian processes using pseudo-inputs," in *Advances in Neural Information Processing Systems*, Y. Weiss, B. Schölkopf, and J. Platt, Eds. Cambridge, MA: MIT Press, 2006, vol. 18, pp. 1257–1264.

[17] C. E. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA: MIT Press, 2006.

[18] A. Schmolck and R. Everson, "Smooth relevance vector machine: A smoothness prior extension of the RVM," *Mach. Learn.*, vol. 68, no. 2, pp. 107–135, Aug. 2007.

[19] M. Tipping and A. Faul, "Fast marginal likelihood maximisation for sparse Bayesian models," in *Proc. 9th Int. Workshop Artif. Intell. Statist.*, Key West, FL, 2003.

[20] M. E. Tipping and N. D. Lawrence, "A variational approach to robust Bayesian interpolation," in *In Neural Networks for Signal Processing*. New York: IEEE, 2003, pp. 229–238.

[21] A. C. Faul and M. E. Tipping, "Analysis of sparse Bayesian learning," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2001, pp. 383–389.

[22] C. C. Holmes and D. G. T. Denison, "Bayesian wavelet analysis with a model complexity prior," in *Bayesian Statistics 6*, J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, Eds. Oxford, U.K.: Oxford Univ. Press, 1999, pp. 972–978.

[23] D. Haussler, "Convolution kernels on discrete structures," Comput. Sci. Dept., Univ. California, Santa Cruz, CA, Tech. Rep. UCSC-CRL-99-10, Jul. 1999.

Dimitris G. Tzikas received the B.Sc. degree in informatics and telecommunications from the University of Athens, Athens, Greece, in 2002 and the M.Sc. degree from the University of Ioannina, Ioannina, Greece, in 2004, where he is currently working towards the Ph.D. degree at the Department of Computer Science.

His research interests include machine learning, Bayesian methods, and statistical image processing.

Aristidis C. Likas (S'91–M'96–SM'03) received the Diploma degree in electrical engineering and the Ph.D. degree in electrical and computer engineering from the National Technical University of Athens, Athens, Greece, in 1990 and 1994, respectively.

Since 1996, he has been with the Department of Computer Science, University of Ioannina, Ioannina, Greece, where he is currently an Associate Professor. His research interests include neural networks, machine learning, statistical signal processing, and bioinformatics.

Nikolas P. Galatsanos (S'84–M'85–SM'95) received the Diploma in electrical engineering from the National Technical University of Athens, Athens, Greece, in 1982 and the M.S.E.E. and Ph.D. degrees from the Electrical and Computer Engineering Department, University of Wisconsin, Madison, in 1984 and 1989, respectively.

He was on the faculty of the Electrical and Computer Engineering Department, Illinois Institute of Technology, Chicago, from 1989 to 2002. Currently, he is with the Department of Computer Science, University of Ioannina, Ioannina, Greece. He coedited a book titled *Image Recovery Techniques for Image and Video Compression and Transmission* (Norwell, MA: Kluwer, 1998). His research interests center around image processing and statistical learning problems for medical imaging and visual communications applications.

Dr. Galatsanos has served as an Associate Editor for the IEEE TRANSACTIONS ON IMAGE PROCESSING and the IEEE SIGNAL PROCESSING MAGAZINE, and he currently serves as an Associate Editor for the *Journal of Electronic Imaging*.