# Μπεϋζιανές Μέθοδοι για Προβλήματα Μηχανικής Μάθησης και Επεξεργασίας Εικόνας

Η ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

υποβάλλεται στην

ορισθείσα από την Γενική Συνέλευση Ειδικής Σύνθεσης

του Τμήματος Πληροφορικής Εξεταστική Επιτροπή

από τον

Δημήτρη Τζίκα

ως μέρος των Υποχρεώσεων για τη λήψη του

ΔΙΔΑΚΤΟΡΙΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΣΤΗΝ ΠΛΗΡΟΦΟΡΙΚΗ

Ιανουάριος 2009

# Ευχαριστιες

Η παρούσα διδακτορική διατριβή εκπονήθηκε στο Τμήμα Πληροφορικής του Πανεπιστημίου Ιωαννίνων με επιβλέποντα τον Αναπληρωτή Καθηγητή κ. Αριστείδη Λύκα και μέλη της τριμελούς συμβουλευτικής επιτροπής τους κ.κ. Νικόλαο Γαλατσάνο, Καθηγητή του Τμήματος Ηλεκτρολόγων και Μηχανικών και Τεχνολογίας Υπολογιστών του Πανεπιστημίου Πάτρας και Ισαάκ Λαγαρή, Καθηγητή του Τμήματος Πληροφορικής του Πανεπιστημίου Ιωαννίνων, τους οποίους θα ήθελα να ευχαριστήσω για την άριστη συνεργασία τους.

Ιδιαίτερα, θα ήθελα να ευχαριστήσω θερμά τους κ.κ. Αριστείδη Λύκα και Νικόλαο Γαλατσάνο για την επιστημονική τους καθοδήγηση κατά την διάρκεια της εκπόνησης αυτής της διατριβής, για την γενικότερη στήριξη τους κατα την διάρκεια των μεταπτυχιακών μου σπουδών, καθώς και για τις ενέργειες τους που εξασφάλισαν την χρηματοδότηση μεγάλου τμήματος της πραγματοποιηθείσας έρευνας, μέσω ερευνητικών προγραμμάτων. Η βοήθειά τους ήταν καθοριστική για την εκπόνηση αυτής της διατριβής.

Θα ήθελα επίσης να ευχαριστήσω τους διδάκτορες του Τμήματος Πληροφορικής του Πανεπιστημίου Ιωαννίνων Κωνσταντίνο Κωνσταντινόπουλο και Ιωάννη Χάντα και τους υποψήφιους διδάκτορες του ίδιου Τμήματος Βασίλη Χασάνη και Αργύρη Καλογεράτο για την συμβολή τους στην δημιουργία ενός εξαιρετικού ερευνητικού περιβάλλοντος. Τέλος, θα ήθελα να ευχαριστήσω τους γονείς μου Γιώργο και Ρένα, την αδερφή μου Μαρία και τη Δήμητρα που μου συμπαραστάθηκαν και με στήριξαν κατά την εκπόνηση της διατριβής αυτής.

Δημήτρης Τζίκας

Ιωάννινα, Ιανουάριος 2009

# Contents

# List of Figures

# List of Tables

# Περιληψη

Η διατριβή εστιάζεται στο *αραιό Μπεϋζιανό γραμμικό μοντέλο* (sparse Bayesian linear model) για προβλήματα *παλινδρόμησης* (regression) και *ταξινόμησης* (classification) και σε εφαρμογές του σε προβλήματα *επεξεργασίας εικόνας*.

Αρχικά, παρουσιάζεται συνοπτικά η μεθοδολογία για *Μπεϋζιανή συμπερασματολογία*. Στη συνέχεια, προτείνεται ένας υπολογιστικά αποδοτικός αλγόριθμος για το πρόβλημα της αραιής Μπεϋζιανής παλινδρόμησης εικόνων. Ο προτεινόμενος αλγόριθμος χρησιμοποιεί λειτουργίες στο πεδίο του *διακριτού μετασχηματισμού Fourier* και τη μέθοδο βελτιστοποίησης *συζυγών κατευθύνσεων* (conjugate gradient) για να επιτύχει παλινδρόμηση εικόνων με εφικτό υπολογιστικό κόστος. Έπειτα, ο αλγόριθμος αυτός χρησιμοποιείται για την επίλυση του προβλήματος ανίχνευσης αντικειμένων σε εικόνες, προτείνοντας μια παραλλαγή του μοντέλου *Relevance Vector Machine (RVM)* που το ονομάζουμε *multikernel* RVM.

Το αραιό Μπεϋζιανό γραμμικό μοντέλο χρησιμοποιείται στη συνέχεια για να εκτιμήσουμε την *συνάρτηση διασποράς σημείου* της θόλωσης (blurring PSF) στο πρόβλημα της *τυφλής αποσυνέλιξης εικόνων* (blind image deconvolution). Προτείνεται ένα στατιστικό μοντέλο, βασικά πλεονεκτήματα του οποίου είναι η εκτίμηση του μεγέθους της συνάρτησης διασποράς σημείου που περιγράφει την θόλωση, η ανακατασκευή των ακμών της εικόνας και η ανθεκτικότητα στο θόρυβο. Η Μπεϋζιανή συμπερασματολογία υλοποιείται με την χρήση της *variational* προσέγγισης.

Κατόπιν, η διατριβή εστιάζεται στο πρόβλημα επιλογής κατάλληλων *συναρτήσεων βάσης* για το αραιό Μπεϋζιανό γραμμικό μοντέλο, το οποίο είναι σημαντικό ζήτημα προκειμένου να κατασκευάσουμε συστήματα με καλή γενικευτική ικανότητα. Τυπικά, η επιλογή κατάλληλων συναρτήσεων βάσης πραγματοποιείται με την χρήση της τεχνικής *cross-validation*, όμως η τεχνική αυτή έχει υψηλές υπολογιστικές απαιτήσεις και έτσι μπορεί να εφαρμοστεί για την επιλογή του καλύτερου σύνολου συναρτήσεων βάσης, μόνο εάν ο αριθμός των υποψήφιων συνόλων είναι μικρός. Προτείνεται ένας *προσαρμοστικός αλγόριθμος μάθησης των συναρτήσεων βάσης*, ο οποίος είναι ανάλογος με το μοντέλο RVM, αλλά εκτιμά τις παραμέτρους των συναρτήσεων βάσης ταυτόχρονα με την εκπαίδευση του μοντέλου. Πιο συγκεκριμένα, η προτεινόμενη μέθοδος εκτιμά διαφορετικές τιμές για τις παραμέτρους κάθε συνάρτησης βάσης και έτσι προκύπτει ένα πολύ ευέλικτο μοντέλο. Για να αποφευχθεί η υπερεκπαίδευση, επιβάλλεται μια εκ των προτέρων κατανομή που οδηγεί σε αραιές λύσεις, ρυθμίζοντας αυτόματα τον *ουσιαστικό αριθμό παραμέτρων* του μοντέλου. Η προτεινόμενη μεθοδολογία εφαρμόζεται σε διάφορα προβλήματα παλινδρόμησης και ταξινόμησης και χρησιμοποιείται για

την ανάλυση εικόνων *λειτουργικού μαγνητικού συντονισμού* (fMRI).

Επίσης, προτείνεται μια τροποποίηση της προηγούμενης μεθόδου, που χρησιμοποιεί ανισοτροπικές Γκαουσσιανές συναρτήσεις βάσης, με ξεχωριστή παράμετρο κλίμακας (πλάτος) για κάθε χαρακτηριστικό, ώστε να επιτύχει *τοπική επιλογή χαρακτηριστικών*. Η επιλογή χαρακτηριστικών είναι τοπική, με την έννοια ότι υποθέτουμε ότι διαφορετικά χαρακτηριστικά είναι σημαντικά σε διαφορετικές περιοχές του χώρου παραδειγμάτων. Για να απαλείψουμε τα χαρακτηριστικά που δεν είναι χρήσιμα, υποθέτουμε μια κατάλληλη εκ των προτέρων κατανομή για τις παραμέτρους κλίμακας. Οι παραπάνω μεθοδολογίες μάθησης των παραμέτρων των συναρτήσεων βάσης (με ή χωρίς ταυτόχρονη τοπική επιλογή χαρακτηρστικών) χρησιμοποιούνται για την ταξινόμηση δεδομένων από *μικροσυστοιχίες* (microarrays) DNA.

# Abstract

In this thesis, we study the *sparse Bayesian linear model* for *regression* and *classification* tasks and for solving *image processing* problems.

We start with an overview of the *Bayesian inference* methodology and its application to *linear regression*. We then develop a computationally efficient training algorithm for sparse Bayesian regression of images. The proposed training algorithm uses operations in the *Fourier domain* and the *conjugate gradients* method, in order to allow regression of large images at reasonable computational cost. We then apply this algorithm to detect objects in images, using a variant of the *relevance vector machine* (RVM), which uses many types of kernels simultaneously and we call the *multikernel* RVM.

Next, we use the sparse Bayesian linear model to estimate the *point spread function* (PSF) in the blind image deconvolution (BID) problem. We propose a Bayesian model that estimates the support of the blurring PSF, allows reconstruction of image edges and achieves noise robustness. Bayesian inference on this model is performed using the *variational approximation*.

Furthermore, we focus on the problem of selecting appropriate basis functions for the sparse Bayesian linear model, which is crucial in order to achieve good generalization performance. Typically, this problem is tackled using *cross-validation* technique, but this technique is computationally expensive and it can be used to select the best out of a small number of candidate basis function sets. Instead, we propose an *adaptive kernel learning* algorithm, which is similar to the RVM but also learns the parameters of the kernels during model training. More specifically, the proposed method estimates different parameter values for each kernel, resulting in a very flexible model. In order to avoid overfitting, a sparsity enforcing prior is imposed that controls the effective number of parameters of the model. The proposed methodology is evaluated on benchmark regression and classification datasets and applied for analysis of *functional magnetic resonance images* (fMRI).

We also propose a modification of this method that performs *local feature selection* by estimating the parameters of appropriate kernel functions. In order to incorporate local feature selection, anisotropic Gaussian kernels are considered, which use a separate scaling factor (width) for each feature. Feature selection is local, in the sense that different features are assumed to be significant at different regions of the input space. This is achieved because we learn different values for the scaling factors of each kernel. In order to eliminate irrelevant features, we assume a sparsity enforcing prior on the scaling factors

of the kernels. The proposed adaptive kernel learning algorithms (with or without local feature selection) are employed to the problem of classifying *DNA microarray* datasets.

# Chapter 1

# Introduction



## 1.1 Machine Learning Basics

*Machine learning* is the area of artificial intelligence that attempts to give machines the ability to learn from their environment. More specifically, in machine learning problems we want to use a set of observations, which we call the *training set*, in order to make predictions for unseen events. Typically, we separate this problem in two phases; first we use the training set to learn a model of the observations and then we make predictions based on this model. For example, in handwritten digit recognition we are given a training set that consists of handwritten images and corresponding labels that identify the digits that appear on each image. Using this training set we can learn a model that captures the correlations between the observed images and their labels. Then, we can use this model to predict the label of a previously unseen image. The importance of machine learning is that it can be used to solve problems (such as handwritten digit recognition or speech recognition) for which it is difficult to design typical algorithms.

But how do we learn a system using the available observations in the training set? A common approach is to consider a parametric function (model) that is used to describe the process that generates the observed data. Then, during the training phase, we estimate the parameters of this model by maximizing some objective function. Under a stochastic framework, it is convenient to assume that the observations are random variables and define an appropriate parametric *probability density function* (pdf) for them. Then, we can define the objective function for learning as the *likelihood* of the observations, which is the probability that the observations have been generated by specific

values of the parameters. Estimation of the parameters is performed by maximizing the likelihood of the observations in a process known as *maximum likelihood* estimation. Alternatively, *Bayesian methods* assume that the parameters of the model are also random variables. This framework provides an elegant way to apply constraints on the parameters by assigning suitable prior distributions to them. More importantly, the Bayesian framework provides a principled methodology to measure uncertainties of the parameter estimates and propagate these uncertainties to the predictions made using this model. These Bayesian learning methodologies are discussed in detail in Chapter 2.

An important issue to note is that in order for a model to generalize the observations of the training set to unseen examples, it needs to make certain assumptions for the mechanism that generates the observations. These assumptions are called the *bias* of the model. For example, a common assumption is that similar inputs should be mapped to similar outputs. The more assumptions a model makes, the larger the bias is. Models with very large bias generally have poor performance, because they make too many assumptions, which are unlikely to be realistic. On the other hand, models with very small bias make too few assumptions and for this reason their predictions are heavily affected by noise that commonly exists in the observations. How much a machine learning model is affected by noise of the training examples is measured by the *variance* of the method. More specifically, the *generalization* performance of a model increases as the bias and variance gets smaller. However, there is a bias-variance tradeoff; the larger the bias of an algorithm, the smaller its variance. For this reason it is important to measure the generalization performance of a machine learning method. This is commonly achieved by comparing the prediction of the method to the known labels of a separate set observations, which is called the *test set* and should be independent of the training set.

Machine learning methods are divided in two broad categories depending on the level of supervision that they require. *Supervised learning* methods assume that the observations have the form of pairs, containing inputs and corresponding outputs. The aim is to build a model that can be used to make predictions for the outputs of previously unseen inputs. On the other hand, *unsupervised learning* methods assume a training set that only consists of observed inputs. They learn a model of these inputs, which can later be used for example to predict *missing values* of some of the observations, or to group similar observations in *clusters*. *Semi-supervised* machine learning methods combine characteristics of both supervised and unsupervised methods. These methods, require that some of the input observations are associated with the corresponding desired output, but they can also take advantage of available input observations whose corresponding output is unknown.

Supervised methods are further divided in two categories depending on the type of the outputs. *Classification* methods assume that outputs are labels that describe the category which the observation belongs to. For example, handwritten digit recognition and speech recognition are examples of classification problems. In contrast, if the outputs are continuous variables, the problem is known as *regression*. For example, predicting the temperature based on some other measurements, predicting the value of a stock based on

its previous values and estimating the value of an image pixel given its neighboring pixels are regression problems.

When designing classification methods there are two general approaches; the *discriminative* and the *generative* approach. In the discriminative approach we attempt to find a function that directly discriminates the categories. For example, in binary classification, where we assume only two categories, we may discriminate the two categories based on the sign of the output of some function, whose parameters are estimated during the training phase. Instead, when using the generative approach we attempt to learn for each category a (probability distribution) function that describes the mechanism that generates its data. Then, we can make predictions for unseen data using the Bayes law. Generative methods solve a more general problem than required, since they also estimate a model of how the observations of each category are generated. Thus, they typically require larger training sets compared to discriminative methods, but they provide a framework to overcome other difficulties, such as missing data in the training set. Furthermore, it is usually straightforward to extend discriminative methods for regression problems. In contrast generative methods cannot be used for regression tasks, since the outputs in regression are continuous and therefore they can take an infinite range of values (a generative method would attempt to estimate the generative model for each possible output value).

The most common assumption that supervised learning methods make is that similar inputs should be mapped to similar outputs. However, it is not always straightforward how to define similar inputs. Many simple algorithms measure similarity between two input points $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ using their inner product $\boldsymbol{x}_1^T\boldsymbol{x}_2$. However, this is usually insufficient and more flexible similarity representations can be obtained using a *mapping function* $\psi(\boldsymbol{x})$ to map the inputs to some feature space of usually higher dimension. Then similarity of the input points can be computed as the inner product $\psi(\boldsymbol{x}_1)^T\psi(\boldsymbol{x}_2)$ in the new feature space. For example, instead of working directly with the two dimensional input $\boldsymbol{x} = (x_1, x_2)^T$ we can map it to the feature vector $\psi(\boldsymbol{x}) = (x_1^2, x_1x_2, x_2^2)^T$. Furthermore, we notice that inner products in the feature space can be efficiently computed using a *kernel function* $K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \psi(\boldsymbol{x}_1)^T\psi(\boldsymbol{x}_2)$, which in the above example is $K(\boldsymbol{x}_1, \boldsymbol{x}_2) = (\boldsymbol{x}_1^T\boldsymbol{x}_2)^2$. In summary, it is often useful to measure similarity using kernel functions rather than the typical inner product approach. Such methods that use kernel functions are known as *kernel methods* and have become very popular for solving regression and classification tasks. However, their performance depends largely on selecting an appropriate kernel function, which generally is an empirical task.

A property of kernel-based machine learning methods that has lately attracted a lot of interest is *sparsity*. Kernel classifiers make predictions for a new input point based on its similarity with the input points of the training set. The key observation in developing sparse methods is that it is typically redundant to consider the similarity with all the input points of the training set. Instead, sparse methods consider the similarity with only a small subset of the input points, which are selected during the training phase. Popular sparse kernel classifiers are the *support vector machine* (SVM), the *relevance vector machine*

(RVM) and several sparse approximations of *Gaussian processes*. In Chapter 3 we describe in detail the linear model and explain how to obtain sparse estimations and in Chapter 6 we propose a modification that learns parameters of the kernel function simultaneously with the parameters of the model.

## 1.2 Image Processing Problems

Nowadays, computer systems have excessive storage and computational capabilities that allow storage and processing of multitmedia content. A lot of interest is paid in applications that process and store images, since images are a very important type of information.

There are several problems of interest in the image processing field. For example, *edge detection* is the problem of detecting the edges (discontinuities) in a given image. *Image segmentation* is the problem of segmenting the image into regions that correspond to different objects. *Image registration* is the problem of aligning two or more images. In *image denoising* we are given an observed image that has been corrupted by additive noise and we want to estimate the initial image. *Image restoration* is a similar problem, where the observed image has been degraded by known blur and addition of some noise source and we want to restore the original image. The problem is known as *blind image deconvolution* when the type of blur is unknown. In *image recognition* we want to identify what type of object is depicted in a given image and in the more specific problem of *face recognition* we wish to recognize individuals using images of their face. *Image detection* is a slightly different problem, were we want to identify all occurrences of a target object in an observed image and we also want to find the locations were they appear. In the *tracking* problem, we are given sequences of images (video), and after detecting an object we attempt to keep track of it. In *image retrieval* we assume to have a database of images and we want to retrieve images that are similar to some other given image. Furthermore, many problems of interest involve processing of *medical images*.

A large variety of image processing methods have been developed in order to treat such problems. Many of these methods are based on the *discrete Fourier trasform* (DFT) that describes images based on their frequency spectrum. Another useful tool that has been more recently used for image processing, is the *discrete Wavelet trasnform* (DWT), which combines information in the spatial and frequency domains. Furthermore, machine learning methods have been used with great success, because many of the previously mentioned problems can be tackled by learning a model using a set of training examples.

An image can be mathematically defined as a function $f : \Omega \to I^c$ that maps input points (pixels) from its support $\Omega \subset \mathbb{R}^2$ to a color space $I^c \subset \mathbb{R}^c$, where $\mathbb{R}$ is the set of real numbers. The support $\Omega$ of the image commonly has rectangular form $\Omega = \{(x, y) : x \in [x_{min}, x_{max}], y \in [y_{min}, y_{max}]\}$, where $x_{min}$, $x_{max}$, $y_{min}$ and $y_{max}$ are minimum and maximum values for both image dimensions. Color images are usually encoded using three channels that correspond to the intensities of the three basic colors (red, green

blue), therefore the color space is $I^c = \mathbb{R}^3$. On the other hand, grayscale images assume only one channel ($I^c = \mathbb{R}$) that corresponds to the grayscale intensity. Computers can only process digital representations of images and for this reason, the image support and color space cannot be infinite sets. This limitation is overcome by quantizing the image support and color space, which can then be assumed to be integer sets, $\Omega \subset \mathbb{Z}^2$ and $I^c \subset \mathbb{Z}^c$, where $\mathbb{Z}$ is the set of integers. The elements of the image support are then called image pixels. A specific characteristic of images, in contrast to arbitrary functions, is that image pixels lie on a uniform grid, which means that they are not randomly distributed, but they have equal distances between their neighbors. For this reason, we can represent images as vectors that contain the intensity values at all the pixels in some specified order, without explicitly representing the pixel location. In this thesis we will only treat grayscale images. However, it should not be difficult to extend the proposed methods in order to deal with the case of color images.

## 1.3 Thesis Contribution

The contribution of this thesis is twofold. On one hand, we focus on the problem of selecting appropriate basis functions for the sparse Bayesian linear model and we propose methods that automatically learn the parameters of the basis functions. On the other hand, we develop computationally efficient training algorithms for applying the sparse Bayesian linear model to regression problems on images and we treat image processing problems, such as object detection, blind image deconvolution and analysis of functional neuroimages.

In Chapter 2 we provide an overview of the Bayesian inference methodology [Tzikas et al., 2008b], which will be used in the following chapters. More specifically, we discuss *graphical models* that provide a powerful approach to visualize the random variables of a stochastic model and the dependencies among them. We then describe how *exact Bayesian inference* can be achieved if we use *conjugate priors*, and how to estimate the parameters of a Bayesian model using the *expectation maximization* (EM) algorithm. Finally, we describe the *maximum a posteriori* (MAP) principle and the *variational Bayesian approximation*, which can be used with rather complex Bayesian models. The application of these methodologies to the linear regression problem is shown in Chapter 3. We first describe the maximum likelihood approach and explain its drawbacks. We then impose conjugate Bayesian priors on the parameters of the linear model to derive a model that can be solved exactly using the EM algorithm. Furthermore, we describe how the prior distribution can be modified in order to obtain *sparse* estimations using the variational Bayesian approximation. Finally, we discuss how the sparse linear model has been used to solve classification problems.

In Chapter 4 we consider a specific case of the linear model, which we call the *multikernel* RVM and which is a variant of the well-known relevance vector machine (RVM)

model, but uses many types of kernels simultaneously [Tzikas et al., 2006b, 2007b]. We then propose a fast algorithm that can be used for sparse Bayesian linear regression of large scale images. The proposed algorithm uses operations in the Fourier domain in order to allow regression of large images with reasonable computational cost. We then use this method to detect objects in images and simultaneously determine their locations.

In Chapter 5 we present a new Bayesian approach for the blind image deconvolution (BID) problem [Tzikas et al., 2006a, 2007a,c, a]. The main novelty of this approach is the use of the sparse Bayesian linear model for the blurring point spread function (PSF) that allows estimation of both PSF shape and support. In the proposed approach, a robust model of the BID errors and an image prior that preserves edges of the reconstructed image are also used. Sparseness, robustness and preservation of edges are achieved by using priors that are based on the Student's t probability density function (PDF). This pdf, in addition to having heavy tails, is closely related to the Gaussian and thus yields tractable inference algorithms. The approximate variational inference methodology is used to solve the corresponding Bayesian model. Numerical experiments are presented that compare this BID methodology to previous ones using both simulated and real data.

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 Chapter 6 we propose an incremental method for kernel-based supervised learning, which is similar to the Relevance Vector Machine (RVM), but it also learns the parameters of the kernels during model training [Tzikas et al., 2008a, b]. 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 datasets. Furthermore, we apply the proposed approach to model spatial correlations of the activation signal in functional neuroimaging. Numerical results with an artificial phantom show that, in contrast to previous approaches, the proposed model can simultaneously detect the presence of activations that are i) strong but small and ii) large but weak.

In Chapter 7 we propose a modification to the kernel learning method of Chapter 6 that performs local feature selection simultaneously with model inference. In order to incorporate local feature selection, appropriate kernels need to be used; we consider Gaussian anisotropic kernels, which use a separate scaling factor (width) for each feature. Because we learn different values for the scaling factors of each kernel, feature selection is local, i.e. different features are assumed to be significant at different regions of the input space. In order to eliminate irrelevant features, we impose a sparsity enforcing prior on the scaling factors of the kernels. Experimental results show that the proposed method has improved performance in several regression and classification benchmark datasets.

Furthermore, we consider the classification task with biological DNA microarray data-

sets, where feature selection is very important, sinse the microarray examples are of very high dimension. The proposed methodology first applies two popular (and computationally efficient) feature selection approaches, namely recursive feature elimination (RFE) and automatic relevance determination (ARD), to initially reduce the number of features. Then, using the remaining features, we apply both the adaptive RVM with kernel learning and the local feature selection approach and examine their performance. Experimental results indicate that the adaptive kernel learning algorithm of Chapter 6 exhibits superior classification performance compared to the commonly used RVM model. Furthermore, the proposed local feature selection approach has similar performance and may be useful in identifying which genes are significant for the classification task.

# Chapter 2

# Bayesian Inference



## 2.1 Introduction

Statistical models are collections of random variables, whose behavior is determined by their joint probability distribution. Typically, these random variables can be distinguished as *observed* $\boldsymbol{x}$ and *hidden* $\boldsymbol{z}$ random variables, depending on whether they are included in the training set or not. Furthermore, it is common to use parameters $\boldsymbol{\theta}$ in order to define the joint distribution model $p(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{\theta})$ of the random variables. In this chapter we will discuss methodologies that can be used to achieve i) *inference* of the hidden random variables and ii) *estimation* of the model parameters. Hereafter, the term inference will be used to refer to the computation of the posterior distribution $p(\boldsymbol{z}|\boldsymbol{x})$ of the hidden random variables $\boldsymbol{z}$, while the term estimation will refer to the process of assigning appropriate values to the parameters $\boldsymbol{\theta}$, using the observations $\boldsymbol{x}$. We will later see that exact inference can be achieved when using conjugate prior distributions. However, in many cases of interest it is advantageous to use more complicated prior distributions, even though we need to resort to approximate inference methods.

Many simple statistical models do not use any hidden variables and it is easy to compute the *likelihood function* $p(\boldsymbol{x}; \boldsymbol{\theta})$ that describes the probabilistic relationship between the observations $\boldsymbol{x}$ and the parameters $\boldsymbol{\theta}$. In this case estimation of the model parameters

is commonly performed using the popular *maximum likelihood* (ML) approach. According to this approach, the ML estimate is obtained as

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, p(\boldsymbol{x};\boldsymbol{\theta}). \tag{2.1}$$

However, in many problems of interest, direct assessment of the likelihood function $p(\boldsymbol{x};\boldsymbol{\theta})$ is complex and is either difficult or impossible to compute it directly or optimize it. In such cases the computation of this likelihood is greatly facilitated by the introduction of *hidden variables* $\boldsymbol{z}$. These random variables act as "links" that connect the observations to the unknown parameters via Bayes' law. The choice of hidden variables is problem dependent. However, as their name suggests, these variables are not observed and they provide sufficient information about the observations so that the conditional probability $p(\boldsymbol{x}|\boldsymbol{z};\boldsymbol{\theta})$ and therefore the joint distribution $p(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\theta})$ are easy to compute.

Once the hidden variables $\boldsymbol{z}$ and a prior probability for them $p(\boldsymbol{z};\boldsymbol{\theta})$ have been introduced, one can obtain the likelihood of the observations $\boldsymbol{x}$ or the *marginal likelihood* as it is usually called, by integrating out (marginalizing) the hidden variables $\boldsymbol{z}$:

$$p(\boldsymbol{x};\boldsymbol{\theta}) = \int p(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{z} = \int p(\boldsymbol{x}|\boldsymbol{z};\boldsymbol{\theta})p(\boldsymbol{z};\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{z} \tag{2.2}$$

This seemingly simple integration is the core of the Bayesian methodology, since in this manner we can obtain both the likelihood function, and by using Bayes' theorem, the posterior of the hidden variables:

$$p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}) = \frac{p(\boldsymbol{x}|\boldsymbol{z};\boldsymbol{\theta})p(\boldsymbol{z};\boldsymbol{\theta})}{p(\boldsymbol{x};\boldsymbol{\theta})}. \tag{2.3}$$

As it will be shown later, if the posterior distribution of the hidden variables $p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta})$ can be analytically computed, the parameters of Bayesian models can be estimated using the EM algorithm, which iteratively maximizes the likelihood function without explicitly computing it.

In many cases of interest the posterior distribution is not available, because the integral in (2.2) is either intractable or very difficult to compute in closed form. Thus, the main effort in Bayesian inference is concentrated on techniques that allow us to bypass or approximately evaluate this integral. Such methods can be classified into two broad categories. The first contains numerical sampling methods also known as *Monte Carlo* [Robert and Casella, 2005, Andrieu et al., 2003] techniques and the second category concerns deterministic approximations of the integral, such as the *Variational Bayes* methodology.

## 2.2 Graphical Models

Graphical Models provide a framework for representing dependencies among the random variables in a statistical modelling problem. They constitute a comprehensive and elegant way to graphically represent the interaction among the entities involved in a probabilistic system. A graphical model is a graph whose nodes correspond to the random variables of a problem and the edges represent the dependencies among the variables. A directed edge from a node $A$ to a node $B$ in the graph indicates that the variable $B$ stochastically depends on the value of the variable $A$. Graphical models can be either directed or undirected. In the second case they are also known as *Markov Random Fields* [Bishop, 2006, Borgelt and Kruse, 2002, Neapolitan, 2003]. We will focus on directed graphical models also called *Bayesian Networks*, where all the edges are considered to have a direction from parent to child denoting the conditional dependency among the corresponding random variables. In addition we assume that the directed graph is acyclic (i.e. it contains no cycles).

Let $G = (V, E)$ be a directed acyclic graph with $V$ being the set of nodes and $E$ the set of directed edges. Let also $x_s$ denote the random variable associated with node $s$ and $\pi_s$ the set of parents of node $s$. Associated with each node $s$ is also a conditional probability density $p(x_s|x_{\pi_s})$ that defines the distribution of $x_s$ given the values of its parent variables. Therefore, for a graphical model to be completely defined, apart from the graph structure, the conditional probability distribution at each node should also be specified. Once these distributions are known, the joint distribution over the set of all variables can be computed as the product:

$$p(x) = \prod_s p(x_s|x_{\pi_s}) \tag{2.4}$$

The above equation constitutes a formal definition of a directed graphical model [Bishop, 2006] as a collection of probability distributions that *factorize* in the way specified in the above equation (which of course depends on the structure of the underlying graph).

In Fig. 2.1 we show an example of a directed Graphical Model. The random variables at the nodes are $a$,$b$,$c$ and $d$. Each node computes a conditional probability density that quantifies the dependency of the node from its parents. The conditional densities at a node $i$ may not be exactly known and may be parameterized by a set of parameters $\theta_i$. Using the chain rule of probability, we would write the joint distribution as:

$$p(a, b, c, d; \boldsymbol{\theta}) = p(a; \theta_1)p(b|a; \theta_2)p(c|a, b; \theta_3)p(d|a, b, c; \theta_4) \tag{2.5}$$

However, we can simplify this expression by taking into account the independencies that the graph structure implies. In general, in a graphical model each node is independent of its ancestors given its parents. This means that node $c$ does not depend on node $a$ given node $b$, and node $d$ does not depend on node $a$ given nodes $b$ and $c$. Thus, from (2.4)

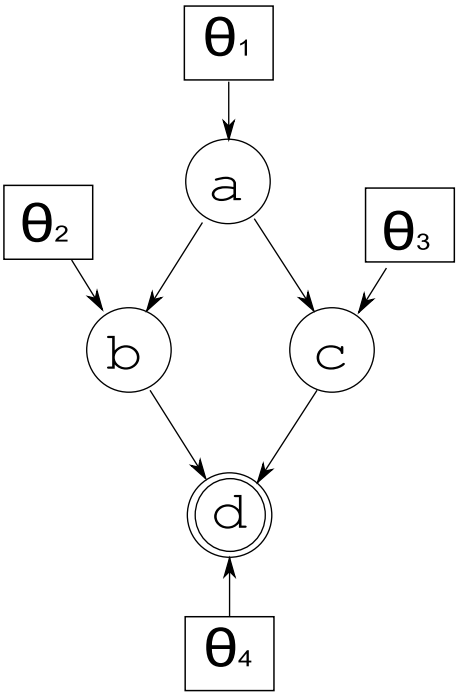



Figure 2.1: Example of directed Graphical Model. Nodes denoted with circles correspond to random variables, while nodes denoted with squares correspond to parameters of the model. Doubly circled nodes represent observed random variables, while single circled nodes represent hidden random variables.

we can write:

$$p(a, b, c, d; \boldsymbol{\theta}) = p(a; \theta_1)p(b|a; \theta_2)p(c|a; \theta_3)p(d|b, c; \theta_4), \tag{2.6}$$

which is also obtained by applying (2.4).

Once a graphical model is completely determined (i.e. all parameters have been specified), then several *inference problems* could be defined, such as computing the marginal distribution of a subset of random variables, computing the conditional distribution of a subset of variables given the values of the rest variables and computing the maximum point in some of the previous densities. In the case where the graphical model is parametric, then we have the problem of *learning* appropriate values of the parameters given some dataset with observations. Usually, in the process of parameter learning, several inference steps are also involved.

## 2.3 Bayesian Inference with Conjugate priors

Conjugate priors play an important role in facilitating Bayesian calculations. More specifically, assume a Bayesian model with hidden variables $\boldsymbol{z}$, observed variables $\boldsymbol{x}$, prior distribution $p(\boldsymbol{z})$ and conditional likelihood $p(\boldsymbol{x}|\boldsymbol{z})$. Then, the marginal likelihood is given

by

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x}, \boldsymbol{z}) \, \mathrm{d}\boldsymbol{z} = \int p(\boldsymbol{x}|\boldsymbol{z}) p(\boldsymbol{z}) \, \mathrm{d}\boldsymbol{z}, \tag{2.7}$$

which cannot always be computed analytically. However, the marginal likelihood is involved in the computation of the posterior distribution $p(\boldsymbol{z}|\boldsymbol{x})$ of the hidden variables $\boldsymbol{z}$, which is computed according to

$$p(\boldsymbol{z}|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}{p(\boldsymbol{x})}, \tag{2.8}$$

and is required in order to proceed with Bayesian inference. Thus, it is important to find a prior $p(\boldsymbol{z})$ such that when multiplied with a likelihood distribution $p(\boldsymbol{x}|\boldsymbol{z})$ allows analytical computation of the marginalization integral of (2.7). A common practice is to choose the prior distribution such that it has the same form as the likelihood, so that the resulting posterior distribution $p(\boldsymbol{z}|\boldsymbol{x})$ has also the same form as the likelihood $p(\boldsymbol{z}|\boldsymbol{x})$, when viewed as a function of the hidden variables. Such prior distributions allow closed form marginalization of the hidden variables and are called *conjugate* to the likelihood distribution.

For example, consider a Gaussian conditional likelihood with zero mean and whose precision (inverse variance) is given by a hidden variable $\alpha$:

$$p(x|\alpha) = (2\pi)^{-1/2} \alpha^{1/2} \exp(-\frac{1}{2}\alpha x^2). \tag{2.9}$$

This likelihood, when viewed as a function of $\alpha$, has the form of a Gamma pdf defined as

$$p(\alpha; a, b) = \frac{b^a}{\Gamma(a)} \alpha^{a-1} \exp(-b\alpha). \tag{2.10}$$

Thus, the marginalization of the precision $\alpha$ of a Gaussian pdf (when a Gamma conjugate prior is used for it) is possible in closed form according to

$$p(x; a, b) = \int p(x|\alpha) p(\alpha; a, b) \, \mathrm{d}\alpha = \frac{\Gamma(a + 1/2)}{\Gamma(a)} \frac{b^a}{(2\pi)^{1/2}} \left( b + \frac{x^2}{2} \right)^{-(a+\frac{1}{2})}, \tag{2.11}$$

and gives the Student's t pdf. Unfortunately, given an arbitrary likelihood distribution, a conjugate prior distribution does not always exist. Table 2.1 shows the formula of some common distributions, and Table 2.2 shows their conjugate prior distributions and the resulting posteriors.

## 2.4 The Expectation Maximization (EM) Algorithm

In the case of statistical models where inference is tractable, therefore the posterior distribution of the hidden variables can be analytically computed, estimation of the parameters

| Distribution | pdf |
|---|---|
| Normal | $\mathrm{N}(\boldsymbol{x}\|\boldsymbol{\mu},\boldsymbol{\Sigma}) = (2\pi)^{-n/2}\|\boldsymbol{\Sigma}\|^{-1/2}\exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right]$ |
| Gamma | $\mathrm{Gamma}(x\|a,b) = b^a\Gamma(a)^{-1}x^{a-1}e^{-bx}$ |
| Wishart | $\mathrm{Wishart}(\boldsymbol{X}\|\nu,\boldsymbol{V}) = \frac{\|\boldsymbol{X}\|^{(\nu-d-1)/2}\exp(\mathrm{trace}-\frac{1}{2}\boldsymbol{V}\boldsymbol{X})}{2^{\nu d/2}\pi^{d(d-1)/4}\boldsymbol{V}^{-n/2}\prod_{i=1}^{d}\Gamma(\nu+1-i)/2}$ |
| Multinomial | $\mathrm{Mult}(\boldsymbol{x}\|\boldsymbol{\pi}) = \frac{(\sum_{i=1}^{n}x_i)!}{\prod_{i=1}^{n}x_i!}\prod_{i=1}^{n}\pi_i^{x_i}$ |
| Dirichlet | $\mathrm{Dirichlet}(\boldsymbol{x}\|\boldsymbol{\alpha}) = \frac{\Gamma\left(\sum_{j=1}^{M}\alpha_j\right)}{\prod_{j=1}^{M}\Gamma(\alpha_j)}\prod_{j=1}^{M}x_j^{\alpha_j-1}$ |

Table 2.1: Formulas for some common probability distribution functions.

| Likelihood | Conjugate Prior | Posterior |
|---|---|---|
| $\mathrm{N}(\boldsymbol{x}\|\boldsymbol{\mu},\boldsymbol{\Sigma})$ | $\mathrm{N}(\boldsymbol{\mu}\|\boldsymbol{\mu}_0,\boldsymbol{\Sigma}_0)$ | $\mathrm{N}(\boldsymbol{\mu}\|\tilde{\boldsymbol{\mu}},\tilde{\boldsymbol{\Sigma}}),\ \tilde{\boldsymbol{\mu}} = \tilde{\boldsymbol{\Sigma}}(\boldsymbol{\Sigma}_0^{-1}\boldsymbol{\mu}_0 + n\boldsymbol{\Sigma}^{-1}\bar{\boldsymbol{x}}),\ \tilde{\boldsymbol{\Sigma}} = (\boldsymbol{\Sigma}_0^{-1} + n\boldsymbol{\Sigma}^{-1})^{-1}$ |
| $\mathrm{N}(x\|\mu,\sigma^2)$ | $\mathrm{Gamma}(\sigma^{-2}\|a,b)$ | $\mathrm{Gamma}\left(\sigma^{-2}\|a+n/2, b+\sum_{i=1}^{n}(x_i-\mu)^2/2\right)$ |
| $\mathrm{N}(\boldsymbol{x}\|\boldsymbol{\mu},\boldsymbol{\Sigma})$ | $\mathrm{Wishart}(\Sigma^{-1}\|\nu,\boldsymbol{V})$ | $\mathrm{Wishart}\left(\Sigma^{-1}\|\nu+n, \boldsymbol{V}+\sum_{i=1}^{n}(\boldsymbol{x}_i-\boldsymbol{\mu})^T(\boldsymbol{x}_i-\boldsymbol{\mu})\right)$ |
| $\mathrm{Mult}(\boldsymbol{x}\|\boldsymbol{\pi})$ | $\mathrm{Dirichlet}(\boldsymbol{\pi}\|\boldsymbol{\alpha})$ | $\mathrm{Dirichlet}(\boldsymbol{\pi}\|\boldsymbol{\alpha}+\sum_{i=1}^{n}x_i)$ |

Table 2.2: Conjugate prior distributions. Here, $n$ denotes the number of observations and $\bar{\boldsymbol{x}} = \sum_{i=1}^{n}\boldsymbol{x}_i$ is the mean of $\boldsymbol{x}$.

can be performed using the *maximum likelihood* principle. Typically the conditional likelihood given the hidden variables $p(\boldsymbol{x}|\boldsymbol{z};\boldsymbol{\theta})$ is readily computable. However, parameter estimation should be carried out by maximizing the marginal likelihood $p(\boldsymbol{x};\boldsymbol{\theta})$, which may be difficult to compute or difficult to maximize. In such cases, the EM algorithm can be used to compute the parameter values $\boldsymbol{\theta}$ that maximize the marginal likelihood $p(\boldsymbol{x};\boldsymbol{\theta})$, without explicitly computing it. The computations involve only the conditional likelihood $p(\boldsymbol{x}|\boldsymbol{z};\boldsymbol{\theta})$ and the posterior of the hidden variables $p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta})$.

Hereafter, we will follow the exposition of the EM in [Neal and Hinton, 1998, Bishop, 2006, Tzikas et al., 2008b]. It is straightforward to show that the log-likelihood can be written as

$$\ln p(\boldsymbol{x};\boldsymbol{\theta}) = F(q,\boldsymbol{\theta}) + KL(q\|p), \tag{2.12}$$

with

$$F(q,\boldsymbol{\theta}) = \int q(\boldsymbol{z})\ln\frac{p(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\theta})}{q(\boldsymbol{z})}\,\mathrm{d}\boldsymbol{z}, \tag{2.13}$$

and

$$KL(q\|p) = -\int q(\boldsymbol{z})\ln\frac{p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta})}{q(\boldsymbol{z})}\,\mathrm{d}\boldsymbol{z}, \tag{2.14}$$

where $q(\boldsymbol{z})$ is any probability density function, $KL(q\|p)$ is the Kullback-Leibler divergence between $p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta})$ and $q(\boldsymbol{z})$, and since $KL(q\|p) \geq 0$, it holds that $\ln p(\boldsymbol{x};\boldsymbol{\theta}) \geq F(q,\boldsymbol{\theta})$. In other words, $F(q,\boldsymbol{\theta})$ is a lower bound of the log-likelihood. Equality holds only when $KL(q\|p) = 0$, which implies $q(\boldsymbol{z}) = p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta})$. The EM algorithm and some recent advances in deterministic approximations for Bayesian inference can be viewed in the

light of the decomposition in (2.12) as the maximization of the lower bound $F(q, \boldsymbol{\theta})$ with respect to the density $q(\boldsymbol{z})$ and the parameters $\boldsymbol{\theta}$.

In particular, EM is a two step iterative algorithm that maximizes the lower bound $F(q, \boldsymbol{\theta})$ and hence the log-likelihood. Assume that the current value of the parameters is $\boldsymbol{\theta}^{OLD}$. In the E-step the lower bound $F(q, \boldsymbol{\theta}^{OLD})$ is maximized with respect to $q(\boldsymbol{z})$. It is easy to see that this happens when $KL(q\|p) = 0$, in other words, when $q(\boldsymbol{z}) = p(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\theta}^{OLD})$. In this case the lower bound is equal to the log-likelihood. In the subsequent M-step, $q(\boldsymbol{z})$ is held fixed and the lower bound $F(q, \boldsymbol{\theta})$ is maximized with respect to $\boldsymbol{\theta}$ to give some new value $\boldsymbol{\theta}^{NEW}$. This will cause the lower bound to increase and as a result, the corresponding log-likelihood will also increase. Because $q(\boldsymbol{z})$ was determined using $\boldsymbol{\theta}^{OLD}$ and is held fixed in the M-step, it will not be equal to the new posterior $p(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\theta}^{NEW})$ and hence the KL distance will not be zero. Thus, the E-step and M-step need to be repeated until the algorithm converges.

If we substitute $q(\boldsymbol{z}) = p(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\theta}^{OLD})$ into the lower bound and expand (2.13) we get

$$\begin{aligned} F(q, \boldsymbol{\theta}) &= \int p(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\theta}^{OLD}) \ln p(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{z} - \int p(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\theta}^{OLD}) \ln p(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\theta}^{OLD}) \, \mathrm{d}\boldsymbol{z} \\ &= Q(\boldsymbol{\theta}, \boldsymbol{\theta}_{OLD}) + constant, \end{aligned} \tag{2.15}$$

where the constant is simply the entropy of $p(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\theta}^{OLD})$ which does not depend on $\boldsymbol{\theta}$. The function

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}_{OLD}) = \int p(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\theta}^{OLD}) \ln p(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{z} = \langle \ln p(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{\theta}) \rangle_{p(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\theta}^{OLD})} \tag{2.16}$$

is the expectation of the log-likelihood of the *complete data* (observations + hidden variables) which is maximized in the M-step. The usual way of presenting the EM algorithm in the literature has been via use of the $Q(\boldsymbol{\theta}, \boldsymbol{\theta}_{OLD})$ function directly [Moon, 1996, Kay, 1997].

In summary, the EM algorithm is an iterative algorithm involving the following two steps:

- E-step: Compute $p(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\theta}^{OLD})$
- M-step: Update $\boldsymbol{\theta}^{NEW} = \mathrm{argmax}_{\boldsymbol{\theta}} \, Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{OLD})$

Furthermore, it is interesting to point out that the EM algorithm requires that the posterior of the hidden variables $p(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\theta})$ is *explicitly* known, or at least we should be able to compute the conditional expectation of its sufficient statistics $\langle \ln p(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{\theta}) \rangle_{p(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\theta})}$, see (2.16). While $p(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\theta})$ is in general much easier to compute than $p(\boldsymbol{x}; \boldsymbol{\theta})$, in many interesting problems this is not possible, thus the EM algorithm is not applicable. For this reason, approximate inference techniques are employed.

## 2.5 The Maximum A Posteriori (MAP) Approximation

One of the most commonly used methodologies in the statistical modeling is the *maximum a posteriori* (MAP) method. MAP can be considered as an approximation of Bayesian inference, since the parameter vector $\boldsymbol{\theta}$ is assumed to be a random variable and a prior distribution $p(\boldsymbol{\theta})$ is imposed on $\boldsymbol{\theta}$. However, this approximation is rather crude, since the posterior distribution is approximated with a degenerate distribution at its mode.

For observations $\boldsymbol{x}$ generated by model $p(\boldsymbol{x}|\boldsymbol{\theta})$, the MAP estimate is defined as

$$\hat{\boldsymbol{\theta}}_{MAP} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, p(\boldsymbol{\theta}|\boldsymbol{x}) \tag{2.17}$$

and using Bayes theorem it can be obtained from

$$\hat{\boldsymbol{\theta}}_{MAP} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, p(\boldsymbol{x}|\boldsymbol{\theta})p(\boldsymbol{\theta}), \tag{2.18}$$

where $p(\boldsymbol{x}|\boldsymbol{\theta})$ is the likelihood of the observations. The MAP estimate is easier to obtain from (2.18) than (2.17). The posterior in (2.17) based on Bayes' theorem is given by

$$p(\boldsymbol{\theta}|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int p(\boldsymbol{x}|\boldsymbol{\theta})p(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta}} \tag{2.19}$$

and requires the computation of the Bayesian integral in the denominator of (2.19) to marginalize $\boldsymbol{\theta}$.

From the above it is clear that both MAP and Bayesian estimators assume that $\boldsymbol{\theta}$ is a random variable and use Bayes' theorem, however, their similarity stops there.

The MAP approach uses only *the mode of the posterior*, which is found by maximizing the posterior with respect to the parameters $\boldsymbol{\theta}$. In fact, the MAP approach can be considered as a simple extension of the maximum likelihood approach, which incorporates penalty terms for the parameters through definition of prior distributions for them. In contrast, for Bayesian inference the posterior is used and thus $\boldsymbol{\theta}$ has to be marginalized. It is important to note that Bayesian inference, unlike MAP, averages over all the available information about $\boldsymbol{\theta}$. Therefore, it is preferable over MAP, because it generally produces more accurate estimations and it also provides measures of the uncertainty of the estimation [Tzikas et al., 2008b].

The EM algorithm can also be used to obtain MAP estimates of $\boldsymbol{\theta}$. Using Bayes' theorem we can write

$$\ln p(\boldsymbol{\theta}|\boldsymbol{x}) = \ln p(\boldsymbol{x}, \boldsymbol{\theta}) - \ln p(\boldsymbol{x}) = \ln p(\boldsymbol{x}|\boldsymbol{\theta}) + \ln p(\boldsymbol{\theta}) - \ln p(\boldsymbol{x}). \tag{2.20}$$

Using a similar framework as for the ML-EM case in Section we can write

$$\ln p(\boldsymbol{\theta}|\boldsymbol{x}) = F(q, \boldsymbol{\theta}) + KL(q\|p) + \ln p(\boldsymbol{\theta}) - \ln p(\boldsymbol{x}) \tag{2.21}$$

$$\geq F(q, \boldsymbol{\theta}) + \ln p(\boldsymbol{\theta}) - \ln p(\boldsymbol{x}), \tag{2.22}$$

where in this context $\ln p(\boldsymbol{x})$ is a constant. The right hand side of (2.21) can be maximized in an alternating fashion as in the EM algorithm. Optimization with respect to $q(\boldsymbol{z})$ gives an identical E-step as in the ML case previously explained. Optimization with respect to $\boldsymbol{\theta}$ gives a different M-step since the objective function now contains also the term $\ln p(\boldsymbol{\theta})$. In general the M-step for the MAP-EM algorithm is more complex than in its ML counterpart [Blekas et al., 2005, Nikou et al., 2007]. Strictly speaking in such a model MAP estimation is used only for the $\boldsymbol{\theta}$ random variables, while Bayesian inference is used for hidden variables $\boldsymbol{z}$.

## 2.6 The Variational Bayes Approximation

Because MAP is a coarse approximation that does not consider uncertainties of the estimation, more flexible approximations are commonly considered. For example, the *Laplacian approximation* [Bishop, 2006] approximates the posterior distribution with a Gaussian distribution whose mean is assumed to be a mode of the true posterior distribution. Then, the covariance of the Gaussian distribution can be determined in terms of the Hessian matrix (matrix of second derivatives) of the true posterior at its mode.

More general approximations have also been considered. *Variational Bayesian inference* is an approximate inference technique that proceeds by assuming an arbitrary approximation $q(\boldsymbol{z})$ for the posterior distribution. Inference proceeds using a EM-like algorithm, which is called *Variational EM* (VEM) and which is based on the decomposition of (2.12). In the E-step $q(\boldsymbol{z})$ is found by maximizing $F(q, \boldsymbol{\theta})$ keeping $\boldsymbol{\theta}$ fixed. To perform this maximization, a particular form of $q(\boldsymbol{z})$ must be assumed. In certain cases it is possible to assume knowledge of the form of $q(\boldsymbol{z}; \boldsymbol{\omega})$, where $\boldsymbol{\omega}$ is a set of parameters. Thus, the lower bound $F(\boldsymbol{\omega}, \boldsymbol{\theta})$ becomes a function of these parameters and is maximized with respect to $\boldsymbol{\omega}$ in the E-step and with respect to $\boldsymbol{\theta}$ in the M-step, see for example [Bishop, 2006].

However, in its general form the lower bound $F(q, \boldsymbol{\theta})$ is a functional in terms of $q$, in other words, a mapping that takes as input a function $q(\boldsymbol{z})$, and returns as output the value of the functional. This leads naturally to the concept of the *functional derivative*, which in analogy to the function derivative, gives the functional changes for infinitesimal changes to the input function. This area of mathematics is called *calculus of variations* [Weinstock, 1974] and has been applied to many scientific areas.

Variational methods can be used to find approximate solutions in Bayesian inference problems. This is done by assuming that the functions over which optimization is performed have specific forms. For example, we can assume only quadratic functions or functions that are linear combinations of fixed basis functions. For Bayesian inference a particular form that has been used with great success is the factorized one, see [Jaakola, 1997, Jordan et al., 1999]. The idea for this factorized approximation stems from theoretical physics where it is called *mean field theory* [Parisi, 1988].

According to this approximation, the hidden variables $\boldsymbol{z}$ are assumed to be partitioned into $M$ partitions $z_i$ with $i = 1, \ldots, M$. Also it is assumed that $q(\boldsymbol{z})$ factorizes with respect to these partitions as

$$q(\boldsymbol{z}) = \prod_{i=1}^{M} q_i(z_i). \tag{2.23}$$

Thus, we wish to find the $q(\boldsymbol{z})$ of the form of (2.23) that maximizes the lower bound $F(q, \boldsymbol{\theta})$. It can be shown that this happens when [Jaakola, 1997, Jordan et al., 1999]:

$$\ln q_j(z_j) = \langle \ln p(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{\theta}) \rangle_{i \neq j} + const, \tag{2.24}$$

and with appropriate normalization the approximate posterior distributions $q_j(z_j)$ are:

$$q_j(z_j) = \frac{\exp\Big(\langle \ln p(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{\theta}) \rangle_{i \neq j}\Big)}{\int \exp\Big(\langle \ln p(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{\theta}) \rangle_{i \neq j}\Big)\, \mathrm{d}z_j}. \tag{2.25}$$

The above equations for $j = 1, \ldots, M$ are a set of consistency conditions for the maximum of the lower bound subject to the factorization of (2.23). They do not provide an explicit solution since they depend on the other factors $q_i(z_i)$ for $i \neq j$. Therefore, a consistent solution is found by cycling through these factors and replacing each in turn with the revised estimate.

In summary, the Variational EM algorithm is given by the following two steps:

1. Variational E-Step
   Evaluate $q^{NEW}(\boldsymbol{z})$ to maximize $F(q, \boldsymbol{\theta}^{OLD})$ solving the system of (2.25)

2. Variational M-Step
   Compute $\boldsymbol{\theta}_{NEW} = \mathrm{argmax}_{\boldsymbol{\theta}}\, F(q^{NEW}, \boldsymbol{\theta})$

At this point it is worth noting that in certain cases a Bayesian model can contain *only hidden* variables and no parameters. In such cases the Variational EM algorithm has *only an E-step* in which $q(\boldsymbol{z})$ is obtained using (2.25). This function $q(\boldsymbol{z})$ constitutes an approximation to $p(\boldsymbol{z}|\boldsymbol{x})$ that can be used for inference of the hidden variables.

# Chapter 3

# Linear Regression



## 3.1 Introduction

In this chapter we will apply the Bayesian Inference methods of the previous chapter on the problem of linear regression. For this problem, we consider an unknown function $y(\boldsymbol{x}) \in \mathbb{R}$, $\boldsymbol{x} \in \Omega \subseteq \mathbb{R}^N$ and want to predict its value $t_* = y(\boldsymbol{x}_*)$ at an arbitrary location $\boldsymbol{x}_* \in \Omega$, using a vector $\boldsymbol{t} = (t_1, \ldots, t_N)^T$ of $N$ noisy observations ($t_n = y(\boldsymbol{x}_n) + \epsilon_n$), at locations $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)^T$, $\boldsymbol{x}_n \in \Omega$, $n = 1, \ldots, N$.

The unknown function $y$ is commonly modelled as the linear combination of $M$ basis functions $\phi_m(\boldsymbol{x})$:

$$y(\boldsymbol{x}) = \sum_{m=1}^{M} w_m \phi_m(\boldsymbol{x}), \tag{3.1}$$

where $\boldsymbol{w} = (w_1, \ldots, w_M)^T$ are the weights of the linear combination. Selection of appropriate basis functions is essential in order to achieve good performance. However, there

is no rigorous methodology in doing so, but cross-validation techniques can be used to compare the performance of several basis function sets and then select the best one.

The additive noise $\boldsymbol{\epsilon} = (\epsilon_1, \ldots, \epsilon_N)^T$ is commonly assumed to be zero-mean, Gaussian distributed

$$p(\boldsymbol{\epsilon}) = \mathrm{N}(\boldsymbol{\epsilon}|\mathbf{0}, \boldsymbol{B}^{-1}), \tag{3.2}$$

where $\boldsymbol{B}$ is the inverse covariance (precision) matrix. Usually, we assume that the observations are independent and identically distributed (i.i.d.), therefore $\boldsymbol{B} = \beta\boldsymbol{I}$. However, here we retain the more general form of the precision matrix, because it is used to derive the classification algorithm and also allows considering non-Gaussian noise distributions. For example, if we assume independent noise, but assign separate precision $\beta_n$ to each data point $t_n$, the precision matrix becomes $\boldsymbol{B} = \mathrm{diag}\{\beta_1, \ldots, \beta_n\}$ and this allows designing robust regression models by selecting an appropriate noise precision prior $p(\beta_n)$. More specifically, assuming a Gamma pdf for the noise precisions:

$$p(\beta_n) = \mathrm{Gamma}(\beta_n|c, d), \tag{3.3}$$

we obtain a Student's t pdf for the noise

$$p(\epsilon_n) = \int p(\epsilon_n|\beta_n)p(\beta_n)\,\mathrm{d}\beta_n = \mathrm{Student}(\epsilon_n|0, \nu, \lambda), \tag{3.4}$$

with $\lambda = c/d$ and $\nu = 2c$. This pdf can provide robustness, because it may have heavy tails [Tipping and Lawrence, 2003].

Here, we consider Gaussian distributed noise, therefore by defining the *design matrix* $\Phi = (\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_M)$, with $\boldsymbol{\phi}_m = (\phi_m(\boldsymbol{x}_1), \ldots, \phi_m(\boldsymbol{x}_N))^T$, the observations $\boldsymbol{t}$ are modelled as

$$\boldsymbol{t} = \boldsymbol{\Phi}\boldsymbol{w} + \boldsymbol{\epsilon}, \tag{3.5}$$

and their likelihood is

$$p(\boldsymbol{t}; \boldsymbol{w}, \boldsymbol{B}) = \mathrm{N}(\boldsymbol{t}|\boldsymbol{\Phi}\boldsymbol{w}, \boldsymbol{B}^{-1}). \tag{3.6}$$

In what follows Bayesian inference is applied to the linear regression problem and we demonstrate three well-known methodologies to compute the unknown weights $\boldsymbol{w}$ of this linear model [Bishop, 2006, Tzikas et al., 2008b]. First, we apply typical maximum likelihood (ML) estimation of the weights which are assumed to be parameters. As it will be demonstrated, if the number of parameters is large (compared to the number of observations), the ML estimates are very sensitive to the noise and overfit the observations. Subsequently, to ameliorate this problem a prior $p(\boldsymbol{w})$ is imposed on the weights, which are assumed to be random variables. First, a simple Bayesian model is used, which is based on a stationary Gaussian prior for the weights. For this model, Bayesian inference is performed using the EM algorithm and the resulting solution is robust to noise. Nevertheless, this Bayesian model is very simplistic, and it is possible to use a more sophisticated non-stationary hierarchical model, which is equivalent to assuming a Student's t prior for

the weights, see Section 3.4.4. This model is too complex to solve using the EM algorithm. Instead, the variational Bayesian methodology described in Section 2.6 is used to infer values for the unknowns of this model. In Fig. 3.1 we show the graphical models for the three approaches to Linear Regression that are described in the following sections.

## 3.2 Maximum Likelihood Estimation

The simplest estimate of the weights $\boldsymbol{w}$ of the linear model is obtained by maximizing the likelihood of the model. This ML estimate assumes the weights $\boldsymbol{w}$ to be parameters, as shown in the graphical model of Fig. 3.1a. The ML estimate is obtained by maximizing the likelihood function of (3.6):

$$p(\boldsymbol{t}; \boldsymbol{w}, \boldsymbol{B}) = (2\pi)^{-N/2} |\boldsymbol{B}|^{1/2} \exp\left( -\frac{1}{2} (\boldsymbol{t} - \boldsymbol{\Phi w})^T \boldsymbol{B}\ (\boldsymbol{t} - \boldsymbol{\Phi w}) \right). \tag{3.7}$$

This is equivalent to minimizing

$$E_{LS} = \|\boldsymbol{t} - \boldsymbol{\Phi w}\|_{\boldsymbol{B}}^2 = (\boldsymbol{t} - \boldsymbol{\Phi w})^T \boldsymbol{B}\ (\boldsymbol{t} - \boldsymbol{\Phi w}). \tag{3.8}$$

Thus, in this case the ML is equivalent with the least squares (LS) estimate

$$\boldsymbol{w}_{LS} = \underset{w}{\operatorname{argmax}}\, p(\boldsymbol{t}; \boldsymbol{w}, \boldsymbol{B}) = \underset{w}{\operatorname{argmin}}\, E_{LS} = (\boldsymbol{\Phi}^T \boldsymbol{B} \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \boldsymbol{B} \boldsymbol{t} \tag{3.9}$$

In many situations and depending on the basis functions that are used, the matrix $\boldsymbol{\Phi}^T \boldsymbol{B} \boldsymbol{\Phi}$ may be ill-conditioned and difficult to invert. This means that if noise $\boldsymbol{\epsilon}$ is included in the observations, it will heavily affect the estimation $\boldsymbol{w}_{LS}$ of the weights. Thus, when using maximum likelihood linear regression, the basis functions should be carefully chosen to ensure that matrix $\boldsymbol{\Phi}^T \boldsymbol{B} \boldsymbol{\Phi}$ can be inverted. This is generally achieved by using a model with few basis functions, which also has the advantage that only few parameters have to be estimated.

## 3.3 The Bayesian Linear Model

A Bayesian treatment of the linear model begins by assigning a prior distribution $p(\boldsymbol{w})$ to the weights of the model. This introduces bias in the estimation, but also greatly reduces its variance, which is a major problem of the maximum likelihood estimate. Here, we consider the common choice of independent, zero-mean, Gaussian prior distribution for the weights of the linear model:

$$p(\boldsymbol{w}; \alpha) = \mathrm{N}(\boldsymbol{w} | \boldsymbol{0}, \alpha^{-1} \boldsymbol{I}). \tag{3.10}$$

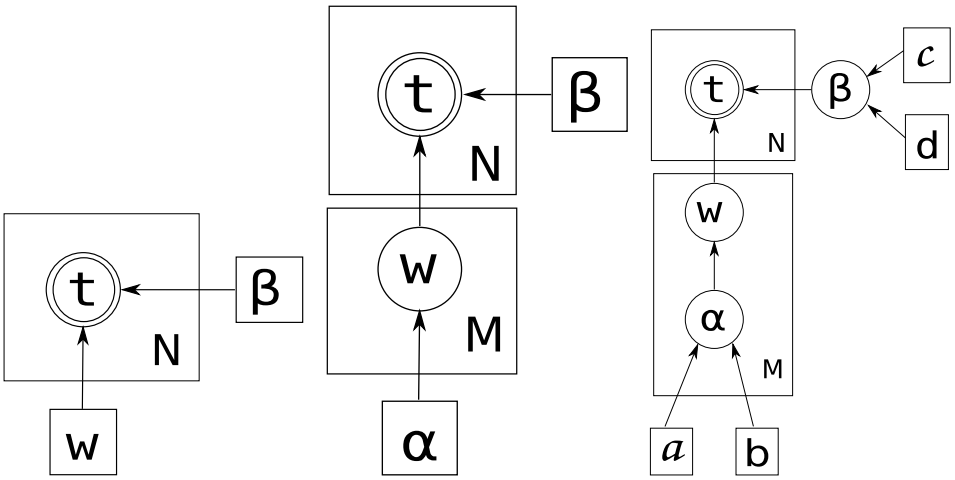



Figure 3.1: Graphical models for linear regression solved using (a) model without prior (direct ML estimation), (b) model with stationary Gaussian prior (EM), (c) model with hierarchical prior (variational EM).

This is a stationary prior distribution, meaning that the distribution of all the weights is identical. The graphical model for this problem is shown in Fig. 3.1b. Notice that here the weights $\boldsymbol{w}$ are hidden random variables and the set of model parameters contains the parameter $\alpha$ of the prior for the weights and the precision $\boldsymbol{B}$ of the additive noise.

Bayesian inference proceeds by computing the posterior distribution of the hidden variables:

$$p(\boldsymbol{w}|\boldsymbol{t};\alpha,\boldsymbol{B}) = \frac{p(\boldsymbol{t}|\boldsymbol{w};\boldsymbol{B})p(\boldsymbol{w};\alpha)}{p(\boldsymbol{t};\alpha,\boldsymbol{B})}. \tag{3.11}$$

Notice, that the marginal likelihood $p(\boldsymbol{t};\alpha,\beta)$ that appears on the denominator can be computed analytically:

$$p(\boldsymbol{t};\alpha,\boldsymbol{B}) = \int p(\boldsymbol{t}|\boldsymbol{w};\boldsymbol{B})p(\boldsymbol{w};\alpha)\,\mathrm{d}\boldsymbol{w} = \mathrm{N}(\boldsymbol{t}|\boldsymbol{0},\boldsymbol{B}^{-1}+\alpha^{-1}\boldsymbol{\Phi}\boldsymbol{\Phi}^T). \tag{3.12}$$

Then, the posterior of the hidden variables is:

$$p(\boldsymbol{w}|\boldsymbol{t};\alpha,\boldsymbol{B}) = \mathrm{N}(\boldsymbol{w}|\boldsymbol{\mu},\boldsymbol{\Sigma}), \tag{3.13}$$

with

$$\boldsymbol{\mu} = \boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{B}\boldsymbol{t}, \tag{3.14}$$

$$\boldsymbol{\Sigma} = (\boldsymbol{\Phi}^T\boldsymbol{B}\boldsymbol{\Phi}+\alpha\boldsymbol{I})^{-1}. \tag{3.15}$$

If we assume that the noise is i.i.d. then $\boldsymbol{B} = \beta\boldsymbol{I}$ and the parameters of the model are the weight and noise precisions $\alpha, \beta$ and they can be estimated by maximizing the logarithm

of the marginal likelihood $p(\boldsymbol{t}; \alpha, \beta)$:

$$(\alpha_{ML}, \beta_{ML}) = \underset{\alpha,\beta}{\operatorname{argmax}} \{\log |\beta^{-1}\boldsymbol{I} + \alpha^{-1}\boldsymbol{\Phi}^T\boldsymbol{\Phi}| + \boldsymbol{t}^T \left(\beta^{-1}\boldsymbol{I} + \alpha^{-1}\boldsymbol{\Phi}\boldsymbol{\Phi}^T\right)^{-1} \boldsymbol{t}\}. \tag{3.16}$$

Optimization of the marginal likelihood can be performed using the EM algorithm, which provides an efficient framework to simultaneously obtain estimates for $\alpha, \beta$ and infer the posterior distribution of $\boldsymbol{w}$. Notice, that although the EM algorithm does not involve computations with the marginal likelihood of (3.12), the algorithm converges to a local maximum of it. After initializing the parameters to some values $(\alpha^{(0)}, \beta^{(0)})$, the algorithm proceeds by iteratively performing the following steps:

- E- step
  Compute the expected value of the logarithm of the complete likelihood:

$$Q^{(t)}(\boldsymbol{t}, \boldsymbol{w}; \alpha, \beta) = \langle \ln p(\boldsymbol{t}, \boldsymbol{w}; \alpha, \beta) \rangle_{p(\boldsymbol{w}|\boldsymbol{t}; \alpha^{(t)}, \beta^{(t)})}. \tag{3.17}$$

  This is computed using equations (3.6) and (3.10) as

$$\begin{aligned} Q^{(t)}(\boldsymbol{t}, \boldsymbol{w}; \alpha, \beta) &= \left\langle \frac{N}{2} \ln \beta - \frac{\beta}{2} \|\boldsymbol{t} - \boldsymbol{\Phi}\boldsymbol{w}\|^2 + \frac{M}{2} \ln \alpha - \frac{\alpha}{2} \|\boldsymbol{w}\|^2 \right\rangle + const \\ &= \frac{N}{2} \ln \beta - \frac{\beta}{2} \left\langle \|\boldsymbol{t} - \boldsymbol{\Phi}\boldsymbol{w}\|^2 \right\rangle + \frac{M}{2} \ln \alpha - \frac{\alpha}{2} \left\langle \|\boldsymbol{w}\|^2 \right\rangle + const. \end{aligned} \tag{3.18}$$

  These expected values are with respect to $p(\boldsymbol{w}|\boldsymbol{t}; \alpha^{(t)}, \beta^{(t)})$ and can be computed from (3.13), giving

$$\begin{aligned} Q^{(t)}(\boldsymbol{t}, \boldsymbol{w}; \alpha, \beta) =& \frac{N}{2} \ln \beta - \frac{\beta}{2} \left( \|\boldsymbol{t} - \boldsymbol{\Phi}\boldsymbol{\mu}^{(t)}\|^2 + \operatorname{trace}(\boldsymbol{\Phi}^T\boldsymbol{\Sigma}^{(t)}\boldsymbol{\Phi}) \right) + \\ & \frac{M}{2} \ln \alpha - \frac{\alpha}{2} \left( \|\boldsymbol{\mu}^{(t)}\|^2 + \operatorname{trace}(\boldsymbol{\Sigma}^{(t)}) \right) + const, \end{aligned} \tag{3.19}$$

  where $\boldsymbol{\mu}^{(t)}$ and $\boldsymbol{\Sigma}^{(t)}$ are computed using the current estimates of the parameters $\alpha^{(t)}$ and $\beta^{(t)}$:

$$\boldsymbol{\mu}^{(t)} = \beta^{(t)}\boldsymbol{\Sigma}^{(t)}\boldsymbol{\Phi}^T\boldsymbol{t}, \tag{3.20}$$

$$\boldsymbol{\Sigma}^{(t)} = (\beta^{(t)}\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \alpha^{(t)}\boldsymbol{I})^{-1}. \tag{3.21}$$

- M-step
  Maximize $Q^{(t)}(\boldsymbol{t}, \boldsymbol{w}; \alpha, \beta)$ with respect to the parameters $\alpha$ and $\beta$:

$$(\alpha^{(t+1)}, \beta^{(t+1)}) = \underset{\alpha,\beta}{\operatorname{argmax}}\, Q^{(t)}(\boldsymbol{t}, \boldsymbol{w}; \alpha, \beta) \tag{3.22}$$

The derivatives of $Q^{(t)}(\boldsymbol{t}, \boldsymbol{w}; \alpha, \beta)$ with respect to the parameters are:

$$\frac{\partial Q^{(t)}(\boldsymbol{t}, \boldsymbol{w}; \alpha, \beta)}{\partial \alpha} = \frac{M}{2\alpha} - \frac{1}{2}\left(\|\boldsymbol{\mu}^{(t)}\|^2 + \text{trace}(\boldsymbol{\Sigma}^{(t)})\right), \tag{3.23}$$

$$\frac{\partial Q^{(t)}(\boldsymbol{t}, \boldsymbol{w}; \alpha, \beta)}{\partial \beta} = \frac{N}{2\beta} - \frac{1}{2}\left(\|\boldsymbol{t} - \boldsymbol{\Phi}\boldsymbol{\mu}^{(t)}\|^2 + \text{trace}(\boldsymbol{\Phi}^T\boldsymbol{\Sigma}^{(t)}\boldsymbol{\Phi})\right). \tag{3.24}$$

Setting these to zero, we obtain the following formulas to update the parameters $\alpha$ and $\beta$:

$$\alpha^{(t+1)} = \frac{M}{\|\boldsymbol{\mu}^{(t)}\|^2 + \text{trace}(\boldsymbol{\Sigma}^{(t)})}, \tag{3.25}$$

$$\beta^{(t+1)} = \frac{N}{\|\boldsymbol{t} - \boldsymbol{\Phi}\boldsymbol{\mu}^{(t)}\|^2 + \text{trace}(\boldsymbol{\Phi}^T\boldsymbol{\Sigma}^{(t)}\boldsymbol{\Phi})}. \tag{3.26}$$

Notice, that the maximization step can be analytically performed, in contrast to direct maximization of the marginal likelihood in (3.12), which would require numerical optimization. Furthermore, equations (3.25) and (3.26) guarantee that positive estimations for the parameters $\alpha$ and $\beta$ are produced, which is a requirement since these represent inverse variance parameters. However, the parameters should be initialized with care, since, depending on the initialization, a different local maximum may be attained. Inference for $\boldsymbol{w}$ is obtained directly, since the sufficient statistics of the posterior $p(\boldsymbol{w}|\boldsymbol{t}|\alpha, \beta)$ are computed in the E-step. The mean of this posterior, given by (3.20), can be used as Bayesian linear minimum mean square error (LMMSE) estimate for $\boldsymbol{w}$.

## 3.4 The Sparse Bayesian Linear Model

In the Bayesian approach described in the previous section, due to the use of a stationary Gaussian prior distribution for the weights of the linear model, exact computation of the marginal likelihood is possible and Bayesian inference is performed analytically. However, in many situations, it is important to allow the flexibility to model local characteristics of the function, which the simple stationary Gaussian prior distribution is unable to do. For this reason, a non-stationary Gaussian prior distribution with a distinct inverse variance $\alpha_m$ for each weight is considered:

$$p(\boldsymbol{w}|\boldsymbol{\alpha}) = \text{N}(\boldsymbol{w}|\boldsymbol{0}, \boldsymbol{A}^{-1}), \tag{3.27}$$

where $\boldsymbol{A} = \text{diag}\{\alpha_1, \ldots, \alpha_M\}$. However, such a model is over-parameterized, since there are as many parameters $\alpha_i$ to be estimated as the number of basis functions. For this purpose the precision parameters $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_M)^T$ are constrained, by treating them

as random variables and imposing a Gamma prior distribution to them according to

$$p(\boldsymbol{\alpha}) = \prod_{m=1}^{M} \text{Gamma}(\alpha_m | a, b). \tag{3.28}$$

This prior is selected because it is conjugate to the Gaussian.

We also assume i.i.d. noise, therefore $\boldsymbol{B} = \beta \boldsymbol{I}$ and we use a Gamma distribution as prior for the noise inverse variance $\beta$:

$$p(\beta) = \text{Gamma}(\beta | c, d). \tag{3.29}$$

The graphical model for this Bayesian approach is shown in Fig. 3.1c, where the dependence of the hidden variables $\boldsymbol{w}$ on the hidden variables $\boldsymbol{\alpha}$ is apparent. Also the parameters $a, b, c$ and $d$ of this model and the hidden variables that depend on them are also depicted.

Bayesian inference requires the computation of the posterior distribution

$$p(\boldsymbol{w}, \boldsymbol{\alpha}, \beta | \boldsymbol{t}) = \frac{p(\boldsymbol{t} | \boldsymbol{\alpha}, \beta) p(\boldsymbol{w} | \alpha) p(\boldsymbol{\alpha}) p(\beta)}{p(\boldsymbol{t})}. \tag{3.30}$$

However, the marginal likelihood $p(\boldsymbol{t}) = \int p(\boldsymbol{t} | \boldsymbol{\alpha}, \beta) p(\boldsymbol{w} | \alpha) p(\boldsymbol{\alpha}) p(\beta) \, \mathrm{d}\boldsymbol{w} \, \mathrm{d}\boldsymbol{\alpha} \, \mathrm{d}\beta$ cannot be computed analytically, and thus the normalization constant in (3.30) cannot be obtained.

### 3.4.1 Variational Bayesian Inference

Because exact Bayesian inference is intractable, approximate Bayesian inference methods are employed and specifically the variational inference methodology of Section 2.6. Assuming posterior independence between the weights $\boldsymbol{w}$ and the variance parameters $\boldsymbol{\alpha}$ and $\beta$,

$$p(\boldsymbol{w}, \boldsymbol{\alpha}, \beta | \boldsymbol{t}) \approx q(\boldsymbol{w}, \boldsymbol{\alpha}, \beta) = q(\boldsymbol{w}) q(\boldsymbol{\alpha}) q(\beta), \tag{3.31}$$

the approximate posterior distributions $q$ can be computed from (2.24) as follows. Keeping only the terms of $\ln q(\boldsymbol{w})$ that depend on $\boldsymbol{w}$, we have:

$$\begin{aligned}
\ln q(\boldsymbol{w}) &= \langle \ln p(\boldsymbol{t}, \boldsymbol{w}, \boldsymbol{\alpha}, \beta) \rangle_{q(\boldsymbol{\alpha}) q(\beta)} + const \\
&= \langle \ln p(\boldsymbol{t} | \boldsymbol{w}, \beta) + \ln p(\boldsymbol{w} | \boldsymbol{\alpha}) \rangle_{q(\boldsymbol{\alpha}) q(\beta)} + const \\
&= \left\langle -\frac{\beta}{2} (\boldsymbol{t} - \boldsymbol{\Phi}\boldsymbol{w})^T (\boldsymbol{t} - \boldsymbol{\Phi}\boldsymbol{w})) - \frac{1}{2} \boldsymbol{w}^T \boldsymbol{A} \boldsymbol{w} \right\rangle_{q(\boldsymbol{\alpha}) q(\beta)} + const \\
&= -\frac{\langle \beta \rangle}{2} (\boldsymbol{t}^T \boldsymbol{t} - 2 \boldsymbol{t}^T \boldsymbol{\Phi} \boldsymbol{w} + \boldsymbol{w}^T \boldsymbol{\Phi}^T \boldsymbol{\Phi} \boldsymbol{w})) - \frac{1}{2} \boldsymbol{w}^T \langle \boldsymbol{A} \rangle \boldsymbol{w} + const \\
&= -\frac{1}{2} \left[ \boldsymbol{w}^T \left( \langle \beta \rangle \boldsymbol{\Phi}^T \boldsymbol{\Phi} + \langle \boldsymbol{A} \rangle \right) \boldsymbol{w} - 2 \langle \beta \rangle \boldsymbol{w}^T \boldsymbol{\Phi}^T \boldsymbol{t} \right] + const \\
&= -\frac{1}{2} \left[ \boldsymbol{w}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{w} - 2 \boldsymbol{w}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} \right] + const.
\end{aligned} \tag{3.32}$$

Notice, that this is the exponent of a Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ given by

$$\boldsymbol{\Sigma} = \left(\langle\beta\rangle\, \boldsymbol{\Phi}^T\boldsymbol{\Phi} + \langle \boldsymbol{A}\rangle\right)^{-1}, \tag{3.33}$$

$$\boldsymbol{\mu} = \langle\beta\rangle\, \boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{t}. \tag{3.34}$$

Therefore, $q(\boldsymbol{w})$ is given by:

$$q(\boldsymbol{w}) = \mathrm{N}(\boldsymbol{w}|\boldsymbol{\mu}, \boldsymbol{\Sigma}). \tag{3.35}$$

The posterior $q(\boldsymbol{\alpha})$ is similarly obtained by computing the terms of $\ln q(\boldsymbol{\alpha})$ that depend on $\boldsymbol{\alpha}$:

$$\begin{aligned}
\ln q(\boldsymbol{\alpha}) &= \langle p(\boldsymbol{t}, \boldsymbol{w}, \boldsymbol{\alpha}, \beta)\rangle_{q(\boldsymbol{w})q(\beta)} + const \\
&= \langle \ln p(\boldsymbol{w}|\boldsymbol{\alpha}) + \ln p(\boldsymbol{\alpha})\rangle_{q(\boldsymbol{w})} + const \\
&= \frac{1}{2}\ln|\boldsymbol{A}| - \frac{1}{2}\left\langle \boldsymbol{w}^T\boldsymbol{A}\boldsymbol{w}\right\rangle + (a-1)\sum_{m=1}^{M}\ln\alpha_m - b\sum m = 1^M\alpha_m + const \\
&= \left(a - \frac{1}{2}\right)\sum_{m=1}^{M}\ln\alpha_m - \sum_{m=1}^{M}\alpha_m\left(\frac{\langle w_m\rangle}{2} + b\right) + const \\
&= \tilde{a}\sum_{m=1}^{M}\ln\alpha_m - \sum_{m=1}^{M}\alpha_m\tilde{b} + const.
\end{aligned} \tag{3.36}$$

This is the exponent of the product of $M$ independent Gamma distributions with parameters $\tilde{a}$ and $\tilde{b}$, given by

$$\tilde{a} = a + \frac{1}{2}, \tag{3.37}$$

$$\tilde{b} = b + \frac{\langle w_m\rangle}{2}. \tag{3.38}$$

Thus, $q(\boldsymbol{\alpha})$ is given by:

$$q(\boldsymbol{\alpha}) = \prod_{m=1}^{M}\mathrm{Gamma}(\alpha_m|\tilde{a}, \tilde{b}). \tag{3.39}$$

The posterior distribution of the noise inverse variance can be similarly computed as:

$$q(\beta) = \mathrm{Gamma}(\beta|\tilde{c}, \tilde{d}). \tag{3.40}$$

with

$$\tilde{c} = c + \frac{N}{2}, \tag{3.41}$$

$$\tilde{d} = d + \frac{\langle\|\boldsymbol{t} - \boldsymbol{\Phi}\boldsymbol{w}\|^2\rangle}{2}. \tag{3.42}$$

The approximate posterior distributions in equations (3.35), (3.39) and (3.40) are then iteratively updated until convergence, since they depend on the statistics of each other, see for details [Bishop and Tipping, 2000].

### 3.4.2 MAP Estimation of Precision Parameters

In this section an alternative training algorithm for the sparse Bayesian linear model is described, which is based on the MAP approximation for estimation of the weight and noise precisions $\boldsymbol{\alpha}$ and $\beta$. Under the MAP approximation, update formulas for the weight precisions $\boldsymbol{\alpha}$ can be obtained by maximizing the logarithm of the marginal likelihood $p(\boldsymbol{t}|\boldsymbol{\alpha}, \boldsymbol{\beta}) = \int p(\boldsymbol{t}|\boldsymbol{w}, \boldsymbol{\beta}) p(\boldsymbol{w}|\boldsymbol{\alpha}) p(\boldsymbol{\alpha}) \, \mathrm{d}\boldsymbol{w}$. Here, we assume an uninformative prior for $\boldsymbol{\alpha}$ ($p(\boldsymbol{\alpha}) = const$), therefore the marginal likelihood is given by [Tipping, 2001]:

$$L = \log p(\boldsymbol{t}|\boldsymbol{\alpha}, \boldsymbol{\beta}) = -\frac{1}{2} \left( N \log 2\pi + |\boldsymbol{C}| + \boldsymbol{t}^T \boldsymbol{C}^{-1} \boldsymbol{t} \right), \tag{3.43}$$

where $\boldsymbol{C} = \boldsymbol{B}^{-1} + \boldsymbol{\Phi} \boldsymbol{A}^{-1} \boldsymbol{\Phi}^T$.

Maximization of the marginal likelihood is typically performed by considering the weights $\boldsymbol{w}$ as hidden variables and then using the EM algorithm. It can be shown that the updates which this approach gives, are equivalent to the updates of the variational algorithm of the previous section. However, instead of using the EM algorithm, [Tipping, 2001] suggests that in this case it is more efficient to maximize the marginal likelihood directly. The derivative of the marginal likelihood with respect to $\log \alpha_i$ is

$$\frac{\partial L}{\partial \log \alpha_i} = \frac{1}{2} (1 - \alpha_i \Sigma_{ii} - \alpha_i \mu_i^2). \tag{3.44}$$

Equating this to zero and setting $\gamma_i = 1 - \alpha_i \Sigma_{ii}$, we obtain the following update formula for $\alpha_i$:

$$\alpha_i = \frac{\gamma_i}{\mu_i^2}. \tag{3.45}$$

We can also compute updates for the noise precision $\beta$. The derivative of the marginal likelihood with respect to $\log \beta$ is

$$\frac{\partial L}{\partial \log \beta} = \frac{1}{2} \left[ \frac{N}{\beta} - \|\boldsymbol{t} - \boldsymbol{\Phi} \boldsymbol{\mu}\|^2 - \mathrm{trace}(\boldsymbol{\Sigma} \boldsymbol{\Phi}^T \boldsymbol{\Phi}) \right], \tag{3.46}$$

and by setting it to zero we obtain the following update formula for $\beta$:

$$\beta = \frac{N - \sum_{i=1}^{N} \gamma_i}{\|\boldsymbol{t} - \boldsymbol{\Phi} \boldsymbol{\mu}\|^2}. \tag{3.47}$$

These updates do not enjoy the theoretical convergence properties of the EM-based update equations. However, it has been experimentally observed that they always converge and, furthermore, they typically converge faster than the EM-based updates.

### 3.4.3 Incremental Training Algorithm

Notice that the computational cost of the sparse Bayesian learning algorithm is high for large datasets, because the computation of $\boldsymbol{\Sigma}$ in (3.33) involves matrix inversion and typically requires $O(N^3)$ operations. An incremental algorithm that is more computationally efficient has been proposed in [Tipping and Faul, 2003]. 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 re-estimated or removed from the current model. When adding a basis functions to the model, the corresponding parameter $\alpha_i$ is set to the value that maximizes the marginal likelihood.

More specifically, the terms of the marginal likelihood (3.43) that depend on a single parameter $\alpha_i$ are [Tipping and Faul, 2003]:

$$l(\alpha_i) = \frac{1}{2}\left(\log \alpha_i - \log(\alpha_i + s_i) + \frac{q_i^2}{\alpha_i + s_i}\right), \tag{3.48}$$

where

$$s_i = \boldsymbol{\phi}_i^T \boldsymbol{C}_{-i}^{-1} \boldsymbol{\phi}_i, \tag{3.49}$$
$$q_i = \boldsymbol{\phi}_i^T \boldsymbol{C}_{-i}^{-1} \hat{\boldsymbol{t}}, \tag{3.50}$$

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 (3.88) and (3.89) respectively.

In order to simplify computations one can define:

$$S_i = \boldsymbol{\phi}_i^T \boldsymbol{C}^{-1} \boldsymbol{\phi}_i, \tag{3.51}$$
$$Q_i = \boldsymbol{\phi}_i^T \boldsymbol{C}^{-1} \hat{\boldsymbol{t}}, \tag{3.52}$$

and compute $s_i$, $q_i$ from:

$$s_i = \frac{\alpha_i S_i}{\alpha_i - S_i}, \tag{3.53}$$
$$q_i = \frac{\alpha_i Q_i}{\alpha_i - S_i}. \tag{3.54}$$

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

$$S_i = \boldsymbol{\phi}_i^T \boldsymbol{B} \boldsymbol{\phi}_i - \boldsymbol{\phi}_i^T \boldsymbol{B} \boldsymbol{\Phi} \Sigma \Phi^T \boldsymbol{B} \boldsymbol{\phi}_i, \tag{3.55}$$
$$Q_i = \boldsymbol{\phi}_i^T \boldsymbol{B} \hat{\boldsymbol{t}} - \boldsymbol{\phi}_i^T \boldsymbol{B} \boldsymbol{\Phi} \Sigma \Phi^T \boldsymbol{B} \hat{\boldsymbol{t}}. \tag{3.56}$$

It has been shown in [Faul and Tipping, 2002] that $l(\alpha_i)$ has a single maximum at:

$$\alpha_i = \frac{s_i^2}{q_i^2 - s_i}, \quad \text{if } q_i^2 > s_i, \tag{3.57}$$

$$\alpha_i = \infty, \quad \text{if } q_i^2 \leq s_i. \tag{3.58}$$

Based on this result, the incremental algorithm proceeds iteratively, adding each time one basis function $\phi_i$ 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 in the marginal likelihood. In the first approach, where we select basis functions at random, the incremental algorithm may require a very large number of iterations to converge. On the other hand, in the second approach, where we select the best basis function for addition, much less iterations are required, but the computational cost of each iteration is significantly increased.

### 3.4.4 Understanding Sparsity

As already mentioned, the 'true' prior distribution of the weights can be computed by marginalizing the hyper-parameters $\boldsymbol{\alpha}$

$$p(\boldsymbol{w}) = \int p(\boldsymbol{w}|\boldsymbol{\alpha})p(\boldsymbol{\alpha})d\boldsymbol{\alpha} \tag{3.59}$$

$$= \int \prod_{m=1}^{M} \left[\mathrm{N}(w_m|0, \alpha_m^{-1})\mathrm{Gamma}(\alpha_m|a, b)d\alpha_m\right] \tag{3.60}$$

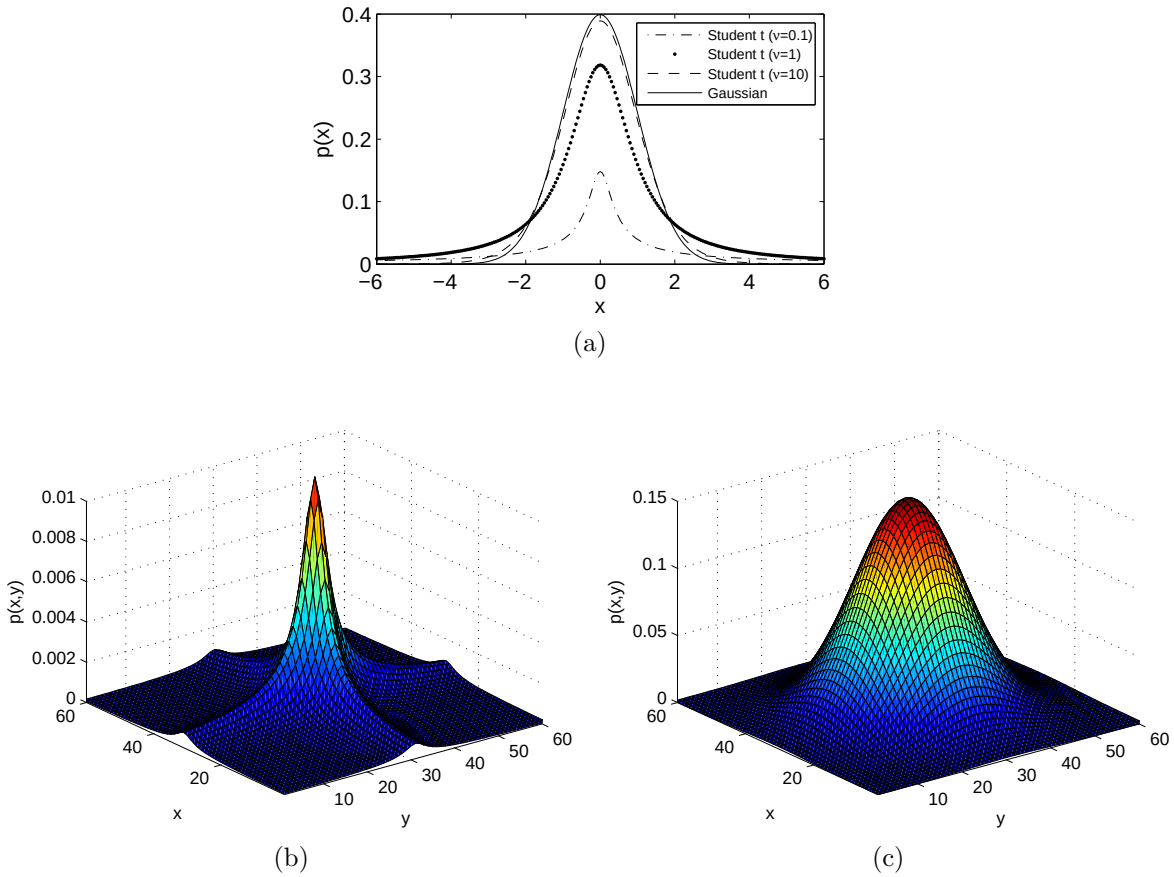
$$= \prod_{m=1}^{M} \mathrm{Student}(w_m|0, \lambda, \nu), \tag{3.61}$$

and is a Student's t pdf,

$$\mathrm{Student}(x|\mu, \lambda, \nu) = \frac{\Gamma((\nu+1)/2)}{\Gamma(\nu/2)} \left(\frac{\lambda}{\pi\nu}\right)^{1/2} \left[1 + \frac{\lambda(x-\mu)^2}{\nu}\right]^{-(\nu+1)/2}, \tag{3.62}$$

with mean $\mu = 0$, parameter $\lambda = a/b$ and degrees of freedom $\nu = 2a$. This distribution can be considered as a generalization of the Gaussian; with appropriate selection of its parameters, it can have heavy tails and in the limit it can become either Gaussian (large $\nu$), or uninformative (small $\nu$), see Fig. 3.2(a).

The important issue is that when the weights of a linear model follow a heavy-tailed distribution (such as the Student's t pdf with few degrees of freedom $\nu$), this results in sparse models, i.e. models with few non-zero parameters $w_i$. The sparsity of such models can be understood by observing the plots of the two-dimensional pdfs in Fig. 3.2(b). Most of the mass of the Student's t pdf is concentrated along the axes and the center, unlike the Gaussian, where it is evenly distributed around ellipses, as shown in Fig. 3.2(c). This observation can be generalized for vectors of arbitrary dimension, where the Student's




(a) (b) (c)

Figure 3.2: (a) The Student's t pdf with 0.1, 1 and 10 degrees of freedom compared to the Gaussian pdf. Two dimensional plot of (b) the Student's-t pdf with 0.1 degrees of freedom and (c) the Gaussian pdf.

t pdf assigns large probability mass to estimations that contain a large number of zero elements. In similar spirit the Laplacian pdf (which also has heavy tails) has been used for obtaining sparse models [Figueiredo, 2003]. Since most of the weights are set to zero, most of the basis functions are pruned and do not contribute to the estimation.

There are several advantages of using sparse priors:

- The complexity of the model is automatically adjusted, thus very complex models may be initially considered.
- The basis functions that remain on the model provide information about which basis functions are relevant with the data. This may be useful in many applications.
- The output of sparse models is computed very efficiently, since only few basis functions are involved in the computation.

Notice, that for simplicity we have assumed fixed the parameters $a, b, c$ and $d$ of the Student's t distributions. In practice we can often obtain good results by assuming uninformative distributions, which are obtained by setting these parameters to very small

values, i.e. $a = b = 10^{-6}$. In the limit, setting $a = b = 0$ gives the improper Jeffrey's prior

$$p(x) = \frac{1}{|x|}, \tag{3.63}$$

which is also known to provide sparse solutions. Alternatively, we can estimate the above parameters using a Variational EM algorithm. Such an approach would add an M-step to the described method, in which the Variational bound would be maximized with respect to these parameters. However, a usual approach in Bayesian modeling is to fix the hyperparameters so that uninformative hyperpriors are defined at the highest level of the model.

## 3.5 The Relevance Vector Machine

The linear model is very efficient provided that appropriate basis functions are used. However, there is no rigorous methodology to select these basis functions. A significant advantage of the sparse linear model is that its estimations are not heavily affected by irrelevant basis functions, since it has the ability to prune them. For this reason, it is possible to consider sparse linear models with a large number of initial candidate basis functions and let the training algorithm prune the irrelevant ones.

The Relevance Vector Machine (RVM) [Tipping, 2001] is an instance of the sparse Bayesian linear model that assumes a particular form for the basis functions. Specifically, it has been motivated by the popular *Support Vector Machine* (SVM) [Schölkopf and Smola, 2001], and it assumes that the basis functions are *kernel functions*. A kernel function $K(\boldsymbol{x}_1, \boldsymbol{x}_2)$ is a function that corresponds to the inner product $\psi(\boldsymbol{x}_1)^T\psi(\boldsymbol{x}_2)$ at some high dimensional *feature space* defined by the mapping function $\psi(\boldsymbol{x})$. In this sense, kernel functions compute a measure of the similarity between two input points, after projecting them to the feature space. A common type of kernel function is the Gaussian kernel function:

$$K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \exp\left(-\frac{1}{2\sigma^2}\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2\right), \tag{3.64}$$

where $\sigma^2$ is the width of the kernel. Notice, that since kernel functions are not probability distribution functions we can omit the normalizing constant.

More specifically, the RVM assumes that the number of the basis functions is equal to the number $N$ of training examples and that each basis function $\phi_i(\boldsymbol{x})$ is a kernel $K(\boldsymbol{x}, \boldsymbol{x}_i)$ that computes the similarity between the input $\boldsymbol{x}$ and the $i$-th training example $\boldsymbol{x}_i$. Then, the output of the RVM model is given by

$$y(\boldsymbol{x}) = \sum_{i=1}^{N} w_i K(\boldsymbol{x}, \boldsymbol{x}_i). \tag{3.65}$$

Because of the sparse prior only a small subset of the available kernels remains in

the final model. The examples of the training set that correspond to the kernels that contribute to the estimation are called *relevance vectors* (RV).

## 3.6 Relation of RVM to other models

### 3.6.1 Gaussian Processes

Gaussian processes [Rasmussen and Williams, 2006] are collections of $N$ random variables $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, any finite number of which have a Gaussian distribution. A Gaussian process is completely specified by its mean $m(\boldsymbol{x}_i)$ and covariance function $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, which are defined as:

$$m(\boldsymbol{x}_i) = \langle f(\boldsymbol{x}_i) \rangle \,, \tag{3.66}$$

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle [f(\boldsymbol{x}_i) - m(\boldsymbol{x}_i)][f(\boldsymbol{x}_j) - m(\boldsymbol{x}_j)] \rangle \,. \tag{3.67}$$

A Gaussian process with these statistics is denoted as

$$f(\boldsymbol{x}) = GP(m(\boldsymbol{x}_i), k(\boldsymbol{x}_i, \boldsymbol{x}_j)). \tag{3.68}$$

It can be seen that the linear model of (3.1) is a special case of the Gaussian process model. Here, we consider a zero mean Gaussian distribution for the weights with arbitrary precision matrix $\boldsymbol{A}$

$$p(\boldsymbol{w}) = \mathrm{N}(\boldsymbol{w}|\boldsymbol{0}, \boldsymbol{A}^{-1}). \tag{3.69}$$

Then, the output $t_* = y(\boldsymbol{x}_*)$ of the model at an arbitrary point $\boldsymbol{x}_*$ is given by $t_* = \phi_*^T \boldsymbol{w} + \boldsymbol{\epsilon}$, where $\boldsymbol{\phi}_* = (\phi_1(\boldsymbol{x}_*), \ldots, \phi_M(\boldsymbol{x}_*))^T$ is a vector that contains all the basis functions evaluated at $\boldsymbol{x}_*$ and it is Gaussian distributed with:

$$\langle t_* \rangle = \boldsymbol{\phi}_*^T \langle \boldsymbol{w} \rangle = 0, \tag{3.70}$$

$$\left\langle t_*^1 t_*^2 \right\rangle = \boldsymbol{\phi}_*^{1T} \left\langle \boldsymbol{w}\boldsymbol{w}^T \right\rangle \boldsymbol{\phi}_*^2 = \boldsymbol{\phi}_*^{1T} \boldsymbol{A}^{-1} \boldsymbol{\phi}_*^2. \tag{3.71}$$

Therefore the Bayesian linear model is a special case of a Gaussian process whose covariance function is determined by the covariance $\boldsymbol{A}$ of the Gaussian prior of the weights and the basis functions $\phi_i(\boldsymbol{x})$. In the Bayesian linear model of Section 3.3 and in the sparse Bayesian linear model of Section 3.4 we assume that $\boldsymbol{A} = \alpha \boldsymbol{I}$ and $\boldsymbol{A} = \mathrm{diag}\{\alpha_1, \ldots, \alpha_M\}$ respectively. The main difference between the typical Gaussian process model and the Bayesian linear model is that in the former we typically assume a fixed covariance function while in the latter we update the covariance function by estimating the precision matrix $\boldsymbol{A}$.

Apart from the RVM, other versions of sparse Gaussian processes have also been developed, most of which are specific cases of a unified view proposed by [Quiñonero-Candela and Rasmussen, 2005].

### 3.6.2 Support Vector Machines

Support vector regression (SVR) [Smola and Schölkopf, 1998] is a regression method based on the popular *Support Vector Machine* (SVM) model [Schölkopf and Smola, 2001]. It is analogous to to the RVM in the sense that they both produce sparse solutions using an initially complex linear model. However, unlike the RVM which is based on the Bayesian framework an therefore sparseness is derived by a suitable weight prior, in SVR sparseness is derived by defining an appropriate penalty function for the noise.

More specifically, in SVR we ideally want to constraint all the errors to be smaller than a constant $\varepsilon$, without penalizing at all any errors that are smaller than $\varepsilon$. However, such a solution does not always exist and, when necessary, we may allow some errors to be larger than $\varepsilon$. Errors are penalized using the function:

$$|\xi|_\varepsilon = \begin{cases} 0 & \text{if } |\xi| \leq \varepsilon, \\ |\xi| - \varepsilon & \text{otherwise.} \end{cases} \tag{3.72}$$

Furthermore, in order to make smooth estimations for the unknown function, we seek the values of the weights that have the smallest magnitude. By introducing auxiliary variables $\xi_i, \xi_i^*$, the SVR training can be formulated as follows:

$$\text{minimize} \qquad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*) \tag{3.73}$$

$$\text{subject to} \qquad \begin{cases} y_i - \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}) - b & \leq \varepsilon + \xi_i \\ \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}) + b - y_i & \leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* & \geq 0 \end{cases} \tag{3.74}$$

$$\text{w.r.t.} \qquad \boldsymbol{w}, \boldsymbol{\xi}, \boldsymbol{\xi}^*, b. \tag{3.75}$$

Here, the constant $C$ determines the strength of the penalty for errors larger then $\varepsilon$. The above maximization can be performed efficiently by constructing the Lagrangian function and then considering its dual function.

There are several drawbacks of SVR compared to the Bayesian sparse linear regression approach. First, in SVR we need to select appropriate values for the parameters $C$ and $\varepsilon$, which is usually achieved with a cross-validation procedure. Furthermore, Bayesian sparse linear modelling is usually sparser than support vector regression or classification, where the number of support vectors scales with the size of the training set [Tipping, 2001]. Another potential advantage of the Bayesian sparse linear model is that, in contrast to SVR, it provides probabilistic predictions and therefore it also offers estimations of the accuracy of predictions. Finally, in SVR the kernel function needs to satisfy Mercer's condition, otherwise the SVR optimization problem might have no solution [Burges, 1998]. In RVM this constraint does not exist.

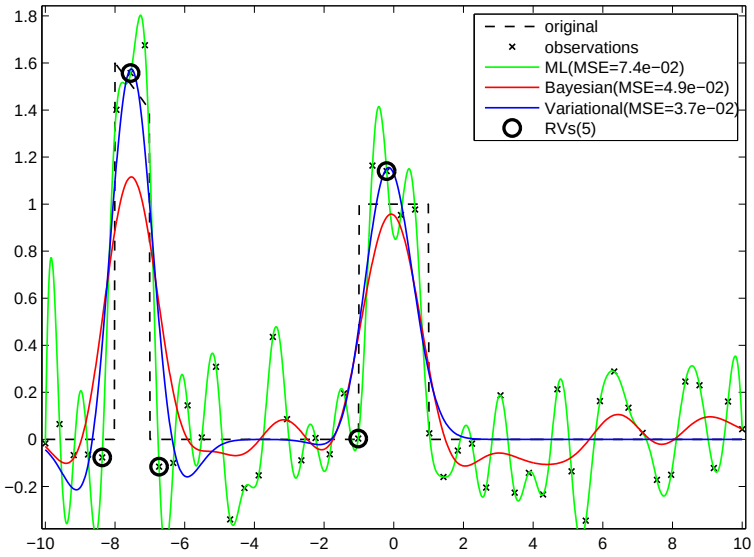



Figure 3.3: Linear regression solutions obtained by ML estimation, EM-based Bayesian inference and variational-EM sparse Bayesian inference.

## 3.7 Linear Regression Examples

Next, we present numerical examples to demonstrate the properties of the previously described linear regression models. We also demonstrate the advantages that can be reaped by using the variational Bayesian inference. An artificially generated signal $y(\boldsymbol{x})$ is used so that the "ground truth" is known. We have obtained $N = 50$ samples of the signal and added Gaussian noise of variance $\sigma^2 = 4 \times 10^{-2}$, which corresponds to signal to noise ratio $SNR = 6.6dB$. We used $N$ basis functions and, specifically, one basis function centred at the location of each observation, similarly to the RVM. The basis functions are Gaussian kernels of the form

$$\phi_i(\boldsymbol{x}) = K(\boldsymbol{x}, \boldsymbol{x}_i) = \exp\left(-\frac{1}{2\sigma_\phi^2}\|\boldsymbol{x} - \boldsymbol{x}_i\|^2\right). \tag{3.76}$$

We then used the observations to build a regression model, using i) ML estimation (3.9) ii) EM-based Bayesian inference (3.14) iii) sparse Bayesian inference (RVM) (3.34). For the case of sparse Bayesian model, we assume uninformative precision prior $p(\boldsymbol{\alpha})$ by setting $a = b = 0$.

Results are shown in Fig. 3.3. Notice, that the ML estimate follows exactly the noisy observations. Thus, it is the worst in terms of mean square error. This should be expected, since in this formulation we use as many basis functions as the observations and there is no constraint on the weights. The Bayesian methodology overcomes this problem since the weights are constrained by the priors. However, since this signal contains some regions with large variance and some with very small variance, it is clear that the stationary

prior does not provide the flexibility to accurately model local signal characteristics. In contrast, the hierarchical non-stationary prior is more flexible and seems to achieve better local fit. Actually, the solution corresponding to the latter prior, uses only a small subset of the basis functions, whose locations are shown as circled observations in Fig. 3.3. This happens because we have set $a = b = 0$, which defines an uninformative student's t distribution. Therefore, most weights are estimated to be exactly zero and only few relevance vectors are finally used for signal estimation (denoted as (RV) in Fig. 3.3).

## 3.8 Classification

The classification exhibits analogy to the regression problem, but in classification the unknown function maps input points $\boldsymbol{x}_n$ to discrete and unordered class labels $t_n$ rather than continuous valued outputs. Assuming $K$ classes, the outputs 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:

$$p(\boldsymbol{t}|\boldsymbol{w}) = \prod_{n=1}^{N}\prod_{k=1}^{K}\sigma(y_k(\boldsymbol{x}_n|\boldsymbol{w}))^{t_{nk}}, \tag{3.77}$$

where $\sigma(z) = \frac{1}{1+e^{-z}}$. For simplicity we only consider binary classification and assume that the outputs are coded so that $t_n \in \{0, 1\}$. Multiclass problems can be solved using the one-vs-all approach, which builds only two-class models. In binary classification the multinomial likelihood in (3.77) simplifies to a Bernoulli likelihood:

$$p(\boldsymbol{t}|\boldsymbol{w}) = \prod_{n=1}^{N} y_n^{t_n}(1 - y_n)^{1-t_n}, \tag{3.78}$$

where $y_n = \sigma(y(\boldsymbol{x}_n|\boldsymbol{w}))$.

Unlike the regression case, we cannot perform exact Bayesian inference with this likelihood. Instead we use the Laplacian approximation that is based on a Gaussian approximation of the posterior distribution around its mode [Tipping, 2001].

We can find the values of the weights $\boldsymbol{w}_{MP}$ that maximize the posterior $p(\boldsymbol{w}|\boldsymbol{t}, \boldsymbol{\alpha}) \propto p(\boldsymbol{t}|\boldsymbol{w})p(\boldsymbol{w}|\boldsymbol{\alpha})$

$$\boldsymbol{w}_{MP} = \underset{\boldsymbol{w}}{\operatorname{argmax}}\, p(\boldsymbol{w}|\boldsymbol{t}, \boldsymbol{\alpha}). \tag{3.79}$$

We consider again a zero mean Gaussian prior distribution for the weights:

$$p(\boldsymbol{w}|\boldsymbol{\alpha}) = \mathrm{N}(\boldsymbol{w}|\boldsymbol{0}, \boldsymbol{A}), \tag{3.80}$$

where $\boldsymbol{A}$ is the precision matrix. Then, the logarithm of the posterior distribution that

appears in (3.79) is given by

$$\log p(\boldsymbol{w}|\boldsymbol{t}, \boldsymbol{A}) = \log p(\boldsymbol{t}|\boldsymbol{w})p(\boldsymbol{w}|\boldsymbol{A}) + const \tag{3.81}$$

$$= \underset{\boldsymbol{w}}{\operatorname{argmax}} \sum_{n=1}^{N} (t_n \log y_n + (1 - t_n) \log(1 - y_n)) - \frac{1}{2}\boldsymbol{w}^T \boldsymbol{A}\boldsymbol{w} + const, \tag{3.82}$$

and its maximization can be efficiently performed using the iteratively reweighted least squares (IRLS) algorithm [Björck, 1996].

The Laplacian approximation is based on a Gaussian approximation of the posterior, or equivalently a quadratic approximation of its logarithm:

$$\log p(\boldsymbol{w}|\boldsymbol{t}, \boldsymbol{A}) \approx -\frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}_{MP})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{w} - \boldsymbol{w}_{MP}). \tag{3.83}$$

We set the covariance matrix $\boldsymbol{\Sigma}$ so that the Hessian matrix $-\frac{1}{2}\boldsymbol{\Sigma}^{-1}$ of the approximation is equal to the Hessian of the log-posterior

$$-\frac{1}{2}\boldsymbol{\Sigma}^{-1} = \nabla_{\boldsymbol{w}} \nabla_{\boldsymbol{w}} \log p(\boldsymbol{w}|\boldsymbol{t}, \boldsymbol{A}) = -(\boldsymbol{\Phi}^T \boldsymbol{B}\boldsymbol{\Phi} + \boldsymbol{A}), \tag{3.84}$$

with $\boldsymbol{B} = \operatorname{diag}\{\beta_1, \ldots, \beta_N\}$ and $\beta_n = \sigma(y(\boldsymbol{x}_n))[1 - \sigma(y(\boldsymbol{x}_n))]$, which gives

$$\boldsymbol{\Sigma} = (\boldsymbol{\Phi}^T \boldsymbol{B}\boldsymbol{\Phi} + \boldsymbol{A})^{-1}. \tag{3.85}$$

At the mode of the posterior $\boldsymbol{w}_{MP}$ its curvature is zero, therefore the mode can be found by setting the gradient of the log-posterior to zero:

$$\nabla_{\boldsymbol{w}} \log p(\boldsymbol{w}|\boldsymbol{t}, \boldsymbol{A})|_{\boldsymbol{w}_{MP}} = \boldsymbol{0}, \tag{3.86}$$

which gives

$$\boldsymbol{w}_{MP} = \boldsymbol{\Sigma}\boldsymbol{\Phi}^T \boldsymbol{B}\hat{\boldsymbol{t}}, \tag{3.87}$$

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

In summary, using the Laplacian approximation, the classification problem is mapped to a regression problem with heteroscedastic noise $p(\epsilon_n) = \mathrm{N}(\epsilon_n|0, \beta_n)$ [Tipping, 2001], whose precision is given by

$$\beta_n = y_n(1 - y_n), \tag{3.88}$$

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

$$\hat{\boldsymbol{t}} = \boldsymbol{\Phi}\boldsymbol{w} + \boldsymbol{B}^{-1}(\boldsymbol{t} - \boldsymbol{y}), \tag{3.89}$$

where $\boldsymbol{y} = (y_1, \ldots, y_N)^T$ and $\boldsymbol{B} = \operatorname{diag}(\beta_1, \ldots, \beta_N)$. Furthermore, depending on the structure of the covariance matrix of the prior of the weights in (3.80) we can obtain either the simple Bayesian linear model of Section 3.3 by setting $\boldsymbol{A} = \alpha\boldsymbol{I}$, or the sparse

Bayesian linear model of Section 3.4 by setting $\boldsymbol{A} = \text{diag}\{\alpha_1, \ldots, \alpha_M\}$.

## 3.9 Conclusions

In this chapter we considered the regression problem using the linear model. In Section 3.2 we described the simplest approach that treats the weights of the linear model as parameters and estimates them using the ML approach. If the linear model contains a large number of basis functions, it is very flexible and the ML estimations may be heavily corrupted by noise. For this reason, in Section 3.3 the Bayesian linear model is described, that treats the weights as random variables. In its simplest form, it uses a stationary Gaussian distribution for the weights, which forces them to small values and allows tractable Bayesian inference. However, there are many advantages in considering more complex prior distributions. In Section 3.4 we presented the sparse Bayesian linear model that uses a non-Gaussian prior distribution to provide sparse solutions that use only few of the available basis functions.

In the next chapters, we use the sparse Bayesian linear model to treat some image processing problems. More specifically, we use the sparse linear model to perform regression of images, which presents several computational difficulties, due to the large scale of the problem. For this reason, in Chapter 4 we propose a training algorithm for sparse Bayesian regression of images that is based on operations in the Fourier domain. Then we use this algorithm to tackle the problem of detecting objects in images and the problem of blind image deconvolution (in Chapter 5). In Chapter 6 we propose a methodology to learn parameters of the basis functions. In contrast to the cross-validation approach that is commonly used for selecting basis function parameters, the proposed incremental approach uses different parameter values for each basis function. We evaluate this methodology in several regression and classification datasets and in we apply it for detecting activations in functional neuroimages. In Chapter 7 we extend the previous method in order to simultaneously perform feature selection and we apply it to the analysis of DNA microarray datasets.

# Chapter 4

# Sparse Multikernel Regression for Image Analysis



## 4.1 Introduction

A major issue with the linear model is how to select appropriate basis functions. Typically, using a large number of basis functions results in a very flexible model which overfits the noise and has poor generalization capability. However, this is not an issue in the sparse linear model, because it computes sparse solutions that use only a small number of the available basis functions. For example, the relevance vector machine (RVM) [Tipping, 2001] initially places one kernel basis function at each point of the training set. In the RVM it is important to select appropriate kernel function, for example Gaussian kernel functions are very commonly used. Parameters of the kernel function, such as the variance of Gaussian kernels are typically selected using cross-validation techniques.

In this chapter we propose an extension of the typical RVM, which is based on a linear model with several different basis functions and we call the *multikernel RVM* [Tzikas et al., 2006b, 2007b]. Similarly to the RVM, each of these basis functions is placed at all

the points of the training set, therefore the multikernel model is given by

$$y(\boldsymbol{x}) = \sum_{m=1}^{M} \sum_{i=1}^{N} w_{mi} \phi_m(\boldsymbol{x}, \boldsymbol{x}_i), \tag{4.1}$$

where $N$ is the number of training points and $M$ is the number of different basis function types and $\phi_m(\boldsymbol{x}, \boldsymbol{x}_i)$ is the $i$-th basis function. For example, we might use Gaussian basis functions of several different widths

$$\phi_m(\boldsymbol{x}, \boldsymbol{x}_i) = \exp\left[-h_m^{-2} \|\boldsymbol{x} - \boldsymbol{x}_i\|^2\right], \tag{4.2}$$

where $h_m$ is the width of the Gaussian kernel.

We then apply this model for modeling images. Unfortunately, since $N$ is large (equal to the number of image pixels) the standard training algorithms are too computationally demanding, even for small images. We notice that if the training points $x_i$ lie on a uniform grid, the linear model can be rewritten as the *convolution* of the weight vector $\boldsymbol{w} = (w_1, \ldots, w_N)^T$ with a vector $\boldsymbol{\phi} = (\phi(\boldsymbol{x_1}), \ldots, \phi(\boldsymbol{x_N}))^T$, which consists of the basis function $\phi(\boldsymbol{x})$ evaluated at the training points $\boldsymbol{x_i}$. The output of the linear model can then be written as:

$$\boldsymbol{y} = \boldsymbol{\phi} * \boldsymbol{w}, \tag{4.3}$$

where $*$ denotes the convolution operator and $\boldsymbol{y} = (y(\boldsymbol{x_1}), \ldots, y(\boldsymbol{x_N}))^T$ is a vector containing the outputs of the model evaluated at the training points. In section 4.2 we present in detail the multikernel RVM model and propose an alternative implementation of the EM-based training algorithm [Tipping, 2001]. Our implementation computes *convolutions in the DFT domain*, improving both time and memory requirements and allows to train RVM models on high resolution images, with reasonable computational costs. The proposed implementation is evaluated in section 4.4.

We then use the proposed algorithm to treat the *object detection* problem, which is the problem of finding the location of an unknown number of occurrences of a given 'target' image in another given 'observed' image, under the presence of noise. The 'target' may appear significantly different in the observed image, as a result of being scaled, rotated, occluded by other objects, different illumination conditions and other effects.

The most common approaches to solve the object detection problem are variants of the matched filter, such as the phase-only [Horner and Gianino, 1984] and the symmetric phase-only [Chen et al., 1994] matched filters. These are based on computing the correlation image between the 'observed' and 'target' images and then using a threshold to determine the locations where the 'target' object is present. Alternatively, the problem can be formulated as an image restoration problem, where the image to restore is considered as an impulse function at the location of the 'target' object. This technique allows many interesting background models to be considered, such as autoregressive models [Abu-Naser et al., 1998]. A different object detection approach, which has been success-

fully applied on face detection [Viola and Jones, 2001], is to split the observed image in several regions and train a classifier with some features of the target 'object' in order to decide which regions contain the 'target' object.

In section 4.5 we propose a method for object detection, which is based on training a multikernel RVM model on the 'observed' image [Tzikas et al., 2007b]. The RVM model consists of two sets of basis functions: basis functions that are used to model the 'target' image and basis functions that are used to model the background. After training the model, each 'target' basis function whose corresponding weight is larger than a specified threshold is considered as a detected 'target' object. Finally, we provide examples of the proposed RVM-based object detection algorithm and a comparison with the autoregressive impulse restoration [Abu-Naser et al., 1998] method.

## 4.2 Multikernel RVM for Image Regression

The linear model is very efficient provided that suitable basis functions $\phi_i$ are selected and that there exist adequate training examples. Thus, finding a basis function set that describes the data well is an important problem, that is difficult to solve. In this paper, we simultaneously consider $M$ different types of basis functions $\phi_1, \ldots, \phi_M$ centered at each training point $\boldsymbol{x}_i$, resulting in the following model with $MN$ basis functions:

$$y(\boldsymbol{x}) = \sum_{m=1}^{M} \sum_{i=1}^{N} w_{mi} \phi_m(\boldsymbol{x} - \boldsymbol{x}_i), \tag{4.4}$$

Although we use so many basis functions (and therefore parameters), overfitting should not be a concern, because of the sparseness in the final RVM model.

In order to model a $N_i \times N_j$ image $t$ using an RVM, we assume that the intensity $t(i,j)$ of the observed image at location $(i,j)$ has been generated from the output $y(i,j)$ of the model (4.4) at the same location, after addition of independent white noise $\epsilon(i,j)$:

$$t(i,j) = y(i,j) + \epsilon(i,j), \tag{4.5}$$

$$\epsilon(i,j) \sim N(0, \beta^{-1}), \tag{4.6}$$

where $\beta$ is the inverse variance of the noise.

Defining $\boldsymbol{t} = (t(1,1), \ldots, t(1,N_j), \ldots, t(N_i,N_j))^T$ to be a vector that contains the intensities of the image pixels in lexicographical order and similarly defining the noise vector $\boldsymbol{\epsilon} = (\epsilon(1,1), \ldots, \epsilon(1,N_j), \ldots, \epsilon(N_i,N_j))^T$, Eq. (4.5) can be rewritten in compact form as:

$$\boldsymbol{t} = \boldsymbol{\Phi}\boldsymbol{w} + \boldsymbol{\epsilon} = \sum_{m=1}^{M} \boldsymbol{\Phi}_m \boldsymbol{w}_m + \boldsymbol{\epsilon}, \tag{4.7}$$

where $\boldsymbol{\Phi}$ is the $N \times (MN)$ design matrix, each column of which is a vector with the values of a basis function at all the training points. The design matrix can be partitioned

as $\boldsymbol{\Phi} = (\boldsymbol{\Phi}_1, \ldots, \boldsymbol{\Phi}_M)$, with $\boldsymbol{\Phi}_m = (\boldsymbol{\phi}_{m1}, \ldots, \boldsymbol{\phi}_{mN})$ being the part of the design matrix corresponding to basis functions of type $\phi_m(\boldsymbol{x})$ and $\boldsymbol{\phi}_{mi} = (\phi_m(\boldsymbol{x_1} - \boldsymbol{x_i}), \ldots, \phi_m(\boldsymbol{x_N} - \boldsymbol{x_i}))^T$ being a vector consisting of the basis function $\phi_m(\boldsymbol{x} - \boldsymbol{x_i})$ evaluated at all the training points. The weight vector $\boldsymbol{w}$ can be similarly partitioned as $\boldsymbol{w} = (\boldsymbol{w_1}^T, \ldots, \boldsymbol{w_M}^T)^T$, with each $\boldsymbol{w_m} = (w_{m1}, \ldots, w_{mN})$, $m = 1, \ldots, M$, consisting of the weights corresponding to basis function $\phi_m(\boldsymbol{x})$. The likelihood of the data set can then be written as:

$$p(\boldsymbol{t}|\boldsymbol{w}, \beta) = (2\pi)^{-N/2} \beta^{N/2} \exp\left\{ -\frac{1}{2}\beta \|\boldsymbol{t} - \boldsymbol{\Phi}\boldsymbol{w}\|^2 \right\}. \tag{4.8}$$

Given that the described model has $M$ times more parameters than the available training examples, it is essential to seek a sparse solution. Under the Bayesian framework sparseness is obtained by assigning suitable prior distributions on the parameters as mentioned in Chapter 2 and Chapter 3. Specifically, independent Gaussian prior distributions with unknown variances are imposed on the weights $\boldsymbol{w}$:

$$p(\boldsymbol{w}) = \prod_{m=1}^{M} \prod_{i=1}^{N} p(w_{mi}) = \prod_{m=1}^{M} \prod_{i=1}^{N} \mathrm{N}(w_{mi}|0, \alpha_{mi}^{-1}), \tag{4.9}$$

where $\alpha_{mi}$ is the inverse variance of the corresponding weight $w_{mi}$. These parameters are assumed unknown and Gamma hyperpriors are assigned to them. The inverse noise variance $\beta$ may also be assumed unknown and similarly, a Gamma prior distribution is assigned to it:

$$p(\boldsymbol{\alpha}) = \prod_{m=1}^{M} \prod_{i=1}^{N} \Gamma(a, b), \tag{4.10}$$

$$p(\beta) = \Gamma(c, d), \tag{4.11}$$

where $\boldsymbol{\alpha} = (\alpha_{11}, \ldots, \alpha_{1N}, \ldots, \alpha_{MN})$.

Since this model is an instance of the Bayesian linear model described in Chapter 3, the posterior weight distribution is:

$$p(\boldsymbol{w}|\boldsymbol{t}, \boldsymbol{\alpha}, \beta) = \mathrm{N}(\boldsymbol{w}|\boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{4.12}$$

with

$$\boldsymbol{\Sigma} = (\beta \boldsymbol{\Phi}^T \boldsymbol{\Phi} + \boldsymbol{A})^{-1}, \tag{4.13}$$

$$\boldsymbol{\mu} = \beta \boldsymbol{\Sigma} \boldsymbol{\Phi}^T \boldsymbol{t}, \tag{4.14}$$

where $\boldsymbol{A} = \mathrm{diag}\{\boldsymbol{\alpha}\}$ and the precision parameters can be updated using:

$$\alpha_{mi} = \frac{1 - \alpha_{mi}\Sigma_{ii}}{\mu_{mi}^2}, \tag{4.15}$$

$$\beta = \frac{N - \sum_{i=1}^{N}(1 - \alpha_i \Sigma_{ii})}{\|\boldsymbol{t} - \boldsymbol{\Phi}\boldsymbol{\mu}\|^2}. \tag{4.16}$$

The learning algorithm proceeds by iteratively computing the posterior statistics $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$ of the weights, given by (4.13) and (4.14) and then updating the hyperparameters using (4.15) and (4.16). Computation of $\Sigma$ involves inverting an $MN \times MN$ matrix which is an $O(M^3N^3)$ procedure. During the training process, many of the hyperparameters are set to infinite values and the corresponding basis functions can be punned, allowing computation of the posterior statistics in $O(L^3)$ time, where $L$ is the number of functions that remain in the model. This results in significant speed-up of the latter iterations of the algorithm, however in the first iteration all the basis functions have to be considered and the overall complexity is still $O(M^3N^3)$. For this reason it is difficult to apply this algorithm on large training sets, such as images. Furthermore, the incremental training algorithm of Section 3.4.3 is an important improvement, but it still cannot be used for large scale problems, such as modeling images. In this chapter we propose an RVM implementation based on *DFT computations*, that successfully resolves the problem of computational complexity.

## 4.3 Sparse Linear Regression in the DFT domain

It can be observed that if the training points are the pixels of an image, or generally uniform samples of a signal, then the RVM given by (4.4) can be written using a *convolution* as:

$$\boldsymbol{y} = \sum_{m=1}^{M} \boldsymbol{\phi}_m * \boldsymbol{w}_m. \tag{4.17}$$

Equation (4.7) still holds, with the additional property that matrices $\boldsymbol{\Phi}_m$ are *circulant*. This means that each row of $\boldsymbol{\Phi}_m$ can be obtained with a circular shift of the elements of the previous row. For this reason, we do not need to store in memory the whole matrix $\boldsymbol{\Phi}_m$, but it is sufficient to store only one of its rows. Furthermore, because $\boldsymbol{\Phi}_m$ is circulant the product $\boldsymbol{\Phi}_m \boldsymbol{w}_m$ is a convolution which can be efficiently computed in the DFT domain by multiplying the DFT $\mathcal{F}_m$ and $\mathcal{W}$ of the basis function $\boldsymbol{\phi}_m$ and the weight vector $\boldsymbol{w}$ respectively.

$$\mathcal{T}_i = \sum_{m=1}^{M} \mathcal{F}_{mi} \mathcal{W}_{mi}, \tag{4.18}$$

where $\mathcal{T}$ is the DFT of the vector $\boldsymbol{t}$. This observation allows computation of the output of the model without using the complete design matrix, but using only one basis vector, improving memory complexity from $O(N^2)$ to $O(N)$ and time complexity from $O(N^2)$ to $O(N \log N)$.

The posterior statistics of the weights $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ can also be computed in the DFT domain, thus obtaining the same advantage. Beginning with (4.14), the posterior mean

of the weights can be found by solving the equation:

$$\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} = \beta\boldsymbol{\Phi}^T\boldsymbol{t}, \tag{4.19}$$

$$(\beta\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \boldsymbol{A})\boldsymbol{\mu} = \beta\boldsymbol{\Phi}^T\boldsymbol{t}. \tag{4.20}$$

Instead of analytically inverting the matrix $\beta\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \boldsymbol{A}$, which is computationally expensive and requires inversion of the large design matrix $\boldsymbol{\Phi}$, we solve equation (4.20) by using the conjugate gradient method [Shewchuk, 1994] to minimize the following quadratic function:

$$\boldsymbol{\mu}^* = \underset{\boldsymbol{\mu}}{argmin}(\boldsymbol{\mu}^T(\beta\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \boldsymbol{A})\boldsymbol{\mu} - \boldsymbol{\mu}^T\beta\boldsymbol{\Phi}^T\boldsymbol{t}). \tag{4.21}$$

The quantities $\beta\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{\mu}$ and $\beta\boldsymbol{\Phi}^T\boldsymbol{t}$ can be easily computed in the DFT domain since $\boldsymbol{\Phi}$ is circulant, while computation of $\boldsymbol{A}\boldsymbol{\mu}$ is straightforward since $\boldsymbol{A}$ is diagonal. In the ideal case, the conjugate gradient method is guaranteed to find the exact minimum after $N$ iterations. In practice, a very good estimate can be obtained in only a few iterations.

Unfortunately, in order to compute the posterior weight covariance $\boldsymbol{\Sigma}$ we have to invert the matrix $\beta\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \boldsymbol{A}$, which is a computational burden. To overcome this problem, we notice that we only need to compute the diagonal elements of $\boldsymbol{\Sigma}$ and consider two approximations.

The simplest approximation is to consider only the main diagonal of the matrix $\beta\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \boldsymbol{A}$, and estimate $\Sigma_{ii}$ as:

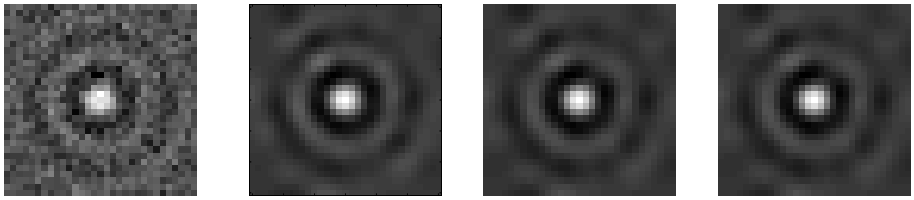
$$\Sigma_{ii} = (\beta\|\boldsymbol{\phi}\|^2 + \alpha_i)^{-1}, \tag{4.22}$$

with $\boldsymbol{\phi} = (\phi_{11}^T, \ldots, \phi_{1M}^T)^T$. Although this approximation is not valid in general, it has been proved very effective in the experiments, because the matrix $\boldsymbol{A}$ has generally very large values and is the dominant term in the sum $\beta\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \boldsymbol{A}$.

An alternative approximation that has been considered is to approximate the matrix $\beta\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \boldsymbol{A}$ with a circulant matrix and estimate $\Sigma_{ii}$ as:

$$\Sigma_{ii} = (\beta\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \alpha_i\boldsymbol{I})^{-1} = \frac{1}{N}\sum_{j=1}^{M}(\beta\mathcal{F}_j^2 + \alpha_i)^{-1}, \tag{4.23}$$

where $F_j$ is the $j$-th element of the DFT of the first row of matrix $\Phi_j$. Notice, that a different (circulant) approximating matrix has to be inverted for the computation of each element of the diagonal of $\boldsymbol{\Sigma}$. For this reason, this approximation requires more computations than the first and may be impractical for large images.



(a) (b) (c) (d)

Figure 4.1: (a) An artificially generated image with added noise. Estimates of (b) the RVM algorithm and the DFT-RVM algorithm using (c) the diagonal approximation of Eq. (4.22) and (d) the circulant approximation of Eq. (4.23).

| Algorithm | $\sigma^2 = 1$ | $\sigma^2 = 2$ | $\sigma^2 = 4$ | $\sigma^2 = 8$ |
|---|---|---|---|---|
| RVM | 0.055(229) | 0.038(84) | 0.040(45) | 0.052(70) |
| DFT-RVM | 0.055(249) | 0.039(120) | 0.048(49) | 0.111(12) |
| DFT-RVM(2) | 0.058(234) | 0.041(105) | 0.077(165) | 0.111(190) |

Table 4.1: Mean square error of the typical RVM algorithm and the DFT-based algorithm with the two approximations for several choices of the kernel width $\sigma^2$. Inside parenthesis is the number of relevance vectors for each case.

## 4.4 Evaluation of the DFT-based RVM implementation

In order to verify the validity and evaluate the performance of the proposed DFT-based implementation we consider the following artificial example. We sampled uniformly the function

$$t(x, y) = \frac{\sin(\|x + y\|)}{\|x + y\|}, \tag{4.24}$$

and added white Gaussian noise of variance 0.1 to generate a 30x30 image shown in Fig. 4.1. We then applied both the typical and the DFT-based algorithm to estimate the parameters of an RVM model, which was then evaluated at each pixel location to produce an estimate of the initial function $t$. Figure 4.1 shows the estimates obtained using the typical RVM algorithm and the DFT-based algorithm with the two different approximations respectively. Averages over 10 runs with different noise realizations of the mean squared error (MSE) of each method and the number of relevance vectors are shown in Table 4.1 for four different widths $\sigma^2$ of the kernel. We notice that the first (diagonal) approximation typically gives better results than the second (circulant) approximation and it also requires less computations. Also notice that the approximation gives excellent results when the size of the kernel is small, because the matrix $\Sigma$ is almost diagonal.

Unfortunately, we can't compare the algorithms for larger images because we can't apply the typical RVM algorithm on larger datasets. However, we demonstrate the effectiveness of the proposed DFT-based algorithm on large scale regression problems, by training a multikernel RVM model with Gaussian kernels of sizes $\sigma_1^2 = 2$, $\sigma_2^2 = 4$ and

(a) (b)

Figure 4.2: (a) An $128 \times 128$ image with added gaussian noise. (b) Estimate of the DFT-RVM algorithm using gaussian kernels of width 2, 4 and 8.

$\sigma_3^2 = 8$ on a $256 \times 256$ image. The estimated image, shown in Fig. 4.2, is improved with respect to the initial noisy image, having $ISNR = 2.2$, where $ISNR$ is defined as $ISNR = 10 \log \left( \|f - g\|^2 / \|f - \hat{f}\|^2 \right)$ and is a measure of the improvement in quality of the estimated image with respect to the initial image.

## 4.5 Object Detection Using the Multikernel RVM Model

In this section, we present an method for object detection, which is based on training a multikernel RVM model on the 'observed' image [Tzikas et al., 2007b]. The RVM model consists of two sets of basis functions: basis functions that are used to model the 'target' image and basis functions that are used to model the background. After training the model, each 'target' basis function that remains in the model can be considered as a detected 'target' object. However, if the background basis functions are not flexible enough, 'target' functions may also be used to model areas of the background. Thus, we should consider only 'target' basis functions whose corresponding weight is larger than a specified threshold.

We denote by $\boldsymbol{t} = (t(1,1), \ldots, t(1,N_j), \ldots, t(N_i,N_j))^T$ a vector consisting of the intensity values of the pixels of the 'observed' image in lexicographical order. We model this image using the following RVM model:

$$\boldsymbol{t} = \sum_{i=1}^{N} w_{ti} \phi_t(\boldsymbol{x} - \boldsymbol{x}_i) + \sum_{i=1}^{N} w_{bi} \phi_b(\boldsymbol{x} - \boldsymbol{x}_i) + \boldsymbol{\epsilon}, \tag{4.25}$$

where $N = N_i N_j$, $\phi_t$ is the 'target' basis function which is a vector consisting of the intensity values of the pixels of the 'target' image, and $\phi_b$ is the background basis function,

which we selected to be a Gaussian function. After training the RVM model, we obtain the vectors $\boldsymbol{\mu}_t$ and $\boldsymbol{\mu}_b$, which are the posterior mean for the kernel and background weights respectively, using (4.14). Ideally, 'target' kernel functions would only be used to model occurrences of the 'target' object. However, because the background basis functions are often not flexible enough to model the background accurately, some 'target' basis functions may have been used to model the background as well. In order to decide which 'target' basis functions actually correspond to 'target' occurrences, the posterior 'target' weight means are thresholded, and only those that exceed a specified threshold $T$ are considered significant:

$$|\mu_{ti}| > T \Rightarrow \text{Target exists at location } i. \tag{4.26}$$

Choosing a low threshold may generate false alarms, indicating that the object is present at locations where it actually doesn't exist. On the other hand, choosing a high threshold may result in failing to detect an existing object. There is no unique optimal value for the threshold, but instead it should be chosen depending on the characteristics of the application.

It must be also noted that the Support Vector Machine (SVM) cannot be used with this approach, since the basis functions used here are the 'target' image and do not correspond to valid kernel functions, since they do not satisfy the Mercer condition.

Next we present experiments that demonstrate the improved performance of the DFT-RVM algorithm compared to autoregressive impulse restoration (ARIR), which is an effective method for object detection [Abu-Naser et al., 1998]. We demonstrate two examples where the 'observed' images have been constructed by adding the 'target' object to a background image and then adding white Gaussian noise. Images consisting of the values of the kernel weights computed with the DFT-RVM algorithm are shown in Fig. 4.3 and compared with the output of the ARIR method. Notice that, because of the RVM sparseness property, the output of the algorithm is zero at most locations where there is no target object. This property of the DFT-RVM detection method, is the main reason for the improved detection performance.

### 4.5.1 Experimental Evaluation

When evaluating a detection algorithm it is important to consider the detection probability $P_D$, which is the probability that an existing 'target' is detected and the probability of false alarm $P_{FA}$, which is the probability that a 'target' is incorrectly detected. Any of these probabilities can be set to an arbitrary level by selecting an appropriate value for the threshold $T$. The receiver operating characteristics (ROC) curve is a plot of the probability of detection $P_D$ versus the probability of false alarm $P_{FA}$ that provides a comprehensive way to demonstrate the performance of a detection algorithm. However, the ROC curve is not suitable for evaluating object detection algorithms because it only considers if an algorithm has detected an object or not; it does not consider if the object was detected in the correct location. Instead, we can use the localized ROC (LROC)

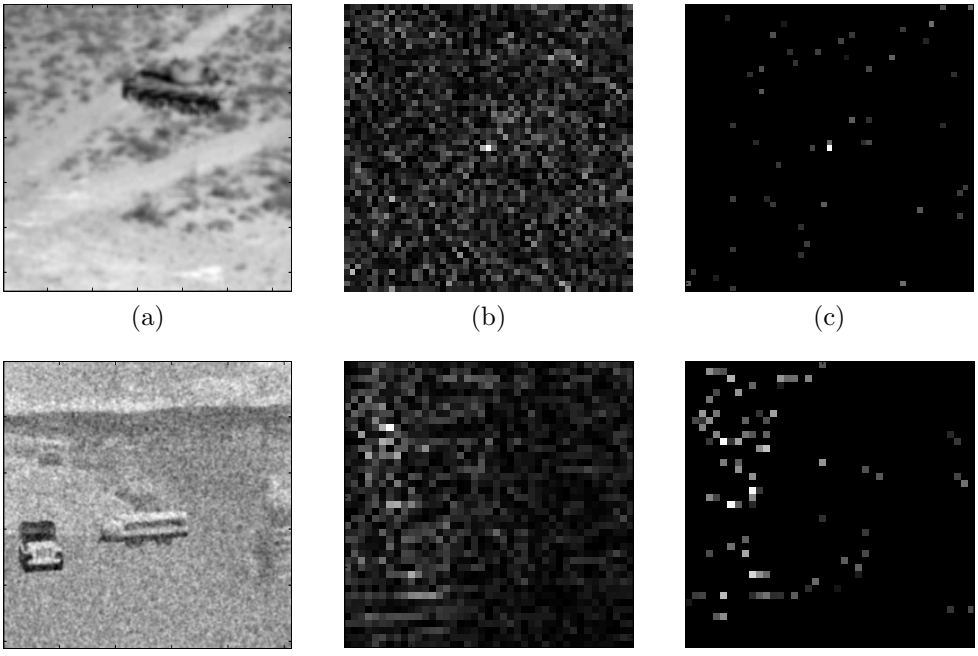



Figure 4.3: Two object detection examples. (a) and (d) are the 'observed' images, (b) and (e) are the results of the ARIR algorithm and (c) and (f) are the results of the DFT-RVM algorithm. The target object is the tank in image (a) and the jeep in image (d). In the results, only a small area around the target is shown. In all cases, the output of both algorithms is maximum at the location of the target. However, at all other locations, where there is no target and the output should ideally be zero, DFT-RVM outperforms the ARIR algorithm, since its output is zero at most locations.

curve which is a plot of the probability of detection and correct localization $P_{DL}$ versus the probability of false alarm and considers also the location where a 'target' has been detected.

In order to evaluate the performance of the algorithm, we created 50 'observed' images by adding a 'target' image at a random location of the background image, and another 50 'observed' images without the 'target' object. White Gaussian noise of variance $\sigma^2 = 20$ was then added to each 'observed' image, that corresponds to signal to noise ratio $22dB$. The DFT-RVM algorithm was then used to estimate the parameters of an RVM model with a 'target' kernel and a Gaussian background kernel for each 'observed' image, generating 100 kernel weight images. The background basis functions were Gaussian functions of the form $\phi_i(\boldsymbol{x}) = \exp(-\frac{1}{r^2}\|\boldsymbol{x} - \boldsymbol{x_i}\|^2)$ with the width parameter set to $r = 6$. The kernel weight images were then thresholded for many different threshold values and estimates of the probabilities $P_{DL}$ and $P_{FA}$ were computed for each threshold value. Similar experiments were also performed for the ARIR algorithm and an LROC curve was plotted for each algorithm. Figure 4.4 shows the LROC curve of each algorithm for the two cases of background and target images shown in Fig. 4.3. It can be observed

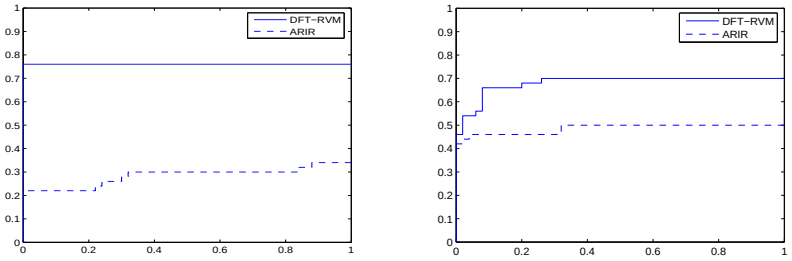



Figure 4.4: LROC curves of the ARIR and DFT-RVM algorithms for the two detection problems shown in Fig. 4.3.

that the area under the LROC curve, which is a common measure of the performance of a detection algorithm, is significantly larger for the DFT-RVM algorithm. Another important observation is that the LROC curve is high for small values of $P_{FA}$, since usually the threshold is chosen so that only a small fraction of false detections is allowed.

## 4.6 Conclusions

We have proposed the multikernel RVM model and an approximate but accelerated inference method for training the RVM model on large scale images, based on fast computation of the posterior statistics in the DFT domain [Tzikas et al., 2006b, 2007b]. Experiments on images demonstrate that the proposed approximation allows inference on large scale images, where the typical RVM algorithm is too computationally demanding to run. We then presented an application of the method to the object detection problem. Experimental results indicate that this approach is more robust than existing methods. Furthermore, the proposed technique can be extended to solve the rotation and scaling invariant object detection problem, by optimizing the model with respect to rotation and scaling of the basis functions.

# Chapter 5

# Bayesian Blind Image Deconvolution with Student's t priors



## 5.1 Introduction

In this chapter we propose the use of the sparse Bayesian linear model to estimate the PSF in the *blind image deconvolution* (BID) problem [Tzikas et al., a]. In order to reduce the computational cost of inference, the proposed algorithm uses operations in the DFT domain, similarly to the algorithm of Chapter 4.

In the BID problem, we observe an image which has been degraded by blurring and addition of some noise source. Such images are commonly observed in many situations; for example motion blur might be induced by motion of the camera during the image acquisition and in astronomy blurring is also induced by the atmosphere. Here, we assume that the *point spread function* (PSF) of the blur is the same at all regions of the image, which happens when the blur caused by motion of objects is negligible. In this case the blurring can be modeled as a *convolution* of the initial image with the blurring PSF. The process that generates the observed image is summarized in Fig. 5.1 and an example is shown in Fig. 5.1.

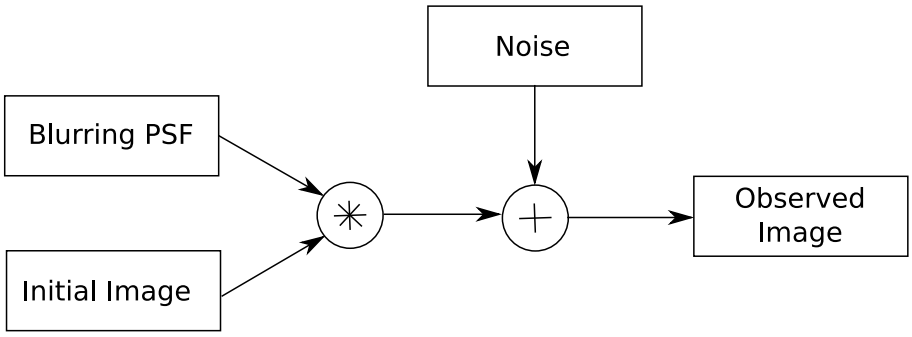



Figure 5.1: Generation mechanism of the observed image in blind image deconvolution.

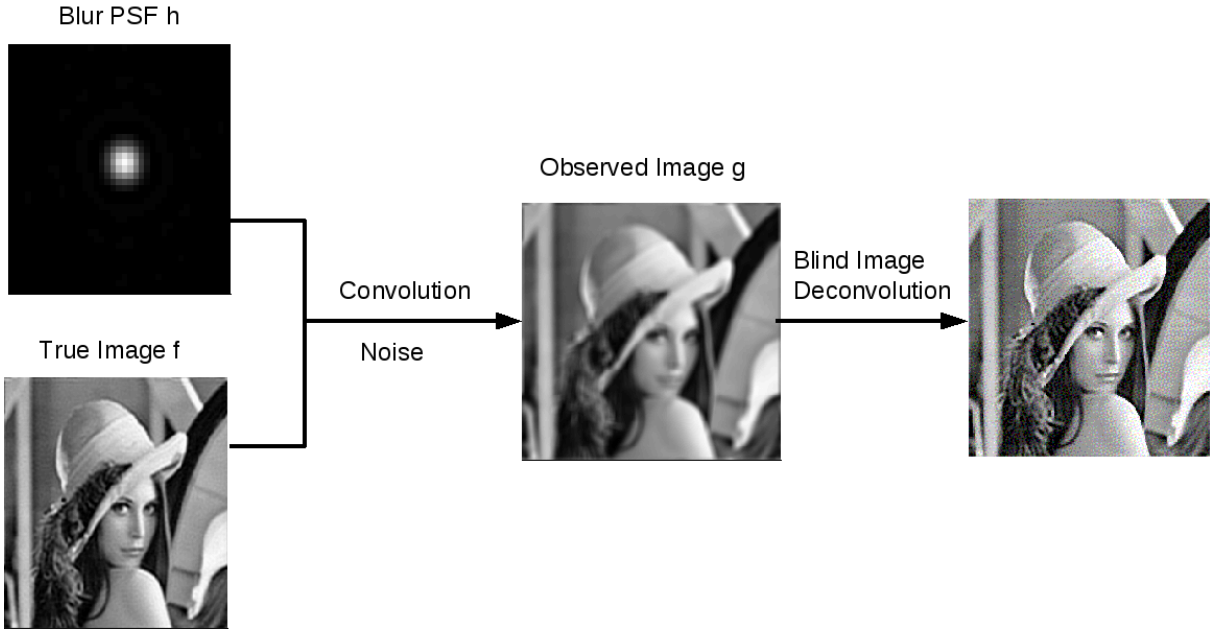



Figure 5.2: Example blind image deconvolution.

Because in blind image deconvolution both the initial image and the point spread function (PSF) are unknown, the observed data are not sufficient to uniquely specify both the unknown image and PSF. In order to resolve this ambiguity, prior knowledge (constraints) has to be used for both the image and the PSF. Over the years a number of methodologies have been employed to introduce constraints in BID. A rather old survey paper on this problem is [Kundur and Hatzinakos, 1996a,b], while a very recent edited book on BID methods is [Campisi and Egiazarian, 2007].

One category of such methods is based on regularization using the total variation (TV) principle. These methods define a distance function based on the data and use smoothness constraints on both the image and the PSF based on the TV principle [Chan and Wong, 1998]. A survey of recent developments on TV methods in image recovery problems and a book containing a review of the recent developments in mathematical tools for low level image processing problems can be found in [Chan et al., 2005a] and

[Chan et al., 2005b] respectively. Methods based on anisotropic diffusion regularization have been also proposed [You and Kaveh, 1999], however they require the choice of the diffusion operator. There are also methods based on soft constraints [Yap et al., 2005, Chen and Yap, 2005], which are very flexible, however, the form and the type of the used soft constraints is ad-hoc. Methods based on sparse image representations and quasi likelihood criteria have been also suggested [Bronstein et al., 2005].

Another way to apply constraints to the image and the PSF, is through the use of the Bayesian methodology. In this approach the unknown quantities are assumed to be random variables and suitable prior distributions are selected to impose the desired characteristics [Jeffs and Christou, 1998, Galatsanos et al., 2002, Miskin and MacKay, 2000, Fergus et al., 2006, Likas and Galatsanos, 2004, Molina et al., 2006]. Unfortunately, since the BID data generation model is non-linear, the posterior distribution of the unknown image and PSF can not be computed analytically. Thus, Bayesian inference using conventional methods, such as Maximum Likelihood (ML) via the Expectation Maximization (EM) algorithm, cannot be applied.

These difficulties can be overcome using the variational Bayesian methodology [Bishop, 2006] and [Jordan et al., 1998] described in Chapter 2. To our knowledge this methodology was first applied to the BID problem in [Miskin and MacKay, 2000]. In that work the PSF and the image were modeled by an exponential and a mixture of exponential distributions, respectively. Furthermore, the support of the PSF was known, and the images were line drawings which are sparse, in the sense that their intensity is zero at most locations. This work was recently extended for natural scene images in [Fergus et al., 2006] with promising results. More specifically, a mixture of Gaussians for the gradient of the image, and a mixture of exponentials for the PSF were used. This PSF model allows only positive PSF intensities and encourages sparsity, all of which are desirable properties for BID. However, it does not model spatial PSF correlations. In another line of work [Likas and Galatsanos, 2004], a simultaneously autoregressive (SAR) prior and a Gaussian prior with unknown mean and spherical covariance have been used for the image and PSF respectively. This methodology was extended in [Molina et al., 2006] to account for spatial PSF correlations using SAR models for both PSF and the image. However, this approach fails to model edges in the image or PSF and does not provide a mechanism to estimate the support of the PSF.

In this chapter we propose a kernel-based Bayesian approach for the BID problem that allows reconstruction of image edges, models spatial PSF correlations and estimates the PSF support [Tzikas et al., 2006a, 2007c,a, a]. The main contribution of this work, is a model that enforces PSF smoothness and simultaneously estimates the PSF support. This is achieved by modeling the PSF as an RVM model, as described in Chapter 3. More specifically, the PSF is modeled as a linear combination of kernel functions that are placed at all the pixels of the image. Thus, the amount of smoothness can be controlled by appropriately selecting the kernel function. The support of the PSF can be arbitrarily large, since we placed kernel functions at all image pixels. However, following the sparse

Bayesian linear model approach, we assume that the distribution of the weights of the kernels that models the PSF is a heavy tailed Student's t distribution. As explained in Section 3.4.4, this distribution favors sparse models, forcing most of the weights to become zero and therefore limiting the support of the PSF. Furthermore, in order to promote smooth image estimates, we constrain the local image differences, by assuming that they follow a zero-mean Student's-t distribution in order to allow reconstruction of edges [Chantas et al., 2006]. Finally, we model the errors of the imaging model with a Student's-t distribution. This is important, not only because the noise in the observed image may not be Gaussian, but also because inaccurate PSF estimates produce heavy tailed errors, since the BID model is non-linear.

The rest of this chapter is organized as follows. In Section 5.2 the Bayesian BID model is presented. In Section 5.3 the variational methodology is applied for inference to the proposed model. In Section 5.4 we present experiments, with artificially blurred images where the ground truth is known and with real astronomical images. In these experiments we compare the proposed methodology with Bayesian methods that use Gaussian priors and TV based methods and the advantages of the proposed methodology are demonstrated. Finally, in Section 5.5 we provide conclusions and directions for future work.

## 5.2 BID Model

We assume that the observed image $g(\boldsymbol{x})$ is given by convolving an unknown image $f(\boldsymbol{x})$ with an unknown PSF $h(\boldsymbol{x})$. To account for errors, additive, independent, identically distributed noise $n(\boldsymbol{x})$ is also assumed. This model is written as

$$g(\boldsymbol{x}) = f(\boldsymbol{x}) * h(\boldsymbol{x}) + n(\boldsymbol{x}), \tag{5.1}$$

where $\boldsymbol{x} = (x_1, x_2) \in \Omega_I$, $\Omega_I \subset \mathbb{R}^2$ is the support of the image and $*$ denotes two-dimensional circular convolution. Equivalently, this can be written in vector form as

$$\boldsymbol{g} = \boldsymbol{f} * \boldsymbol{h} + \boldsymbol{n}, \tag{5.2}$$

where $\boldsymbol{g}$, $\boldsymbol{f}$, $\boldsymbol{h}$ and $\boldsymbol{n}$ are $M \times 1$ lexicographically ordered vectors ($M$ is the number of pixels) of the intensities of the degraded image, observed image, PSF and additive noise respectively. Here, we introduce the $M \times M$ block-circulant matrices $\boldsymbol{F}$ and $\boldsymbol{H}$ that implement two-dimensional convolution with the vectors $\boldsymbol{f}$ and $\boldsymbol{h}$ respectively, so that $\boldsymbol{F}\boldsymbol{h} = \boldsymbol{H}\boldsymbol{f} = \boldsymbol{f} * \boldsymbol{h}$. Then, the BID model in (5.2) can be written as

$$\boldsymbol{g} = \boldsymbol{F}\boldsymbol{h} + \boldsymbol{n} = \boldsymbol{H}\boldsymbol{f} + \boldsymbol{n}. \tag{5.3}$$

The blind image deconvolution problem is difficult because there are too many unknown parameters that have to be estimated. More specifically, the number of unknown parameters $\boldsymbol{h}$ and $\boldsymbol{f}$ is larger than the number of observations $\boldsymbol{g}$, and thus reliable estima-

tion of these parameters can only be achieved by exploiting prior knowledge of the characteristics of the unknown quantities. Following the Bayesian framework, the unknown parameters are treated as hidden random variables and prior knowledge is expressed by assuming that they have been sampled from specific prior distributions.

### 5.2.1 PSF kernel model

We model the PSF as a kernel-based linear model:

$$h(\boldsymbol{x}) = \sum_{i=1}^{M} w_i \phi_i(\boldsymbol{x}), \tag{5.4}$$

where $\phi_i(\boldsymbol{x}) = R(\boldsymbol{x}, \boldsymbol{x}_i)$ is a kernel function centered at $\boldsymbol{x}_i = (x_{i1}, x_{i2}) \in \Omega_I$ and $w_i \in \mathbb{R}$. We denote as $\boldsymbol{h} = (h(\boldsymbol{x}_1), \ldots, h(\boldsymbol{x}_M))^T$ the vector of values of the PSF $h(\boldsymbol{x})$ at each $\boldsymbol{x}_i$ and with $\boldsymbol{\phi}_i = (\phi_i(\boldsymbol{x}_1), \ldots, \phi_i(\boldsymbol{x}_M))^T$ the corresponding basis vector for $\phi_i(\boldsymbol{x})$. Then the PSF vector $\boldsymbol{h}$ can be written as

$$\boldsymbol{h} = \sum_{i=1}^{M} w_i \boldsymbol{\phi}_i. \tag{5.5}$$

We further assume that the kernel is invariant to translations, i.e. $R(\boldsymbol{x}, \boldsymbol{x}_i) = R(\boldsymbol{x} - \boldsymbol{x}_i)$, thus (5.5) can be further written as

$$\boldsymbol{h} = \boldsymbol{\phi} * \boldsymbol{w} = \boldsymbol{\Phi}\boldsymbol{w} = \boldsymbol{W}\boldsymbol{\phi}, \tag{5.6}$$

where $\boldsymbol{w} = (w_1, \ldots, w_M)^T$ are the weights of the linear combination and $\boldsymbol{\Phi}$, $\boldsymbol{W}$ are $M \times M$ block-circulant matrices that implement two-dimensional convolution with $\boldsymbol{\phi} = \boldsymbol{\phi_1}$ and $\boldsymbol{w}$ respectively, so that $\boldsymbol{\Phi}\boldsymbol{w} = \boldsymbol{W}\boldsymbol{\phi} = \boldsymbol{w} * \boldsymbol{\phi}$. Thus, the BID data generation model (5.2) can be written as

$$\boldsymbol{g} = \boldsymbol{F}\boldsymbol{\Phi}\boldsymbol{w} + \boldsymbol{n} = \boldsymbol{\Phi}\boldsymbol{W}\boldsymbol{f} + \boldsymbol{n}. \tag{5.7}$$

Here, we consider Gaussian kernel function of the form $R(\boldsymbol{x}, \boldsymbol{x}_0) = \exp[-\frac{1}{2\sigma_\phi^2}\|\boldsymbol{x} - \boldsymbol{x}_0\|^2]$ (RBF kernels), which produces smooth estimates of the PSF. However, any other type of kernel could be used as well. It is even possible to model the PSF using a multikernel RVM that considers many different types of kernels simultaneously, at a small additional computational cost, as described in the previous chapter.

### 5.2.2 PSF sparseness

A hierarchical prior that enforces sparsity is imposed on the weights $\boldsymbol{w}$ [Tipping, 2001]:

$$p(\boldsymbol{w}|\boldsymbol{\alpha}) = \mathrm{N}(\boldsymbol{w}|0, \boldsymbol{A}^{-1}), \tag{5.8}$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_M)^T$, $\boldsymbol{A} = \mathrm{diag}\{\boldsymbol{\alpha}\}$. Each weight is assigned a separate local precision parameter $\alpha_i$, which is treated as a random variable that follows a Gamma distribu-

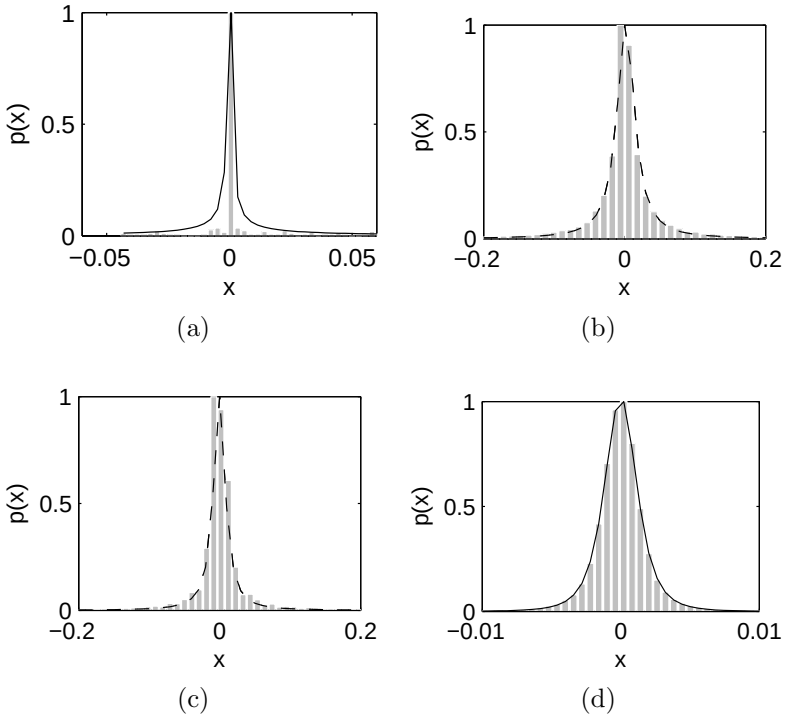



Figure 5.3: Histograms of (a) estimated weights of the PSF sparse linear model, assuming the true PSF is known, (b) horizontal and (c) vertical local differences of the "Lenna" image and (d) model errors of an image restoration method using incorrect PSF estimation. Solid lines show fits by the Student's t pdf with parameters (a) $\mu = 2.51 \times 10^{-34}, \lambda = 9.05 \times 10^{37}, \nu = 0.043$ (b) $\mu = 1.7 \times 10^{-3}, \lambda = 4.59 \times 10^{3}, \nu = 1.09$ (c) $\mu = -4 \times 10^{-4}, \lambda = 1.03 \times 10^{4}, \nu = 1.132$ and (d) $\mu = 2.39 \times 10^{-6}, \lambda = 6.68 \times 10^{5}, \nu = 3.12$.

tion:

$$p(\boldsymbol{\alpha}) = \prod_{i=1}^{M} \text{Gamma}(\alpha_i | a^\alpha, b^\alpha). \tag{5.9}$$

This hierarchical prior is equivalent to a Student's t pdf. In order to demonstrate why sparse estimations of the PSF weights are appropriate, Fig. 5.3(a) shows a histogram of the PSF weights. This histogram was obtained using a $7 \times 7$ uniform square-shaped PSF function. Estimates of the PSF weights were obtained using the sparse Bayesian linear model of Chapter 3. It is apparent that the pdf of the weights is very heavy tailed and that there are only few non-zero weights. For this reason, we set $a^\alpha = b^\alpha = 0$ that define a very heavy tailed, uninformative Student's-t distribution. It is interesting that the hidden variables $\alpha_i$ of this Student's-t distribution provide an estimate of the support of the PSF. Specifically, the local precision $\alpha_i$ that corresponds to kernels outside the support of the PSF obtains very large values, therefore those kernels are pruned by setting $w_i = 0$. This is demonstrated in Fig. 5.4(a) where we show the estimated local variances for a BID problem with a $7 \times 7$ uniform PSF. Notice that outside a limited area that captures the support of this PSF these variances are zero.

### 5.2.3 Image model

The image prior that we use is based on $K$ filtered versions of the image: $\boldsymbol{\epsilon}^k = \boldsymbol{Q}^k \boldsymbol{f}$, where $\boldsymbol{Q}^k$ are $M \times M$ convolutional operators of the filters ($k = 1, \ldots, K$). Specifically, we use horizontal and vertical first order local differences, by defining $K = 2$, $\boldsymbol{Q}^1$ and $\boldsymbol{Q}^2$ so that:

$$\epsilon^1(x, y) = f(x, y) - f(x + 1, y), \tag{5.10}$$

$$\epsilon^2(x, y) = f(x, y) - f(x, y + 1). \tag{5.11}$$

Without any changes in the method, we could also use other convolutional operators $\boldsymbol{Q}^k$ [Chantas et al., 2007]. In practice, we join all operators $\boldsymbol{Q}^k$ in the $KM \times M$ operator $\tilde{\boldsymbol{Q}} = (\boldsymbol{Q}^{1^T}, \ldots, \boldsymbol{Q}^{K^T})^T$ that produces the $KM \times 1$ vector $\tilde{\boldsymbol{\epsilon}} = (\boldsymbol{\epsilon}^{1^T}, \ldots, \boldsymbol{\epsilon}^{K^T})^T$:

$$\tilde{\boldsymbol{\epsilon}} = \tilde{\boldsymbol{Q}} \boldsymbol{f} = ((\boldsymbol{Q}^1 \boldsymbol{f})^T, \ldots, (\boldsymbol{Q}^K \boldsymbol{f})^T)^T. \tag{5.12}$$

We assume that $\epsilon_i^k$ is Gaussian distributed with distinct precision $\gamma_i^k$:

$$p(\epsilon_i^k | \gamma_i^k) = \mathrm{N}(\epsilon_i^k | 0, (\gamma_i^k)^{-1}). \tag{5.13}$$

Assuming the $\epsilon_i^k$ independent with respect to $i$, induces a prior for the image, which is given by

$$p_k(\boldsymbol{f} | \boldsymbol{\gamma}^k) = \mathrm{N}(\boldsymbol{f} | 0, (\boldsymbol{Q}^{k^T} \boldsymbol{\Gamma}^k \boldsymbol{Q}^k)^{-1}), \tag{5.14}$$

with $\boldsymbol{\gamma}^k = (\gamma_1^k \ldots \gamma_M^k)^T$ and $\boldsymbol{\Gamma}^k = \mathrm{diag}\{\boldsymbol{\gamma}^k\}$. In order to combine the information captured by each prior $p_k$, we define a composite prior, which is the product of them [Welling et al., 2003]:

$$p(\boldsymbol{f} | \tilde{\boldsymbol{\gamma}}) = \frac{1}{Z} \prod_{k=1}^{K} p_k(\boldsymbol{f} | \boldsymbol{\gamma}^k) = \mathrm{N}(\boldsymbol{f} | 0, (\tilde{\boldsymbol{Q}}^T \tilde{\boldsymbol{\Gamma}} \tilde{\boldsymbol{Q}})^{-1}), \tag{5.15}$$

with $\tilde{\boldsymbol{\gamma}} = (\boldsymbol{\gamma}^{1^T}, \ldots, \boldsymbol{\gamma}^{K^T})^T$ and $\tilde{\boldsymbol{\Gamma}} = \mathrm{diag}\{\tilde{\boldsymbol{\gamma}}\}$. Unfortunately, it is not possible to analytically compute the determinant $|\tilde{\boldsymbol{Q}}^T \tilde{\boldsymbol{\Gamma}} \tilde{\boldsymbol{Q}}|$ that is required to estimate the normalization constant $Z$ in (5.15), since $\tilde{\boldsymbol{Q}}$ is not square. Instead we approximate it as $|\tilde{\boldsymbol{Q}}^T \tilde{\boldsymbol{\Gamma}} \tilde{\boldsymbol{Q}}| \approx |\tilde{\boldsymbol{\Gamma}}| |\tilde{\boldsymbol{Q}}^T \tilde{\boldsymbol{Q}}|$, giving:

$$p(\boldsymbol{f} | \tilde{\boldsymbol{\gamma}}) \propto \prod_{k=1}^{K} \prod_{i=1}^{M} (\gamma_i^k)^{\frac{1}{2}} \exp\left[ -\frac{1}{2} \boldsymbol{f}^T \tilde{\boldsymbol{Q}}^T \tilde{\boldsymbol{\Gamma}} \tilde{\boldsymbol{Q}} \boldsymbol{f} \right]. \tag{5.16}$$

Notice, that the approximation only affects the normalizing constant of the pdf. Therefore, this is an *improper* pdf, whose integral is not necessarily unity. Improper pdfs have been used in many other Bayesian methods [Bernardo and Smith, 1994]. The local precision parameters $\gamma_i^k$ are assumed to be independent identically distributed, Gamma random

(a) (b) (c)

Figure 5.4: Example of the estimated local variances (a) $\alpha^{-1}$ of the PSF weights for a uniform $7 \times 7$ square-shaped PSF, (b) and (c) $(\gamma^1)^{-1}$ and $(\gamma^2)^{-1}$ of the image model residuals.

variables:

$$p(\tilde{\boldsymbol{\gamma}}) = \prod_{k=1}^{K} \prod_{i=1}^{M} \text{Gamma}(\gamma_i^k | a^\gamma, b^\gamma). \tag{5.17}$$

Thus, the prior on the first order local differences $\boldsymbol{\epsilon}^k$ is equivalent to a Student's t pdf.

### 5.2.4 Noise model

The noise $\boldsymbol{n}$ of the BID model (5.3) is assumed to be zero mean Gaussian distributed, given by:

$$p(\boldsymbol{n}|\boldsymbol{\beta}) = \prod_{i=1}^{M} \text{N}(n_i|0, \beta_i^{-1}) = \text{N}(\boldsymbol{n}|0, \boldsymbol{B}^{-1}), \tag{5.18}$$

with $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_M)$ and $\boldsymbol{B} = \text{diag}\{\boldsymbol{\beta}\}$. The local precision parameters $\beta_i$ are also assumed to be random variables with a Gamma prior:

$$p(\boldsymbol{\beta}) = \prod_{i=1}^{M} \text{Gamma}(\beta_i | a^\beta, b^\beta). \tag{5.19}$$

This two-level hierarchical prior for noise is equivalent to a Student's t pdf.

## 5.3 Variational Bayesian Inference

The observed variables of the proposed model are $\boldsymbol{D} = \{\boldsymbol{g}\}$, the hidden variables are $\boldsymbol{\theta} = \{\boldsymbol{w}, \boldsymbol{f}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}\}$ and the parameters of the model are $\boldsymbol{\xi} = \{a^\alpha, b^\alpha, a^\beta, b^\beta, a^\gamma, b^\gamma\}$. The dependencies among the random variables that define the proposed Bayesian model are shown in the graphical model of Fig. 5.5.

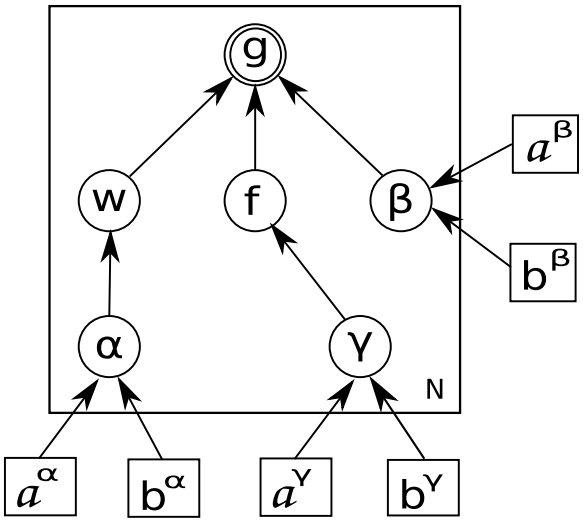



Figure 5.5: Graphical model that describes the dependencies between the random variables of the proposed model. Circular nodes represent random variables, while square nodes represent parameters of the model. The observed variables are represented by double circled nodes.

Because the BID model is non-linear, the posterior distribution of the parameters $p(\boldsymbol{\theta}|\boldsymbol{D})$ cannot be computed. Thus, we can not apply exact inference methods, such as maximum likelihood via the EM algorithm. Instead, we resort to approximate inference and specifically to the variational Bayesian methodology described in Chapter 2.

### 5.3.1 Approximate Posterior Distributions

Using the mean field approximation (2.23), the posterior distribution of the parameters is given by (2.24). Because we have used conjugate priors, the approximate posteriors have the same form as the priors. Specifically, the approximate posterior distributions of the PSF weights $\boldsymbol{w}$ and the image $\boldsymbol{f}$ are Gaussian and the distributions of the precision parameters $\boldsymbol{\alpha}, \boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are Gamma:

$$q(\boldsymbol{w}) = \mathrm{N}(\boldsymbol{w}|\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w), \tag{5.20}$$

$$q(\boldsymbol{f}) = \mathrm{N}(\boldsymbol{f}|\boldsymbol{\mu}_f, \boldsymbol{\Sigma}_f), \tag{5.21}$$

$$q(\boldsymbol{\alpha}) = \prod_{i=1}^{M} \mathrm{Gamma}(\alpha_i|\tilde{a}^\alpha, \tilde{b}_i^\alpha), \tag{5.22}$$

$$q(\boldsymbol{\beta}) = \prod_{i=1}^{M} \mathrm{Gamma}(\beta_i|\tilde{a}^\beta, \tilde{b}_i^\beta), \tag{5.23}$$

$$q(\boldsymbol{\gamma}) = \prod_{k=1}^{K} \prod_{i=1}^{M} \mathrm{Gamma}(\gamma_i^k|\tilde{a}^\gamma, \tilde{b}_i^{\gamma^k}), \tag{5.24}$$

where

$$\boldsymbol{\mu}_w = \boldsymbol{\Sigma}_w \boldsymbol{\Phi}^T \langle \boldsymbol{F}^T \rangle \langle \boldsymbol{B} \rangle \boldsymbol{g}, \tag{5.25}$$

$$\boldsymbol{\Sigma}_w = \left( \boldsymbol{\Phi}^T \langle \boldsymbol{F}^T \boldsymbol{B} \boldsymbol{F} \rangle \boldsymbol{\Phi} + \langle \boldsymbol{A} \rangle \right)^{-1}, \tag{5.26}$$

$$\boldsymbol{\mu}_f = \boldsymbol{\Sigma}_f \boldsymbol{\Phi}^T \langle \boldsymbol{W}^T \rangle \langle \boldsymbol{B} \rangle \boldsymbol{g}, \tag{5.27}$$

$$\boldsymbol{\Sigma}_f = \left( \boldsymbol{\Phi}^T \langle \boldsymbol{W}^T \boldsymbol{B} \boldsymbol{W} \rangle \boldsymbol{\Phi} + \tilde{\boldsymbol{Q}}^T \langle \tilde{\boldsymbol{\Gamma}} \rangle \tilde{\boldsymbol{Q}} \right)^{-1}, \tag{5.28}$$

$$\tilde{a}^\alpha = a^\alpha + 1/2, \tag{5.29}$$

$$\tilde{b}_i^\alpha = b^\alpha + \frac{1}{2} \langle w_i^2 \rangle, \tag{5.30}$$

$$\tilde{a}^\beta = a^\beta + M/2, \tag{5.31}$$

$$\tilde{b}_i^\beta = b^\beta + \frac{1}{2} \langle \boldsymbol{n} \boldsymbol{n}^T \rangle_{ii}, \tag{5.32}$$

$$\tilde{a}^\gamma = a^\gamma + 1/2, \tag{5.33}$$

$$\tilde{b}_i^{\gamma^k} = b^\gamma + \frac{1}{2} \left( \boldsymbol{Q}^k \langle \boldsymbol{f} \boldsymbol{f}^T \rangle \boldsymbol{Q}^k \right)_{ii}. \tag{5.34}$$

The required expected values can be computed as:

$$\langle \boldsymbol{w} \rangle = \boldsymbol{\mu}_w, \tag{5.35}$$

$$\langle w_i^2 \rangle = \mu_{w_i}^2 + \Sigma_{w_{ii}}, \tag{5.36}$$

$$\langle \boldsymbol{f} \rangle = \boldsymbol{\mu}_f, \tag{5.37}$$

$$\langle \boldsymbol{f} \boldsymbol{f}^T \rangle = \boldsymbol{\mu}_f \boldsymbol{\mu}_f^T + \boldsymbol{\Sigma}_f, \tag{5.38}$$

$$\langle \alpha_i \rangle = \tilde{a}^\alpha / \tilde{b}_i^\alpha, \tag{5.39}$$

$$\langle \beta_i \rangle = \tilde{a}^\beta / \tilde{b}_i^\beta, \tag{5.40}$$

$$\langle \gamma_i^k \rangle = \tilde{a}^\gamma / \tilde{b}_i^{\gamma_k}, \tag{5.41}$$

$$\langle \boldsymbol{n} \boldsymbol{n}^T \rangle = \boldsymbol{g} \boldsymbol{g}^T - 2 \boldsymbol{\Phi} \langle \boldsymbol{F} \boldsymbol{w} \rangle \boldsymbol{g}^T + \boldsymbol{\Phi} \langle \boldsymbol{F} \boldsymbol{w} \boldsymbol{w}^T \boldsymbol{F}^T \rangle \boldsymbol{\Phi}^T. \tag{5.42}$$

The approximate posterior distributions of (5.20) to (5.24) can be computed as follows. In order to find the posterior distribution of the weights $q(\boldsymbol{w})$ we start from (2.24) and keeping only the terms that depend on $\boldsymbol{w}$ we have:

$$\begin{aligned} \ln q(\boldsymbol{w}) &= \langle \ln p(\boldsymbol{g} | \boldsymbol{\beta}, \boldsymbol{w}, \boldsymbol{f}) p(\boldsymbol{w} | \boldsymbol{\alpha}) \rangle_{q(\boldsymbol{f}) q(\boldsymbol{\alpha}) q(\boldsymbol{\beta}) q(\boldsymbol{\gamma})} \\ &= \langle \ln p(\boldsymbol{g} | \boldsymbol{\beta}, \boldsymbol{w}, \boldsymbol{f}) + \ln p(\boldsymbol{w} | \boldsymbol{\alpha}) \rangle_{q(\boldsymbol{f}) q(\boldsymbol{\alpha}) q(\boldsymbol{\beta})} \\ &= \left\langle -\frac{1}{2} \boldsymbol{n}^T \boldsymbol{B} \boldsymbol{n} - \frac{1}{2} \sum_{i=1}^{M} \alpha_i w_i^2 \right\rangle_{q(\boldsymbol{f}) q(\boldsymbol{\alpha}) q(\boldsymbol{\beta})} + const. \end{aligned}$$

Then, because $\boldsymbol{n} = \boldsymbol{g} - \boldsymbol{\Phi w}$ and $\boldsymbol{B}$ is diagonal and therefore symmetric, we have:

$$\begin{aligned}\ln q(\boldsymbol{w}) &= -\frac{1}{2}[\boldsymbol{g}^T \langle \boldsymbol{B} \rangle \boldsymbol{g} - 2\boldsymbol{g}^T \langle \boldsymbol{BF} \rangle \boldsymbol{\Phi w} + \boldsymbol{w}^T \boldsymbol{\Phi}^T \left\langle \boldsymbol{F}^T \boldsymbol{BF} \right\rangle \boldsymbol{\Phi w}] - \frac{1}{2} \sum_{i=1}^{M} \langle \alpha_i \rangle w_i^2 \\ &= -\frac{1}{2} \boldsymbol{w}^T \left( \boldsymbol{\Phi}^T \left\langle \boldsymbol{F}^T \boldsymbol{BF} \right\rangle \boldsymbol{\Phi} + \langle \boldsymbol{A} \rangle \right) \boldsymbol{w} - \boldsymbol{w}^T \boldsymbol{\Phi}^T \left\langle \boldsymbol{F}^T \boldsymbol{B} \right\rangle \boldsymbol{g} + const.\end{aligned}$$

We can easily see that this is the exponent of a Gaussian distribution, therefore $q(\boldsymbol{w})$ is a Gaussian distribution given by (5.20). Similarly, we can obtain the posterior $q(\boldsymbol{f})$ which is also a Gaussian distribution given by (5.21).

The posterior $q(\boldsymbol{\alpha})$ is similarly obtained by computing the terms of $\ln q(\boldsymbol{\alpha})$ that depend on $\boldsymbol{\alpha}$:

$$\begin{aligned}\ln q(\boldsymbol{\alpha}) &= \langle \ln p(\boldsymbol{w}|\boldsymbol{\alpha}) p(\boldsymbol{\alpha}) \rangle_{q(\boldsymbol{f}) q(\boldsymbol{w}) q(\boldsymbol{\beta}) q(\boldsymbol{\gamma})} \\ &= \frac{1}{2} \sum_{i=1}^{M} \ln \alpha_i - \sum_{i=1}^{M} \alpha_i \left\langle w_i^2 \right\rangle + (a^\alpha - 1) \sum_{i=1}^{M} \ln \alpha_i - b^\alpha \sum_{i=1}^{M} \alpha_i \\ &= \left( a^\alpha - \frac{1}{2} \right) \sum_{i=1}^{M} \ln \alpha_i - \sum_{i=1}^{M} \left( \frac{1}{2} \left\langle w_i^2 \right\rangle + b^\alpha \right) + const.\end{aligned}$$

This is the exponent of a Gamma distribution, and therefore $q(\boldsymbol{\alpha})$ is a Gamma distribution given by (5.22). The posterior distributions $q(\boldsymbol{\beta})$ and $q(\boldsymbol{\gamma})$ are also Gamma distributions given by (5.23) and (5.24) and their computation is very similar.

### 5.3.2 Parameter Estimation

The parameters $a^\beta$, $b^\beta$ and $a^\gamma$, $b^\gamma$ of the noise and image Gamma hyperpriors can be estimated by optimizing the variational bound $F$ (2.13), which is given by:

$$\begin{aligned}F(\theta) &= \left\langle \ln \frac{p(\boldsymbol{g}, \boldsymbol{f}, \boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma})}{q(\boldsymbol{w}, \boldsymbol{f}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma})} \right\rangle_{q(\boldsymbol{w}, \boldsymbol{f}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma})} \\ &= \langle \ln p(\boldsymbol{g}|\boldsymbol{w}, \boldsymbol{\beta}, \boldsymbol{f}) \rangle + \langle \ln p(\boldsymbol{w}|\boldsymbol{\alpha}) \rangle + \langle \ln p(\boldsymbol{f}|\boldsymbol{\gamma}) \rangle \\ &+ \langle \ln p(\boldsymbol{\alpha}) \rangle + \langle \ln p(\boldsymbol{\beta}) \rangle + \langle \ln p(\boldsymbol{\gamma}) \rangle - \langle \ln q(\boldsymbol{w}) \rangle \\ &- \langle \ln q(\boldsymbol{f}) \rangle - \langle \ln q(\boldsymbol{\alpha}) \rangle - \langle \ln q(\boldsymbol{\beta}) \rangle - \langle \ln q(\boldsymbol{\gamma}) \rangle\end{aligned}$$

and the required expected values can be computed as:

$$\langle \ln p(\boldsymbol{g}|\boldsymbol{w}, \boldsymbol{\beta}, \boldsymbol{f}) \rangle = -\frac{M}{2} \ln(2\pi) + \frac{1}{2} \sum_{i=1}^{M} \langle \ln \beta_i \rangle - \frac{1}{2} \langle (\boldsymbol{g} - \boldsymbol{F\Phi w})^T \boldsymbol{B} (\boldsymbol{g} - \boldsymbol{F\Phi w}) \rangle$$

$$\langle \ln p(\boldsymbol{w}|\boldsymbol{\alpha}) \rangle = -\frac{M}{2} \ln(2\pi) + \frac{1}{2} \sum_{i=1}^{M} \langle \ln \alpha_i \rangle - \frac{1}{2} \sum_{i=1}^{M} \langle \alpha_i \rangle \left\langle w_i^2 \right\rangle,$$

$$
\begin{aligned}
\langle \ln p(\boldsymbol{f}|\boldsymbol{\gamma})\rangle &= -\frac{M}{2}\ln(2\pi) + \frac{1}{2}\sum_{k=1}^{K}\sum_{i=1}^{M}\left\langle \ln \gamma_i^k \right\rangle - \frac{1}{2}\sum_{k=1}^{K}\sum_{i=1}^{M}\left\langle \gamma_i^k \right\rangle (\boldsymbol{Q}^k \boldsymbol{f})_i^2, \\
\langle \ln p(\boldsymbol{\alpha})\rangle &= Ma^\alpha \ln b^\alpha + (a^\alpha - 1)\sum_{i=1}^{M}\langle \ln \alpha_i\rangle - b^\alpha \sum_{i=1}^{M}\langle \alpha_i\rangle - M\ln\Gamma(a^\alpha), \\
\langle \ln p(\boldsymbol{\beta})\rangle &= Ma^\beta \ln b^\beta + (a^\beta - 1)\sum_{i=1}^{M}\langle \ln \beta_i\rangle - b^\beta \sum_{i=1}^{M}\langle \beta_i\rangle - M\ln\Gamma(a^\beta), \\
\langle \ln p(\boldsymbol{\gamma})\rangle &= MKa^\gamma \ln b^\gamma + (a^\gamma - 1)\sum_{k=1}^{K}\sum_{i=1}^{M}\left\langle \ln \gamma_i^k \right\rangle - b^\gamma \sum_{k=1}^{K}\sum_{i=1}^{M}\left\langle \gamma_i^k \right\rangle - MK\ln\Gamma(a^\gamma), \\
\langle \ln q(\boldsymbol{w})\rangle &= -\frac{M}{2}(\ln(2\pi)+1) - \frac{1}{2}\ln|\boldsymbol{\Sigma}_w|, \\
\langle \ln q(\boldsymbol{f})\rangle &= -\frac{M}{2}(\ln(2\pi)+1) - \frac{1}{2}\ln|\boldsymbol{\Sigma}_f|, \\
\langle \ln q(\boldsymbol{\alpha})\rangle &= \sum_{i=1}^{M}[\tilde{a}^\alpha \ln \tilde{b}_i^\alpha + (\tilde{a}^\alpha - 1)\langle \ln \alpha_i\rangle - \tilde{b}_i^\alpha \langle \alpha_i\rangle - \ln\Gamma(\tilde{a}^\alpha)], \\
\langle \ln q(\boldsymbol{\beta})\rangle &= \sum_{i=1}^{M}[\tilde{a}^\beta \ln \tilde{b}_i^\beta + (\tilde{a}^\beta - 1)\langle \ln \beta_i\rangle - \tilde{b}_i^\beta \langle \beta_i\rangle - \ln\Gamma(\tilde{a}^\beta)], \\
\langle \ln q(\boldsymbol{\gamma})\rangle &= \sum_{k=1}^{K}\sum_{i=1}^{M}[\tilde{a}^\gamma \ln \tilde{b}_i^{\gamma^k} + (\tilde{a}^\gamma - 1)\left\langle \ln \gamma_i^k \right\rangle - \tilde{b}_i^{\gamma^k}\left\langle \gamma_i^k \right\rangle - \ln\Gamma(\tilde{a}^\gamma)].
\end{aligned}
$$

The derivatives of $F$ with respect to the above parameters are:

$$
\begin{aligned}
\frac{\partial F}{\partial a^\beta} &= M\ln b^\beta - M\psi(a^\beta) + \sum_{i=1}^{M}\langle \ln \beta_i\rangle, && (5.43)\\
\frac{\partial F}{\partial b^\beta} &= M\frac{a^\beta}{b^\beta} - \sum_{i=1}^{M}\langle \beta_i\rangle, && (5.44)\\
\frac{\partial F}{\partial a^\gamma} &= MK\ln b^\gamma - MK\psi(a^\gamma) + \sum_{k=1}^{K}\sum_{i=1}^{M}\left\langle \ln \gamma_i^k \right\rangle, && (5.45)\\
\frac{\partial F}{\partial b^\gamma} &= MK\frac{a^\gamma}{b^\gamma} - \sum_{k=1}^{K}\sum_{i=1}^{M}\left\langle \gamma_i^k \right\rangle, && (5.46)
\end{aligned}
$$

where $\psi(x)$ is the digamma function given by $\psi(x) = \frac{d\ln\Gamma(z)}{dz} = \frac{\Gamma'(z)}{\Gamma(z)}$ and $\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}\,\mathrm{d}t$. We can obtain updates for these parameters by setting the above derivatives to zero. This cannot be done analytically for the parameters $a^\beta$ and $a^\gamma$, thus we find a numerical solution using a combination of bisection, secant, and inverse quadratic interpolation methods, as implemented by matlab's fzero function.

### 5.3.3 Computational issues

The computations in equations (5.25) - (5.42) involve matrix operations, whose dimension is $M \times M$, where $M$ is the number of pixels in the image. Unfortunately, computation of $\boldsymbol{\Sigma}_f$ and $\boldsymbol{\Sigma}_w$ involves inversion of matrices that contain both diagonal and circulant matrices and cannot be performed explicitly for large $M$. However, diagonal and circulant matrices are easy to invert. For this reason, we approximate $\boldsymbol{\Sigma}_w$ (5.26) with a diagonal matrix and $\boldsymbol{\Sigma}_f$ (5.28) with a circulant matrix, as:

$$\bar{\boldsymbol{\Sigma}}_w = \left(\text{diag}\{\boldsymbol{\Phi}^T \langle \boldsymbol{F}^T \boldsymbol{B} \boldsymbol{F} \rangle \Phi\} + \langle \boldsymbol{A} \rangle \right)^{-1}, \tag{5.47}$$

$$\bar{\boldsymbol{\Sigma}}_f = \left( \langle \bar{\beta} \rangle \boldsymbol{\Phi}^T \langle \boldsymbol{W}^T \boldsymbol{W} \rangle \boldsymbol{\Phi} + \langle \bar{\gamma} \rangle \tilde{\boldsymbol{Q}}^T \tilde{\boldsymbol{Q}} \right)^{-1}, \tag{5.48}$$

with $\bar{\gamma} = \frac{1}{MK} \sum_{k=1}^{M} \sum_{i=1}^{M} \gamma_i^k$, $\bar{\beta} = \frac{1}{M} \sum_{i=1}^{M} \beta_i$ and

$$\langle \boldsymbol{W}^T \boldsymbol{W} \rangle = \langle \boldsymbol{W}^T \rangle \langle \boldsymbol{W} \rangle + I \sum_{i=1}^{M} \langle \bar{\Sigma}_{w_{ii}} \rangle, \tag{5.49}$$

$$\langle \boldsymbol{F}^T \boldsymbol{B} \boldsymbol{F} \rangle = \langle \boldsymbol{F}^T \rangle \langle \boldsymbol{B} \rangle \langle \boldsymbol{F} \rangle + \bar{\boldsymbol{\Sigma}}_f \sum_{i=1}^{M} \langle \beta_i \rangle. \tag{5.50}$$

The diagonal approximation for matrix $\boldsymbol{\Sigma}_w$ is justified because parameters $\alpha_i$ that appear in the diagonal were found to dominate in (5.26). On the other hand, $\boldsymbol{\Sigma}_f$ is approximated with a circulant matrix because both the parameters $\beta_i$ and $\gamma_i^k$ obtain values in the same range. The above approximations are used for computation of $\tilde{b}_i^{\alpha}$, $\tilde{b}_i^{\beta}$, and $\tilde{b}_i^{\gamma^k}$, in (5.30), (5.32) and (5.34) respectively, where the elements of the matrices $\boldsymbol{\Sigma}_w$ and $\boldsymbol{\Sigma}_f$ appear directly. Furthermore, they are used for computing the expected value $\langle \boldsymbol{F} \boldsymbol{w} \boldsymbol{w}^T \boldsymbol{F}^T \rangle$ that appears in (5.42) as:

$$\langle \boldsymbol{F} \boldsymbol{w} \boldsymbol{w}^T \boldsymbol{F}^T \rangle = \langle \boldsymbol{F} \rangle \langle \boldsymbol{w} \boldsymbol{w}^T \rangle \langle \boldsymbol{F}^T \rangle + \bar{\boldsymbol{\Sigma}}_f \sum_{i,j} \langle \boldsymbol{w} \boldsymbol{w}^T \rangle_{ij}. \tag{5.51}$$

For the posterior image and weight means $\boldsymbol{\mu}_f$ and $\boldsymbol{\mu}_w$, we do not use the above approximations, since we can exactly obtain them by solving the following linear systems:

$$\boldsymbol{\Sigma}_f^{-1} \boldsymbol{\mu}_f = \boldsymbol{\Phi}^T \langle \boldsymbol{W} \rangle^T \langle \boldsymbol{B} \rangle \boldsymbol{g}, \tag{5.52}$$

$$\boldsymbol{\Sigma}_w^{-1} \boldsymbol{\mu}_w = \boldsymbol{\Phi}^T \langle \boldsymbol{F} \rangle^T \langle \boldsymbol{B} \rangle \boldsymbol{g}. \tag{5.53}$$

These linear systems are solved iteratively with the conjugate gradient method, using the approximation matrices $\bar{\boldsymbol{\Sigma}}_f$ and $\bar{\boldsymbol{\Sigma}}_w$ as preconditioners. In these iterations, products of circulant matrices are efficiently computed in the DFT domain, while products of diagonal matrices in the spatial domain. Specifically, each conjugate gradient iteration requires $O(M \log M)$ iterations. Theoretically, an exact solution of the linear system is obtained after $C = N$ iterations, however, we typically obtain a good approximation after

only few iterations, e.g. $C = 20$. The overall computation cost is $O(CM \log M)$.

### 5.3.4 Variational Optimization Algorithm

Each iteration of the optimization algorithm proceeds as follows. First we compute the parameters of the approximate posterior probabilities, as given in (5.25) - (5.34) and then we compute the expected values using (5.35) - (5.42). Finally, we may update the parameters of the noise and image prior distributions, using equations (5.43) - (5.46). The means of the posteriors $q(\boldsymbol{w})$ and $q(\boldsymbol{f})$ are used to obtain estimates of the PSF $\hat{\boldsymbol{h}}$ and the image $\hat{\boldsymbol{f}}$: $\hat{\boldsymbol{h}} = \boldsymbol{\Phi}\boldsymbol{\mu}_w$ and $\hat{\boldsymbol{f}} = \boldsymbol{\mu}_f$.

## 5.4 Numerical Experiments

Several numerical experiments have been carried out both with artificially generated observations where the ground truth is known and with real observations in order to demonstrate the properties of the proposed method. We compare the proposed method with previous Bayesian BID formulations based on Gaussian PSF and image models [Likas and Galatsanos, 2004], with the TV-based blind deconvolution method in [Chan and Wong, 1998] and another recent variational Bayesian method in [Molina et al., 2006].

Hereafter, we will refer to the proposed method as the StStSt method, to imply that three Student's t priors are used to model the PSF weights, the BID model errors and the image local differences. We also considered several simpler versions of this Bayesian model that use Gaussian distributions in place of the Student's t distributions. Specifically, we consider Gaussian distributions for the PSF weights, $p(\boldsymbol{w}) = \mathrm{N}(\boldsymbol{w}|0, \alpha^{-1}\boldsymbol{I})$, the additive noise, $p(\boldsymbol{n}) = \mathrm{N}(\boldsymbol{n}|0, \beta^{-1}\boldsymbol{I})$, and the image local differences, $p(\boldsymbol{f}) = \mathrm{N}(\boldsymbol{f}|0, (\gamma\boldsymbol{Q}^T\boldsymbol{Q})^{-1})$. The names of these simplified versions consist of three parts that express the distributions of the PSF weights, the additive noise and the image local differences. For example, the method that uses Gaussian distribution for the image local variances but Student's t distributions for the PSF weights and noise is denoted as StStG.

The GGG is very similar to the VAR1 method described in [Likas and Galatsanos, 2004], which also assumes that the PSF weights, the imaging model errors and the image local differences are Gaussian. The only difference between VAR1 and GGG is that VAR1 does not use a kernel model for the PSF, i.e. $\boldsymbol{h} = (w_1, \ldots, w_M)^T$. Thus, the VAR1 method is identical to the GGG, when a Gaussian kernel of very small size is used.

In the simplified models GGG, GStG, StGG and StStG, where Gaussian stationary image priors are used, we consider the typical simultaneously autoregressive (SAR) prior that has been used extensively in image restoration [Likas and Galatsanos, 2004, Molina et al., 2006]. This prior assumes a pdf for the image residuals $\epsilon(x, y)$ given by:

$$\epsilon(x, y) = \sum_{(k,l) \in D(x,y)} \left( f(x, y) - f(k, l) \right), \tag{5.54}$$

where $D(x,y)$ is the set of four neighbors of $(x,y)$, given by $D(x,y) = \{(x+1,y),(x-1,y),(x,y-1),(x,y+1)\}$. The Bayesian method in [Molina et al., 2006] uses the SAR prior for both the image and PSF and then uses the variational methodology to achieve inference, similarly to the proposed method.

Furthermore, we provide a detailed comparison with the TV blind deconvolution method [Chan and Wong, 1998]. This method provides estimates of the image and PSF by solving the following minimization problem:

$$\min_{f,h} \frac{1}{2}\|h * f - g\|^2 + \alpha_f TV(f) + \alpha_h TV(h), \tag{5.55}$$

where $TV(x) = \int |\nabla x(z)|\, \mathrm{d}z$ is a total variation regularization term.

### 5.4.1 Experiments with artificially blurred images

In the first experiment, we compared all the methods using artificially degraded images. We generated a degraded image $\boldsymbol{g}$ by blurring the true image $\boldsymbol{f}$ with a known PSF $\boldsymbol{h}$ and then adding Gaussian noise with variance $\sigma^2 = 10^{-6}$. The signal to noise ratio (SNR) of the observed image $\boldsymbol{g}$ is $SNR = 10\log_{10}\frac{\|\boldsymbol{f}\|^2}{M\sigma^2} = 45dB$. In all methods, the initial PSF $\boldsymbol{h_{in}}$ was set to a Gaussian-shaped function with variance $\sigma^2_{h_{in}} = 3$. Since the true image is known, we can measure the quality of a recovered image $\hat{\boldsymbol{f}}$, by computing the improved signal to noise ratio $ISNR_f = 10\log_{10}\frac{\|\boldsymbol{f}-\boldsymbol{g}\|^2}{\|\boldsymbol{f}-\hat{\boldsymbol{f}}\|^2}$ which is a measure of the improvement of the quality of the estimated image with respect to the initial degraded image. We can also measure the quality of a PSF estimation $\hat{\boldsymbol{h}}$, by computing $ISNR_h = 10\log_{10}\frac{\|\boldsymbol{h}-\boldsymbol{h_{in}})\|^2}{\|\boldsymbol{h}-\hat{\boldsymbol{h}}\|^2}$.

The PSF that was used in this experiment was a $7\times 7$ uniform, square-shaped PSF. However, we initialized the PSF as a Gaussian-shaped function with variance $\sigma^2_{h_{in}} = 3$. The kernel function that we used was set to a Gaussian with variance $\sigma^2_\phi = 0.1$, which is flexible enough to model the boundaries of the square. The ISNR values for the image and PSF estimates of all methods are shown in Table 5.1. Furthermore, the degraded image and restored images for some of these methods are shown in Fig. 5.6 along with the restoration in [Chantas et al., 2006], which was obtained by assuming that the PSF is known and a similar in spirit image prior.

Inspection of these results reveals that in general, improvement in the accuracy of the estimated PSF implies improvement in the quality of the recovered image. Furthermore, using a Student's t distribution to model the weights of the kernel model of the PSF gives *significantly better* PSF estimates as compared to using a Gaussian distribution for the same task. This demonstrates beyond any doubt the importance of this selection for the BID problem. The image estimates are also improved when using Student's t distributions for either the image local differences or noise. Finally, the StStSt model seems to produce visually more pleasing restored images with "sharper" edges than either the StGSt and StStG models, even though the $ISNR_f$ might be slightly lower. However, it is well known that $ISNR_f$ does not always capture accurately the human perception of image quality.

Table 5.1: ISNR for image and PSF for the experiments on the degraded lenna image with a uniform, $7 \times 7$ square-shaped PSF.

| Method | $ISNR_f$ | $ISNR_h$ |
|---|---|---|
| GGG | 0.47 | 0.88 |
| GGSt | 0.58 | 0.79 |
| GStG | 0.05 | 1.53 |
| GStSt | 1.11 | 1.64 |
| StGG | 2.17 | 6.69 |
| StGSt | 5.87 | 8.12 |
| StStG | 5.57 | 10.91 |
| StStSt | 5.29 | 9.44 |
| Method in [Chan and Wong, 1998] | 3.13 | 5.64 |
| Method in [Molina et al., 2006] | 0.54 | 2.44 |
| Known PSF in [Chantas et al., 2006] | 8.63 | — |

### 5.4.2 Comparison with other BID methods

In this subsection, we describe another experiment, where we compare the method based on the StStSt model with methods in [Chan and Wong, 1998] and [Molina et al., 2006]. In these experiments, we use the $256 \times 256$ "Cameraman" image, degraded with several PSFs and noise levels. Specifically, we used three different PSFs; a Gaussian-shaped PSFs with variance 5, a uniform square-shaped PSFs of size $7 \times 7$ and a rectangular non-symmetric, accelerated motion blur [Yitzhaky et al., 1998] given by

$$h(x,y) = \begin{cases} (u_0^2 + 2a(x+s_x))^{-1/2} & \text{if } |x| \leq s_x \text{ and } |y| \leq s_y \ , \\ 0 & \text{otherwise,} \end{cases}$$

with $s_x = 4$, $s_y = 1$, $u_0 = 0.5$ and $a = 0.1$. We also used two levels of noise; low noise with $SNR = 40dB$ and high noise with $SNR = 20dB$. The PSF was initialized as a Gaussian-shaped function with variance $\sigma^2_{h_{in}} = 3$. For the StStSt method we used a Gaussian-shaped kernel function with variance $\sigma^2_\phi = 2$, in all cases except for the case of accelerated motion PSF, where we used a Gaussian-shaped kernel with variance $\sigma^2_\phi = 1$. The degraded images are shown in Fig. 5.4.2 and the restored images are shown in Fig. 5.8 and Fig. 5.9. The parameters of all the methods were selected in a trial and error manner in order to optimize the resulting images.

We can observe here that in all cases the StStSt method outperforms both the methods in [Chan and Wong, 1998] and [Molina et al., 2006], especially in the case of low noise with $SNR = 40dB$. Specifically, the method in [Chan and Wong, 1998] fails to estimate the Gaussian-shaped and motion PSFs, which is explained by the fact that the TV constraint on the PSF has the tendency to create flat areas and discontinuities, that are in contrast with the smooth PSFs that were used.

In terms of computational cost, the method in [Molina et al., 2006] is the most

(a) Degraded Image (b) GGG $0.47, 0.88$ (c) StStSt $5.29, 9.44$

(d) [Chan and Wong, 1998] $3.13, 5.64$ (e) [Molina et al., 2006] $0.54, 2.44$ (f) Known PSF [Chantas et al., 2006] $8.63, -$

Figure 5.6: Comparison of the proposed methods on the (a) lenna image degraded with a uniform, $7 \times 7$ square-shaped PSF. Estimated images using the (b) GGG method, (c) StStSt method (d) method in [Chan and Wong, 1998], (e) method in [Molina et al., 2006] and (f) Known PSF restoration method in [Chantas et al., 2006]. In all cases the PSF was initialized as a Gaussian with $\sigma^2_{h_{in}} = 3$ and the kernel was a Gaussian with variance $\sigma^2_\phi = 0.1$. The numbers below each image are the ISNR values of the image ($ISNR_f$) and the corresponding PSF ($ISNR_h$).

efficient, since each iteration involves $O(M \log M)$ operations. On the other hand, each iteration of both the proposed method and the method in [Chan and Wong, 1998] require the solution of a $M \times M$ linear system that is solved using the conjugate gradient method and require $O(CM \log M)$ computations, where $C$ is the number of conjugate gradient iterations.

### 5.4.3 Experiments with real astronomical images

We also applied the proposed methodology on a real astronomical image of the Saturn planet, which has previously been used in [Molina et al., 2006]. Astronomical measure-

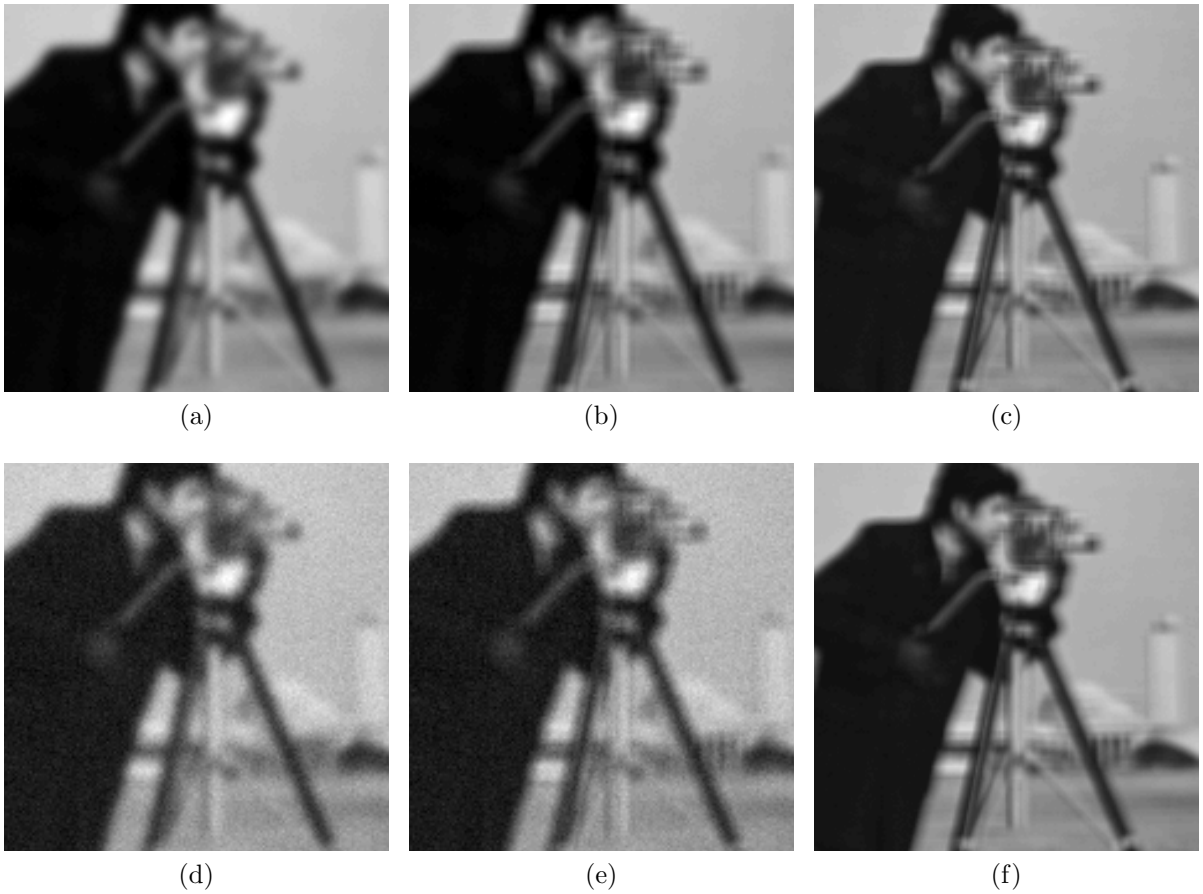


Figure 5.7: Degraded cameraman images with (a)-(c) $SNR = 40dB$ and (d)-(f) $SNR = 20dB$. The PSF was (a),(d) Gaussian-shaped with variance $\sigma_h^2 = 5$, (b),(e) uniform, square-shaped $7 \times 7$ and (c),(f) accelerated motion blur.

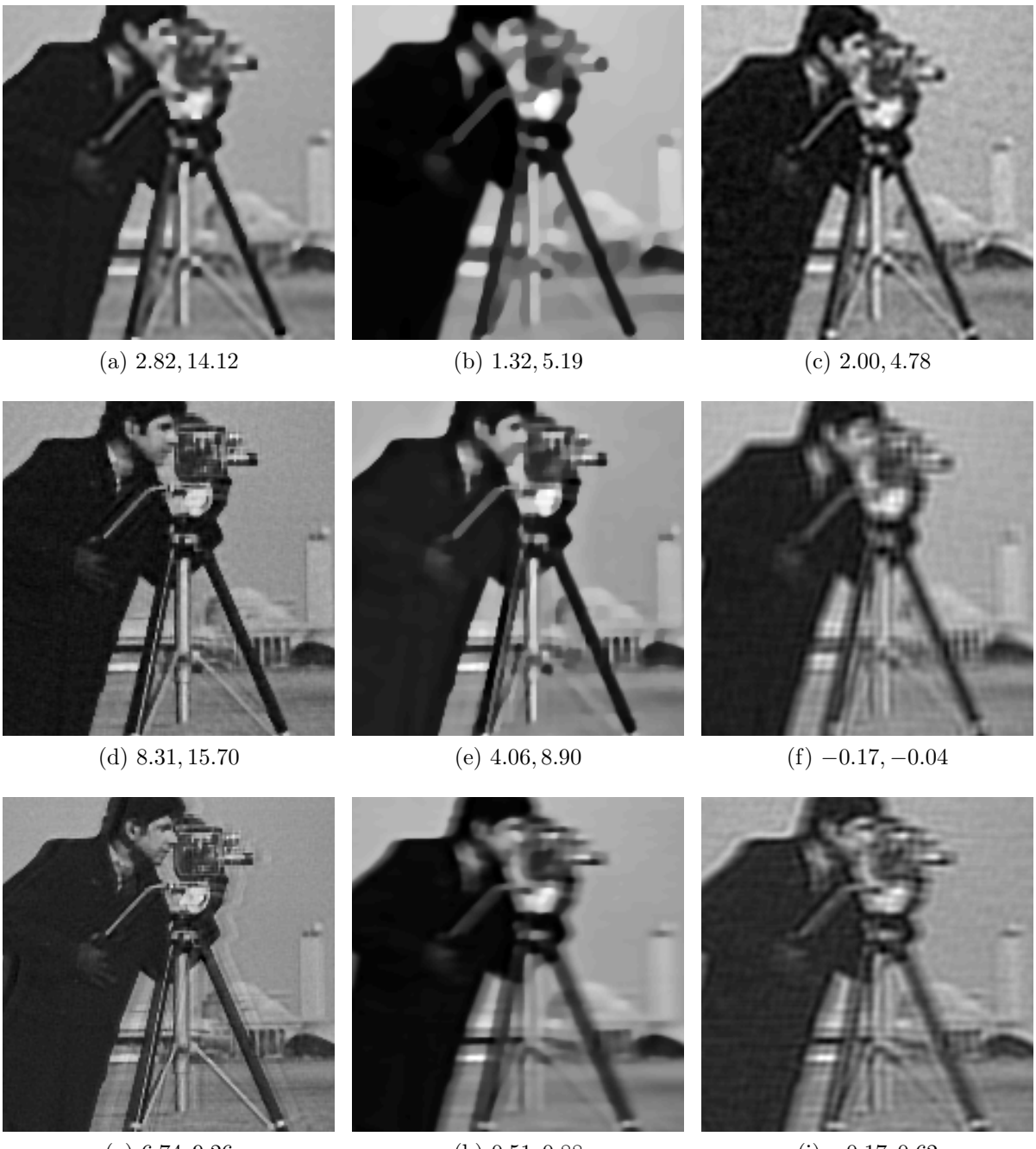
ments suggest the following PSF model for ground based telescopes:

$$h(r) \propto (1 + \frac{r^2}{R^2})^{-\delta}. \tag{5.56}$$

The parameters $\delta$ and $R$ can be measured [Molina et al., 2006] and $\delta \approx 3$ and $R \approx 3.4$. The recovered images by the different methods are shown in Fig. 5.10 and the resulting PSFs in Fig. 5.11.

From these images it is clear again that the models with two or more Student's t priors give visually superior results. In these images there is less ringing at the edges, noise in flat areas and the Saturn bands are better separated. Furthermore, the StStSt model produces again "sharper" images. It is interesting to notice that the StGG model does not yield good recovered images although it estimates well the measured PSF. This demonstrates the inappropriateness of the Gaussian to model the errors of the BID model and the image model. Notice also, that again, the TV-based methodology fails to estimate the smooth PSF and creates edges in areas where they do not exist in the original PSF, see Fig. 5.11.

StStSt [Chan and Wong, 1998] [Molina et al., 2006]



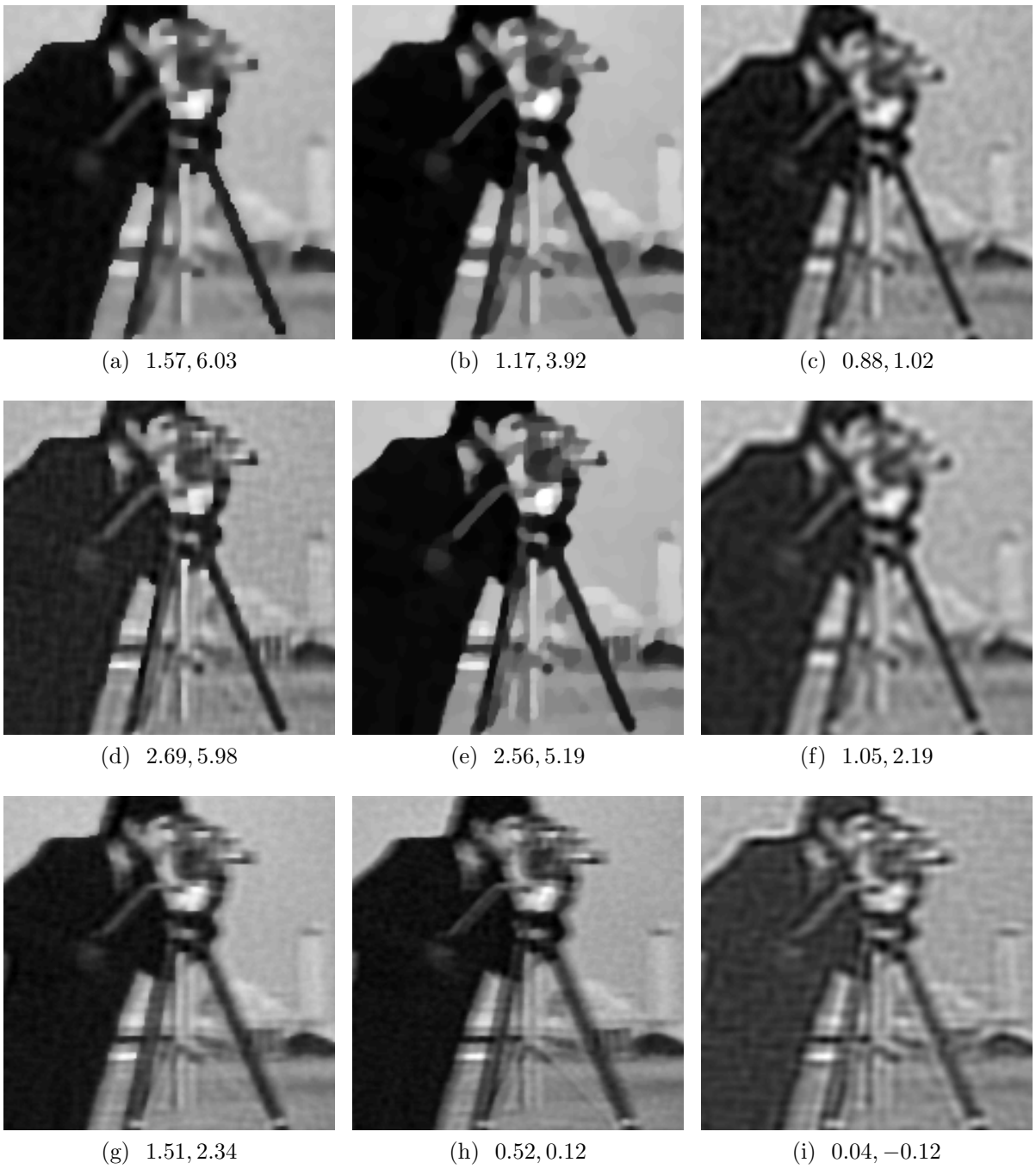
(a) $2.82, 14.12$ (b) $1.32, 5.19$ (c) $2.00, 4.78$

(d) $8.31, 15.70$ (e) $4.06, 8.90$ (f) $-0.17, -0.04$

(g) $6.74, 9.26$ (h) $0.51, 0.88$ (i) $-0.17, 0.62$

Figure 5.8: Comparison on cameraman image with SNR = 40dB and (a)-(c) Gaussian-shaped PSF with variance $\sigma_h^2 = 5$, (d)-(f) uniform, square-shaped $7 \times 7$ PSF (g)-(i) motion-blur PSF. Estimates obtained with (b), (f), (g) the proposed StStSt method, (c), (g), (k) method in [Chan and Wong, 1998] and (d), (h), (l) method in [Molina et al., 2006]. The numbers below each image are the ISNR values of the image ($ISNR_f$) and the corresponding PSF ($ISNR_h$).

StStSt [Chan and Wong, 1998] [Molina et al., 2006]

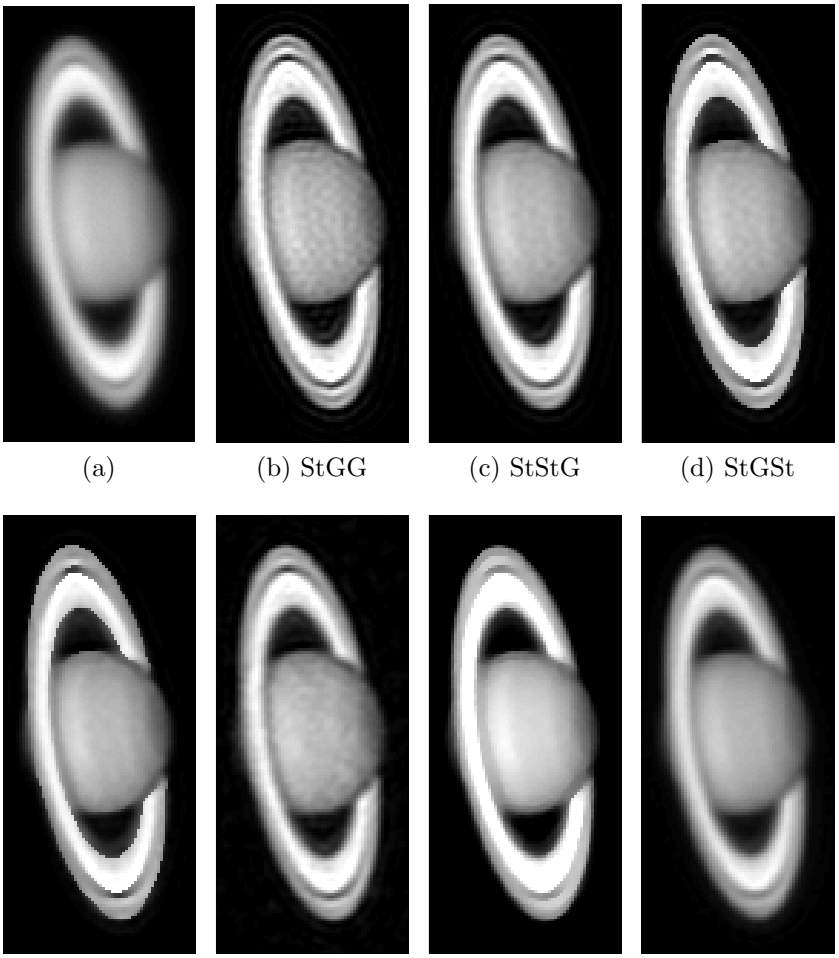


(a) $1.57, 6.03$ (b) $1.17, 3.92$ (c) $0.88, 1.02$

(d) $2.69, 5.98$ (e) $2.56, 5.19$ (f) $1.05, 2.19$

(g) $1.51, 2.34$ (h) $0.52, 0.12$ (i) $0.04, -0.12$

Figure 5.9: Comparison on cameraman image with SNR = 20dB and (a)-(c) Gaussian-shaped PSF with variance $\sigma_h^2 = 5$, (d)-(f) uniform, square-shaped $7 \times 7$ PSF (g)-(i) motion-blur PSF. Estimates obtained with (b), (f), (g) the proposed StStSt method, (c), (g), (k) method in [Chan and Wong, 1998] and (d), (h), (l) method in [Molina et al., 2006]. The numbers below each image are the ISNR values of the image ($ISNR_f$) and the corresponding PSF ($ISNR_h$).



(a) (b) StGG (c) StStG (d) StGSt

(e) StStSt (f) GGG (g) [Chan and Wong, 1998] (h) [Molina et al., 2006]

Figure 5.10: Comparison on real astronomical image of Saturn. (a) Degraded image. Estimated images using the methods (b) StGG, (c) StStG (d) StGSt, (e) StStSt, (f) GGG and the methods in (g) [Chan and Wong, 1998] and (h) [Molina et al., 2006]. The PSF was initialized as a Gaussian with $\sigma^2_{h_{in}} = 3$ in all cases and the kernel was a Gaussian with variance $\sigma^2_\phi = 1$.

### 5.4.4 Selecting the kernel width and initial values for the parameters

The proposed method uses a sparse kernel model to estimate the PSF. The significance of the kernel model is that it favors smooth estimations of the PSF, by forcing neighboring pixels to have similar values. This is important in order to enforce PSF smoothness and prevent the noise in the observed image to corrupt the PSF estimate. However, selecting an appropriate kernel is not straightforward. Here, we have considered several Gaussian kernels of different widths in order to determine how the proposed method is affected by the width of the Gaussian kernel. We have applied the proposed method on the artificially blurred images of the first experiment and considered degradation with Gaussian PSF or

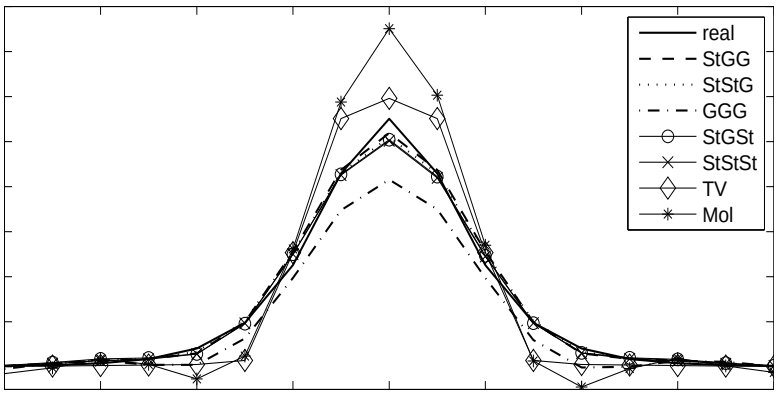



Figure 5.11: One dimensional slice of the true and estimated PSFs for the images of Fig. 5.10. The true PSF has been estimated as $h(r) \propto (1 + \frac{r^2}{R^2})^{-\delta}$, with $\delta \approx 3$ and $R \approx 3.4$. The kernel was Gaussian with variance $\sigma_\phi^2 = 1$.

Table 5.2: ISNR for image and PSF for various values of the kernel width for the case of Gaussian-shaped PSF with $\sigma_h^2 = 5$.

| Method | $\sigma_\phi^2 = 0.1$ | | $\sigma_\phi^2 = 1$ | | $\sigma_\phi^2 = 2$ | | $\sigma_\phi^2 = 3$ | |
|---|---|---|---|---|---|---|---|---|
| | $ISNR_f$ | $ISNR_h$ | $ISNR_f$ | $ISNR_h$ | $ISNR_f$ | $ISNR_h$ | $ISNR_f$ | $ISNR_h$ |
| GGG | 1.62 | -0.57 | 1.92 | 0.47 | 2.57 | 2.62 | 2.90 | 5.12 |
| StGG | 3.53 | 6.58 | 3.53 | 7.49 | 3.47 | 7.95 | 2.39 | 1.78 |
| StStG | 3.19 | 7.15 | 3.21 | 7.40 | 3.77 | 10.55 | 2.33 | 0.36 |
| StGSt | 3.69 | 8.86 | 3.96 | 10.33 | 4.24 | 12.30 | 1.55 | 2.88 |
| StStSt | 4.00 | 11.32 | 3.98 | 11.36 | 3.94 | 12.31 | 2.48 | 0.71 |

Table 5.3: ISNR for image and PSF for various values of the kernel width for the case of uniform, $7 \times 7$ square-shaped PSF.

| Method | $\sigma_\phi^2 = 0.1$ | | $\sigma_\phi^2 = 1$ | | $\sigma_\phi^2 = 2$ | | $\sigma_\phi^2 = 3$ | |
|---|---|---|---|---|---|---|---|---|
| | $ISNR_f$ | $ISNR_h$ | $ISNR_f$ | $ISNR_h$ | $ISNR_f$ | $ISNR_h$ | $ISNR_f$ | $ISNR_h$ |
| GGG | 0.70 | -4.71 | 0.64 | -3.41 | 0.12 | -0.64 | 0.13 | -3.87 |
| StGG | 2.17 | 6.69 | 1.20 | 9.09 | -0.37 | 1.67 | -2.31 | -0.43 |
| StStG | 5.57 | 10.91 | 5.45 | 9.27 | -0.29 | 1.87 | -2.12 | -0.20 |
| StGSt | 5.87 | 8.12 | 5.62 | 7.80 | 4.22 | 6.72 | 0.20 | -0.12 |
| StStSt | 5.29 | 9.44 | 4.56 | 8.17 | -0.51 | 2.01 | -1.58 | 0.09 |

uniform-square shaped PSF. Tables 5.2 and 5.3 show the ISNRs of the image and PSF for several values of the kernel width, for the case where the true PSF is Gaussian-shaped and square-shaped, respectively. Notice that in all cases, selecting a very large kernel leads to very smooth estimates of the PSF that provide poor results. In case of uniform square true PSF (Table 5.3) the best results are obtained when using a very small kernel. This is because the square PSF is not smooth at the edges of the rectangle. On the other hand, in the case of Gaussian-shaped true PSF (Table 5.2), it is favorable to select a kernel that produces smooth PSF estimation.

It must be also noted that the performance of all the variational algorithms generally depends on the initialization of the parameters. This happens because the variational bound is a non-convex function and therefore, depending on the initialization, a different local maximum may be attained. In order to apply the proposed method, the following parameters have to be initialized:

**The weights $w$ of the kernel model that define the PSF**

In BID, having a good estimate of the PSF is usually very important and many BID methods fail when they are badly initialized. This is a significant limitation, because in many situations there is no available estimate of the PSF. The proposed method does not rely on a good initial PSF estimation. Instead, the sparse kernel based PSF model, can successfully estimate the PSF from the observed image. This is demonstrated in the previous experiments, where we successfully estimated Gaussian-shaped, square-shaped and accelerated motion PSFs using an initial PSF that was Gaussian-shaped with variance $\sigma^2_{h_{in}} = 3$. The weights $w$ were initialized by solving the PSF model given in (5.6), which gives $\boldsymbol{w} = \beta \boldsymbol{\Sigma}_w \boldsymbol{\Phi}^T h$ with $\boldsymbol{\Sigma}_w = \left(\beta \boldsymbol{\Phi}^T \boldsymbol{\Phi} + \alpha \boldsymbol{I}\right)^{-1}$.

**The weight normalization parameters $\alpha_i$ of the PSF model and the hyperparameters $a^\alpha$, $b^\alpha$**

Initially, we set all these parameters to very small values, e.g. $\alpha_i = 10^{-16}$, which corresponds to a very flexible linear model. This is desirable in order to obtain an initial estimate of the support of the PSF using all the available kernels. The hyperparameters $a^\alpha$ and $b^\alpha$ are set to zero, thus assuming an uninformative distribution for the parameters $\alpha$. During inference, the parameters $\alpha_i$ for most kernels tend to infinity, thus the support of the PSF is limited.

**The noise precision $\beta$ and the hyperparameters $a^\beta$, $b^\beta$**

The noise precision $\beta$ is initially set to $\beta = 10^3$. The hyperparameters $a^\beta$, $b^\beta$ are initially set to values that define a Gamma distribution with mean $10^3$ and variance $10^2$, which is a flat and rather uninformative distribution. Their values are then updated using (5.43) and (5.44).

**The strength of the image prior $\gamma$ and the hyperparameters $a^\gamma$, $b^\gamma$**

The parameter $\gamma$ that defines the strength of the image prior is initially set to $\gamma = 10^2$. The hyperparameters $a^\gamma$ and $b^\gamma$ are set to values that define a Gamma distribution with mean $10^2$ and variance $10^4$. Updating $a^\gamma$ and $b^\gamma$ (Section 5.3.2), usually improves the performance of the algorithm, at least in the first few iterations. However, we have empirically found that at convergence, these hyperparameters attain very small values, thus defining an uninformative distribution. This leads to very noisy image estimates and for this reason we do not update the hyperparameters $a^\gamma$, $b^\gamma$ but keep them fixed to their initial values. An explanation for the failure to estimate these parameters is that we use an improper prior for the image (5.16). Although selecting values for these parameters may seem arbitrary they actually depend on the characteristics of the image. Specifically, small values of the parameter $b^\gamma$ lead to very smooth solutions, while small values of the parameter $a^\gamma$ allow few hard edges by defining a heavy tailed distribution for the image local differences.

## 5.5 Conclusions and Future Work

We presented a Bayesian approach to the BID problem where the PSF is modeled as a superposition of kernel functions, i.e. as a kernel-based linear model. We assumed a suitable heavy tailed prior distribution on this kernel model, in order to obtain a sparse estimate of the support and shape of the PSF. We also used a heavy tailed pdf both for the noise, in order to achieve robustness to BID model errors and for the local image differences, in order to allow the reconstruction of edges. The Student's t pdf was our choice as a heavy tailed pdf, due to its close relationship with the Gaussian. Because of the complexity of this model, the variational framework was used for approximate Bayesian inference.

Several experiments were carried out, to test the proposed methodology. These experiments indicated beyond doubt that the use of a Student's t pdf to model the weights of the PSF kernel-based model in crucial to the success of this approach. Furthermore, Bayesian BID models that use at least two Student's t priors, one for the PSF, are clearly superior to BID models that use two or more Gaussian priors. It is also interesting to notice that the StStSt model that uses only Student's-t priors seems to produce visually superior images compared to models that use a combination of two Student's t and Gaussian priors.

We also compared this methodology with TV-based and Bayesian as implemented in [Molina et al., 2006] BID in a number of different scenarios. From these comparisons it is clear that the proposed methodology is always superior to the Gaussian model based methodology in [Molina et al., 2006]. As far as TV-based BID is concerned, the proposed method is clearly superior for scenarios with small sized PSFs and low noise. In the case of large PSFs and high noise the two methods produce different in nature results.

The proposed methodology produces image where image details were better preserved. It also yields better $ISNR$ values. However, it produces "ringing" artifacts in image edges. TV-based BID gave no "ringing", however, many image details were eliminated.

In the future it's interesting to explore the possibility of learning the filters $Q^k$ in a manner analogous to [Welling et al., 2003]. Furthermore, it is possible to explore extending the *constrained variational* methodology in [Chantas et al., 2007] to BID in order to avoid using the approximation of the partition function in (5.16). Finally, it might be interesting to learn the width parameter of the kernel function, possibly using the methodology described in Chapter 6.

# Chapter 6

# Adaptive Kernel Learning for the Relevance Vector Machine



## 6.1 Introduction

As mentioned in Chapter 3 the Relevance Vector Machine (RVM) constitutes a special case of the sparse Bayesian linear model that assumes that the basis functions are kernels placed at the training points. Recently, it has been used successfully in many applications; for example in recognition of hand motions [Wong and Cipolla, 2005], recovery of 3D human pose from silhouettes [Agarwal and Triggs, 2004], detection of clustered microcalcifications for mammography [Wei et al., 2005], classification of gene expression data [Li et al., 2002, Yang et al., 2004], detection of activations for neuroimaging [Lukic et al., 2007], real time tracking [Williams et al., 2005], etc. 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 [Quiñonero-Candela and Hansen, 2002] that the width parameter of

Gaussian kernels can be learned by maximizing the marginal likelihood of the model. Furthermore, in [Lanckriet et al., 2004, Girolami and Rogers, 2005, Sonnenburg et al., 2006] the kernel has been modeled as a linear combination of other basis functions. In [Krishnapuram et al., 2004] feature selection has been achieved by learning the variances of anisotropic Gaussian kernel functions after applying to them a sparsity enforcing prior. Also, in [Snelson and Ghahramani, 2006] an alternative to the Gaussian process model has been proposed that learns a set of pseudo-inputs, which are similar to the relevance vectors, 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 share the same parameter values. 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 this chapter, we propose a new methodology to automatically learn the basis functions of a sparse linear model [Tzikas et al., b, 2008a]. 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 [Schmolck and Everson, 2007], 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* training algorithm [Tipping and Faul, 2003] described in Section 3.4.3. 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 case of 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 [Tipping, 2001]:

- 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 the chapter is organized as follows. In Section 6.2 we review the sparsity prior of [Schmolck and Everson, 2007] and generalize it for non-orthogonal basis function sets. In Section 6.3 we present an algorithm for learning the basis function set. In Section 6.4 we provide experiments on artificial datasets that demonstrate the advantages of

the proposed method, we compare the proposed algorithm with the typical RVM algorithm on benchmark datasets and we apply the proposed method for analysis of functional neuroimages. In Section 6.5 we discuss the computational cost of the method and provide a probabilistic interpretation of the kernel function and finally in Section 6.7 we provide some conclusions.

## 6.2 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 (3.27), sparsity depends on selecting an appropriate distribution $p(\boldsymbol{\alpha})$. The typical RVM [Tipping, 2001] suggests the use of 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 $a$, $b$ of the Gamma distribution, they are typically set to $a = b = 0$. These values define an improper uninformative distribution for $\alpha_i$ 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 [Schmolck and Everson, 2007]. 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 trace($\boldsymbol{S}$), measure the effective number of parameters of the model. This motivates the following sparsity prior [Schmolck and Everson, 2007]:

$$p(\boldsymbol{\alpha}) \propto \exp(-c\,\mathrm{trace}(\boldsymbol{S})), \tag{6.1}$$

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 [Holmes and Denison, 1999]:

$$c = \begin{cases} 0 & \text{None (typical RVM),} \\ 1 & \text{AIC (Akaike information criterion),} \\ \log(N)/2 & \text{BIC (Baysian information criterion),} \\ \log(N) & \text{RIC (Risk inflation criterion).} \end{cases} \tag{6.2}$$

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

for each data point ($\boldsymbol{B} = \beta \boldsymbol{I}$) this prior reduces to:

$$p(\alpha_i) \propto \exp(-\frac{c}{1 + \alpha_i/\beta}). \tag{6.3}$$

Assuming an uninformative prior for the noise ($p(\beta) = const$), the use of the sparsity prior of (6.3) leads to the addition of a normalization term to the marginal log-likelihood of (3.43):

$$L^o = L - \sum_{i=1}^{M} \frac{c}{1 + \alpha_i/\beta}. \tag{6.4}$$

Keeping only the terms that depend on a single parameter $\alpha_i$ we can write:

$$l^o(\alpha_i) = l(\alpha_i) - \frac{c}{1 + \alpha_i/\beta}. \tag{6.5}$$

Based on this decomposition, an incremental algorithm that maximizes the marginal likelihood has been proposed in [Schmolck and Everson, 2007], which is similar to the typical incremental RVM algorithm [Tipping and Faul, 2003]. However, because of the sparsity prior, setting the derivative of (6.5) to zero does not provide analytical updates (such as (3.57)) for the weight precisions $\alpha_i$, but instead a numerical solution is required to update them.

In the proposed method, we consider the general case of non-orthogonal basis functions and heteroscedastic noise with different noise precision $\beta_n$ at each data point $\boldsymbol{x}_n$. Since $\text{trace}(\boldsymbol{\Phi\Sigma\Phi}^T\boldsymbol{B}) = M - \sum_{i=1}^{M} \alpha_i \Sigma_{ii}$ we can write the proposed sparsity prior as:

$$p(\alpha_i) \propto \exp\left(-c\left(M - \sum_{i=1}^{M} \alpha_i \Sigma_{ii}\right)\right). \tag{6.6}$$

Learning is again performed by maximizing the posterior $p(\boldsymbol{\alpha}, \boldsymbol{\beta}|\boldsymbol{t}) \propto p(\boldsymbol{t}|\boldsymbol{\alpha}, \boldsymbol{\beta}) p(\boldsymbol{\alpha}) p(\boldsymbol{\beta})$, which leads to adding to the marginal log-likelihood of (3.43) an additional term that is obtained from (6.6):

$$L^s = L - c(M - \sum_{i=1}^{M} \alpha_i \Sigma_{ii}). \tag{6.7}$$

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

$$\frac{\partial L^s}{\partial \log \alpha_i} = \frac{1}{2}(1 - \alpha_i \Sigma_{ii} - \alpha_i \mu_i^2) + c(1 - \alpha_i \Sigma_{ii}) \alpha_i \Sigma_{ii} = 0, \tag{6.8}$$

we obtain the following update formula for $\alpha_i$:

$$\alpha_i = \frac{\gamma_i}{\mu_i^2 - 2c\gamma_i \Sigma_{ii}}. \tag{6.9}$$

In the regression case assuming that $\boldsymbol{B} = \beta \boldsymbol{I}$ we can also update $\beta$ by setting the

derivative of $L^s$ with respect to $\log\beta$ to zero:

$$\frac{\partial L^s}{\partial \log\beta} = \frac{1}{2}\left[\frac{N}{\beta} - \|\boldsymbol{t} - \boldsymbol{\Phi\mu}\|^2 - \operatorname{trace}(\boldsymbol{\Sigma\Phi}^T\boldsymbol{\Phi})\right] - \beta c \operatorname{trace}(\boldsymbol{\Phi\Sigma\Phi}) = 0. \tag{6.10}$$

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

Regarding the incremental algorithm, keeping only the terms of $L$ that depend on a single parameter $\alpha_i$ and because $\Sigma_{ii} = 1/(\alpha_i + s_i)$ [Tipping and Faul, 2003], we obtain:

$$l^s(\alpha_i) = l(\alpha_i) - c(1 - \frac{\alpha_i}{\alpha_i + s_i}), \tag{6.11}$$

whose gradient is given by:

$$\frac{\partial l^s(\alpha_i)}{\partial \alpha_i} = \frac{1}{2}\left[\frac{1}{\alpha_i} - \frac{1}{\alpha_i + s_i} - \frac{q_i^2 - 2cs_i}{(\alpha_i + s_i)^2}\right]. \tag{6.12}$$

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

$$\begin{aligned} \alpha_i &= \frac{s_i^2}{q_i^2 - (2c+1)s_i} && \text{if } q_i^2 > (2c+1)s_i, \\ \alpha_i &= \infty && \text{if } q_i^2 \leq (2c+1)s_i. \end{aligned} \tag{6.13}$$

## 6.3 Kernel Learning

### 6.3.1 Sparse infinite linear models

Consider a linear model of the form

$$y(\boldsymbol{x}|\boldsymbol{w}) = \sum_{i=1}^{M} w_i \phi_i(\boldsymbol{x}). \tag{6.14}$$

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:

$$y[\boldsymbol{x}|w(\boldsymbol{\xi})] = \int w(\boldsymbol{\xi})\phi(\boldsymbol{x};\boldsymbol{\xi})\,\mathrm{d}\boldsymbol{\xi}, \tag{6.15}$$

which are defined by using a family of basis functions $\phi_i(\boldsymbol{x}) = \phi(\boldsymbol{x};\boldsymbol{\xi})$ 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:

$$w(\boldsymbol{\xi}) = \sum_{i=1}^{M} w_i \delta(\boldsymbol{\xi}, \boldsymbol{\theta}_i), \tag{6.16}$$

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 (6.16), 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 $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_M)^T$:

$$y(\boldsymbol{x}|\boldsymbol{w}) = \sum_{i=1}^{M} w_i \phi(\boldsymbol{x}; \boldsymbol{\theta}_i). \tag{6.17}$$

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

### 6.3.2 Learning algorithm

In this section we propose an algorithm for learning the model of (6.17). 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, we use 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 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 re-estimates 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.

The steps of the proposed learning method are summarized in Algorithm 1 and we

next discuss them in detail.

**Algorithm 1** Sparse Infinite Linear Model Learning Algorithm.

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 $\beta$.
4. Remove from the model any unnecessary active basis functions.
5. Repeat steps 1 to 4 until convergence.

### Select an inactive basis function to add to the model

Addition of a basis functions to the model should always be performed in a way that increases the marginal likelihood. This search is an optimization procedure in the space defined by the hyperparameters $\alpha_i$ and the basis function parameters $\theta_i$. In contrast, in the typical incremental RVM method, where the set of candidate basis functions is discrete and finite, selecting a basis function to add to the model requires discrete optimization which is performed by evaluating the marginal likelihood for each candidate basis function.

In our continuous optimization framework, the required derivative for $\alpha_i$ is given by (6.12) and for $\theta_{ik}$ is given by:

$$\frac{\partial l^s(\alpha_i)}{\partial \theta_{ik}} = -\left(\frac{1}{\alpha_i + s_i} + \frac{q_i^2 + c\alpha_i}{(\alpha_i + s_i)^2}\right) r_i + \frac{q_i}{\alpha_i + s_i} w_i, \tag{6.18}$$

where

$$r_i \equiv \frac{1}{2}\frac{\partial s_i}{\partial \theta_{ik}} = \boldsymbol{\phi}_i(\boldsymbol{\theta}_i)^T \boldsymbol{C}_{-i}^{-1} \frac{\partial \boldsymbol{\phi}_i(\boldsymbol{\theta}_i)}{\partial \theta_{ik}}, \tag{6.19}$$

$$w_i \equiv \frac{\partial q_i}{\partial \theta_{ik}} = \boldsymbol{t}^T \boldsymbol{C}_{-i}^{-1} \frac{\partial \boldsymbol{\phi}_i(\boldsymbol{\theta}_i)}{\partial \theta_{ik}}. \tag{6.20}$$

These derivatives can be efficiently computed in a similar manner as in (3.55):

$$r_i = \frac{\alpha_i R_i}{\alpha_i - R_i}, \tag{6.21}$$

$$w_i = \frac{\alpha_i W_i}{\alpha_i - W_i}, \tag{6.22}$$

where

$$R_i = \boldsymbol{\phi}_i^T \boldsymbol{C}^{-1} \frac{\partial \boldsymbol{\phi}_i(\boldsymbol{\theta}_i)}{\partial \theta_{ik}}, \tag{6.23}$$

$$W_i = \boldsymbol{\phi}_i^T \boldsymbol{C}^{-1} \frac{\partial \boldsymbol{\phi}_i(\boldsymbol{\theta}_i)}{\partial \theta_{ik}}, \tag{6.24}$$

which gives:

$$R_i = \boldsymbol{\phi}_i^T \boldsymbol{B} \frac{\partial \boldsymbol{\phi}_i(\boldsymbol{\theta}_i)}{\partial \theta_{ik}} - \boldsymbol{\phi}_i^T \boldsymbol{B} \boldsymbol{\Phi} \boldsymbol{\Sigma} \boldsymbol{\Phi}^T \boldsymbol{B} \frac{\partial \boldsymbol{\phi}_i(\boldsymbol{\theta}_i)}{\partial \theta_{ik}}, \tag{6.25}$$

$$W_i = \boldsymbol{\phi}_i^T \boldsymbol{B} \frac{\partial \boldsymbol{\phi}_i(\boldsymbol{\theta}_i)}{\partial \theta_{ik}} - \boldsymbol{\phi}_i^T \boldsymbol{B} \boldsymbol{\Phi} \boldsymbol{\Sigma} \boldsymbol{\Phi}^T \boldsymbol{B} \frac{\partial \boldsymbol{\phi}_i(\boldsymbol{\theta}_i)}{\partial \theta_{ik}}. \tag{6.26}$$

Notice that since we use a local optimization method (in our case the quasi-Newton BFGS 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 $\epsilon_n^2$:

$$p(\boldsymbol{m} = \boldsymbol{x}_n) = \frac{\epsilon_n^2}{\sum_{n=1}^N \epsilon_n^2}. \tag{6.27}$$

### 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 parameters of the already existing basis functions 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 $\alpha_i$ and $\theta_i$ of all the active basis functions of the current model. Specifically, the weight precision parameters $\alpha_i$ are updated using (3.57), while the basis function parameters $\theta_i$ are updated using an optimization algorithm. Instead of computing separately the derivative for each $\theta_i$ from (6.18), we use the following formula [Tipping, 2001]:

$$\frac{\partial L^s}{\partial \theta_{ik}} = \sum_{n=1}^N \frac{\partial L^s}{\partial \phi_i(\boldsymbol{x}_n; \boldsymbol{\theta}_i)} \frac{\partial \phi_i(\boldsymbol{x}_n; \boldsymbol{\theta}_i)}{\partial \theta_{ik}}, \tag{6.28}$$

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

$$\boldsymbol{D} = (\boldsymbol{C}^{-1}\boldsymbol{t}\boldsymbol{t}^T\boldsymbol{C}^{-1} - \boldsymbol{C}^{-1})\boldsymbol{\Phi}\boldsymbol{A}^{-1} + 2c\boldsymbol{B}\boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{A}\boldsymbol{\Sigma} \tag{6.29}$$

$$= \boldsymbol{B}\left[(\boldsymbol{t} - \boldsymbol{\Phi}\boldsymbol{\mu})\boldsymbol{\mu}^T - \boldsymbol{\Phi}\boldsymbol{\Sigma}\right] + 2c\boldsymbol{B}\boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{A}\boldsymbol{\Sigma}. \tag{6.30}$$

**Optimize hyperparameters and noise precision**

The hyperparameters $\boldsymbol{\alpha}$ of the active basis functions are updated at each iteration using (6.9). Similarly, in regression the noise precision $\beta$ is updated by numerically solving (6.10).

**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.

**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.

## 6.4 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 i) the typical RVM with Gaussian kernel [Tipping, 2001] and ii) the RVM with a smoothness prior and orthogonal wavelet basis functions (denoted with sRVM) [Schmolck and Everson, 2007]. Notice that the sRVM approach is based on wavelet analysis requiring that the training data points are equally spaced. Therefore, it can not be used for arbitrary multidimensional regression and classification problems and we test it only on one-dimensional artificial regression example.

More specifically, we consider Gaussian kernel functions of the form

$$\phi_i(\boldsymbol{x}; \boldsymbol{m}_i, h_i) = \exp\left[-h_i^{-2}\|\boldsymbol{x} - \boldsymbol{m}_i\|^2\right], \tag{6.31}$$

whose derivatives with respect to the mean and variance parameters are:

$$\frac{\partial \phi_i(\boldsymbol{x}_n; \boldsymbol{m}_i, h_i)}{\partial \boldsymbol{m}_i} = 2h_i^{-2}\phi_i(\boldsymbol{x}_n; \boldsymbol{m}_i, h_i)(\boldsymbol{x}_n - \boldsymbol{m}_i), \tag{6.32}$$

$$\frac{\partial \phi_i(\boldsymbol{x}_n; \boldsymbol{m}_i, h_i)}{\partial h_i} = 2h_i^{-3}\phi_i(\boldsymbol{x}_n; \boldsymbol{m}_i, h_i)\|\boldsymbol{x}_n - \boldsymbol{m}_i\|^2. \tag{6.33}$$

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, as proposed in Chapter 4.

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 10 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.

### 6.4.1 Experiments on Artificial Data

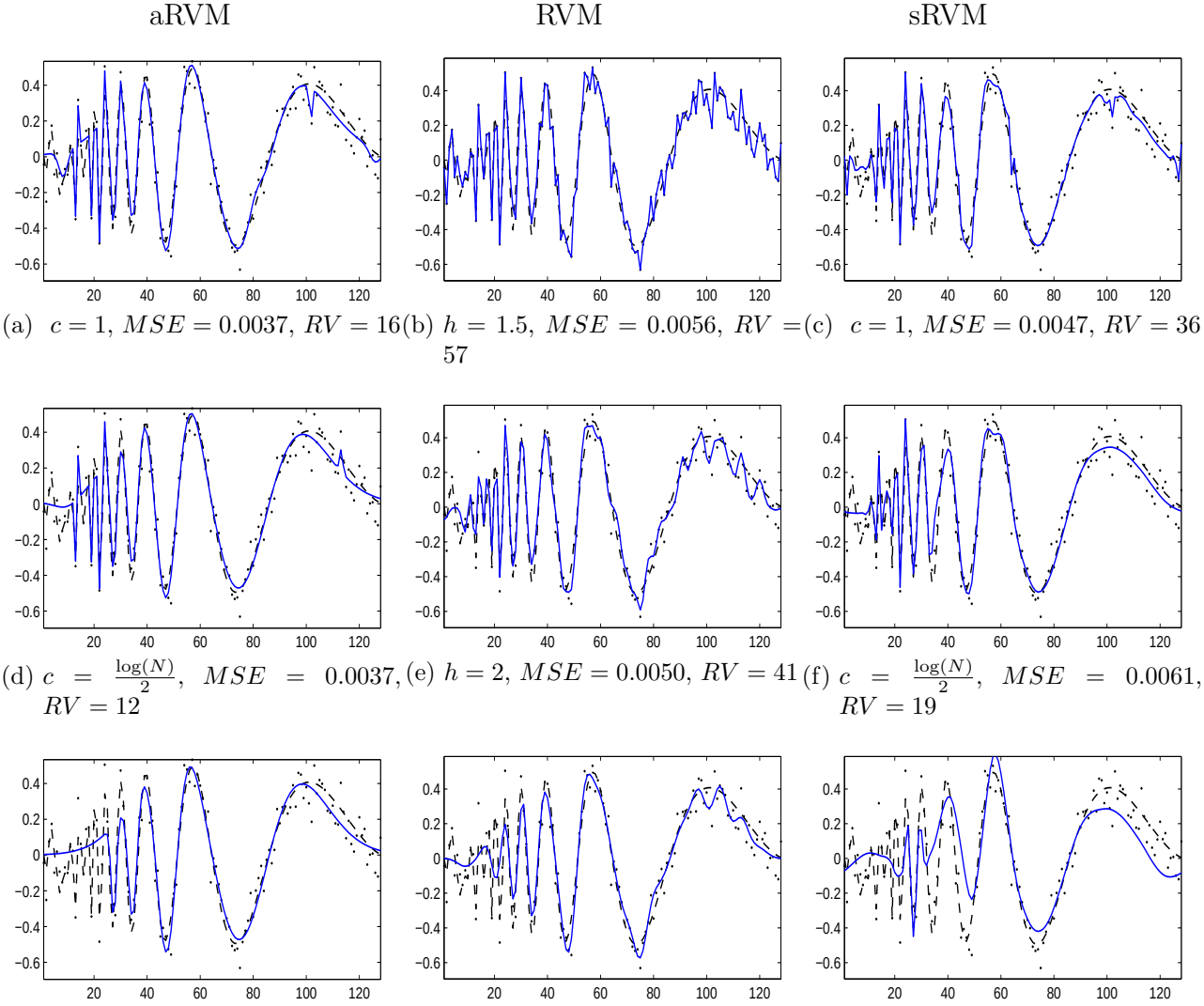
#### Regression

In the first experiment we generated $N = 128$ points from the well-known 'Doppler' function [Schmolck and Everson, 2007]:

$$g(x) = \sqrt{x(1-x)}\sin\frac{2\pi(1+\delta)}{x+\delta}, \tag{6.34}$$

with $\delta = 0.01$ and added white Gaussian noise of variance $\sigma^2 = 0.1$. We then applied the three compared methods and evaluated the estimated model on the same 128 points. In order to measure the quality of the estimates we compute the *mean square error* $MSE = \sum_n (g(x_n) - \hat{y}_n)^2/N$, where $\hat{y}_n$ 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. 6.1.

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. 6.1b), noisy estimates are obtained, while when using a large kernel size (shown in Fig. 6.1h), large fluctuations of the function (high frequencies) cannot be adequately estimated. Instead, the adaptive RVM and the sRVM (shown in Fig. 6.1d and Fig. 6.1f) can successfully es-




(a) $c = 1$, $MSE = 0.0037$, $RV = 16$ (b) $h = 1.5$, $MSE = 0.0056$, $RV = 57$ (c) $c = 1$, $MSE = 0.0047$, $RV = 36$

(d) $c = \frac{\log(N)}{2}$, $MSE = 0.0037$, $RV = 12$ (e) $h = 2$, $MSE = 0.0050$, $RV = 41$ (f) $c = \frac{\log(N)}{2}$, $MSE = 0.0061$, $RV = 19$

(g) $c = \log(N)$, $MSE = 0.0092$, $RV = 5$ (h) $h = 4$, $MSE = 0.0090$, $RV = 20$ (i) $c = \log(N)$, $MSE = 0.0254$, $RV = 4$

Figure 6.1: Regression example with Doppler signal. Estimates obtained (a),(d),(g) with aRVM, (b),(e),(h) with RVM and (c),(f),(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 relevance vectors ($RV$) are shown.

timate functions that exhibit smoothness in some regions and large fluctuations in other regions. However, the sRVM gives worst 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 (6.34). 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. 6.2 we provide the

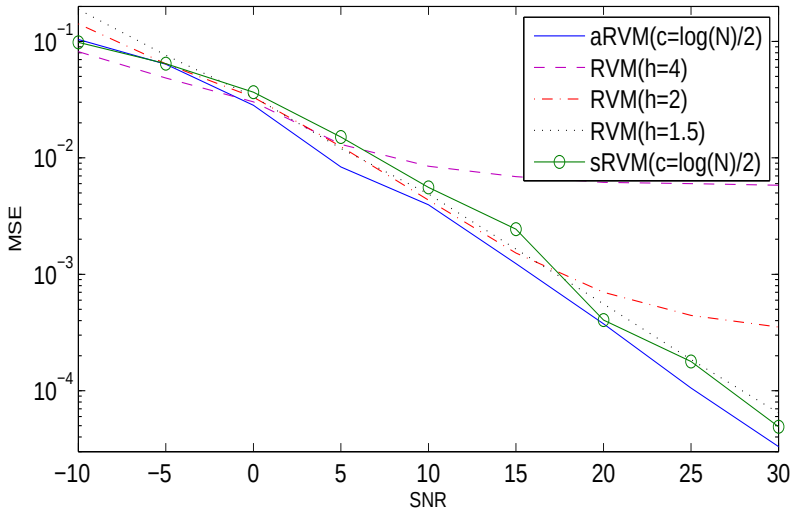



Figure 6.2: Comparison of the performance of aRVM, typical RVM and sRVM for several noise values.

MSE of the estimate of each method for various signal to noise (SNR) ratios. 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.

Next, we applied the proposed method on a two-dimensional generalization of the 'Doppler' function:

$$g(x_1, x_2) = g(x_1)g(x_2), \tag{6.35}$$

where $g(x)$ is given by 6.34. We then set $\delta = 0.01$ and generated a $128 \times 128$ image by sampling this function on a grid. We trained the compared methods using a subset of these samples, containing a proportion of $r = 0.5$ randomly selected samples. Furthermore, we added to the observations white Gaussian noise of variance $\sigma^2 = 0.1$.

We consider two approaches for two dimensional regression. In the first, we use isotropic Gaussian kernels, which assume the same variance for each dimension of the input space and are given by

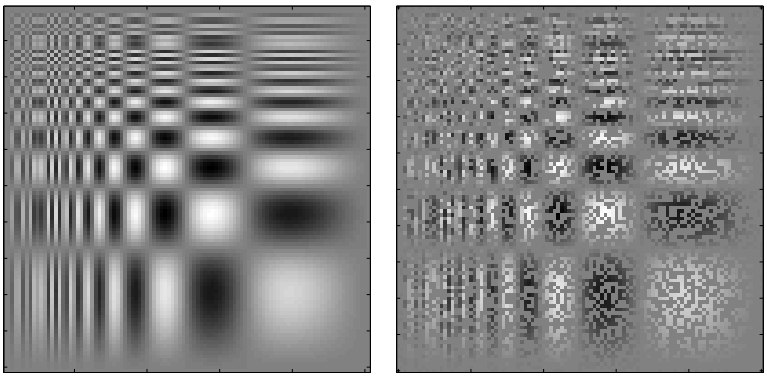
$$\phi(\boldsymbol{x}; \boldsymbol{m}, h) = \exp\left[-h^{-2}\|\boldsymbol{x} - \boldsymbol{m}\|^2\right]. \tag{6.36}$$

The second approach uses anisotropic Gaussian kernels, which use a separate variance for each dimension of the input space:

$$\phi(\boldsymbol{x}; \boldsymbol{m}, \boldsymbol{h}) = \exp\left[-\sum_{j=1}^{d}(h^j)^{-2}(x^j - m^j)^2\right]. \tag{6.37}$$

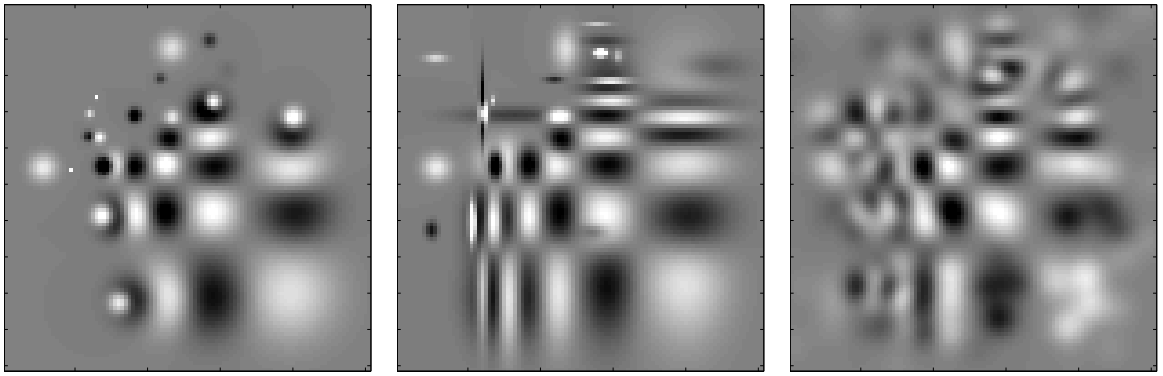
We denote the second approach as aRVM$^d$.

We then applied on the two-dimensional 'Doppler' function (i) the proposed aRVM



(a) true function (b) noisy samples

Figure 6.3: (a) True signal and (b) noisy samples of the two-dimensional 'Doppler' signal that was used for training.




(a) $MSE = 0.0036$ (b) $MSE = 0.0031$ (c) $MSE = 0.0038$

Figure 6.4: Estimation of the aRVM method with (a)isotropic and (b)anisotropic Gaussian kernel functions.

method with Gaussian kernels (ii) aRVM with anisotropic kernels (aRVM$^d$) and (iii) the typical RVM method with a fixed Gaussian kernel that was selected using cross-validation. The result of each method was evaluated by measuring the mean square error with respect to the true function (without noise) on the whole $128 \times 128$ image. For aRVM, we set the sparsity parameter to $c = \log(N)/2$ and for the kernel width of RVM we test several values and select to illustrate the case $h = 2$, which is the value that produced the smallest MSE among all tested values of $h$. The samples of the training set are shown in Fig. 6.3 and the estimations of the algorithms are shown in Fig. 6.4. Observing these results, it is obvious that in this case using anisotropic kernels improves the accuracy of the estimation.

### Classification

In this subsection we compare the typical RVM and adaptive RVM (aRVM) models on classification problems (sRVM has been proposed only for regression problems). We gen-

erated two-class, two-dimensional, artificial data by obtaining 50 samples from each of the following Gaussian mixture distributions:

$$p(x|C_1) = 0.25\mathrm{N}(\boldsymbol{\mu}_1, \sigma_1^2\boldsymbol{I}) + 0.75\mathrm{N}(\boldsymbol{\mu}_2, \sigma_2^2\boldsymbol{I}), \tag{6.38}$$

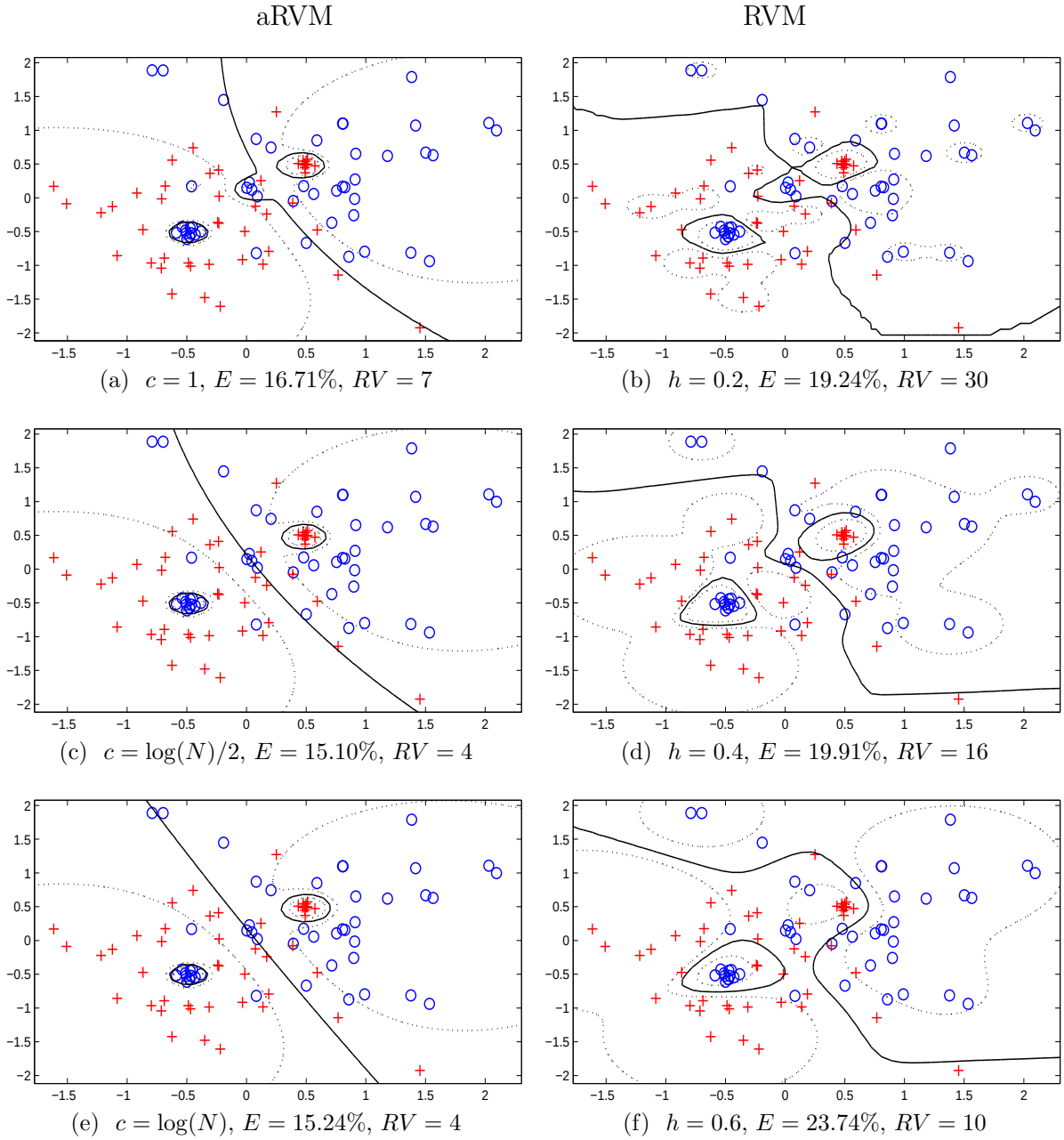
$$p(x|C_2) = 0.25\mathrm{N}(\boldsymbol{\mu}_2, \sigma_1^2\boldsymbol{I}) + 0.75\mathrm{N}(\boldsymbol{\mu}_1, \sigma_2^2\boldsymbol{I}), \tag{6.39}$$

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 $N_t = 10000$ test points drawn from the mixture distributions of (6.38) and (6.39). 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. 6.5 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 $c$ (Fig. 6.5c).

### 6.4.2 Experiments on Real Datasets

In this section we compare the performance of the proposed method (aRVM) with the typical RVM method on several regression and classification datasets[1]. In what follows, we describe the experimental setup that we followed. For each dataset, in order to estimate the generalization error of each method, we perform ten-fold 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 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, $\hat{y}_n$ the predicted value and $N$ the number of test examples. In classification the error is the percentage of misclassified

[1]Computer hardware, concrete and pima datasets were obtained from the UCI machine learning repository at `http://archive.ics.uci.edu/ml/`, the Boston housing dataset was obtained from `http://lib.stat.cmu.edu/datasets/boston`, and the banana, titanic, image and breast-cancer datasets from `http://ida.first.fraunhofer.de/projects/bench/`.




(a) $c = 1$, $E = 16.71\%$, $RV = 7$

(b) $h = 0.2$, $E = 19.24\%$, $RV = 30$

(c) $c = \log(N)/2$, $E = 15.10\%$, $RV = 4$

(d) $h = 0.4$, $E = 19.91\%$, $RV = 16$

(e) $c = \log(N)$, $E = 15.24\%$, $RV = 4$

(f) $h = 0.6$, $E = 23.74\%$, $RV = 10$

Figure 6.5: Classification example on artificial Gaussian clusters. Estimates obtained with (a),(c),(e) aRVM, (b),(d),(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 relevance vectors ($RV$) are shown.

Table 6.1: 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 6.2: 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 |

examples in the test set.

The results in Tables 6.1 and 6.2 show the cross-validation error and the number of relevance vectors (averaged over 10 folds) that were obtained by applying the RVM and aRVM methods on several regression and classification datasets. We can observe that in both regression and classification problems, the solutions obtained with aRVM use much less relevance vectors (RV) 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 datasets, the accuracy of the two methods is generally comparable, but the aRVM solution is considerably sparser.

## 6.5 Discussion

### 6.5.1 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 (3.33), 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 $\mathbf{\Sigma}$ is $M \times M$, where $M$ is the number of active 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 $O(N^2M)$.

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 [Schmolck and Everson, 2007]). 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.

### 6.5.2 Probabilistic Kernel Interpretation

As noted in Chapter 3, a Gaussian process (GP) [Rasmussen and Williams, 2006] 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 Gaussian process 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 [Tipping, 2001], 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 (3.43), therefore the sparse linear model is a special case of GP, with covariance function given by:

$$K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sum_{i=1}^{M} \alpha_i^{-1} \phi_i(\boldsymbol{x}_1)\phi_i(\boldsymbol{x}_2). \tag{6.40}$$

This covariance function depends directly to 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:

$$p(\boldsymbol{x}) = \sum_{i=1}^{M} p(i)p(\boldsymbol{x}|i), \tag{6.41}$$

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

$$p(\boldsymbol{x}|i) \propto \phi_i(\boldsymbol{x}) \tag{6.42}$$

and $p(i) \propto \alpha_i^{-1}$. Then the covariance function $K(\boldsymbol{x}_1, \boldsymbol{x}_2)$ of (6.40) is proportional to the probability to generate two inputs $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ from the same component $i_1 = i_2$ of the mixture model:

$$K(\boldsymbol{x}_1, \boldsymbol{x}_2) \propto p(\boldsymbol{x}_1, \boldsymbol{x}_2|i_1 = i_2) = \sum_{i=1}^{M} p(i)p(\boldsymbol{x}_1|i)p(\boldsymbol{x}_2|i). \tag{6.43}$$

Such probabilistic interpretation of the kernel function has been used to construct kernels in [Haussler, 1999]. 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. However, it is important to use the sparsity prior [Schmolck and Everson, 2007] in order to avoid overfitting.

## 6.6 Statistical Models for Analysis of Functional Neuroimages

In this section we apply the proposed adaptive kernel learning methodology to detect activations in *functional neuroimages*, which are brain images whose intensity measures the neural activity of the brain [Friston et al., 2007]. Although neural activity cannot be directly measured, there are techniques to measure it indirectly. PET imaging measures the blood flow and Functional MRI the BOLD (Blood Oxygenation Level Dependent) signal in a brain area, which are both proportional to the neural activity in that area. Thus, brain regions which are activated can be identified by finding regions in a PET or fMRI image where the blood flow or BOLD signal is elevated in comparison to a baseline or control state. The baseline is the measurement of blood flow or BOLD signal when the brain does not perform any task. Similarly, brain regions which have lower activity than the baseline state are said to be inhibited.

A brain activation study aims in recording brain activity during performing a specific task, such as cognition, memory, sensory stimulation and motor activity or studying the effects of diseases or drugs to normal brain activity. A typical activation study consists of four parts: experimental design, image acquisition, preprocessing and analysis.

The *experimental design* is the step where all the parameters of the experiment are defined. There are mainly two types of designs: 1) block design and 2) event-related design. In block related design, the experiment consists of alternating periods in which a specific event or task is performed and periods of rest. Neuroimages are obtained continuously, and can be split into two sets, depending on whether the task or event is performed at the time or not. Event- related activation studies consist of a brief stimulus performed only once. Furthermore, other parameters, such as the subjects that will be tested and the machinery that will be used are determined. Usually, the signal to noise ratio of the obtained images is very poor and in order to get robust results many images are required.

In the *image acquisition* step several scans of the brain of each subject are obtained and in the *preprocessing* step the data are prepared for analysis. The main objective of this step is to eliminate the differences in the images that are caused by extraneous factors. For example, the position of the head of the subjects cannot be perfectly repeated among scans, so an image processing technique (image registration) is used to correct any misalignments. If images are obtained from more than one subjects, differences in the anatomy of the subject's brain should also be eliminated before the image analysis step. In this case, piecewise linear transformations based on a brain atlas are used to bring the brain images into anatomical alignment in a standard coordinate system. Then, the images are usually spatially smoothed by a low-pass filter.

The final step is *image analysis*. The aim of neuroimaging analysis are: i) characterization of the spatio-temporal activation pattern induced in the brain by the stimulus, and ii) estimation of data model parameters that can be used to accurately predict the values of experimental design parameters (e.g. state labels) given the brain scans not previously analyzed.

There are several important factors that make it difficult to relate specific changes in brain activity to the experimental conditions being studied. First, the brain is always active therefore the experiment must be designed carefully to isolate the effect of the stimulus. Furthermore, the degree of activation with respect to the baseline state may be very slight and difficult to detect. Another problem is that the images often suffer from poor quality (low resolution, blur and noise). In experiments that involve many subjects additional errors may be introduced because of anatomical and functional differences among subjects. Finally, it is difficult to validate the results of the analysis, because very little prior knowledge is available about human brain activity.

These challenges have inspired the development of several image processing and statistical tools to detect and establish statistical significance of studies. The predominant approach [Friston et al., 2007] is based on the t-test from statistics and uses pixel-wise comparisons between images of the control and test states of the brain to detect the local changes in activity. More recent methods, which have gained lower acceptance so far, are based on pairwise pixel correlations. Recently, Bayesian techniques have been proposed that model spatial correlations of the activation signal. More specifically, [Penny et al., 2005] models spatial correlations using a Bayesian prior based on the Laplacian operator and [Flandin and Penny, 2007] uses a Wavelet-based prior. Furthermore, in [Lukic et al., 2007] kernel methods, such as the RVM of Section 3.5, have been used to account for spatial correlations of the activation signal.

In simple fMRI studies two sets of volumes are acquired during an experiment: i) *baseline* volumes that are obtained when the subject rests and ii) *activated* volumes that are obtained when the subject is exposed to the examined stimulus. The problem of interest is to statistically compare activated and baseline volumes to find activated regions in the brain, i.e. regions where neural activity significantly changes when the subject is exposed to the studied stimulus. More complicated fMRI studies may study the effects

of many simultaneous stimuli.

### 6.6.1 The t-test

Typical statistical analysis of functional neuroimages is performed separately for each voxel [Friston et al., 2007], without modeling any correlations between neighboring voxels. For this reason, smoothing is commonly performed as a preprocessing step. A common assumption is that the intensity of each voxel is generated by the addition of i) a constant baseline value, ii) possibly an activation and iii) some noise source. We denote with $X^b_{in}$ and $X^a_{in}$ the intensities in the $i$-th voxel of the $n$-th volume that has been acquired at the baseline and activation states respectively. Then, we can write

$$X^b_{in} = B_i + \epsilon^n_i, \tag{6.44}$$

$$X^a_{in} = B_i + A_i + \epsilon^n_i, \tag{6.45}$$

where $B_i$ and $A_i$ are the unknown intensities in the $i$-th voxel of the baseline and activation respectively and $\epsilon^n_i$ is the noise source. The noise is typically assumed zero-mean Gaussian distributed, with different variance $\sigma^2_i$ at each voxel, $p(\epsilon^n_i) = \mathrm{N}(\epsilon^n_i|0, \sigma^2_i)$. Based on this assumption, we can compute the likelihood that a voxel has been generated from this model:

$$p(X|A_i, B_i, \sigma_i) = \prod_i \left( \prod_{n=1}^{N^b} \mathrm{N}(X^b_{in}|B_i, \sigma^2_i) \prod_{n=1}^{N^a} \mathrm{N}(X^a_{in}|B_i + A_i, \sigma^2_i) \right), \tag{6.46}$$

where $N^b$ and $N^a$ are the numbers of volumes acquired in the baseline and activation states respectively.

The maximum likelihood estimates of the baseline $\hat{B}_i$, the activation $\hat{A}_i$ and the variances $\hat{\sigma}^2_i$ are given by:

$$\hat{B}_i = \frac{1}{N^b} \sum_{n=1}^{N^b} X^b_{in}, \tag{6.47}$$

$$\hat{A}_i = \frac{1}{N^a} \sum_{n=1}^{N^b} X^a_{in} - \hat{B}_i, \tag{6.48}$$

$$\hat{\sigma}^2_i = \frac{1}{N-1} \left( \sum_{n \in N^b} (X^b_{in} - \hat{B}_i)^2 + \sum_{n \in N^a} (X^a_{in} - \hat{B}_i - \hat{A}_i)^2 \right), \tag{6.49}$$

where $N = N^b + N^a$ is the total number of acquired volumes in both baseline and activation states. Furthermore, it can be shown that the baseline and activation estimations $\hat{B}_i$ and $\hat{A}_i$ are Gaussian distributed, with $\hat{B}_i \sim \mathrm{N}(\hat{B}_i|B_i, \frac{\sigma^2_i}{N^b})$ and $\hat{A}_i \sim \mathrm{N}(\hat{A}_i|A_i, \frac{\sigma^2_i}{N^a})$ and that $V \equiv (N-1)\hat{\sigma}^2_i/\sigma^2_i$ follows a $\chi^2$ distribution.

Then, the quantity

$$t = \frac{\hat{A}_i}{\hat{\sigma}_i/\sqrt{N}} \sim t_{N-1}, \tag{6.50}$$

follows a Student's t distribution with $N-1$ degrees of freedom. Therefore, in order to test the hypothesis that there is no activation in the $i$-th voxel ($A_i = 0$), we use the t-test

$$t \leq T, \tag{6.51}$$

where $T$ is a threshold that is defined by selecting the probability $P_{FA}$ of incorrectly identifying a pixel as activated (typically $P_{FA} = 0.05$).

### 6.6.2 Application of the Sparse Linear Model with Kernel Learning to Detect fMRI Activations

In this chapter, we apply the sparse Bayesian linear model of Chapter 6 to detect activations in functional neuroimages. More specifically, we use the sparse linear model to estimate the unknown activation signal $A_i$. Sparsity is desirable because typically only a small proportion of the voxels are activated. Because this approach models spatial correlations of the activation signal, statistical inference is not performed voxel-wise. Instead, after computing the voxel estimates $\hat{A}_i$ from (6.48) and their variance $\hat{\sigma}_i^2$ from (6.49), we refine this estimation using the sparse linear model.

The main advantage of learning the kernel parameters of the sparse linear model is that activations of different sizes and shapes can be simultaneously detected. More specifically, spatial correlations are not assumed to be identical in all brain regions. For this reason, this method allows simultaneous detection of activations with small size and high intensity, or large size and small intensity.

When detecting activations in fMRI, it is desirable to know the probability of incorrectly detecting an activation. Using the sparsity prior (6.1) we can adjust this probability by setting an appropriate value for the sparsity parameter $c$. Assuming that we train the sparse linear model using the incremental algorithm of Section 3.4.3, activation will be detected only if a basis function is added to the model, which happens using (6.13) when

$$q_i^2 > (2c+1)s_i, \tag{6.52}$$

$$(\boldsymbol{\phi}_i^T \boldsymbol{C}_{-i}^{-1} \hat{\boldsymbol{t}})^2 > (2c+1)\boldsymbol{\phi}_i^T \boldsymbol{C}_{-i}^{-1} \boldsymbol{\phi}_i, \tag{6.53}$$

$$\beta^2 (\boldsymbol{\phi}_i^T \hat{\boldsymbol{t}})^2 > (2c+1)\beta \boldsymbol{\phi}_i^T \boldsymbol{\phi}_i, \tag{6.54}$$

$$T = \frac{(\boldsymbol{\phi}_i^T \hat{\boldsymbol{t}})^2}{\beta^{-1} \boldsymbol{\phi}_i^T \boldsymbol{\phi}_i} > (2c+1), \tag{6.55}$$

where we assume an initially empty model, thus $\boldsymbol{C}_{-i} = \beta^{-1}\boldsymbol{I}$. Assuming that there is no activation in the observed signal we have:

$$\boldsymbol{t} \sim \mathrm{N}(\boldsymbol{t}|\boldsymbol{0}, \sigma^2 \boldsymbol{I}), \tag{6.56}$$

$$\boldsymbol{\phi}_i^T \boldsymbol{t} \sim \mathrm{N}(\boldsymbol{\phi}_i^T \boldsymbol{t} | 0, \sigma^2 \boldsymbol{\phi}_i^T \boldsymbol{\phi}_i), \tag{6.57}$$

$$\frac{\boldsymbol{\phi}_i^T \boldsymbol{t}}{\sigma \sqrt{\boldsymbol{\phi}_i^T \boldsymbol{\phi}_i}} \sim \mathrm{N}(\boldsymbol{\phi}_i^T \boldsymbol{t} | 0, 1), \tag{6.58}$$

$$\frac{(\boldsymbol{\phi}_i^T \boldsymbol{t})^2}{\sigma^2 \boldsymbol{\phi}_i^T \boldsymbol{\phi}_i} \sim \chi^2. \tag{6.59}$$

We assume that the noise estimate $\beta^{-1} \approx \sigma^2$ is accurate, therefore $T$ follows a $\chi^2$ distribution

$$T = \frac{(\boldsymbol{\phi}_i^T \hat{\boldsymbol{t}})^2}{\beta^{-1} \boldsymbol{\phi}_i^T \boldsymbol{\phi}_i} \sim \chi^2. \tag{6.60}$$

Based on this result we can compute the probability of incorrectly detecting activation as a function of the sparsity parameter $c$. For example $P_{FA} = P(T > 2c + 1) = 0.05$ gives $c = 1.42$ and $P_{FA} = 0.01$ gives $c = 2.82$.

### 6.6.3 Experimental Setup

It is widely accepted that evaluation of the detection performance of statistical analysis methods is a difficult task, because in fMRI datasets the actual activation signal is generally unknown. For this reason, the evaluation of statistical analysis methods is typically performed with simulated data (phantoms). However, in order for the evaluation to be realistic the simulated data must have similar statistical properties with real fMRI data.

Typically, in order to generate simulated data, we first generate a baseline volume and then add several instances of noise based on a stochastic noise model. Finally, activations are added to the volumes that are assumed to belong to the active state. Here, in order to generate data that have similar statistical properties to real fMRI data, we do not generate artificial realization of the noise. Instead, we use real baseline volumes that have been obtained using an fMRI scanner with the subject at the resting state. Then, we add known artificial activations to some of the baseline volumes, in order to generate the activated ones.

The activations that we add are Gaussian-shaped, given by

$$A(x_i, y_i) = V \exp\left( -\frac{1}{\sigma_x^2}(x_i - x_0)^2) - \frac{1}{\sigma_y^2}(y_i - y_0)^2 \right), \tag{6.61}$$

where $V$ is the activation intensity, $(x_0, y_0)$ is the center of the activation and $\sigma_x, \sigma_y$ are the variances at each direction.

### 6.6.4 Numerical Results

Next, we compare three methods for detecting activations in fMRI signals using the simulated data described in the previous section. The first method, which we denote with SPM, is based on estimating the activation signal using (6.48) and then denoting a voxel as activated or not based on the t-statistic of (6.50). For this method we use

the popular software package statistical parametric mapping (SPM[2]). Instead of using the t-test of (6.50), the other two methods that we compare attempt to model spatial correlations of the activation signal. They are based on training a sparse Bayesian linear model using the activation estimates of (6.48). More specifically, one method is denoted with RVM and uses the RVM with Gaussian kernels [Lukic et al., 2007]. The other method is denoted with aRVM and again uses an RVM with Gaussian kernels but also uses the methodology presented in Chapter 6 to learn the location and scale parameters of the Gaussian kernels. Furthermore, we assume separate variance parameter for each direction of the kernel, in order to allow for elliptical shaped activations.

The increased flexibility of the aRVM method is demonstrated in Fig. 6.6, where the estimated activations by each method are shown. We can see in Fig. 6.6b that the RVM method with a small kernel size results in a large number of falsely detected activations. On the other hand, in Fig. 6.6c we observe that using a larger kernel, we fail to estimate the top right part of the activation, because it is not large enough. In Fig. 6.6d we see that by learning parameters of the kernel we can detect all the regions of the activation signal, while very few regions are incorrectly identified.

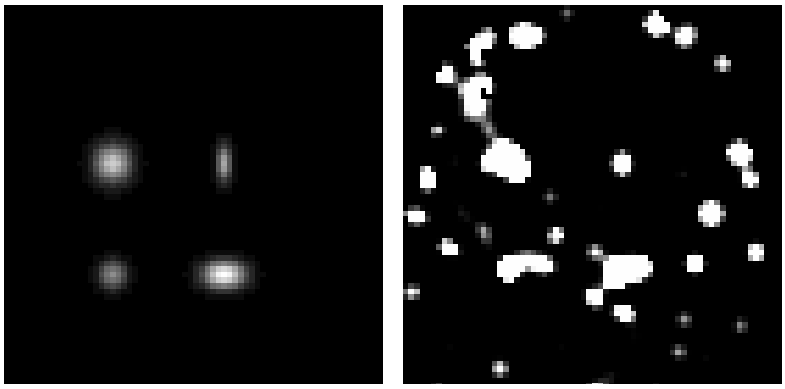
We have also performed a more detailed evaluation of the detection performance of the methods. For this purpose, we added 40 Gaussian artificial activations given by (6.61), at the baseline image and used each of the three methods to estimate them. The activations were added in distant locations in order to make detection of each activation independent to the others. Then ROC curves were generated by varying the probability threshold over which voxels are detected as activated. The obtained ROC curves are shown in Fig. 6.7, for probability of false alarm less than 0.1. In this figure we observe that aRVM has the overall best detection performance. However, because of the sparsity enforcing prior, it does not provide probabilities of false alarm larger than 0.05.

## 6.7 Conclusions

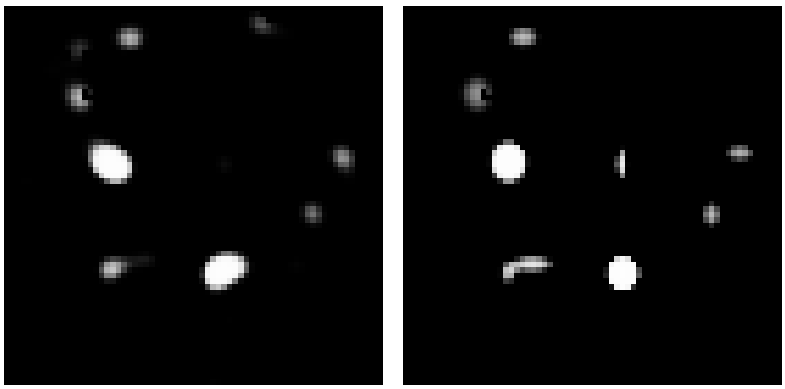
We have presented a learning methodology according to which the parameters of the basis functions of sparse linear models can be determined automatically. More specifically, we assume that the basis functions of this model are kernels and, unlike most kernel methods, for each kernel we learn distinct 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 avantages. 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

[2]SPM can be obtained from `http://www.fil.ion.ucl.ac.uk/spm/software/`.



(a) (b)



(c) (d)

Figure 6.6: (a) Example activation pattern and its estimation with (b) RVM with small kernel ($\sigma^2 = 2.5$) (c) RVM with large kernel ($\sigma^2 = 4$) (d) RVM with adaptive kernel learning.

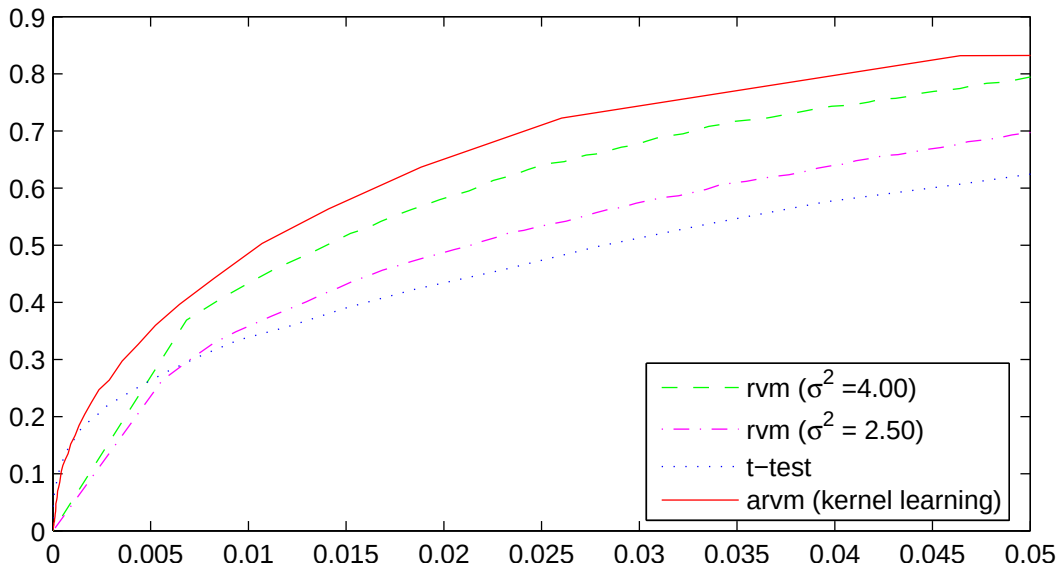



Figure 6.7: ROC curves that summarize the performance of t-test, RVM and aRVM methods.

demonstrated in Section 6.4 where we considered regression of a function with varying frequencies and classification of data drawn from a mixture of distributions with very different characteristics. Because of the sparsity prior that we use, the obtained models are typically much sparser than the models obtained using the typical RVM. Furthermore, we used the proposed kernel learning algorithm to model spatial correlations of the activation signal in functional neuroimages. The proposed method, unlike previous ones, can simultaneously detect activations that are small in size but have high intensity and activations that have low intensity but large size.

In this method, we have assumed that the basis functions are Gaussian kernels and we learn the location and their width parameters. However, the proposed methodology can be also used for selecting other types of bases. Furthermore, it is possible to simultaneously use several types of kernel functions and the appropriate kernel should be automatically selected, in a similar spirit as in Chapter 4.

# Chapter 7

# Local Feature Selection with Adaptive Kernel Learning: Application to the Analysis of DNA Microarray Datasets



## 7.1 Introduction

In several regression and classification problems the examples in the training set contain a very large number of features. For example, in biological microarray datasets, the number of features may be up to $100,000$. In such cases, it is often useful to preselect some of the features, in a process known as *feature selection*, and then build regression or classification models using only the selected features. This approach has several advantages. First, the computational cost of training a model is usually greatly reduced, because of the reduction in the number of features. More importantly, removing irrelevant features improves the generalization performance. This happens because irrelevant features only add noise to the observations, and many methods tend to overfit this noise. Finally, in several applications it is interesting to know which features are relevant to make decisions, for example in

biological microarray experiments it is important to identify the gene expressions that are related with some condition.

An introduction to feature selection methods can be found in [Guyon and Elisseeff, 2003]. Feature selection methods can be divided in two broad categories. First, there are methods that are based on variable ranking, i.e. they identify the relevance of each feature independently. Then there are methods based on subset selection that assess the discriminative capability of subsets of features. The second category is generally more powerful, because it can identify correlated features; for instance two features that independently seem irrelevant to the problem, might turn out to have significant discriminative capability when examined together. However, subset selection methods are more computationally demanding and for this reason feature selection in biological experiments with microarray datasets is usually performed with variable ranking methods.

In this chapter we propose a *local* feature selection method that is based on *learning kernel parameters* of a sparse Bayesian linear model using the approach of Chapter 6. More specifically, a sparse Bayesian liner model is assumed, whose basis functions are Gaussian anisotropic kernels. Local feature selection is then achieved by estimating for each kernel the separate scaling factor (width) parameter that corresponds to each feature. Because we learn different values for the scaling factors of each kernel, feature selection is *local*. This means that different features are assumed to be relevant at different regions of the input space. In order to eliminate irrelevant features, we assume a sparsity enforcing prior on the scaling factors of the kernels.

Furthermore, we treat the problem of analyzing DNA microarray datasets. We consider some typical feature preselection approaches in order to eliminate irrelevant features and reduce the dimensionality of the dataset to manageable size. Then we apply i) the typical RVM, ii) the RVM with adaptive kernel learning classifier of Chapter 6 and iii) the proposed RVM with simultaneous feature selection. Experimental results demonstrate that the adaptive kernel learning algorithm of Chapter 6 exhibits superior classification performance compared to the commonly used RVM model. Furthermore, the proposed local feature selection approach has similar performance and may be useful in identifying which genes are significant for the classification task.

## 7.2 Feature Selection based on Linear Models

Next we present an overview of two feature selection approaches based on variable ranking, namely *recursive feature elimination* (RFE) and *automatic relevance determination* (ARD). These approaches are based on training linear models on the available data. However, ARD builds linear models that are sparse in the number of features, while RFE builds linear models that may be sparse in the number of input points.

### 7.2.1 Recursive Feature Elimination

A common approach in feature selection is *recursive feature elimination* (RFE) [Kohavi and John, 1997]. This approach initially builds an appropriate classification or regression model using all the available features, and uses the produced model to assess the significance of each feature. The least significant feature is then eliminated and this process is repeated until the desired number of features are obtained.

More specifically, assume that we are given a two-class classification training set $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$, where $y_n \in \{-1, 1\}$ determines the category that the example $\boldsymbol{x}_n \in \mathbb{R}^d$ belongs to. RFE is commonly used with linear classifiers, which make decisions based on a decision function of the form

$$D(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b, \tag{7.1}$$

where $\boldsymbol{x} = (x_1, \ldots, x_d)^T$ and $\boldsymbol{w} = (w_1, \ldots, w_d)^T$ is the vector of weights. Using a linear classifier, a straightforward method to compute the *significance* $s_i$ of the $i$-th feature is:

$$s_i = w_i^2. \tag{7.2}$$

In recent works, the popular SVM linear classifiers have been used [Guyon et al., 2002]. In such case, the parameters $\boldsymbol{w}$ of the final model are given by

$$\boldsymbol{w} = \sum_{n \in SV} \alpha_n y_n \boldsymbol{x}_n, \tag{7.3}$$

where the parameters $\alpha_n$ are estimated during SVM training and $SV$ is the set of support vectors. In the method proposed here, we estimate the parameters $\alpha_n$ using an RVM classifier. This classifier is very similar but it has no parameters to select, unlike the SVM that needs the a priori specification of the soft margin parameter $C$.

In datasets that contain a very large number of features, the above sequential elimination approach may be very computationally demanding. In order to reduce the computations, we can eliminate more than one features at each iteration. Especially in the first few iterations, irrelevant features should be easily identified, therefore we can begin by eliminating a large number of features and at subsequent iterations reduce the number of features eliminated at each step. In [Ding and Wilkins, 2006] it is suggested to eliminate $\frac{1}{i+1}$ features at the $i$-th iteration.

### 7.2.2 Automatic Relevance Determination

A different approach to feature selection is based on the Bayesian framework using an appropriate prior distribution and it is known as *automatic relevance determination* [Neal, 1996]. This approach employes a special type of the Sparse Bayesian Linear Regression model described in Section 3.4 that does not use a kernel function. The output of the

linear model for input $\boldsymbol{x} = (x_1, \ldots, x_d)^T$ is given by:

$$y(\boldsymbol{x}) = \sum_{i=1}^{d} w_i x_i = \boldsymbol{w}^T \boldsymbol{x}, \tag{7.4}$$

and a prior is assumed for the weights $w$:

$$p(\boldsymbol{w}) = \prod_{i=1}^{d} \mathrm{N}(w_i | 0, \alpha_i^{-1}). \tag{7.5}$$

As explained in Section 3.4, this prior favors sparse the estimations for the weights, meaning that most of the weights are set to zero. As a consequence, the features $x_i$ that are associated with zero-valued weights are automatically eliminated.

## 7.3 Adaptive Kernel Learning for Feature Selection

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, Krishnapuram et al. [2004] suggest the JCFO classification method 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 $\theta_i$ is estimated for each feature $x_i$, which measures the significance of that feature. For example, Gaussian kernels can be used, if they are parameterized as:

$$\boldsymbol{K}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}_n) = \exp\left[ -\sum_{i=1}^{d} \theta_i (x_i - x_{ni})^2 \right], \tag{7.6}$$

Then a Laplacian sparsity prior is enforced on the scaling factors $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_d)^T$ in order to eliminate irrelevant features.

In this section we propose a method to incorporate a feature selection mechanism in the adaptive kernel learning approach for the RVM (aRVM) proposed in Chapter 6. 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 of Chapter 6, 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 Section 7.4.1.

### 7.3.1 A Bayesian Model for Feature Selection

We consider the sparse Bayesian linear model of Chapter 6:

$$y(\boldsymbol{x}) = \sum_{n=1}^{M} w_n \phi(\boldsymbol{x}; \boldsymbol{\theta}_n), \tag{7.7}$$

where $\boldsymbol{w} = (w_1, \ldots, w_M)^T$ is the weight vector and $\phi(\boldsymbol{x}; \boldsymbol{\theta}_n)$ is the $n$-th basis function whose parameters are $\boldsymbol{\theta}_n$. In order to obtain sparsity, we assume a zero mean Gaussian prior for the weights $\boldsymbol{w}$, with separate variance parameter for each weight $w_n$:

$$p(\boldsymbol{w}) = \mathrm{N}(\boldsymbol{w}|0, \boldsymbol{A}^{-1}), \tag{7.8}$$

where $\boldsymbol{A} = \mathrm{diag}\{\boldsymbol{\alpha}\}$ and $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_M)^T$, and a prior distribution may be assumed on $\boldsymbol{\alpha}$, following the approach of Section 3.4.

In (7.7) we assumed 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 such that it incorporates the scaling factors. Here, we consider anisotropic Gaussian kernel functions, which have a separate precision parameter $h_{ni}$ for each feature $i$:

$$\phi(\boldsymbol{x}; \boldsymbol{m}_n, \boldsymbol{h}_n) = \exp\left[ -\sum_{i=1}^{d} (h_{ni})^2 (x_i - m_{ni})^2 \right], \tag{7.9}$$

where $\boldsymbol{h}_n = (h_1, \ldots, h_d)^T$ and $\boldsymbol{m}_n = (m_1, \ldots, m_d)^T$. Estimation of the parameters $\boldsymbol{\theta}_n = (\boldsymbol{m}_n, \boldsymbol{h}_n)^T$ can be performed using the method proposed in Chapter 6.

We 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. Therefore, elimination of irrelevant features can be motivated by assuming a prior distribution for the scaling factors $\boldsymbol{h} = (\boldsymbol{h}_1^T, \ldots, \boldsymbol{h}_M^T)^T$ that enforces sparsity. The distribution that we use is the Student's t distribution, which is known to give sparse solutions for few degrees of freedom, see Section 3.4.4. Furthermore, as mentioned in previous chapters, a Student's t distributed random variable is equivalent to a Gaussian distributed random variable whose precision parameter is assumed Gamma distributed. Therefore, we can write:

$$p(\boldsymbol{h}|\boldsymbol{\delta}) = \mathrm{N}(\boldsymbol{h}|\boldsymbol{0}, \boldsymbol{\Delta}^{-1}), \tag{7.10}$$

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

$$p(\boldsymbol{\delta}) = \prod_{n=1}^{M} \prod_{i=1}^{d} \text{Gamma}(\delta_{ni}|a, b), \tag{7.11}$$

where we set $a = b = 0$ that define an uninformative Gamma distribution.

### 7.3.2 Parameter Estimation

The learning method is similar to the adaptive kernel learning algorithm of Chapter 6. It incrementally adds basis functions to an initially empty model and at the same time it assigns appropriate values to their parameters. Estimation of these parameters is performed using a numerical optimization method, which is greatly assisted by the availability of the parameter derivatives. The only difference of the proposed model that incorporates feature selection is that the basis function parameters $\boldsymbol{h}$ are treated as random variables. This is necessary since a prior distribution is assigned on them in order to encode sparsity. However, exact Bayesian inference is not possible and we will attempt to obtain MAP estimates.

In order to obtain MAP estimates, we need to maximize the posterior distribution of $\boldsymbol{h}$, or equivalently its logarithm that can be decomposed in two terms. The first is a likelihood term and is identical to the case of Chapter 6, given by (7.12). The second term comes from the newly defined prior on $\boldsymbol{h}$ and is $-\frac{1}{2}\boldsymbol{h}^T\boldsymbol{\Delta}\boldsymbol{h}$. Therefore, we now want to maximize $L^{fs}$, which is given by

$$L^{fs} = L^s - \frac{1}{2}\boldsymbol{h}^T\boldsymbol{\Delta}\boldsymbol{h}. \tag{7.12}$$

This optimization is performed using a general purpose optimization technique, such as the quasi-Newton BFGS method. The required gradient with respect to the kernel location $\boldsymbol{m}_n$ is identical to the case of Chapter 6 and is given by (6.18). However, the gradient with respect to the scaling factors $\boldsymbol{h}_n$ is given by adding the corresponding prior term to (6.18):

$$\frac{\partial L^{fs}}{\partial h_{ni}} = \frac{\partial L^s}{\partial h_{ni}} - \delta_{ni} h_{ni}. \tag{7.13}$$

Furthermore, we need to consider estimation of the parameters $\boldsymbol{\delta}$ that define the precision of $\boldsymbol{h}$. By setting $a = b = 0$ in (7.11), we assume an uninformative prior distribution for them. Then, maximization of the likelihood with respect to $\boldsymbol{\delta}$ gives

$$\delta_{ni} = \frac{1}{h_{ni}^2}. \tag{7.14}$$

**Algorithm 2** 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 $\beta$.
4. Update hyperparameters $\boldsymbol{\delta}$ using (7.14).
5. Remove from the model any unnecessary active basis functions.
6. Repeat steps 1 to 5 until convergence.

### 7.3.3 The Proposed Algorithm

The proposed learning algorithm, see Algorithm 2, is based on the incremental adaptive kernel learning algorithm of Chapter 6, but it also updates the parameters $\boldsymbol{\delta}$ that have been introduced to facilitate feature selection. Initially an empty model is considered. Then, basis functions are iteratively added to the model until convergence. Basis functions are added in a way that the marginal likelihood of the model is increased. The main differences from the adaptive kernel learning algorithm of Chapter 6 are:

1. An additional term is added to the derivative of the precision parameters in order to achieve feature selection
2. The existence of parameters $\delta_{ni}$, $n = 1, \ldots, M$, $i = 1, \ldots, d$ that measure the significance of feature $j$ for kernel $i$ and are updated at each iteration.

## 7.4 Numerical Experiments

### 7.4.1 Artificial Example

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

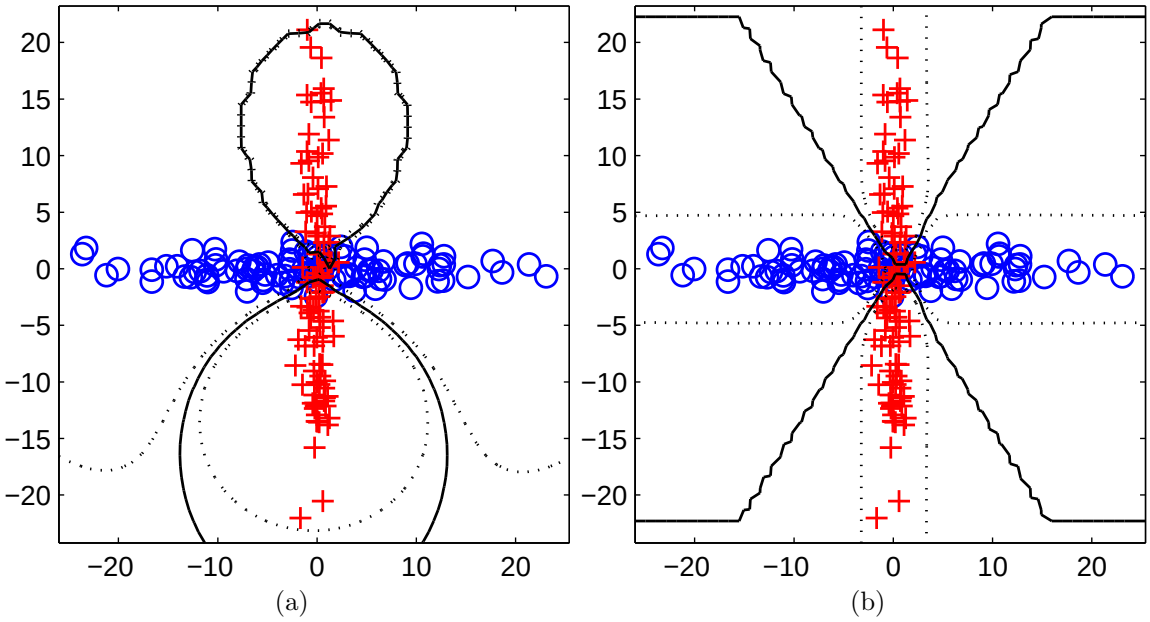



Figure 7.1: Example aRVM classifiers (a) without feature selection (b) with feature selection. Solid lines show the decision boundary and dotted lines show the areas where the probability of misclassification is 0.25.

Fig. 7.1 we show the estimated models using i) the RVM with adaptive kernel learning algorithm of Chapter 6 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.

### 7.4.2 Evaluation on Common Benchmark Datasets

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 Chapter 6. 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*, $MSE = \sum_n (t_n - \hat{y}_n)^2/N$, 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 proposed in Chapter 6 (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 7.1 and Table 7.2 respectively It can be observed that the proposed approach, which incorporates feature selection, provides improved performance compared to both the typical RVM model and the aRVM method of Chapter 6.

Table 7.1: Comparison on regression datasets.

| | RVM | | aRVM | | aRVM$^d$ | |
|---|---|---|---|---|---|---|
| Dataset | error | RVs | error | RVs | error | RVs |
| computer | 30004 | 140.5 | 22379 | 5.0 | 4089 | 13.6 |
| Boston | 12.48 | 69.5 | 11.53 | 13.27 | 13.66 | 17.5 |
| concrete | 44.204 | 140.2 | 34.515 | 9.10 | 28.868 | 42.3 |

Table 7.2: Comparison on classification datasets.

| | RVM | | aRVM | | aRVM$^d$ | |
|---|---|---|---|---|---|---|
| Dataset | error | RVs | error | RVs | error | RVs |
| banana | 0.1092 | 12.1 | 0.1126 | 6.3 | 0.0994 | 4.4 |
| titanic | 0.2292 | 31.0 | 0.2270 | 2.0 | 0.2254 | 4.0 |
| image | 0.0390 | 34.6 | 0.0387 | 6.9 | 0.0342 | 21.3 |
| breast | 0.2818 | 9.6 | 0.2844 | 4.4 | 0.2629 | 3.0 |
| pima | 0.243 | 27.9 | 0.2303 | 5.6 | 0.2276 | 5.1 |

### 7.4.3 Evaluation on DNA Microarray Datasets

DNA microarray experiments have recently attracted a lot of interest. In these experiments datasets are constructed, which simultaneously describe the expression levels of a very large number of genes of several tissues. Typical experiments involve two subsets of tissues, one of which is associated with some disease and the other is not. The goal of these experiments is not only to build classifiers with good generalization properties based on these datasets, but it is also important to identify which are the important features for discriminating the categories. Usually, analysis of the obtained datasets require special treatment, because they contain an extremely large number of features (up to 100,000 features).

Several approaches have been used in the literature. Recursive feature elimination with support vector machines has been proposed in [Guyon et al., 2002]. Also, in [Cawley and Talbot, 2006] sparse logistic regression has been used to identify significant features. In [Li et al., 2006] a relevance vector machine model that implements automatic relevance determination has been proposed, while a Gaussian process based classifier has been used in [Chu et al., 2005]. These approaches build classifiers directly on the feature space. Instead in [Krishnapuram et al., 2004] the JCFO classification method is presented that uses polynomial kernels and jointly identifies the optimal classifier and the relevant features.

In this section we evaluate the proposed RVM-based feature selection method on the task of classification of DNA microarray datasets. In this context, we performed several experiments using datasets that have been previously studied in the literature. The first dataset (Leukemia) contains 72 examples with $7,129$ features that correspond to

Table 7.3: Average Classification Error after feature selection with ARD.

| Method<br>Number of selected features | Colon<br>6 | Leukemia<br>5 | Prostate<br>6 | AML Prognosis<br>7 |
|---|---|---|---|---|
| RVM (no kernel) | 0.0322581 | 0.0138889 | 0.0 | 0.0185185 |
| RVM (linear kernel) | 0.16129 | 0.180556 | 0.0196078 | 0.0925926 |
| RVM (Gaussian kernel ) | 0.225806 | 0.0 | 0.00980392 | 0.166667 |
| aRVM | 0.016129 | 0.0138889 | 0.0 | 0.0 |
| aRVM$^d$ | 0.0645161 | 0.0277778 | 0.0196078 | 0.0555556 |

expression levels of genes from 47 patients with acute myeloid leukemia (AML) and 25 patients with acute lymphoblastic leukemia (ALL). The second dataset (Colon) contains 62 examples with $2,000$ features that correspond to genes from 40 tumor and 22 normal colon tissues. The third dataset (Prostate) contains 102 examples with $12,533$ features that correspond to genes from 52 tumor tissues and 50 normal tissues. Finally the last dataset (AML Prognosis), contains 54 examples with 12625 features that correspond to gene expressions from 28 remission and 26 relapse cases of acute myeloid leukemia[1].

In these experiments we compare the classification performance of the following methods i) RVM without kernel $y(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x}$, ii) typical RVM $y(\boldsymbol{x}) = \sum_{i=1}^{M} w_i K(\boldsymbol{x}, \boldsymbol{x}_i)$ with linear kernel $K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \boldsymbol{x}_1^T\boldsymbol{x}_2$, iii) typical RVM with Gaussian kernel $K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \exp(-h^{-2}\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2)$, where the width $h$ was appropriately selected in order to minimize the classification error, iv) adaptive RVM with learning of Gaussian kernel parameters proposed in Chapter 6 (denoted as aRVM) and v) the proposed adaptive RVM with simultaneous feature selection by learning the parameters of anisotropic Gaussian kernels (denoted as aRVM$^d$).

We consider two feature selection strategies. In the first strategy, we initially select 2000 features using the RFE approach and then we use ARD considering only these features. ARD typically selects 5–10 features. Then we train all classifiers using the few selected features and we evaluate their classification performance using leave-one-out cross validation. The results are reported in Table 7.3. We can observe that the aRVM method gives the best results in all datasets. The proposed aRVM$^d$ method, which incorporates feature selection, performs worse than the kernel learning aRVM without the feature selection extension, probably because appropriate features have already been selected by the ARD approach.

In the second strategy, we initially use the RFE feature selection approach to select 20 relevant features. Then we apply the compared methods, using the 20 selected features. The classification performance of each classifier (computed using leave-one-out cross validation) is reported in Table 7.4. In this table we can observe that using RFE for feature

[1]The Leukemia dataset can be obtained at `http://www-genome.wi.mit.edu/mpr/table_AML_ALL_samples.rtf`, the Colon dataset can be obtained at `http://microarray.princeton.edu/oncology/affydata/index.html` and the datasets Prostate and AML Prognosis can be obtained from `http://www.ailab.si/supp/bi-cancer/projections/index.htm`.

Table 7.4: Average Classification Error after selecting 20 features with RFE.

| Method | Colon | Leukemia | Prostate | AML Prognosis |
|---|---|---|---|---|
| RVM (no kernel) | 0.112903 | 0.0694444 | 0.117647 | 0.277778 |
| RVM (linear kernel) | 0.370968 | 0.0694444 | 0.0392157 | 0.277778 |
| RVM (Gaussian kernel) | 0.516129 | 0.0833333 | 0.0882353 | 0.222222 |
| aRVM | 0.180328 | 0.112676 | 0.029703 | 0.207547 |
| aRVM$^d$ | 0.193548 | 0.0694444 | 0.0686275 | 0.240741 |

selection, all the classifiers generally exhibited worse performance compared to when using ARD for feature selection. Also, aRVM exhibited the best performance in two out of four datasets. Furthermore, when using RFE for feature selection, aRVM$^d$ was a close competitor to aRVM, probably because the RFE feature selection approach was relatively inaccurate (compared to ARD).

## 7.5 Conclusions

In this chapter we have presented an approach to incorporate feature selection to the sparse Bayesian linear model, using the adaptive kernel learning approach of Chapter 6. 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. Furthermore, feature selection is performed simultaneously with model estimation, which is expected to lead to improved performance but at higher computational cost.

Furthermore, we have used i) the adaptive kernel learning RVM of Chapter 6 and ii) the proposed adaptive RVM with simultaneous feature selection to perform classification with DNA microarray datasets. Experiments showed that aRVM exhibits excellent classification performance. Although the performance of the proposed approach that incorporates feature selection was inferior, the approach has the advantage of identifying which features (genes) are significant, which is important in such applications.

# Chapter 8

# Conclusions



## 8.1 Concluding Remarks

In this thesis we have studied the sparse Bayesian linear model and its application on regression and classification problems. First, we considered sparse Bayesian regression of images, which presents several computational problems because of the size of typical images. We then used sparse Bayesian image regression on some image processing problems, namely object detection, blind image deconvolution and analysis of functional neuroimages. Furthermore, we studied the problem of selecting parameters of the basis functions, which is commonly performed using the computationally expensive cross-validation technique.

In order to apply the sparse Bayesian linear model for regression of images, in Chapter 4 we proposed an algorithm that is based on operations in the discrete Fourier transform (DFT) domain. The conjugate gradient method was used to efficiently compute the posterior mean, and for the computation of the posterior covariance we considered two simple but rather efficient approximations. Furthermore, we considered a variant of the Relevance Vector Machine (RVM), which we call the *multikernel* RVM and uses simultaneously many types of kernels. Finally, we used the proposed algorithm to detect objects in images and simultaneously find their locations. Experimental results indicate that the proposed method has improved detection performance compared to some common alternatives [Tzikas et al., 2006b, 2007b].

In Chapter 5 we presented a Bayesian approach to the blind image deconvolution (BID) problem [Tzikas et al., 2006a, 2007c,a, a], where the sparse Bayesian linear model was used to obtain smooth PSF estimates with limited support. We used the Student's

t pdf for the noise, in order to achieve robustness to BID model errors and for the local image differences, in order to allow the reconstruction of edges. Because of the complexity of this model, the variational framework was used for approximate Bayesian inference. Several experiments were carried out, to test the proposed methodology. These experiments indicated that the use of the sparse Bayesian linear model to model the PSF is crucial to the success of this approach. Furthermore, the importance of using heavy tailed distributions, such as the Student's t distribution for modelling the BID noise and image local differences is apparent. We have also compared this methodology with a TV-based approach and an alternative Bayesian approach implemented in [Molina et al., 2006]. It is clear that the proposed methodology is always superior to the Gaussian based methodology in [Molina et al., 2006]. As far as TV-based BID is concerned, the proposed method is clearly superior for scenarios with small sized PSFs and low noise.

In Chapter 6 an adaptive kernel learning algorithm has been proposed to learn parameters of the basis functions for the sparse Bayesian linear model [Tzikas et al., 2008a, b]. More specifically, the proposed algorithm learns different parameter values for each kernel and for this reason it is very flexible. We have also imposed a prior distribution that controls the effective number of parameters of the model, in order to force sparse estimations and avoid overfitting the noise. Experimental results on artificial data demonstrate the advantages of the proposed method. We also provide a comparison with the typical RVM on several commonly used regression and classification datasets. Furthermore, the proposed approach has been applied to model spatial correlations of the activation signals in functional neuroimaging. Numerical results with an artificial phantom indicate that, in contrast to previous approaches, the proposed method can simultaneously detect activations that are i) strong but small and ii) large but weak.

In Chapter 7 the adaptive kernel learning method of Chapter 6 was further extended in order to perform local feature selection, simultaneously with model inference. To achieve this behavior, we used anisotropic Gaussian kernels, which assume a separate scaling factor (width) for each feature. Because the proposed approach estimates different values for the scaling factors of each kernel, feature selection is local, i.e. different features are assumed to be significant at different regions of the input space. We have then imposed a sparsity enforcing prior on the scaling factors that results in eliminating features from kernels to which they are irrelevant. We have conducted several experiments with common regression and classification benchmark datasets showing that the performance of the proposed method is improved. Furthermore, we considered the classification task with biological DNA microarray datasets, where feature selection is very important. We have applied two common (and computationally efficient) feature selection approaches, recursive feature elimination (RFE) and automatic relevance determination (ARD) and evaluated both the performance of the adaptive RVM with kernel learning of Chapter 6 and the proposed extension to incorporate local feature selection.

## 8.2 Directions for Future research

In future work it would be interesting to consider more efficient approximations in the DFT-based algorithm of Chapter 4. The main computational difficulty arises in the computation of the diagonal elements of the posterior covariance matrix (4.13). An approximation that is based on the Lanczos process [Chantas et al.] appears to have superior performance and should be examined. Also, a large scale version of the *Expectation Propagation* algorithm [Seeger and Nickisch, 2008] has been used for training sparse linear models on images.

In the adaptive kernel learning methodology of Chapter 6 the basis functions were assumed to be Gaussian kernels and we learn the location and their width parameters. However, the proposed methodology can be also used for selecting other types of bases. Also, it is possible to simultaneously use several types of kernel functions and the appropriate kernel should be automatically selected, in a similar spirit as in Chapter 4. Furthermore, we have used an uninformative (uniform) distribution for sampling the parameters of the basis function (Algorithm 1, step 1a). However, it might be useful to develop methodologies that appropriately select more informative distributions for sampling the basis function parameters. Moreover, instead of obtaining maximum likelihood estimations of the basis function parameters, it would be interesting to treat them as random variables and use approximate Bayesian inference techniques. This approach was followed in order to perform local feature selection in Chapter 7, however the MAP approximation was used, which is very crude. There may be advantages in using sampling-based approximate Bayesian inference methods, such as *Markov Chain Monte Carlo* (MCMC).

The adaptive kernel learning methodology of Chapter 6 could be used as well for the problem of object detection, following an approach similar to Chapter 4. Using this method, it would be interesting to introduce parameters that control the scale and rotation of the target basis functions. Then, using the adaptive kernel learning methodology we would be able to estimate these parameters, in order to adjust for scaling and rotation of the target objects. This is very important in most object detection problems, where the target objects may appear scaled and rotated. It would also be interesting to apply the adaptive kernel learning methodology in real-world regression and classification problems and compare its performance to other methods.

Furthermore, in the blind image deconvolution problem, it would be interesting to consider estimating the width parameter of the kernel function, possibly using the methodology of Chapter 6. Also, it would be interesting to explore the possibility of learning the filters $Q^k$ in a manner analogous to [Welling et al., 2003]. It is also possible to explore extending the *constrained variational* methodology in [Chantas et al., 2007] to BID to avoid using the approximation of the partition function in (5.16). The sparse Bayesian linear model could also be used to model the blurring PSF in the related problem of super resolution [He et al., 2006, Yang et al., 2008], where we want to fuse several low-resolution images of the same scene, in order to obtain a high-resolution image.

It would also be interesting to further examine the proposed local feature selection

approach of Chapter 7 for classification using DNA microarray datasets. In Chapter 7 we have presented experiments evaluating the classification performance of the method. It would be interesting to evaluate the performance of the local feature selection approach in selecting those genes that are significant for the classification task in DNA microarray datasets.

# Bibliography


A. Abu-Naser, N. P. Galatsanos, M. N. Wernick, and D. Shonfeld. Object recognition based on impulse restoration using the expectation-maximization algorithm. *Journal of the Optical Society of America*, 15(9):2327–2340, September 1998.

A. Agarwal and B. Triggs. 3D human pose from silhouettes by relevance vector regression. In *Proceedings of Computer Vision and Pattern Recognition*, volume 2, pages 882–888, 2004.

C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50(1):5–43, January 2003.

J. Bernardo and A. Smith. *Bayesian Theory*. John Wiley and Sons, 1994.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, August 2006.

C. M. Bishop and M. E. Tipping. Variational relevance vector machines. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 46–53, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

Å. Björck. *Numerical Methods for Least Squares Problems*. SIAM, Philadelphia, 1996.

K. Blekas, A. Likas, N. P. Galatsanos, and I. E. Lagaris. A spatially-constrained mixture model for image segmentation. *IEEE Transactions on Neural Networks*, 16:494–498, 2005.

C. Borgelt and R. Kruse. *Graphical Models: Methods for Data Analysis and Mining*. John Wiley & Sons, Inc., New York, NY, USA, 2002.

M. M. Bronstein, A. M. Bronstein, M. Zibulevsky, and Y. Zeevi. Blind image deconvolution of images using optimal sparse representations. *IEEE Transactions on Image Processing*, 14(6):726–736, June 2005.

C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

P. Campisi and K. Egiazarian, editors. *Blind image deconvolution: theory and applications*. CRC press, May 2007.

G. C. Cawley and N. L. C. Talbot. Gene selection in cancer classification using sparse logistic regression with Bayesian regularization. *Bioinformatics*, 22(19):2348–2355, 2006.

T. F. Chan and C. K. Wong. Total variation blind deconvolution. *IEEE Transactions on Image Processing*, 7(3):370–375, March 1998.

T. F. Chan, S. Esedoglu, F. Park, and M. H. Yip. Recent developments in total variation image restoration. In *Handbook of Mathematical Models in Computer Vision*. Springer, 2005a.

T. F. Chan, S. Esedoglu, F. Park, and M. H. Yip. *Image Processing and Analysis - Variational, PDE, wavelet, and stochastic methods*. SIAM, 2005b.

G. Chantas, A. Likas N. P. Galatsanos, and M. Saunders. Variational bayesian image restoration based on a product of t-distributions image prior. *IEEE Transactions on Image Processing*. to appear.

G. Chantas, N. P. Galatsanos, and A. Likas. Bayesian restoration using a new nonstationary edge preserving image prior. *IEEE Transactions on Image Processing*, 15(10): 2987–2997, October 2006.

G. Chantas, N. P. Galatsanos, and A. Likas. Bayesian image restoration based on variational inference and a product of Student-t priors. In *Proceedings of the IEEE International Workshop on MLSP 2007*, Thessaloniki, Greece, August 2007.

L. Chen and K. H. Yap. A soft double regularization approach to parametric blind image deconvolution. *IEEE Transactions on Image Processing*, 14(5):624–633, May 2005.

Q. Chen, M. Defrise, and F. Deconinck. Symmetric phase-only matched filtering of Fourier-Mellin transforms for image registration and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(12):1156–1168, 1994.

W. Chu, Z. Ghahramani, F. Falciani, and D. L. Wild. Biomarker discovery in microarray gene expression data with Gaussian processes. *Bioinformatics*, 21(16):3385–3393, 2005.

Y. Ding and D. Wilkins. Improving the performance of SVM-RFE to select genes in microarray data. *BMC Bioinformatics*, 7(2), 2006.

A. C. Faul and M. E. Tipping. Analysis of sparse Bayesian learning. In *Proceedings of Advances in Neural Information Processing Systems*, pages 383–389. MIT Press, 2002.

R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis, and W. T. Freeman. Removing camera shake from a single photograph. *ACM Transactions on Graphics*, 25(3):787–794, 2006.

M. A. T. Figueiredo. Adaptive sparseness for supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(9):1150–1159, 2003.

G. Flandin and W.D. Penny. Bayesian fMRI data analysis with sparse spatial basis function priors. *NeuroImage*, 34(3):1108–1125, 2007.

K.J. Friston, J. Ashburner, S.J. Kiebel, T.E. Nichols, and W.D. Penny, editors. *Statistical Parametric Mapping: The Analysis of Functional Brain Images*. Academic Press, 2007.

N. P. Galatsanos, V. Z. Mesarovic, R. Molina, A. K. Katsaggelos, and J. Mateos. Hyperparameter estimation in image restoration problems with partially-known blurs. *Optical Engineering*, 41(8):1845–1854, 2002.

M. Girolami and S. Rogers. Hierarchic Bayesian models for kernel learning. In *Proceedings of the 22nd international conference on machine learning*, pages 241–248, New York, NY, USA, 2005. ACM.

I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1–3):389–422, 2002.

D. Haussler. Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, University of California, Santa Cruz, Computer Science Department, July 1999.

Y. He, K.-H. Yap, L. Chen, and L.-P. Chau. Blind super-resolution image reconstruction using a maximum a posteriori estimation. In *Proceedings of IEEE International Conference on Image Processing*, pages 1729–1732, Oct. 2006.

C. C. Holmes and D. G. T. Denison. Bayesian wavelet analysis with a model complexity prior. In José M. Bernardo, James O. Berger, A. P. Dawid, and Adrian F. M. Smith, editors, *Bayesian Statistics 6: Proceedings of the Sixth Valencia International Meeting*. Publisher: Oxford University Press, 1999.

J. L. Horner and P. D. Gianino. Phase-only matched filtering. *Journal of Applied Optics*, 23(6):812–816, 1984.

T. S. Jaakola. *Variational Methods for Inference and Learning in Graphical Models*. PhD thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 1997.

B. D. Jeffs and J. C. Christou. Blind Bayesian restoration of adaptive optics telescope images using generalized Gaussian markov random field models. In D. Bonaccini and R. K. Tyson, editors, *Proceedings of SPIE. Adaptive Optical System Technologies*, volume 3353, pages 1006–1013, 1998.

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

M.I. Jordan, Z. Gharamani, T.S. Jaakkola, and L.K. Saul. An introduction to variational methods for graphical models. In M.I. Jordan, editor, *Learning in Graphical Models*, pages 105–162. Kluwer, 1998.

S. M. Kay. *Fundamentals of Statistical Signal Processing: Estimation Theory*. Prentice Hall, 1997.

R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1–2):273–324, 1997.

B. Krishnapuram, A. J. Hartemink, and M. A. T. Figueiredo. A Bayesian approach to joint feature selection and classifier design. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1105–1111, 2004.

D. Kundur and D. Hatzinakos. Blind image deconvolution. *IEEE Signal Processing Magazine*, 13(3):43–64, 1996a.

D. Kundur and D. Hatzinakos. Blind image deconvolution revisited. *IEEE Signal Processing Magazine*, 13(6):61–63, 1996b.

G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

Y. Li, C. Campbell, and M. Tipping. Bayesian automatic relevance determination algorithms for classifying gene expression data. *Bioinformatics*, 18(10):1332–1339, 2002.

Y. Li, K. K. Lee, S. Walsh, C. Smith, S. Hadingham, K. Sorefan, G. Cawley, and M. W. Bevan. Establishing glucose- and ABA-regulated transcription networks in arabidopsis by microarray analysis and promoter classification using a relevance vector machine. *Genome Research*, January 2006.

A. Likas and N. P. Galatsanos. A variational approach for Bayesian blind image deconvolution. *IEEE Transactions on Signal Processing*, 52(8):2222–2233, 2004.

A. Lukic, M. Wernick, D. Tzikas, X. Chen, A. Likas, N. Galatsanos, Y. Yang, F. Zhao, and S. Strother. Bayesian kernel methods for analysis of functional neuroimages. *IEEE Transactions on Medical Imaging*, 26(12):1613–1622, December 2007.

J. Miskin and D. MacKay. Ensemble learning for blind image separation and deconvolution. In M. Girolami, editor, *Advances in Independent Component Analysis*. Springer-Verlag, 2000.

R. Molina, J. Mateos, and A.K. Katsaggelos. Blind deconvolution using a variational approach to parameter, image, and blur estimation. *IEEE Transactions on Image Processing*, 15(12):3715–3727, 2006.

T. K. Moon. The expectation-maximization algorithm. *Signal Processing Magazine, IEEE*, 13(6):47–60, 1996.

R. M. Neal. *Bayesian Learning for Neural Networks (Lecture Notes in Statistics)*. Springer, 1 edition, August 1996.

R. M. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Proceedings of Learning in Graphical Models*, pages 355–368. Kluwer Academic Publishers, 1998.

R. E. Neapolitan. *Learning Bayesian Networks*. Prentice Hall, April 2003.

C. Nikou, N. P. Galatsanos, and A. Likas. A class-adaptive spatially variant mixture model for image segmentation. *IEEE Transactions on Image Processing*, 16(4):1121–1130, 2007.

G. Parisi. *Statistical Field Theory*. Addison-Wesley, 1988.

W.D. Penny, N. Trujillo-Bareto, and K.J. Friston. Bayesian fMRI time series analysis with spatial priors. *NeuroImage*, 24(2):350–362, 2005.

J. Quiñonero-Candela and L. K. Hansen. Time series prediction based on the relevance vector machine with adaptive kernels. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 985–988, Piscataway, New Jersey, 2002. IEEE.

J. Quiñonero-Candela and C. E. Rasmussen. A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959, 2005.

C. E. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

C. P. Robert and G. Casella. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

A. Schmolck and R. Everson. Smooth relevance vector machine: a smoothness prior extension of the RVM. *Machine Learning*, 68(2):107–135, August 2007.

B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning)*. The MIT Press, December 2001.

M. Seeger and H. Nickisch. Large scale variational inference and experimental design for sparse generalized linear models. Technical report, Max Planck Institute for Biological Cybernetics, 2008.

J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. http://www.cs.cmu.edu/ quake-papers/painless-conjugate-gradient.ps, 1994.

A. J. Smola and B. Schölkopf. A tutorial on support vector regression. Technical report, Statistics and Computing, 1998.

E. Snelson and Z. Ghahramani. Sparse Gaussian processes using pseudo-inputs. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Proceedings of Advances in Neural Information Processing Systems 18*, pages 1257–1264, Cambridge, MA, 2006. MIT Press.

S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7:1531–1565, 2006.

M. E. Tipping. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, 2001.

M. E. Tipping and A. Faul. Fast marginal likelihood maximisation for sparse Bayesian models. In *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, 2003.

M. E. Tipping and N. D. Lawrence. A variational approach to robust Bayesian interpolation. In *Proceedings of Neural Networks for Signal Processing*, pages 229–238. IEEE, 2003.

D. Tzikas, A. Likas, and N. Galatsanos. Variational bayesian sparse kernel-based blind image deconvolution with Student's-t priors. *IEEE Transactions on Image Processing*, a. to appear.

D. Tzikas, A. Likas, and N. Galatsanos. Sparse bayesian modeling with adaptive kernel learning. *IEEE Transactions on Neural Networks*, b. to appear.

D. Tzikas, A. Likas, and N. Galatsanos. Variational Bayesian blind image deconvolution based on a sparse kernel model for the point spread function. In *Proceedings of European Signal Processing Conference*, Florence, Italy, September 2006a.

D. Tzikas, A. Likas, and N. Galatsanos. Large scale multikernel RVM for object detection. In *Hellenic Conference on Artificial Intelligence*, pages 389–399, Heraclion, Crete, Greece, May 2006b.

D. Tzikas, A. Likas, and N. Galatsanos. Bayesian bid based on a kernel model for the point spread function. In *Proceedings of International Conference on Image Processing*, San Antonio, TX, USA, September 2007a.

D. Tzikas, A. Likas, and N. Galatsanos. Large scale multikernel relevance vector machine for object detection. *International Journal on Artificial Intelligence Tools*, 16(6):967–979, December 2007b.

D. Tzikas, A. Likas, and N. Galatsanos. Robust variational Bayesian kernel based blind image deconvolution. In *Proceedings of International Conference on Computer Vision Theory and Applications*, Barcelona, Spain, March 2007c.

D. Tzikas, A. Likas, and N. Galatsanos. Incremental relevance vector machine with kernel learning. In *Proceedings of Hellenic Conference on Artificial Intelligence*, pages 301–312, Syros, Greece, October 2008a.

D. Tzikas, A. Likas, and N. Galatsanos. The variational approximation for Bayesian inference. *IEEE Signal Processing Magazine*, 25(6):131–146, December 2008b.

P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of Computer Vision and Pattern Recognition*, Hawaii, 2001.

L. Wei, Y. Yang, R. Nishikawa, M. Wernick, and A. Edwards. Relevance vector machine for automatic detection of clustered microcalcifications. *IEEE Transactions on Medical Imaging*, 24(10):1278–1285, 2005.

R. Weinstock. *Calculus of Variations*. Dover Publications, 1974.

M. Welling, G. E. Hinton, and S. Osindero. Learning sparse topographic representations with products of Student-t distributions. In MIT Press, editor, *Proceedings of Advances in Neural Information Processing Systems*, volume 15, Cambridge, MA, 2003.

O. Williams, A. Blake, and R Cipolla. Sparse Bayesian learning for efficient visual tracking. *Pattern Analysis and Machine Intelligence*, 27:1292–1304, 2005.

S.-F. Wong and R. Cipolla. Real-time adaptive hand motion recognition using a sparse Bayesian classifier. In *Proceedings of IEEE Workshop on Human-Computer Interaction*, pages 170–179, 2005.

D. Yang, S. O. Zakharkin, G. P. Page, J. P. L. Brand, J. W. Edwards, A. A. Bartolucci, and D.B. Allison. Applications of Bayesian statistical methods in microarray data analysis. *American Journal of PharmacoGenomics*, 4(1):53–62, 2004.

H. Yang, J. Gao, and Z. Wu. Blur identification and image super-resolution reconstruction using an approach similar to variable projection. *Signal Processing Letters, IEEE*, 15: 289–292, 2008.

K. H. Yap, L. Guan, and W. Liu. A recursive soft-decision approach to blind image deconvolution. *IEEE Transactions on Image Processing*, 14(5):624–633, May 2005.

Y. Yitzhaky, I. Mor, A. Lantzman, and N. S. Kopeika. Direct method for restoration of motion-blurred images. *Journal of the Optical Society of America-A*, 15:1512–1519, June 1998.

Y. L. You and M. Kaveh. Blind image restoration by anisotropic regularization. *IEEE Transactions on Image Processing*, 8(3):396–407, March 1999.

# Curriculum Vitae

Dimitris Tzikas was born in Athens, Greece in 1981. He received the B.Sc. Degree in Informatics and Telecommunications from the University of Athens, Greece in 2002 and the M.Sc. Degree from the Department of Computer Science, University of Ioannina, Greece in 2004. Since 2004 he has been a Ph.D. candidate in the same Department.

He has been involved in several research projects and has published six papers in scientific journals and eight papers in refereed conference proceedings. He has also given a tutorial at an international conference. His research interests include machine learning, Bayesian models and statistical image processing.

# Author's Publications

## Journal Publications


[1] D. Tzikas, A. Likas, and N. Galatsanos. Variational bayesian sparse kernel-based blind image deconvolution with Student's-t priors. IEEE Transactions on Image Processing, to appear.

[2] D. Tzikas, A. Likas, and N. Galatsanos. Sparse bayesian modeling with adaptive kernel learning. IEEE Transactions on Neural Networks, to appear.

[3] D. Tzikas, A. Likas, and N. Galatsanos. The variational approximation for Bayesian inference. IEEE Signal Processing Magazine, 25(6):131–146, November, 2008.

[4] D. Tzikas, A. Likas, and N. Galatsanos. Large scale multikernel relevance vector machine for object detection. International Journal on Artificial Intelligence Tools, 16(6):967-979, December 2007.

[5] A. Lukic, M. Wernick, D. Tzikas, X. Chen, A. Likas, N. Galatsanos, Y. Yang, F. Zhao, and S. Strother. Bayesian kernel methods for analysis of functional neuroimages. IEEE Transactions on Medical Imaging, 26(12):1613–1622, December 2007.

[6] D. Tzikas, L. Wei, A. Likas, Y. Yang, and N. Galatsanos. A tutorial on relevance vector machines for regression and classification with applications. EURASIP NEWS LETTER, 17(2):4–23, June 2006.


## Conference Publications


[1] D. Tzikas, A. Likas, and N. Galatsanos. Incremental relevance vector machine with kernel learning. In Proceedings of Hellenic Conference on Artificial Intelligence, pp. 301–312, Syros, Greece, October 2008.

[2] D. Tzikas, A. Likas, and N. Galatsanos. Bayesian bid based on a kernel model for the point spread function. In Proceedings of International Conference on Image Processing, San Antonio, TX, USA, September 2007.

[3] D. Tzikas, A. Likas, and N. Galatsanos. Robust variational Bayesian kernel based blind image deconvolution. In Proceedings of International Conference on Computer Vision Theory and Applications, Barcelona, Spain, March 2007.

[4] D. Tzikas, M. Kukar, and A. Likas. Transductive reliability estimation for kernel based classifiers. In Proceedings of International Symposium on Intelligent Data Analysis, Ljubljana, Slovenia, September 2007.

[5] D. Tzikas, A. Likas, and N. Galatsanos. Variational Bayesian blind image deconvolution based on a sparse kernel model for the point spread function. In Proceedings of European Signal Processing Conference, Florence, Italy, September 2006.

[6] D. Tzikas, A. Likas, and N. Galatsanos. Large scale multikernel RVM for object detection. In Proceedings of Hellenic Conference on Artificial Intelligence, pp. 389–399, Heraclion, Crete, Greece, May 2006.

[7] D. Tzikas, A. Likas, and N. Galatsanos. Bayesian regression of functional neuroimages. In Proceedings of European Signal Processing Conference, Vienna, Austria, September 2004.

[8] D. Tzikas, A. Likas, and N. Galatsanos. Relevance vector machine analysis of functional neuroimages. In Proceedings of IEEE International Symposium on Biomedical Imaging, Arlington, VA, USA, April 2004.

## Tutorials

[1] N. Galatsanos, D. Tzikas. New Tools for Bayesian Inference: The Variational Approximation. In *European Signal Processing Conference*, Lausanne, Switzerland, August 2008.