ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μεταπτυχιακή Εργασία Ειδίκευσης με Θέμα:

ΑΡΑΙΕΣ ΜΠΕΥΖΙΑΝΕΣ ΜΕΘΟΔΟΙ ΓΙΑ ΠΡΟΒΛΗΜΑΤΑ ΠΑΛΙΝΔΡΟΜΗΣΗΣ: ΕΦΑΡΜΟΓΗ ΣΤΗΝ ΑΝΑΛΥΣΗ ΕΙΚΟΝΩΝ ΛΕΙΤΟΥΡΓΙΚΟΥ ΜΑΓΝΗΤΙΚΟΥ ΣΥΝΤΟΝΙΣΜΟΥ

ΔΗΜΗΤΡΗΣ ΤΖΙΚΑΣ

Σεπτέμβριος 2004

Περίληψη

Οι Μπεϋζιανές μέθοδοι είναι ιδιαίτερα δημοφιλείς για την επίλυση προβλημάτων μάθησης με επίβλεψη, στα οποία, δοθέντος ενός συνόλου παρατηρήσεων της εξόδου ενός συστήματος για διάφορες τιμές των εισόδων του, ζητείται η πρόβλεψη των τιμών της εξόδου για πιθανές άλλες τιμές των εισόδων. Οι μέθοδοι αυτές μοντελοποιούν το σύστημα ως μια τυχαία διαδικασία, που επηρεάζεται από ένα σύνολο τυχαίων μεταβλητών. Οι προβλέψεις γίνονται με βάση πιθανοτικούς κανόνες, όμως συχνά είναι απαραίτητη η χρήση αρκετών προσεγγίσεων.

Το Relevance Vector Machine (RVM) [24] είναι ένα τέτοιο Μπεϋζιανό μοντέλο, η έξοδος του οποίου εκφράζεται ως γραμμικός συνδιασμός κάποιων στάθερών συναρτήσεων που ονομάζονται συναρτήσεις βάσης και η τιμή των οποίων εξαρτάται από την είσοδο του συστήματος. Το μοντέλο αυτό είναι αραιό, δηλαδή στο γραμμικό συνδιασμό των συναρτήσεων βάσης περιέχονται μόνο λίγες από αυτές. Η πρόβλεψη της εξόδου για όποιαδήποτε νέα τιμή της εισόδου μπορεί να γίνει αν υπολογιστεί η κατανομή των παράμετρων του γραμμικού συνδιασμού, που ονομάζονται βάρη. Στα βάρη ανατίθεται κάποια αρχική συνάρτηση κατανομής πιθανότητας, που εκφράζει την επιθυμία για αραιά μοντέλα, και με βάση αυτή υπολογίζεται η κατανομή των βαρών δεδομένων των παρατηρήσεων.

Στη συνέχεια το μοντέλο RVM χρησιμοποιείται για την ανάλυση εικόνων λειτουργικού μαγνητικού προσανατολισμού. Οι εικόνες αυτές παρέχουν ένα μέτρο της λειτουργίας των νευρώνων του εγκεφάλου σε κάθε σημείο του και μπορούν να χρησιμοποιηθούν για την αντιστοίχηση μιας περιοχής του εγκεφάλου σε κάποια λειτουργία ή κάποιο ερέθισμα. Συνήθως, στα πειράματα που εκτελούνται συλλέγονται δύο σύνολα εικόνων, στό ένα μόνο από τα οποία υπάρχει το ερέθισμα. Έτσι συγκρίνοντας τα δύο αυτά σύνολα εικόνων μπορεί κανείς να αποφανθεί για την περιοχή στην οποία επηρεάζεται η λειτουργία του εγκεφάλου εξαιτίας του συγκεκριμένου ερεθίσματος. Η περιοχή αυτή ονομάζεται περιοχή ενεργοποίησης..

Υπάρχουν διάφορες μέθοδοι για την ανάλυση εικόνων λειτουργικού μαγνητικού συντονισμού. Οι πιο διαδεδομένες από αυτές (t-test, covariance thresholding, SVD) αποφασίζουν εάν ένα σημείο της εικόνας είναι ενεργοποιημένο ή όχι βασιζόμενες μόνο σε τιμές του σημείου αυτού. Καλύτερα αποτελέσματα μπορούν να επιτευχθούν αν ληφθεί υπόψιν η δομή του σήματος ενεργοποίησης, συμπεριλαμβάνοντας στην απόφαση και τις τιμές των γειτονικών σημείων της εικόνας. Αυτό επιτυγχάνεται υποθέτωντας ότι το σήμα ενεργοποίησης έχει κάποια συγκεκριμένη δομή. Μια τέτοια μέθοδος ακολουθείται στο [13], η οποία όμως βασίζεται στον αλγόριθμο Reversible Jump Markov Chain Monte Carlo και έχει σημαντικό

υπολογιστικό κόστος. Στην εργασία αυτή προτείνεται η χρήση του RVM για την μοντελοποίηση του σήματος ενεργοποίησης, και δημιουργούνται τεχνητά δεδομένα για την αξιολόγηση της μεθόδου.

UNIVERSITY OF IOANNINA
DEPARTMENT OF COMPUTER SCIENCE

SPARSE BAYESIAN METHODS FOR REGRESSION PROBLEMS: APPLICATION TO THE ANALYSIS OF FUNCTIONAL MAGNETIC RESONANCE IMAGES

DIMITRIS TZIKAS

September 2004

Table Of Contents

1 Introduction

1.1 Supervised Learning

Supervised learning is the task of estimating an unknown function $f: X \to T$ given a training set $\{(x_n, t_n)\}_{n=1}^{N}$ of input vectors $x_n \in X$ and output values $t_n \in T$, so that $t_n = f(x_n)$. The domain X of the function f can be any set, but usually it is the set of vectors with length D of real numbers, $X = \Re^D$. In a wide variety of problems the domain T of the function f is the set of real numbers. Such problems are called regression problems, and examples of these are the prediction of stock market values and approximating some quantity of interest given a set of relative measurements. In another class of problems, which are called classification problems, T is a discrete set of class labels. Some classification problems are optical character recognition (OCR), pattern recognition and object recognition in computer vision. In practice, noise is added to the output of the function, thus the target values t_n are given by $t_n = f(x_n) + \varepsilon_n$, where ε_n is the noise component and behave as random variables. This noise is usually modeled as additive, Gaussian distributed and independent among the training examples.

In the general case, where f can be any function, the problem described above cannot be solved, since knowing the value of the function f for a given input vector x_n, does not provide any knowledge for the function at other input vectors, even if they are close to x_n. However, we can make predictions based on assumptions that we make on the form of function f. These assumptions are called the inductive bias of the learning algorithm. Usually, the inductive bias is established by modeling the function f with a known parametric function $y(x;\theta)$, where $\theta = (\theta_1, \theta_2, ..., \theta_M)$ is a set of unknown parameters. Examples of commonly used parametric forms are the neural networks, polynomials and mixture models. The model of f is chosen according to its properties, the size of the training set and the amount of noise added to its output values. It is desired that the model has as few parameters as possible so that they can be easily estimated from the training data. However, models with few parameters are more probable to fail to fit the function f. Generally, selecting a model that describes the training data satisfactorily is a very difficult problem.

1.2_ Maximum Likelihood

After modeling the function f with a parametric function $y(x;\theta)$ we have to use an algorithm to fit the model to the training set, by selecting the appropriate values for the parameters θ. A general method that is commonly used is called maximum likelihood and, as its name implies, it suggests selecting the values of the parameters that maximize the likelihood of the training data:

$$\theta_{ML} = \arg\max_{\theta} L(t;\theta). \tag{1.1}$$

The likelihood is the probability that the training data is generated for the specific values θ of the model parameters.

If we assume that training examples are independent and the noise is additive, white and Gaussian, then

$$t_n = y(x_n;\theta) + \varepsilon_n, \tag{1.2}$$

with

$$\varepsilon_n \sim N(0,\sigma^2), \tag{1.3}$$

where σ^2 is the variance of the noise. Then, the training data is also Gaussian distributed, with

$$p(t;\theta) = N(y(x;\theta),\sigma^2). \tag{1.4}$$

If the training data are assumed independent, the likelihood can be written as:

$$L(t;\theta) = \prod_{n=1}^{N} p(t_n;\theta). \tag{1.5}$$

Equivalently to maximizing the likelihood in (1.5) we can minimize the negative logarithm of it, since it is a monotonically decreasing function. Thus we minimize:

$$\begin{aligned} -\ln L(t;\theta) &= -\ln \prod_{n=1}^{N} p(x_n;\theta) = -\sum_{n=1}^{N} \ln p(x_n;\theta) \\ &= -\sum_{n=1}^{N} \ln \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}\left(y(x_n;\theta)-t_n\right)^2\right\} \\ &= \frac{1}{2\sigma^2}\sum_{n=1}^{N}\left(y(x_n;\theta)-t_n\right)^2 + \frac{1}{2}\ln\frac{N}{2\pi\sigma^2}, \end{aligned} \tag{1.6}$$

which is equivalent to minimizing the mean squared error, given by

$$\theta_{ML} = \arg\min_{\theta} \sum_{n=1}^{N}\left(y(x_n;\theta)-t_n\right)^2. \tag{1.7}$$

Unfortunately, for most models $y(x;\theta)$, we cannot find analytically the parameters θ_{ML} that minimize the mean squared error. Instead we can use multidimensional optimization methods, such as for example gradient descent or Quasi-Newton methods.

Although maximizing the likelihood of the training set seems intuitively to be the best way to select the parameter values, it is possible, in the presence of noise, that the values that minimize the likelihood are not the ones that best fit the function. This will happen if the model is so flexible that it will fit the noise instead of the function and it is known as overfitting. In order to avoid overfitting we have to choose a model with few parameters, so that it is not flexible enough to fit the noise. Alternatively, we can use a validation set to detect overfitting and stop the training process when the performance on the validation set starts decreasing. Another popular way to avoid overfitting is to set constraints on the parameters of the model. These constraints can either be hard constraints, such as assigning an upper and lower bound to a parameter, or soft constraints such as assigning a probability density function to a parameter (prior).

A common approach to avoid overfitting is weight decay. This is an extension of the maximum likelihood method that adds a penalty term to the objective function which is minimized:

$$\theta_{ML} = \arg\min_{\theta} \left\{ \sum_{n=1}^{N} \left(y(x_n;\theta) - t_n \right)^2 + \lambda Q(\theta) \right\}, \tag{1.8}$$

where

$$Q(\theta) = \sum_{i=1}^{M} \theta_i^2 , \tag{1.9}$$

and $\theta = \{\theta_i\}_{i=1}^{M}$ is the set of parameters of the model. The term $Q(\theta)$ prevents the parameters from taking very large values, which results in smooth models that are less likely to overfit the data. Parameter λ controls how much we want to emphasize on the smoothness of the model and therefore to avoid overfitting. Very small values of λ result in very flexible models that are likely to overfit the data, while very large values will result in setting all parameters close to zero, and the model will fail to fit the training data satisfactorily.

1.3 Bayesian methods

Bayesian methods also assume that the unknown function f is modeled with a parametric form $y(x;\theta)$. However, the parameters θ are now assumed to be random variables, and the corresponding probability density functions are assumed known. Although, usually the parameters of the model do not have a physical meaning and thus their distributions are actually unknown, assuming a specific prior distribution on the model parameters can establish preference for certain values of the parameters which is frequently desired. In Bayesian methods, we do not compute specific values for the parameters, but compute their posterior probability density function. Predictions are made using the posterior distribution, which usually provides more knowledge than a specific value (e.g. the expected value of the posterior) would provide.

Having defined these prior probability distributions we can compute the posterior probability density of the parameters given the training data using the Bayes theorem

$$p(\theta \mid t) = \frac{p(t \mid \theta)\, p(\theta)}{p(t)}, \tag{1.10}$$

where $p(t \mid \theta)$ is the probability that the training data are created from the model with parameters θ and can easily be computed if we know the pdf of the noise affecting the training set. The denominator $p(t) = \int p(t \mid \theta)\, p(\theta)\, d\theta$ is a normalizing constant.

Predictions in Bayesian method are made by marginalizing over all unknown parameters. Then, given a new point x_*, the corresponding value of the unknown function t_* is predicted as follows:

$$p(t_* \mid t) = \int p(t_* \mid x_*, \theta)\, p(\theta \mid t)\, d\theta. \tag{1.11}$$

Unfortunately, this integral is very rarely computed analytically and usually it is necessary to make some approximations.

Generally, Bayesian methods give better results, because they make predictions based on the whole posterior distribution of the parameters. However, there are two important problems. First, it is not always obvious what prior probabilities should be chosen for the parameters of the model and second, the posterior probability in (1.10) can be very rarely computed analytically and approximations have to be considered. Commonly used techniques to approximate these integrals are the variational techniques where a lower bound of the integral is found, and sampling techniques such as Monte Carlo integration where the integral is approximated by a finite sum of suitably sampled values of the integrated function.

In order to avoid the difficulties of the Bayesian methodology, sometimes instead of computing $p(\theta|t)$ and the integral of (1.11), we only estimate the values of the parameters θ that maximize $p(\theta|t)$ and use them to make predictions. This is called the maximum a posteriori (MAP) methodology and is similar to the maximum likelihood method, but it also considers prior information for the parameters of the model.

It can be shown that the MAP estimate is equivalent to a maximum likelihood solution with weight decay. If we assume that noise is additive, Gaussian distributed and independent among the training examples the pdf of the training set given the parameters is:

$$p(t|\theta)=\prod_{n=1}^{N} N\left(y(x_n|\theta),\sigma^2\right), \tag{1.12}$$

and assuming zero-mean Gaussian prior distributions for the parameters of the model,

$$p(\theta_i)=N(0,\alpha_i), \tag{1.13}$$

we can apply Bayes's theorem as:

$$p(\theta|t)=\frac{p(t|\theta)p(\theta)}{p(t)}. \tag{1.14}$$

The MAP estimate is then computed as:

$$\theta_{MAP}=\arg\max_{\theta}\left(p(t|\theta)p(\theta)\right). \tag{1.15}$$

Since the samples and parameters are independent:

$$\theta_{MAP}=\arg\max_{\theta}\left(\prod_{n=1}^{N} p(t_n|\theta)\prod_{i=1}^{M} p(\theta_i)\right), \tag{1.16}$$

which (taking the negative logarithm) is equivalent to:

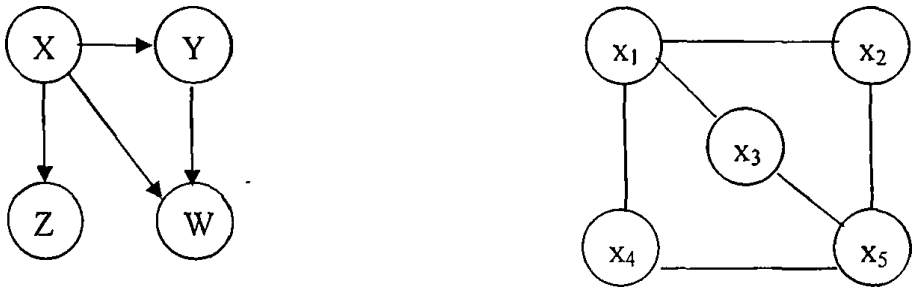
$$\theta_{MAP}=\arg\min_{\theta}\left(\sum_{n=1}^{N}\left(y(x_n;\theta)-t_n\right)^2+\frac{\sigma^2}{\alpha_i}\sum_{i=1}^{M}\theta_i^2\right) \tag{1.17}$$

Equation (1.17) is similar to (1.8) when the parameter λ is set to the ratio of the noise variance to the variance of the parameter prior distribution $\frac{\sigma^2}{\alpha_i}$. Thus, the weight decay method used to avoid overfitting is identical to the MAP method when assuming Gaussian distributions for the model parameters.

1.4 Graphical Models

Although, in most problems the probability distributions of most variables are not known, usually they can be efficiently approximated using common types of distributions. However, in problems with many (generally not independent) variables, computation of the joint distribution of the whole variable set is not always straightforward. Graphical models are then used to express the joint probability density function over a set of variables.

Graphical models consist of nodes that represent random variables and arcs between some of the nodes, that may be directed or not, and represent relations between the variables. Graphical models that contain only undirected arcs are also called Markov networks or Markov random fields (MRFs) and those that contain only directed arcs are also called Bayesian networks or belief networks. Sometimes both directed and undirected arcs are contained in the same graphical model. This class of graphs is known as chain graphs.



(a) a directed graph *(b) an undirected graph*

Figure 1.1. Examples of Graphical Models

In Markov Random Fields two sets of nodes A, B are conditionally independent given a third set of nodes C, if all paths between A and B are separated by a node in C. Obviously, a node is independent of all other non-neighboring nodes given the values of its immediate neighbors. The probability distribution of the set of variables can be expressed as

$$p(z)=\frac{1}{Z}\prod_{j=1}^{J}\psi_j\left(C_j(z)\right), \tag{1.18}$$

where z is the set of variables, $C_j(z), j=1..J$ are the cliques of the graph, $\psi_j, j=1..J$ is a set of clique potential functions and Z is a normalizing constant to ensure that $p(z)$ is a probability function. The set of cliques $\left\{C_j(z)\right\}_{j=1}^{J}$ is defined so that the cliques cover all the variables, $\left\{C_1(z)\cup...\cup C_J(z)\right\}=z$ and they may be overlapping. The clique potential functions

ψ_j are arbitrary real valued positive functions that assign a positive number to each possible assignment of values to the variables of the clique.

In Bayesian networks an arc from node X to node Y informally expresses the fact that A causes B. Conditional independence between two nodes X and Y, given a set of nodes C, is easy to detect, since a node is always independent of its ancestors given its parents. The ancestors of a node are recursively defined as union of the set of parents of that node and the ancestors of the node's parents. Similarly, the descendents of a node are defined as the union of the set of children of that node and the descendents of the node's children. Two sets of nodes A and B are independent given the set of nodes C if they are d-separated by C. Two nodes X and Y are said to be d-separated if every undirected path form X to Y contains a node d that satisfies one of the following conditions: 1) d has converging arrows and neither d nor its descendants are in C 2) d does not have converging arrows and is in C. A node has converging arrows when it is the child of the previous node and parent of the following node in the path (e.g. Node W in Figure 1.1a).

The probability distribution over the set of variables can be computed as:

$$p(z) = \prod_{j=1}^{J} p\left(z_j \mid z_{pa(j)}\right), \tag{1.19}$$

where $z = \{z_j\}_{j=1}^{J}$ is the set of variables and $z_{pa(j)}$ is the set of parents of node z_j.

Given a set of nodes A, we are often interested in the minimal set of nodes C such that the nodes in A are independent with all other nodes that do not belong in C, given the nodes in C. The set C is called the Markov blanket of the set A. The Markov blanket of a single node is the union of the node's parents, its children and the parents of its children.

In the field of machine learning directed networks are more popular because they usually provide an easier way to express the process underlying the creation of the observations. On the other hand, undirected graphs are more popular in the physics and vision communities because the systems being studied can usually be expressed in terms of many localized potential functions.

1.5 Exact Inference in Graphical Models

Nodes in graphical models represent random variables which may be observed, if we can observe the value of the variable, or hidden otherwise. Inference is the process of estimating the posterior distribution of hidden variables, given the values of the observed variables. If all nodes of the model are observed, it is relatively straightforward to estimate the parameters of the

model, using a maximum likelihood technique as discussed in section 1.2. However, when the model contains hidden variables, application of the maximum likelihood technique is not trivial and usually requires the use of Expectation-Maximization (E-M) algorithms, that iteratively maximize the likelihood by finding estimates of the hidden variables and then using them to compute and maximize the likelihood of the observations. The estimation of the hidden variables requires the computation of their posterior distribution.

Let H represent the hidden variables of the graphical model and E represent the observed variables (also called evidence). We are interested in computing the posterior distribution of the hidden variables:

$$p(H \mid E) = \frac{p(E,H)}{p(E)}, \tag{1.20}$$

$p(E,H)$ is computed as:

$$p(E,H) = p(E \mid H)\, p(H). \tag{1.21}$$

and $p(E)$ is computed by marginalizing over the hidden variables:

$$p(E) = \int p(E \mid H)\, p(H)\, dH, \tag{1.22}$$

Computational difficulties arise when trying to compute the marginal distribution $p(E)$. Thus, inference algorithms focus on computation of marginal probabilities. Then, all other quantities are easily computed.

Let $x = \{x_1, x_2, x_3, x_4, x_5\}$ be the set of discrete variables of the undirected graphical model shown in Figure 1.1b. The joint probability distribution of all variables is given by equation (1.18). The marginal distribution of variable x_1 is computed as:

$$p(x_1) = \frac{1}{Z} \sum_{x_2} \sum_{x_3} \sum_{x_4} \sum_{x_5} \psi(x_1, x_2)\psi(x_1, x_3)\psi(x_1, x_4)\psi(x_2, x_5)\psi(x_3, x_5)\psi(x_4, x_5) \tag{1.23}$$

This generally requires $O(k^n)$ computations where n is the number of nodes and k the number of different values that a variable takes and is practically computable only for very small number of variables. However, the marginal distribution can be more efficiently computed by exploiting the independence properties of the graphical model, as follows:

$$p(x_1) = \frac{1}{Z} \sum_{x_3} \psi(x_1, x_3) \sum_{x_5} \psi(x_3, x_5) \sum_{x_4} \psi(x_1, x_4)\psi(x_4, x_5) \sum_{x_2} \psi(x_1, x_2)\psi(x_2, x_5) \tag{1.24}$$

$$= \frac{1}{Z} \sum_{x_3} \psi(x_1, x_3) \sum_{x_5} \psi(x_3, x_5) \sum_{x_4} \psi(x_1, x_4)\psi(x_4, x_5)\, m_2(x_2, x_5) \tag{1.25}$$

$$= \frac{1}{Z} \sum_{x_3} \psi(x_1, x_3) \sum_{x_5} \psi(x_3, x_5) m_4(x_1, x_5) m_2(x_2, x_5) \tag{1.26}$$

$$= \frac{1}{Z} \sum_{x_3} \psi(x_1, x_3) m_5(x_1, x_3) \tag{1.27}$$

$$= \frac{1}{Z} m_1(x_1) \tag{1.28}$$

where each message $m_j(x,...)$ is a new potential obtained by eliminating the jth variable, and is a function of all the variables linked to the eliminated variable.

In general, the process to compute the marginal likelihood of a set of variables is described by the following algorithm:

1. Choose a node of the model x_j to eliminate
2. Define a new message m_j
3. Remove node x_j. Connect its neighbors with edges.
4. Repeat from step 1 until only the requested nodes remain in the model.

Inference in directed graphical models generally follows the same principles. Every directed graphical model can be converted to an equivalent undirected graphical model, with a procedure called moralisation. Thus, inference on directed graphical models can be done by applying the inference procedure on the equivalent undirected graph. However, if the graph is a tree, it is possible to apply an exact analogous algorithm, called belief propagation, on the directed graph immediately.

1.6 Approximate Inference Methods for Graphical Models

In many problems, where the graphical model is complicated, exact inference methods as described in section 1.5 are computationally intractable and approximate methods have to be considered. Most important methods that perform approximate inference in graphical models are based either on Monte Carlo sampling algorithms or variational techniques.

1.6.1 Monte Carlo Integration

Monte Carlo integration is a method for approximating integrals with finite sums as follows:

$$\int f(x)p(x)dx \approx \frac{1}{N}\sum_{i=1}^{N} f(x^i), \tag{1.29}$$

where $x^i, i = 1..., N$ are samples generated from the distribution $p(x)$.

In many cases it is difficult to draw samples from $p(x)$. Rejection sampling and importance sampling [7] are algorithms that draw samples from an arbitrary distribution $p(x)$ and can be used in this case. Both these algorithms are based on drawing samples from a proposal distribution, and their performance depends on the choice of this proposal distribution. Unfortunately, for high dimensional spaces it is very difficult to find a suitable proposal distribution, and these algorithms cannot be used. Instead, there is a variety of Markov Chain Monte Carlo (MCMC) algorithms that achieve good performance. These algorithms, define a Markov Chain that is guaranteed to converge to a desired distribution $p(x)$.

The most popular MCMC algorithm is the Metropolis-Hastings algorithm. The algorithm randomly selects an initial state x^0 for the Markov Chain and then iteratively selects the next state based on a proposal distribution $q(x^{i+1} \mid x^i)$. In order to converge to the desired target distribution $p(x)$, samples x^{i+1} generated from the proposal distribution $q(x^{i+1} \mid x^i)$ are accepted with probability

$$R = \min\left\{1, \frac{p(x^{i+1})}{p(x^i)} \frac{q(x^i \mid x^{i+1})}{q(x^{i+1} \mid x^i)}\right\}, \tag{1.30}$$

otherwise the chain remains at state x^i. The performance of the algorithm highly depends on the choice of a proposal distribution. Generally, it is desirable that the proposal distribution is similar to the target distribution.

Within the context of graphical models, Monte Carlo integration can be used to approximate marginal likelihoods, as follows:

$$p(E) = \int p(E \mid H)p(H)dH = \frac{1}{N}\sum_{i=1}^{N} p(E \mid H^i), \tag{1.31}$$

where H^i, $i = 1...N$ are samples generated from the prior distribution $p(H)$.

1.6.2 Variational methods

Variational methods are used as approximation methods in a wide variety of problems. They introduce a set of variational parameters that is used to express a class of bounds for the

quantity of interest. Then, the variational parameters are set to values that obtain the tighter of the bounds, which is used to approximate the unknown function.

Variational methods depend on the observation that a concave function $f(x)$ can be represented via a conjugate or dual function as follows:

$$f(x) = \min_{\lambda} \{\lambda^T x - f^*(\lambda)\}, \tag{1.32}$$

where λ is a set of variational parameters and $f^*(\lambda)$ is the conjugate function of $f(x)$ which can be computed by:

$$f^*(\lambda) = \min_{x} \{\lambda^T x - f(x)\}. \tag{1.33}$$

Thus, $\lambda^T x - f^*(\lambda)$ is a class of lower bounds for function $f(x)$. The benefit of the expression (1.32) is that given a value for the variational parameters λ, it is linear to the variables x. This is an important simplification, but also introduces the additional cost of finding appropriate values of the variational parameters λ for each possible instance of the variables x.

Similarly, upper bounds can be found for a convex function. If the function is neither concave nor convex, the same methodology can be applied after transforming the argument of the function so that it becomes convex or concave. This will generally lead to non-linear bounds for $f(x)$.

When used for approximate inference in graphical models, variational methods obtain a class of lower bounds for the logarithm of the marginal likelihood $\ln P(E)$, as follows:

$$\begin{aligned}
\ln P(E) &= \ln \int P(E,H)\,dH \\
&= \ln \int Q(H \mid E) \frac{P(E,H)}{Q(H \mid E)}\,dH \\
&= \ln E\left\{\frac{P(E,H)}{Q(H \mid E)}\right\} \\
&\geq E\left\{\ln \frac{P(E,H)}{Q(H \mid E)}\right\} \qquad \text{(applying Jensen's inequality)} \\
&= \int Q(H \mid E) \ln \frac{P(E,H)}{Q(H \mid E)}\,dH \\
&= L(Q).
\end{aligned} \tag{1.34}$$

The difference between the true log marginal likelihood $\ln P(E)$ and its lower bound $L(Q)$ is

$$D(Q\|P) = -\int Q(H|E) \ln \frac{P(H|E)}{Q(H|E)} dH, \tag{1.35}$$

which is the Kullback-Leibler (KL) divergence between the approximating distribution $Q(H|E)$ and the true posterior $p(H|E)$. Thus, in order to obtain the tightest bound for the marginal likelihood it is sufficient to minimize the KL divergence. If we assume that $Q(H|E)$ may be any function, minimization of the KL divergence happens when $Q(H|E) = P(H|E)$. However this does not simplify the problem at all. Instead we have to choose a family of distributions that are flexible enough to approximate the target distribution efficiently, and yet simple enough that the lower bound $L(Q)$ can be easily evaluated. Usually, we choose a parametric form $Q(H|E,\lambda)$ for the family Q of distributions, where λ is a set of variational parameters. Then, the variational parameters λ^* are set to the values that minimize the Kullback-Leibler (KL) divergence between the approximating distribution $Q(H|E)$ and the true posterior $p(H|E)$.

$$\lambda^* = \arg\min_{\lambda} D\big(Q(H|E,\lambda)\|P(H|E)\big). \tag{1.36}$$

By choosing an approximating distribution $Q(H|E,\lambda)$ that is flexible enough, we can obtain bounds that can efficiently approximate the desired distribution $p(H|E)$.

1.7 Kernel Methods

As discussed in section 1.2, in order to achieve good generalization it is essential that the learning algorithm implies the appropriate inductive bias. However, it is usually very difficult to identify what the proper inductive bias is, and equally difficult to design an algorithm that implements that inductive bias.

Kernel methods are based on the concept of similarity between training examples to perform generalization. Given a training set $\{(x_n, t_n)\}_{n=1}^{N}$ the prediction for the target value t_* at a new point x_* is made based on training examples that are similar to the new point x_*. The similarity between two training vectors is measured by a class of functions that are called kernel

functions. Depending on the type of the training vectors, an appropriate kernel function has to be chosen, that describes suitably the similarity between training examples.

A kernel function is a symmetric function

$$k : X \times X \to \Re \tag{1.37}$$

that, given two vectors $x_1, x_2 \in X$, returns a real number that expresses a measure of the similarity between the vectors x_1 and x_2. Common kernel functions are:

1. Polynomial kernels

$$k(x_1, x_2) = \langle x_1, x_2 \rangle^d, \tag{1.38}$$

where $\langle x_1, x_2 \rangle$ denotes the dot product between x_1 and x_2.

2. Gaussian kernels

$$k(x_1, x_2) = \exp\left\{ -\frac{\|x_1 - x_2\|^2}{2\sigma^2} \right\} \tag{1.39}$$

3. Sigmoid kernels

$$k(x_1, x_2) = \tanh\left(k\langle x_1, x_2 \rangle + \vartheta \right) \tag{1.40}$$

4. Inhomogenous polynomial kernels

$$k(x_1, x_2) = \left(\langle x_1, x_2 \rangle + c \right)^d \tag{1.41}$$

5. B-Spline kernels

$$k(x_1, x_2) = B_{2p+1}\left(\|x_1 - x_2\| \right), \text{ with } B_n = \bigotimes_{i=1}^{n} I_{\left[-\frac{1}{2}, \frac{1}{2}\right]}. \tag{1.42}$$

The most popular class of kernel functions are positive definite kernel functions. A kernel function is said to be positive definite if for any set of training points $\{x_1, x_2, ..., x_N\}$ and any set of real numbers $\{a_1, a_2, ..., a_N\} \in \Re^N$ it satisfies

$$\sum_{i,j} a_i a_j k(x_i, x_j) \geq 0. \tag{1.43}$$

This is equivalent to saying that the Gram matrix defined by $[K]_{ij} = k(x_i, x_j)$ is positive definite.

Positive definite kernel functions have the property that they represent similarity as the dot product in some linear space H [18], so there exists a map Φ for which:

$$k(x_1, x_2) = \langle \Phi(x_1), \Phi(x_2) \rangle, \tag{1.44}$$

where

$$\Phi : X \to H, \quad x \mapsto \Phi(x). \tag{1.45}$$

An interesting mapping is :

$$\Phi : X \to \mathbb{R}^X, \quad x \mapsto k(.,x), \tag{1.46}$$

where $\mathbb{R}^X := \{f : X \to \mathbb{R}\}$ is the set of functions mapping X into $\mathbb{R}$. Thus, $\Phi(x)$ is a function that assigns the value $k(x',x)$ to $x' \in X$. This mapping constructs a feature space consisting of functions on the domain X. Each of these functions represents similarity between a vector of the training set and vectors in the rest feature space. This feature space is a very useful representation of the training set, since learning in kernel methods is achieved based on similarity between training examples. Though initially it might seem very difficult to apply a learning algorithm on the feature space defined in (1.46), surprisingly the dot product defined in this space can be computed by evaluation of the kernel function:

$$\langle \Phi(x_1), \Phi(x_2) \rangle = k(x_1, x_2). \tag{1.47}$$

Kernel methods benefit from this observation, and apply simple algorithms based on dot product distances after mapping the data on a new feature space. This mapping can be done explicitly by applying the algorithm on the training set $\{\Phi(x_i)\}_{i=1}^N$, but it is much more efficient to just substitute the dot product operations with evaluation of the kernel function corresponding to the desired mapping. The same idea can be applied on any algorithm that is based on a positive definite kernel k and substitute it with another positive definite kernel $\tilde{k}$. This procedure is known as the kernel trick.

1.8 Support Vector Machines

A popular algorithm for regression and classification problems that is based on the kernel trick is the Support Vector Machine (SVM) [17]. Considering regression problems and given a training set $\{x_n, t_n\}_{n=1}^N$ the goal of the SVM algorithm is to find a function $f(x)$ that is as smooth as possible and its distance $|f(x_n) - t_n|$ from each training point is bounded by a constant ε. The function $f(x)$ is assumed to be of the form:

$$f(x) = x^T w + b. \tag{1.48}$$

Then, the wanted function can be found by solving the following optimization problem:

$$\text{minimize:} \quad \frac{1}{2}\|w\|^2, \quad \text{(smoothness criterion)} \tag{1.49}$$

$$\text{subject to:} \quad |f(x_n)-t_n| \le \varepsilon, \quad n=1,...,N. \tag{1.50}$$

However, given a constant ε the above optimization problem does not always have a solution. In order to cope with situations like this, some errors may be allowed but we seek the function that minimizes them. Thus the optimization problem can be written as:

$$\text{minimize:} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*) \tag{1.51}$$

$$\begin{aligned} \text{subject to:} \quad & t_n - f(x_n) \le \varepsilon + \xi_n, \quad n=1,...,N. \\ & f(x_n) - t_n \le \varepsilon + \xi_n^*, \quad n=1,...,N \\ & \xi_n, \xi_n^* \ge 0 \end{aligned}$$

This formulation allows errors smaller than ε without penalizing them, but if this is not feasible greater errors may be accepted. Moreover, errors greater than ε may be accepted if this would result in a much more smooth function $f(x)$. The parameter C determines the trade-off between the smoothness of $f(x)$ and the cost of errors larger than ε.

In order to solve the above constrained minimization problem we form the Lagragian function as:

$$\begin{aligned} L = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) - \sum_{i=1}^{N}\alpha_i(\varepsilon + \xi_i - t_i + f(x_i)) \\ -\sum_{i=1}^{N}\alpha_i^*(\varepsilon + \xi_i^* - t_i + f(x_i)) - \sum_{i=1}^{N}(\eta_i\xi_i + \eta_i^*\xi_i^*) \end{aligned} \tag{1.52}$$

with $\alpha_i, \alpha_i^*, \eta_i, \eta_i^* \ge 0$.

It can be shown that the optimal solution of (1.51) can be found by finding the saddle points of the Lagragian.

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{N}(\alpha_i^* - \alpha_i) = 0 \tag{1.53}$$

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)x_i = 0 \tag{1.54}$$

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \eta_i = 0 \tag{1.55}$$

$$\frac{\partial L}{\partial \xi_i^*} = C - \alpha_i^* - \eta_i^* = 0 \tag{1.56}$$

Substituting (1.53) - (1.56) into (1.52) yields the dual optimization problem:

$$\textit{minimize:} \quad -\frac{1}{2}\sum_{i=1}^{N}\sum_{i=1}^{N}\left(\alpha_i - \alpha_i^*\right)\left(\alpha_j - \alpha_j^*\right)x_i^T x_j - \varepsilon\sum_{i=1}^{N}\left(\alpha_i + \alpha_i^*\right) + \sum_{i=1}^{N} t_i\left(\alpha_i - \alpha_i^*\right) \tag{1.57}$$

$$\text{subject to:} \quad \sum_{i=1}^{N}\left(\alpha_i - \alpha_i^*\right) = 0,$$

$$\alpha_i, \alpha_i^* \in [0, C].$$

Many optimization algorithms can be used to solve the problem of (1.57), such as gradient descent and quasi-Newton algorithms.

From (1.54) we can write

$$w = \sum_{i=1}^{N}\left(\alpha_i - \alpha_i^*\right)x_i \tag{1.58}$$

which gives:

$$f(x) = \sum_{i=1}^{N}\left(\alpha_i - \alpha_i^*\right)x_i^T x + b \tag{1.59}$$

Thus the weight vector can completely be described as the linear combination of the training patterns x_i. Furthermore, it can be shown [17] that the weights are independent of many input patterns x_i. Actually, the training process only requires those training patterns that the weights depend on, which are called Support Vectors (SV).

Moreover, the complete algorithm can be described in terms of dot products between the data. This motivates the use of the kernel trick in order to extend the algorithm for more general than linear functions. The use of the kernel trick allows to apply the SVM learning algorithm on projections of the training data to some other space. Although particular projections may be very hard to compute, using the kernel trick bypasses this difficulty by finding effective ways to compute dot products in such spaces. The kernel trick suggests projecting the training data to another space using a map Φ, and then, running the dot product SVM algorithm, is equivalent to running the SVM algorithm and substitute dot products with a proper kernel function, so that:

$$K(x_1, x_2) = \langle \Phi(x_1), \Phi(x_2) \rangle \tag{1.60}$$

Thus, assuming the more general model:

$$f(x) = \Phi(x)w + b, \tag{1.61}$$

the weights can be found from the solutions of the optimization problem in (1.57) by substituting the dot product with a kernel function that satisfies (1.60).

2 The Relevance Vector Machine

2.1 Introduction

A model that is frequently used in supervised learning is that of the form:

$$y(x;w)=\sum_{i=1}^{M} w_i\phi_i(x)=w^T\phi(x), \tag{2.1}$$

where the output is the linear combination of M basis functions $\phi(x)=(\phi_1(x),\phi_2(x),...,\phi_M(x))^T$. These basis functions are fixed and our goal is to estimate the values of the weights $w=(w_1,w_2,...,w_M)$ in order to fit the given training set.

The Support Vector Machine (SVM) [17], [28] is a specialization of the model defined in equation (2.1), where the set of basis functions, consists of one kernel function $K(x,x_i)$ for each training point. Furthermore, a bias is used by adding one constant basis function, giving:

$$y(x;w)=\sum_{i=1}^{N} w_iK(x,x_i)+w_0. \tag{2.2}$$

To achieve good generalization, it is desirable that the trained model is sparse, meaning that most weights are set to zero, effectively pruning the corresponding kernel functions. This is desired because it speeds up computations on the trained model, and also reduces the effective number of parameters, which otherwise is equal to the number of the training examples, resulting in better generalization performance.

Inference on this model using the SVM learning algorithm is quite satisfactory, but there are some practical limitations, such as:

- The SVM algorithm produces relatively sparse solutions, but the number of support vectors generally grows linearly with the number of training examples, which implies that the sparseness of the estimated model depends on the number of training examples.
- Predictions are not probabilistic, meaning that they are hard point estimates for regression and class labels in classification. Ideally, we wish to compute the conditional probability $p(t|x,X)$ of the output at a given point x given the training set X, which provides also a measure of uncertainty in the prediction.
- Parameters of the SVM training method have to be estimated using methods such as cross-validation that are generally computationally expensive
- The kernel functions must be symmetric and positive definite

The Relevance Vector Machine (RVM) is an alternative method of learning in models like (2.2). Learning is achieved by following a Bayesian inference procedure, after assuming appropriate prior distributions on the parameters of the model. These prior distributions are chosen to encode our preference for sparse models. This is achieved by using automatic relevance determination (ARD) priors [15] on the weights. A set of hyperparameters is introduced, one associated with each weight and their most probable values are iteratively estimated from the training data. Then, basis functions that are irrelevant with the data are pruned from the model by setting their associated weight to zero, while emphasis is given on those basis functions that are supported by the data.

Selecting the appropriate basis functions for modeling the training data is essential in order to achieve good generalization performance. However, there is no methodology to find the optimal basis functions. Instead, one has to select arbitrary sets of basis functions, compute the performance of each of the resulting models using a method such as cross validation [2] and then select the model with the best performance.

2.2 RVM Model Description

The Relevance Vector Machine (RVM) has been proposed in [24]. Given a training set $\{x_n, t_n\}_{n=1}^{N}$, the target values t_n are assumed to derive from the model $y(x_n; w)$ defined in (2.2) with additive noise,

$$t_n = y(x_n; w) + \varepsilon_n . \tag{2.3}$$

Target values may be either continuous variables or discrete 'class labels', but we will only study regression problems and thus assume that t_n are continuous random variables. Classification using a RVM is very similar and requires only small modifications. The noise samples are assumed to be independent and Gaussian distributed with zero mean and deviation σ^2 which is assumed unknown and will be estimated from the training data. Thus the target values are also assumed independent and Gaussian distributed,

$$p(t_n \mid w, \sigma^2) = \mathrm{N}(\Phi w, \sigma^2), \tag{2.4}$$

where Φ is the $N \times (N+1)$ design matrix with $\Phi = [\phi(x_1), \phi(x_2), ... \phi(x_N)]^T$ and $\phi(x_n) = [1, K(x_n, x_1), K(x_n, x_2), ..., K(x_n, x_N)]^T$.

Priors are assumed over the model parameters in order to follow a Bayesian inference procedure. Automatic Relevance determination (ARD) priors [15] are used for the weights in order to encode the preference for smooth models. Thus, the weights w_i of the model are assigned Gaussian prior distributions:

$$p(w_i \mid \alpha_i) = \mathrm{N}(0, a_i^{-1}), \tag{2.5}$$

where $\alpha = (\alpha_0, \alpha_1, ..., \alpha_N)$ is a vector of $N+1$ hyperparameters which are also treated as random variables and allow pruning of irrelevant basis functions by setting the corresponding hyperparameter to infinity. These hyperparameters α_i and the noise variance σ^2 are scale parameters, and suitable priors for them are Gamma distributions [1]:

$$p(\alpha_i) = Gamma(a,b), \tag{2.6}$$

$$p(\sigma^2) = Gamma(c,d), \tag{2.7}$$

where

$$Gamma(a,b) = \Gamma(\alpha)^{-1} b^{\alpha} a^{\alpha-1} e^{-b\alpha} \tag{2.8}$$

and

$$\Gamma(\alpha) = \int_0^{\infty} t^{\alpha-1} e^{-t} dt . \tag{2.9}$$

Parameters a, b, c and d of the Gamma distributions allow us to set the mean and variance of the Gamma, and usually should be set to very small values making the Gamma distributions uninformative. By setting appropriate values for these parameters, we can strengthen or weaken our preference for sparser models. However, a too sparse model is unlikely to be flexible enough to fit the training data well, while a too big model will probably overfit it.

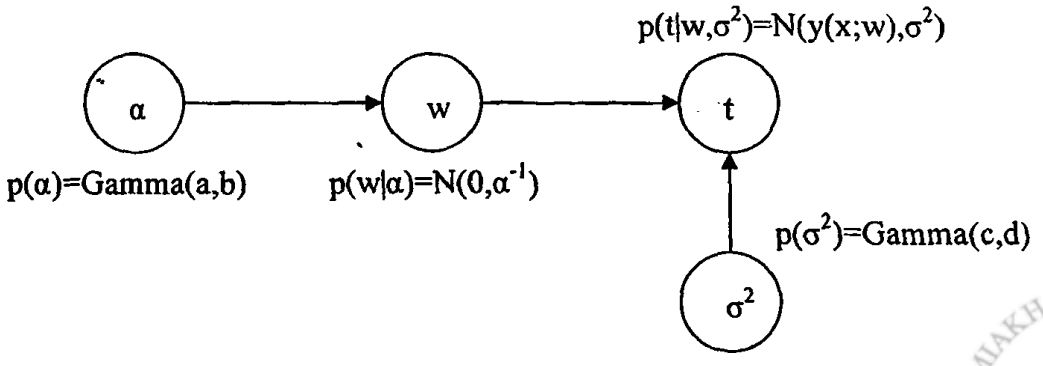


Figure 2.1 Graphical model of the Relevance Vector Machine

The sparseness of the model relies on the prior distributions defined above. Although the weight prior $p(w|a)$, which is a Gaussian distribution, does not appear to encourage sparse models, it can be seen by integrating out the weights, that $p(w)$ is a Student-t distribution which has most probability mass concentrated at sparse models. This is why by selecting different values for the parameters of the Gamma distributions over the hyperparameters, the sharpness of $p(w)$ can be modified, adjusting at the same time our preference for sparser models.

In the described model the weight parameters are as many as the training examples are. However, the priors defined over the parameters, prevent the model from overfitting the data. The assignment of one additional hyperparameter for each weight of the model is the key feature of the RVM, and the sparseness of the model relies on these additional hyperparameters. To introduce additional $N+1$ parameters may seem counter-intuitive, since there are already too many parameters. However, from a Bayesian perspective, these parameters can be intergraded out and they do not present a problem.

2.3 Inference

The posterior probability over the unknowns can be computed using Bayes' rule:

$$p(w,\alpha,\sigma^2|t)=\frac{p(t|w,\alpha,\sigma^2)p(w,\alpha,\sigma^2)}{p(t)} \tag{2.10}$$

Unfortunately, the normalizing integral $p(t)=\int p(t|w,\alpha,\sigma^2)p(w,\alpha,\sigma^2)dwd\alpha d\sigma^2$ in the denominator of (2.10) cannot be analytically computed and an effective approximation is used instead. The posterior is decomposed as:

$$p(w,\alpha,\sigma^2|t)=p(w|t,\alpha,\sigma^2)p(\alpha,\sigma^2|t). \tag{2.11}$$

Then, the posterior over the weights $p(w|t,\alpha,\sigma^2)$ can be analytically computed, since its denominator $p(t|\alpha,\sigma^2)=\int p(t|w,\sigma^2)p(w|\alpha)dw$ is now a convolution of two Gaussian functions. As described in Appendix A the posterior over the weights can be computed by

$$p(w|t,\alpha,\sigma^2)=\frac{p(t|w,\sigma^2)p(w|\alpha)}{p(t|\alpha,\sigma^2)}=\mathrm{N}(\mu,\Sigma), \tag{2.12}$$

where

$$\Sigma = \left(\sigma^{-2}\Phi^T\Phi + A\right)^{-1}, \tag{2.13}$$

$$\mu = \sigma^{-2}\Sigma\Phi^T t, \tag{2.14}$$

with $A = diag\left(a_1, a_2, ..., a_M\right)$.

The posterior $p\left(\alpha, \sigma^2 \mid t\right)$ cannot be computed analytically and it is approximated by a delta function at its mode.

$$p\left(\alpha, \sigma^2 \mid t\right) \approx \delta\left(\alpha_{MP}, \sigma^2_{MP}\right), \tag{2.15}$$

where α_{MP} and σ^2_{MP} are the most probable values of $p\left(\alpha, \sigma^2 \mid t\right)$,

$$a_{MP} = \arg\max_a\left(p\left(a \mid t\right)\right), \tag{2.16}$$

$$\sigma^2_{MP} = \arg\max_{\sigma^2}\left(p\left(\sigma^2 \mid t\right)\right). \tag{2.17}$$

This approximation has been frequently used and in this problem it is very effective.

We can find α_{MP} and σ^2_{MP} by maximizing $p\left(\alpha, \sigma^2 \mid t\right) \propto p\left(t \mid \alpha, \sigma^2\right) p\left(\alpha\right) p\left(\sigma^2\right)$. The first term $p\left(t \mid \alpha, \sigma^2\right)$ is known as the marginal likelihood and is computed by marginalizing over the weights (see Appendix A) according to

$$p\left(t \mid \alpha, \sigma^2\right) = \int p\left(t \mid w, \sigma^2\right) p\left(w \mid \alpha\right) dw. \tag{2.18}$$

This gives

$$p\left(t \mid \alpha, \sigma^2\right) = \mathrm{N}\left(0, C\right), \tag{2.19}$$

with $C = \sigma^2 I + \Phi A^{-1}\Phi^T$. The marginal likelihood is also referred to as the 'evidence for the hyperparameters' [14]. In order to maximize it with respect to the hyperparameters α_i many of these have to be set to infinity. This leads to setting the corresponding weights w_i to zero, and pruning the corresponding basis functions. The basis functions that get pruned this way are those which are not supported by the training set. On the other hand, the hyperparameters corresponding to basis functions which are well supported by the data are set to small values, broadening the prior distribution of the corresponding weights and allowing them to be set to large values.

2.4 Hyperparameter Optimization

Unfortunately a_{MP} cannot be computed analytically. Instead we use iterative formula for its re-estimation. Maximization of the marginal likelihood is equivalent to maximizing its logarithm with respect to the logarithm of the hyperparameters:

$$L = \log p\left(t \mid \log \alpha, \log \sigma^2\right) + \sum_{i=0}^{N} \log p\left(\log \alpha_i\right) + \log p\left(\log \sigma^2\right), \tag{2.20}$$

which is computed as

$$L = -\frac{1}{2}\left[\log|C| + t^T C^{-1} t\right] + \sum_{i=0}^{N}\left(a \log \alpha_i - b\alpha_i\right) + c \log \sigma^2 - d\sigma^2, \tag{2.21}$$

ignoring terms that are independent of α and σ^2.

$|C|$ and C^{-1} are computed using the determinant identity and matrix inversion lemma respectively:

$$\log|C| = -\log|\Sigma| - N \log \sigma^2 - \log|A|, \qquad t^T C^{-1} t = \sigma^2 t^T \left(t - \Phi\mu\right) \tag{2.22}$$

Substituting to (2.21) we get:

$$L = \frac{1}{2}\left[\log|\Sigma| + N \log \sigma^2 + \log|A| - \sigma^2 t^T \left(t - \Phi\mu\right)\right] + \sum_{i=0}^{N}\left(a \log \alpha_i - b\alpha_i\right) + c \log \sigma^2 - d\sigma^2. \tag{2.23}$$

Differentiation of (2.23) with respect to $\log \alpha_i$ and equating to zero (see Appendix B), gives

$$\alpha_i^{new} = \frac{1 + 2a}{\mu_i^2 + \Sigma_{ii} + 2b} \tag{2.24}$$

where μ_i is the i-th element of the posterior mean weight and Σ_{ii} is the i-th diagonal element of the posterior weight covariance. μ_i and Σ_{ii} are evaluated from (2.13) and (2.14) respectively using the current estimate for a_{MP}. This update formula for the hyperparameters is equivalent to an expectation-maximization update [24], and so is guaranteed to converge to a local maximum. However, following the procedure described by MacKay in [14] results in the update formula:

$$\alpha_i^{new} = \frac{\gamma_i + 2a}{\mu_i^2 + 2b}, \tag{2.25}$$

with

$$\gamma_i = 1 - \alpha_i \Sigma_{ii}, \tag{2.26}$$

which converges much faster, although convergence cannot be proved theoretically.

The update formula for the noise variance σ^2 is computed through differentiation of (2.23) with respect to $\log \sigma^2$ and equating to zero, which gives

$$\left(\sigma^2\right)^{new} = \frac{\left\|t - \Phi\mu\right\|^2 + 2d}{N - \sum_{i=0}^{N} \gamma_i + 2c}. \tag{2.27}$$

2.5 Making Predictions Using RVM

Given a new test point x_*, the posterior probability of the target value t_* is computed as follows:

$$p\left(t_* \mid t, \alpha_{MP}, \sigma^2_{MP}\right) = \int p\left(t_* \mid w, \sigma^2_{MP}\right) p\left(w \mid t, \alpha_{MP}, \sigma^2_{MP}\right) dw. \tag{2.28}$$

Both terms in the integral are Gaussians, so this is easily computed giving:

$$p\left(t_* \mid t, \alpha_{MP}, \sigma^2_{MP}\right) = N\left(y_*, \sigma^2_*\right) \tag{2.29}$$

with

$$y_* = \mu^T \phi\left(x_*\right), \tag{2.30}$$

$$\sigma^2_* = \sigma^2_{MP} + \phi\left(x_*\right)^T \Sigma \phi\left(x_*\right). \tag{2.31}$$

Equation (2.30) gives the expected value of t_*, which intuitively is equal to $y\left(x_*; \mu\right)$ and equation (2.31) gives the expected error which comprises the sum of two variance components: the estimated noise in the data, and that due to uncertainty in the prediction of the weights. In practice, the parameters w are set to fixed value μ while Σ can be used for computing error bars.

The learning algorithm proceeds by the repeated application of (2.25) and (2.27), along with the update of the posterior statistics Σ and μ from (2.13) and (2.14), until some suitable convergence criterion have been satisfied. Computation of Σ in (2.13) requires the computation of the inverse of an MxM matrix , which is done using Cholesky decomposition with complexity $O\left(M^3\right)$. During re-estimation, many of the hyperparameters α_i are found to tend to infinity, meaning that the distribution of the associated weight is a delta function peaked at zero. These basis functions can be pruned from the model, reducing dramatically the number of basis functions M after a few iterations, therefore computations become significantly faster. However, initially there are N basis functions and computation of Σ is time consuming.

Figure 2.2 demonstrates the relevance vectors of an RVM model fitted on a sample sinusoidal function with additive Gaussian noise.

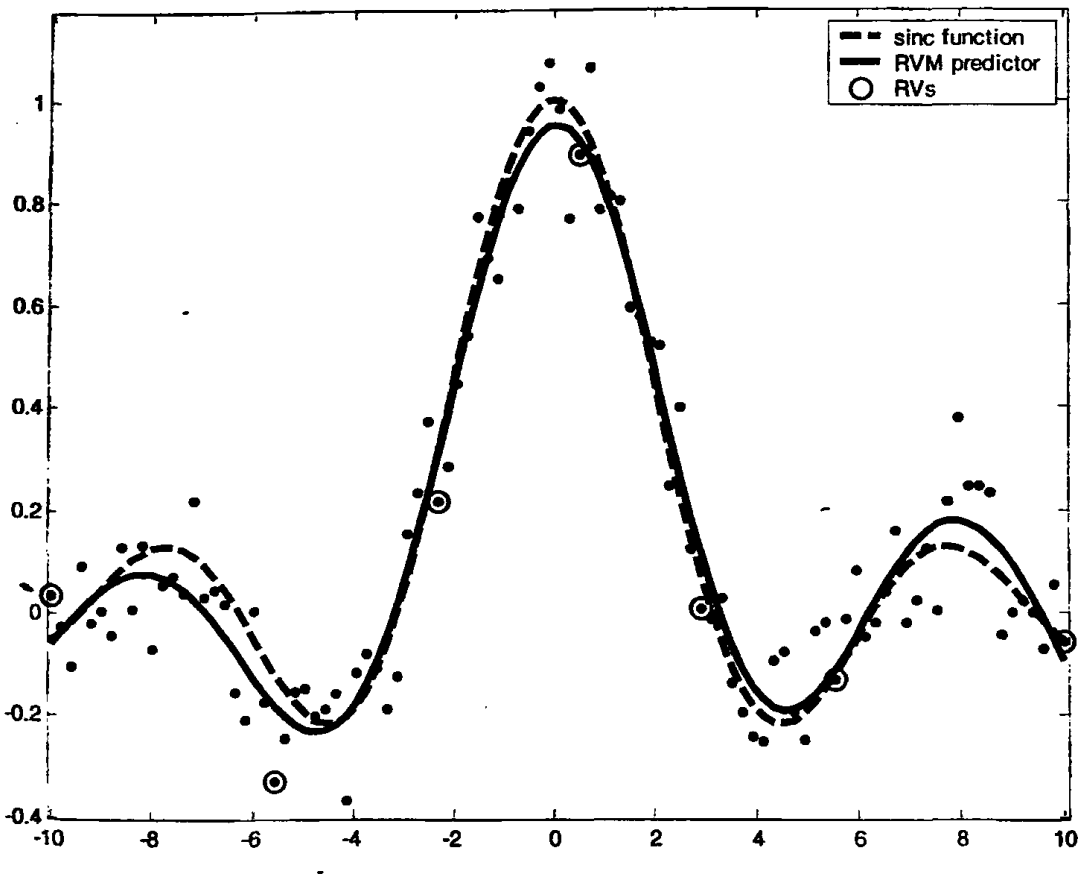


Figure 2.2. Relevance Vectors and the RVM estimate of a sample sinusoidal function

2.6 Discussion

The RVM can also be formulated as a Gaussian process model. Learning in RVM is performed by maximizing the marginal likelihood in (2.19). This can be re-expressed as:

$$p(t) = N(0, C), \tag{2.32}$$

with

$$C = \sigma^2 I + \sum_{i=0}^{N} \alpha_i^{-1} v_i v_i^T . \tag{2.33}$$

and $v_i = (\phi_i(x_1), \phi_i(x_2), ..., \phi_i(x_N))^T$.

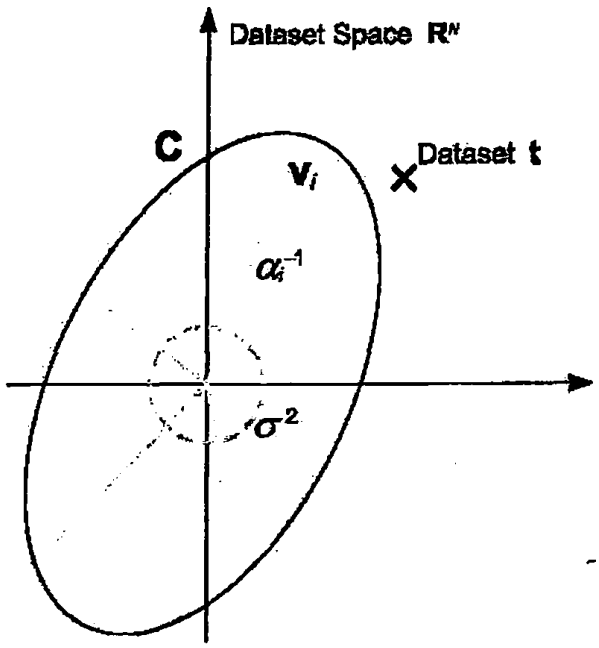


Figure 2.3. Gaussian process view of the RVM.

Figure 2.3 demonstrates the projection of such a model in two dimensions. The shape and size of C can be changed by adjusting the hyperparameters α_i and σ^2. Specifically, adjusting a hypermarameter α_i stretches the shape of C along the direction of the corresponding basis vector v_i, while adjusting the noise variance σ^2 changes the size of C by growing or shrinking it equally in all directions.

When learning a data set t, the objective is to find values for the hyperparameters α_i and σ^2 that best explain the data. This is achieved by maximizing the probability of observing the data set, given in (2.32). If we assume that the data set is to be explained by noise alone, the noise variance σ^2 would be set equal to the variance of the data set, and the C would have the shape of a circle. If we consider adding to the model a basis vector v_i, the shape of C would become an ellipse. Whether the model with the basis function is preferred over the empty model depends on the alignment between the basis function and the observed data set. Generally, in order to maximize the probability of the observations $p(t)$ we have to prune those basis functions for which the data are better explained by the noise variance.

In Bayesian models, like the RVM, the marginalized probability of a given model M, which is given by:

$$p(t|M) = \int p(t|\alpha,\sigma^2,M)\,p(\alpha)\,p(\sigma^2)\,d\alpha\,d\sigma^2, \tag{2.34}$$

can be used as a measure of the generalization performance of the model and can be used in order to select the appropriate basis functions for a given data set. However, computation of the marginal probability $p(t|M)$ can not be done analytically, and approximations are either computationally expensive or ineffective. Instead a cross validation criterion can be used to select the basis functions, which is given by:

$$\hat{\sigma}^2_{LOO} = \frac{\hat{y}^T P(diag(P))^{-2} P\hat{y}}{N}, \tag{2.35}$$

where $P = I_N - \Phi\Sigma\Phi^T$ is known as the projection matrix.

Alternatively, considering basis functions of a given family (e.g. gaussian), we can optimize the parameters of the basis functions such as the width of the kernel, along with the training process. This approach has the advantage that a different width parameter can be associated with each dimension of the data set, but optimization of the extra width parameters is usually significantly time consuming.

2.7 Applications

Although the RVM has been introduced very recently, there are already a number of applications where it has been used. Many of these applications exploit its property to produce sparse models. Other applications use it as a normal regression technique that overcomes the problem of setting the parameters C and ε of the SVM.

2.7.1 Robust regression

In [4] an extension of the RVM algorithm is described that can be used to perform generalization tasks in data sets that may contain several outliers. Outliers are data points that are so noisy that should not be modelled with a standard Gaussian distribution. This robust regression algorithm defines a mixture model with two components. One of the components is an uninformative distribution describing the outlier data points and the other is a RVM model describing the rest data points. The uninformative distribution for the outliers may be either a Gaussian probability function centred at the estimated function value $y(x;w)$, or a uniform distribution between the minimum and maximum of the data points. The method considers a fixed percentage of outlier datapoints and performs a variational inference algorithm to achieve learning.

2.7.2 Image compression

Another application of the RVM algorithm is in constructing approximate image compression algorithms. A suitable set of basis functions has to be chosen, and then a RVM model with these basis functions is trained with the values of the image pixels as target values. The image can then be represented by the values of the weights of the RVM model and it can easily be reconstructed given the weights and the basis functions. Although the weights are as many as the pixels of the image, compression is succeeded because a large percentage of the weights will be set to zero since the model is sparse. Thus, the image can be represented by the non-zero values of the weights. In order to control the compression ratio, the noise variance σ^2 can be set to a suitable fixed value.

2.7.3 Image super-resolution

The RVM algorithm can also be used to construct a high resolution image from a set of lower resolution images, which is known as the super-resolution problem [27]. A RVM model is trained on the set of low resolution images. Then, the high resolution image is constructed by evaluating the value of the trained model at each pixel.

2.7.4 Sparse kernel PCA

In [25] the RVM methodology is used to modify the kernel PCA algorithm so that it produces sparse solutions. The kernel PCA algorithm is a non-linear extension of the well-known Principal Component Analysis (PCA) algorithm. Unfortunately, the principal components obtained with the kernel PCA method are expressed in terms of kernels associated with every training vector. Sparse components can be obtained by approximating the covariance matrix in the feature space by a reduced number of example vectors, using a methodology very similar to the RVM algorithm.

2.7.5 Time-series prediction

The RVM algorithm has also been used for time-series prediction in [3]. In order to find suitable values for the width of the Gaussian basis functions that were used, an iterative optimization was performed. Since all basis functions are assumed to have the same width, this is a one dimensional optimization problem and was solved using a simple direct search method [8]. Direct search is more efficient in this case, because evaluating the derivative of the marginal likelihood with respect to the width of the kernel functions is much more computationally expensive than evaluating the marginal likelihood. Generalization results prove

that the RVM estimator outperforms all the other methods that were tested, including a simple linear model, five nearest neighbour approach and a carefully optimized multilayer perceptron.

3 Incremental Marginal Likelihood Maximization

In this chapter an alternative method for maximizing the marginal likelihood is described [26]. In the original marginal likelihood maximization algorithm, initially all basis functions were assumed in the model, and then many of these basis functions were pruned. Computation of the posterior statistics was very expensive in the first few steps because all basis functions had to be considered, and improved dramatically after the first few iterations, because many basis functions are pruned, since the model is sparse.

The incremental algorithm works by initially defining a model with only one basis function. Then, basis functions are iteratively added until the marginal likelihood is maximized. Of course, since the model is modified when a basis function is added, the values of the hyperparameters may have to be re-estimated after adding some basis function. It is even possible, that a basis function added in some step of the algorithm may eventually have to be deleted. Thus, at each iteration a basis function may be added, removed, or a hyperparameter may be re-estimated. The algorithm in each step selects an operation on a basis function so that the marginal likelihood is increased. Eventually, when no further increase is possible, the algorithm has converged to a local maximum.

When constructing the model by adding basis functions, it is very difficult to choose which basis function should be added. In contrast, when pruning a function from the full model it is much easier to select the basis function that should be pruned. Thus, the incremental algorithm, may add basis functions that will later be deleted, and this causes the algorithm to require more iterations until it converges. However, the posterior statistics are now computed significantly faster, since the model will typically consist of a small number of basis functions. When the noise variance is assumed fixed, the posterior statistics can be computed even more efficiently using the result of the previous iteration and an update formula that requires very little computations.

3.1 Marginal likelihood analysis

Assuming flat priors over the hyperparameters, maximization of the marginal likelihood in equation (2.21) is equivalent to maximizing the quantity

$$L = -\frac{1}{2}\left[\log|C| + t^T C^{-1} t\right]. \tag{3.1}$$

Considering the dependence of L on a single hyperparameter a_i, the matrix $C = \sigma^2 I + \Phi A^{-1} \Phi^T$ can be decomposed into two terms as:

$$\begin{aligned} C &= \sigma^2 I + \sum_{m \neq i} \alpha_m^{-1} \phi_m \phi_m^T + \alpha_i^{-1} \phi_i \phi_i^T \\ &= C_{-i} + \alpha_i^{-1} \phi_i \phi_i^T, \end{aligned} \tag{3.2}$$

where C_{-i} denotes C with the contribution of basis vector i removed.

Now, using the determinant and matrix inversion identities, L can be decomposed as

$$\begin{aligned} L &= -\frac{1}{2}\left[\log|C_i| + t^T C_i t - \log \alpha_i + \log\left(\alpha_i + \phi_i^T C_{-i}^{-1} \phi_i\right) - \frac{\left(\phi_i^T C_{-i}^{-1} t\right)^2}{\alpha_i + \phi_i^T C_{-i}^{-1} \phi_i}\right] \\ &= L(\alpha_{-i}) + \frac{1}{2}\left[\log \alpha_i - \log(\alpha_i + s_i) + \frac{q_i^2}{\alpha_i + s_i}\right] \end{aligned} \tag{3.3}$$

$$= L(\alpha_{-i}) + l(\alpha_i), \tag{3.4}$$

where $L(\alpha_{-i})$ is independent of hyperparameter a_i and

$$l(\alpha_i) = \frac{1}{2}\left[\log \alpha_i - \log(\alpha_i + s_i) + \frac{q_i^2}{\alpha_i + s_i}\right] \tag{3.5}$$

is independent of all hyperparameters except a_i.

The quantities s_i and q_i introduced in equation (3.3) are defined as

$$s_i \triangleq \phi_i^T C_{-i}^{-1} \phi_i \text{ and } q_i \triangleq \phi_i^T C_{-i}^{-1} t. \tag{3.6}$$

The 'sparseness factor' s_i provides a measure of the overlap of basis vector ϕ_i with the other basis vectors, while the 'quality factor' q_i can be written as $q_i = \sigma^{-2} \phi_i^T (t - y_i)$, and thus provides a measure of the alignment of basis vector ϕ_i with the error of the model. Intuitively, in order to maximize the marginal likelihood, the basis functions included in the model should have large quality factors and small sparseness factors. This observation can be used when selecting a basis function to add to the model or remove from it.

Since $L(\alpha_{-i})$ is independent of basis function ϕ_i, the maximum of the marginal likelihood with respect to α_i is found from equation (3.4) by maximizing $l(\alpha_i)$. By differentiating equation (3.5) we find that $l(\alpha_i)$ is maximized when:

$$\alpha_i = \frac{s_i^2}{q_i^2 - s_i}, \qquad \text{if } q_i^2 > s_i, \tag{3.7}$$

$$\alpha_i = \infty, \qquad \text{if } q_i^2 \le s_i. \tag{3.8}$$

Computation of s_i and q_i from equation (3.6) is relatively straightforward since $C = \sigma^2 I + \Phi A^{-1} \Phi^T$ has as many rows as the basis functions in the model, and thus its inverse can be computed. However, it is more efficient to maintain values of

$$S_m \triangleq \phi_m^T C^{-1} \phi_m \quad \text{and} \quad Q_m \triangleq \phi_m^T C^{-1} t, \tag{3.9}$$

and compute s_m and q_m from

$$s_m = \frac{\alpha_m S_m}{\alpha_m - S_m}, \qquad q_m = \frac{\alpha_m Q_m}{\alpha_m - S_m}. \tag{3.10}$$

Using the Woodbury identity, we can efficiently compute S_m and Q_m:

$$S_m = \sigma^2 \phi_m^T \phi_m - \sigma^4 \phi_m^T \Phi \Sigma \Phi^T \phi_m, \tag{3.11}$$

$$Q_m = \sigma^2 \phi_m^T t - \sigma^4 \phi_m^T \Phi \Sigma \Phi^T t. \tag{3.12}$$

Thus, maximization of the marginal likelihood with respect to only one hyperparameter α_i is straightforward from equations (3.7) and (3.8). It is also implied that:

- A basis function ϕ_i inside the model, should be removed if $q_i^2 \le s_i$.
- A basis function ϕ_i excluded from model, should be added if $q_i^2 > s_i$.

This observation is useful when selecting a function to add or remove from the model.

3.2 A sequential hyperparameter optimization algorithm

Based on the observations of the previous section, the following algorithm is suggested, that finds a local maximum of the marginal likelihood.

1. Initialize σ^2.
2. Assume an initial model with only one basis function ϕ_i. From equation (3.7), set $\alpha_i = \dfrac{\|\phi_i\|^2}{\|\phi_i^T t\|^2 / \|\phi_i\|^2 - \sigma^2}$, which maximizes the initial marginal likelihood.
3. Compute Σ and μ from equations (2.13) and (2.14), and s_m and q_m from equation (3.6) for all basis functions.

4. Select a candidate basis vector ϕ_i, either included in the model or not.
5. Compute $\theta_i = q_i^2 - s_i$.
6. If $\theta_i > 0$ and $\alpha_i < \infty$, re-estimate α_i.
7. If $\theta_i > 0$ and $\alpha_i = \infty$, add ϕ_i to the model and update α_i.
8. If $\theta_i < 0$ and $\alpha_i < \infty$, delete ϕ_i from the model and set $\alpha_i = \infty$.
9. Update the noise variance estimate $\sigma^2 = \dfrac{\|t - y\|^2}{N - M + \sum_m \alpha_m \Sigma_{mm}}$.
10. Recompute Σ, μ and all s_m and q_m.
11. If converged terminate, otherwise repeat from step 4. Convergence is detected when the marginal likelihood can no longer be increased by adding or removing a basis function, and re-estimation of the hyperparameters does not significantly change their values.

In step 2 of the algorithm, though any basis function can be selected as initial, the best choice is to choose the one that maximizes the initial marginal likelihood, which is the one that has the largest normalized projection onto the target vector, $\|\phi_i^T t\|^2 / \|\phi_i\|^2$. Alternatively, a bias may be selected.

In step 4 of the algorithm, the selection of a candidate function can be done in several ways. The simplest is to select it at random, or from an ordered list in sequence. However, better performance may be obtained by selecting the basis function that would cause the largest increase in the marginal likelihood. This would require computing in each step the change in the marginal likelihood for each basis function, which can be done efficiently only when the noise variance is fixed.

3.3 Incremental updates

If the noise variance is fixed, quantities S_m, Q_m, Φ, Σ and the change in the marginal likelihood ΔL can be computed using their values at the previous iteration, from the following update formula:

- Adding a new basis function:

$$2\Delta L = \frac{Q_i^2 - S_i}{S_i} + \log\frac{S_i}{Q_i^2}, \tag{3.13}$$

$$\tilde{\Sigma} = \begin{bmatrix} \Sigma + \sigma^{-4}\Sigma_{ii}\Sigma\Phi^T\phi_i\phi_i^T\Phi\Sigma & -\sigma^{-2}\Sigma_{ii}\Sigma\Phi^T\phi_i \\ -\sigma^{-2}\Sigma_{ii}\left(\Sigma\Phi^T\phi_i\right)^T & \Sigma_{ii} \end{bmatrix}, \tag{3.14}$$

$$\tilde{\mu} = \begin{bmatrix} \mu - \mu_i\sigma^{-2}\Sigma\Phi^T\phi_i \\ \mu_i \end{bmatrix}, \tag{3.15}$$

$$\tilde{S}_m = S_m - \Sigma_{ii}\left(\sigma^{-2}\phi_m^T e_i\right)^2, \tag{3.16}$$

$$\tilde{Q}_m = Q_m - \mu_i\left(\sigma^{-2}\phi_m^T e_i\right), \tag{3.17}$$

where $\Sigma_{ii} = \left(\tilde{\alpha}_i + S_i\right)^{-1}$, $\mu_i = \Sigma_{ii}Q_i$ and $e_i = \phi_i - \sigma^{-2}\Phi\Sigma\Phi^T\phi_i$.

- Re-estimating a basis function:

$$2\Delta L = \frac{Q_i^2}{S_i + \left[\tilde{\alpha}_i^{-1} - \alpha_i^{-1}\right]^{-1}} - \log\left\{1 + S_i\left[\tilde{\alpha}_i^{-1} - \alpha_i^{-1}\right]\right\}, \tag{3.18}$$

$$\tilde{\Sigma} = \Sigma - \kappa_j\Sigma_j\Sigma_j^T \tag{3.19}$$

$$\tilde{\mu} = \mu - \kappa_j\mu_j\Sigma_j \tag{3.20}$$

$$\tilde{S}_m = S_m + \kappa_j\left(\sigma^{-2}\Sigma_j^T\Phi^T\phi_m\right)^2 \tag{3.21}$$

$$\tilde{Q}_m = Q_m + \kappa_j\mu_j\left(\sigma^{-2}\Sigma_j^T\Phi^T\phi_m\right) \tag{3.22}$$

where $\kappa_j = \left(\Sigma_{jj} + \left(\tilde{\alpha}_i - \alpha\right)^{-1}\right)^{-1}$.

- Deleting a basis function:

$$2\Delta L = \frac{Q_i^2}{S_i - \alpha_i} - \log\left(1 - \frac{S_i}{\alpha_i}\right), \tag{3.23}$$

$$\tilde{\Sigma} = \Sigma - \frac{1}{\Sigma_{jj}}\Sigma_j\Sigma_j^T \tag{3.24}$$

$$\tilde{\mu} = \mu - \frac{\mu_j}{\Sigma_{jj}}\Sigma_j \tag{3.25}$$

$$\tilde{S}_m = S_m + \frac{1}{\Sigma_{jj}}\left(\sigma^{-2}\Sigma_j^T\Phi^T\phi_m\right)^2 \tag{3.26}$$

$$\tilde{Q}_m = Q_m + \frac{\mu_j}{\Sigma_{jj}}\left(\sigma^{-2}\Sigma_j^T\Phi^T\phi_m\right) \tag{3.27}$$

Following updates (3.24) and (3.25) the appropriate row and/or column is removed from $\tilde{\Sigma}$ and $\tilde{\mu}$.

In the above equations, the integer i is used to index the single basis function for which α_i is to be updated, and the integer j to denote the index within the current basis, that corresponds to i. Updated quantities are denoted by a tilde (e.g. $\tilde{\alpha}_i$). These update formulas are much more efficient than recomputing these quantities on each iteration.

3.4 Conclusions

The incremental algorithm is computationally much more efficient than the initially approach. Specifically, the computation of the posterior weight distribution initially needed $O(M^3)$ operations, where M is the total number of basis functions, while the incremental algorithms needs $O(M_*^3)$, where M_* is the number of basis functions included in the model. These are expected to be significantly less than the total number of basis functions.

The improved performance of the incremental algorithm is also proved experimentally in [16]. Figure 3.1 shows the run time required for different numbers of training examples. For instance, given 1000 training examples, the initial algorithm needs about 18 times more run-time than the incremental algorithm. Also, it is shown experimentally that the performance of the two algorithms, in terms of generalization error, is approximately the same, and that the incremental algorithm produces more sparse models.

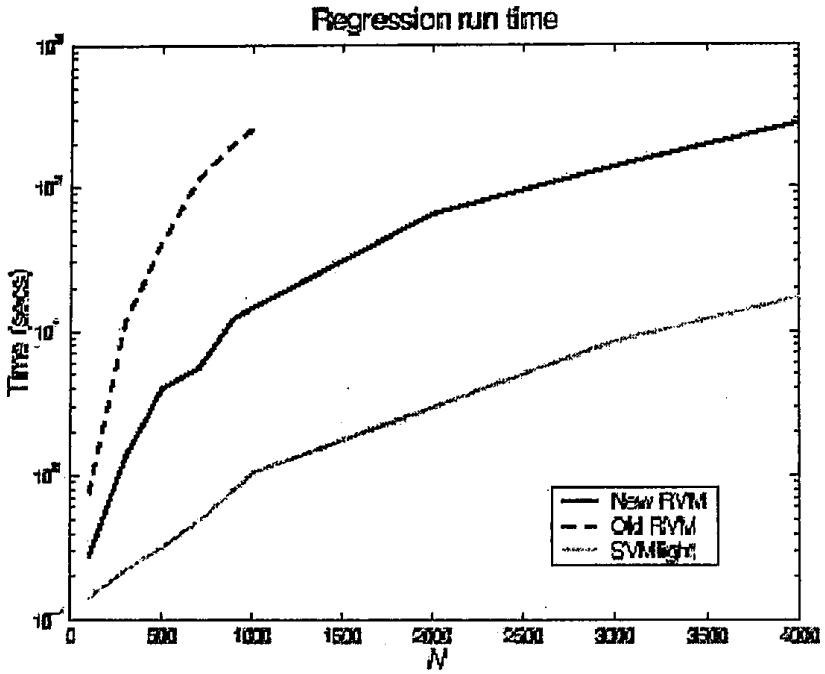


Figure 3.1 . Runtime comparison between the initial and incremental RVM algorithm and an implementation of the SVM learning algorithm

4 Functional Magnetic Resonance imaging

4.1 Introduction

Functional neuroimages are brain images, in which the intensity of the image at each point measures the neural activity in that brain area. 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. However, due to safety considerations, these have to be obtained from different subjects. Generally, the subjects should be chosen to be of the same age and gender and with similar medical history in order to eliminate extraneous factors that might produce irrelevant results. Of course it is impossible to perform a perfectly controlled study in practice.

In the Image Acquisition step several scans of the brain of each subject are obtained. 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 known as 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 the Talairach brain atlas [21] 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: 1) characterization of the spatio-temporal activation pattern induced in the brain by the stimulus, and 2) 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: 1) the brain is always active therefore the experiment must be designed carefully to isolate the effect of the stimulus, 2) the degree of activation with respect to the baseline state can be very slight and difficult to detect, 3) the images often suffer from poor quality (low resolution, blur and noise), 4) anatomical and functional differences among subjects introduce additional errors, and 5) very little prior knowledge is available about human brain activity.

These challenges have inspired to develop an arsenal of image processing and statistical tools to detect and establish statistical significance of studies. The predominant approach [5], [29] based on the t-test from statistics [10], uses pixel-wise comparisons between images of the control and test states of the brain to detect the local changes in activity. Newer methods, which have gained lower acceptance so far, are based on pairwise pixel correlations. The recent advances in signal processing have opened a number of possibilities for further developments in this field.

We consider a two-state activation study, in which activation and control state images are obtained and used to locate activated regions. We consider only two-dimensional images, but the analysis is the same for three-dimensional images. We model the images of the brain in the control and activation states, respectively, as:

$$g_j^{(c)}(x,y) = b(x,y) + n_j^{(c)}(x,y),$$
$$g_j^{(a)}(x,y) = b(x,y) + s(x,y) + n_j^{(a)}(x,y), \quad (4.1)$$

for $j = 1,\ldots,N$, where, at each spatial location (x,y) in the brain, the activation pattern we wish to determine is $s(x,y)$. The baseline value is $b(x,y)$ and $n_j^{(c)}(x,y)$ and $n_j^{(a)}(x,y)$ are the imaging noise contributions for the two cases. We model the noise as additive colored Gaussian noise, the covariance of which is proportional to the value of the baseline image at the corresponding pixel and is assumed known [20]. This noise is given by $n_j^{(c,a)}(x,y) \sim N(0,C_n)$ where C_n is the covariance matrix of the noise. We define the mean activation image as:

$$g^{(s)}(x,y) = \sum_{j=1}^{N}\left(g_j^{(a)}(x,y) - g_j^{(c)}(x,y)\right). \tag{4.2}$$

Then, from equation (4.1) follows that:

$$g^{(s)}(x,y) = s(x,y) + n^{(s)}(x,y) \tag{4.3}$$

where $n^{(s)}(x,y) = \sum_{i=1}^{N}\left[n^{(a)}(x,y) - n^{(c)}(x,y)\right]$ is the average noise in the images and $n^{(s)}(x,y) \sim N(0,\frac{2}{N}C_n)$.

In the activation state, the image differs from the control state only by addition of the activation pattern.

4.2 t-test

The most popular approach to detect activations in an activation study as the one described above uses the t-test. The value of the t-statistic image in pixel j is calculated as follows:

$$t(x,y) = \frac{g^{(s)}(x,y)}{\hat{S}(x,y)/\sqrt{N}} \tag{4.4}$$

where $\hat{S}(x,y)$ is a standard deviation estimate for pixel (x,y). It can either be a single pixel estimate, given by

$$\hat{S}(x,y) = \left[\frac{1}{N-1}\sum_{i=1}^{N}\left\{\left(g_i^a(x,y) - g_i^c(x,y)\right) - g^s(x,y)\right\}^2\right]^{1/2}, \tag{4.5}$$

or a pooled variance estimate, given by

$$\hat{S} = \left[\frac{1}{xy}\sum_{x=1}^{X}\sum_{y=1}^{Y}S(x,y)^2\right]^{1/2}. \tag{4.6}$$

Then, detection of activations is performed, by thresholding the t-statistic image, as

$$|t(x,y)| > T, \tag{4.7}$$

where T is a threshold. Selecting a very small value for the threshold results in detecting successfully most activations, but will probably lead to detecting false activated regions as well. On the other hand, a very large value of the threshold T will fail to detect all activations. Generally, there is a trade-off between the probability of correctly detecting an activated pixel (probability of detection), and the probability of falsely detecting as activated a non activated pixel (probability of false alarm).

4.3 Singular Value Decomposition

Another method for detecting activations is the Singular Value Decomposition (SVD) method. It proceeds by computing the vector direction along which the data set exhibits the greatest variance, known as the principal eigenimage. Activated regions are expected to have the greatest variance, and they can be detected by thresholding the principal eigenimage. A matrix X containing the data is formed, as

$$X = \left[x_1^{(a)} \ldots x_N^{(a)} x_1^{(c)} \ldots x_N^{(c)} \right], \tag{4.8}$$

where $x_i^{(a)}$ and $x_i^{(c)}$ are vectors containing the intensity at all pixels of images $g_i^{(a)}$ and $g_i^{(c)}$ respectively. The SVD of X is then computed, i.e.

$$X = \tilde{U}\tilde{\Lambda}\tilde{V}^T. \tag{4.9}$$

The columns of $\tilde{U}$ are eigenimages of the pooled covariance matrix of the data set. The eigenimage corresponding to the largest singular value in $\tilde{\Lambda}$ serves as the test-statistic image for this method. Detection performance is improved by normalizing the data matrix X, using either row-centering, column-centering, or double-centering.

Row centering of matrix X_0 is defined as:

$$X = X_0 - [c \ldots c], \tag{4.10}$$

where the elements of column vector c are

$$c_j = \frac{1}{2N} \sum_{i=1}^{2N} [X_0]_{j,i}, j = 1, \ldots, J, \tag{4.11}$$

J is the number of pixels of each image, and $[X]_{i,j}$ denotes the ij element of X. Each row of the row-centered data matrix sums to zero.

Similarly, column centering of matrix X_0 is defined by

$$X = X_0 - \begin{bmatrix} r \\ \dots \\ r \end{bmatrix}, \tag{4.12}$$

where the elements of the row vector r are

$$r_j = \frac{1}{J}\sum_{i=1}^{J}[X_0]_{j,i}, j = 1,\dots,2N. \tag{4.13}$$

and each column the column-centered data matrix sums to zero.

- Double centering consists of the successive application of both row centering and column centering.

4.4 Covariance thresholding and correlation-coefficient thresholding

Unlike the t-test that detects activations based on the mean activation differences of each location, covariance thresholding and correlation-coefficient thresholding detect activations by finding regions where the activation signal is highly correlated. Methods like these, which focus exclusively on the analysis of the correlation structure of the activation signal, are very important in fMRI studies. The main advantage of such methods is that they require one set of images in the activation state, and do not need images in a control state like most methods.

The correlation coefficient between pixels p and q is defined as:

$$R_{p,q} = \frac{\sum_{i=1}^{N}\left(x_{i,p}^{a} - \hat{\mu}_p\right)\left(x_{i,q}^{a} - \hat{\mu}_q\right)}{\left[\sum_{i=1}^{N}\left(x_{i,p}^{a} - \hat{\mu}_p\right)^2\right]^{1/2}\left[\sum_{i=1}^{N}\left(x_{i,q}^{a} - \hat{\mu}_q\right)^2\right]^{1/2}}, \tag{4.14}$$

where $\hat{\mu}_p$ is the sample mean of pixel p.

Detection is performed by thresholding the test-statistic image, defined as:

$$r_p = \max_{q \in I_q}\left\{R_{p,q}\right\} \tag{4.15}$$

where I_q is the set of indices of pixels whose distance from the qth pixel is sufficiently large, so that noise between these two pixels is not correlated. The value of the test-statistic image in a pixel is equal to the greatest correlation that this pixel has with any other pixel. Since activated regions are highly correlated they can be detected by thresholding the correlation-coefficient test-statistic image.

Covariance thresholding, is essentially similar to the correlation-coefficient method, but omits the sample standard deviation factors used for normalization in the denominator of (4.14).

This produces improved detection performance because it eliminates variance caused by these noisy standard deviation estimates.

4.5 Fisher discriminant applied to the SVD of the data matrix

In this method, a Fisher discriminant vector, which serves as the test-statistic image, is computed from N activation-state and N control-state images. The Fisher discriminant is derived from 2N feature vectors, each consisting of a subset of the SVD coefficients for an image vector in the data matrix X . Using a subset of the SVD coefficients as a feature vector serves to reduce dimensionality and to regularize the solution [11]. The number of eigenvectors to include was selected in two ways: 1) fixed at one-third the total number of eigenvectors (Fisher-Fixed); or 2) using a Scree plot and the graphical Cattell-Jaspers rule (identified as Fisher-Adaptive [9]). This graphical method uses the empirical observation that, beyond the significant low-order values, the eigenvalue spectrum is usually roughly linear. The method seeks to determine the lowest-order eigenvalue that does not fit a linear model of the high-order portion of the eigenvalue spectrum.

The basis vectors of the SVD feature space are the eigenimages in $\tilde{U}$ that correspond to the largest singular values in $\tilde{\Lambda}$. This subset of the eigenimages and their corresponding singular values will be denoted by matrices U and Λ, respectively. The coordinates of the images in this reduced-dimension feature space are contained in the columns of the matrix Z given by

$$Z = U^T X = U^T U \Lambda V^T = \Lambda V^T \tag{4.16}$$

Let z_i^a and z_i^c denote the feature vectors for images x_i^a and x_i^c, respectively, and let $y_i^a = {z_i^a}^T d_z$ and $y_i^c = {z_i^c}^T d_z$ denote, respectively, the projections of z_i^a and z_i^c onto vector d_z. The Fisher discriminant is the vector d_z that yields the greatest separation of the values $y_i^a, i = 1,...,N$, from the values $y_i^c, i = 1,...,N$, as measured by the Fisher ratio [16]:

$$\frac{N\left(\bar{y}^a - \bar{y}\right)^2 + N\left(\bar{y}^c - \bar{y}\right)^2}{\sum_{i=1}^{N}\left(\bar{y}_i^a - \bar{y}^a\right)^2 + \sum_{i=1}^{N}\left(\bar{y}_i^c - \bar{y}^c\right)^2}. \tag{4.17}$$

In (4.17) $\bar{y}^a$ and $\bar{y}^c$ are the sample of $\bar{y}_i^a$ and $\bar{y}_i^c$, respectively, and $\bar{y}$ is the grand mean of the two groups. Once the feature-space Fisher discriminant d_z has been computed, it must be

transformed back into image space to take the form of a test-statistic image d_x . The transformation is performed as follows:

$$d_x = Ud_z \tag{4.18}$$

4.6 General Linear Model (SPM)

The Statistical Parametric Mapping (SPM) [6], which is a popular application for analysis of functional neuroimages, is based on the general linear model. The general linear model considers an arbitrary number of K effects during the experiment. The general linear model is defined as:

$$X = G\beta + \varepsilon. \tag{4.19}$$

Here, we assume that we have obtained N images, each of which contains M pixels. X is the $N \times M$ data matrix whose element X_{ij} at position i, j is the value of the j-th pixel of the i-th image, G is the $N \times K$ design matrix whose i-th row determines the effects under which the i-th image was taken, β is the $K \times M$ parameter matrix whose element β_{ij} at position i, j is an unknown parameter describing how the pixel j is affected by effect i, and ε is a matrix of independent and identically distributed error values.

The use of a general linear model allows studying complex experiments, which may combine more than one stimuli simultaneously. The design matrix G has one column for each stimulus, designating for each image if this stimulus is present or not. If the stimulus has many levels of intensity, a real number can be used to express how intense the stimulus was when the image was taken, otherwise an integer in $\{0,1\}$ can be used. The design matrix can involve many other design parameters as well. For instance, if the study involves many images and we are interested in the differences between the subjects, the design matrix may also have columns that designate from which subject each image was taken. Generally, the parameters of the design matrix are chosen according to the design of the experiment and the effects that are being studied.

Assuming that the noise is normally distributed the maximum likelihood solution of (4.19) is equivalent to the least squares solution and can be found by:

$$\hat{\beta} = \arg\min_{\beta} (X - G\beta)^T (X - G\beta), \tag{4.20}$$

which gives:

$$\hat{\beta} = \left(G^T G\right)^{-1} GX . \tag{4.21}$$

Then, in order to test whether stimulus described by the k-th parameter β_k is present at some pixel j of the image we test the for the hypothesis

$$\begin{aligned} H_0 &: \beta_k = 0 \\ H_1 &: \beta_k \neq 0 \end{aligned} \tag{4.22}$$

Thresholding the likelihood ratio of the two hypotheses:

$$L = \frac{p\left(X; H_1\right)}{p\left(X; H_0\right)}, \tag{4.23}$$

is equivalent to thresholding:

$$t_j = \frac{c\hat{\beta}_j}{\varepsilon_j}, \tag{4.24}$$

where c is a vector that describes the tested stimulus, and ε_j is an estimate of the standard deviation of voxel j.

Consider for example a simple experiment consisting of one image set under a control state and another image set under an activated state. In this case the design matrix would consist of two columns, one of which would correspond to the control state image, and the other would correspond to the activation signal induced by some stimulus.

$$G = \begin{bmatrix} g_1 & g_2 \end{bmatrix} \tag{4.25}$$

with

$$g_{1i} = 1, \tag{4.26}$$

since the control signal is present in all images and

$$g_{2i} = \begin{cases} 1 & \text{if } X_i \text{ is in the activation state} \\ 0 & \text{otherwise} \end{cases} \tag{4.27}$$

In order to test if activation is present at voxel j we define $c = \begin{bmatrix} -1 & 1 \end{bmatrix}$. Then the test-statistic is given by (4.24) and it can be proven that it is equivalent to the t-test defined in (4.4). Furthermore, the correlation coefficient thresholding can be considered as a special case of a general linear model.

4.7 Markov Chain Monte Carlo

In the general linear model described above, decisions about the activation state of a voxel are made based on intensity values of that voxel only. However, better performance can be achieved if the intensity values of the neighboring voxels are also considered when deciding about the activation state of a voxel, since activated regions usually consist of several neighboring voxels. In [13] a model is suggested that considers the spatial correlation of the activation signal. Specifically, it is modeled as the superposition of K two dimensional circular functions, as

$$s(x,y\,|\,\theta)=\sum_{i=1}^{K} a^i h\left(x,y;c_x^i,c_y^i,r^i\right), \tag{4.28}$$

where $h\left(x,y;c_x^i,c_y^i,r^i\right)$ is a circular function centered at location $\left(c_x^i,c_y^i\right)$, with diameter r^i and activation strength a^i. The activation function $s(x,y)$ is determined by the following set of parameters:

$$\theta=\left\{K,\bigcup_{i=1}^{K} a^i,c_x^i,c_y^i,r^i\right\}. \tag{4.29}$$

The method proceeds by computing the maximum a posteriori (MAP) estimate $\hat{\theta}$ of the parameters θ of the activation signal:

$$\hat{\theta}=\arg\max_{\theta} L(g\,|\,\theta)\,p(\theta), \tag{4.30}$$

where $p(\theta)$ is the prior distribution of θ, g is a collection of the activation-state and control-state images and $L(g\,|\,\theta)$ is the likelihood of observing data g given the parameters in θ. Assuming that noise is independent across observed images, equivalently we minimize the negative logarithm of (4.30), which gives:

$$\hat{\theta}=\arg\min_{\theta}\left\{\sum_{j=1}^{N}\left[g_j^{(a)}-g_j^{(c)}-s(\theta)\right]^T C_n^{-1}\left[g_j^{(a)}-g_j^{(c)}-s(\theta)\right]-\log\left(p(\theta)\right)\right\}. \tag{4.31}$$

The location, size and amplitude parameters of one source, are assumed to be independent of each other, and also the parameters of each source are assumed independent to the parameters of each other source. The prior over the number of sources K and the prior over the locations of the activations $\left\{\left[c_x^i \quad c_y^i\right]\right\}_{i=1}^{K}$ are assumed uniform. The diameter r and amplitude a of an activation source are assigned truncated Gaussians prior distributions.

The MAP estimate of the parameter vector θ is difficult to be computed because the length of the parameter vector is not fixed. However, samples from the posterior distribution $p(\theta|g)$ can be generated using the Reversible Jump Markov Chain Monte Carlo (RJMCMC) [13] algorithm. Then the one that has the maximum posterior probability is chosen. Because the number of sources is unknown, the algorithm has to be able to explore a space with varying dimensionality. The RJMCMC algorithm operates similarly to the standard MCMC algorithm, generating samples from a given proposal distribution and then accepting or rejecting them in order to approximate the target distribution. Furthermore, it uses 'jumps' to move between spaces of different dimensionality.

The algorithm proceeds by randomly choosing one of the following steps at each iteration: 1) creation ('birth') of a new source, 2) deletion ('death') of a source, 3) merge of two sources into one ('merge'), split of a source into two ('split'), or improving the estimate of the parameters without changing the parameter vector length ('update'). In each iteration of RJMCMC a new parameter sample vector is proposed. The acceptance ratio that governs the probability of acceptance of a proposed sample at iteration I is:

$$R' = \frac{\pi(\theta^{l+1}, g)}{\pi(\theta^{l}, g)} \frac{\zeta(\theta^{l}, step^{-1} | \theta^{l+1}, g)}{\zeta(\theta^{l+1}, step | \theta^{l}, g)} \left| \frac{\partial \theta^{l+1}}{\partial \theta^{l}} \right|, \tag{4.32}$$

where π is the so called target distribution from where we want to sample. In our application this is the posterior distribution. Therefore, the target ratio term is composed of likelihood-ratio and prior-ratio terms:

$$\frac{\pi(\theta^{l+1}, g)}{\pi(\theta^{l}, g)} = \frac{L(\theta^{l+1} | g)}{L(\theta^{l} | g)} \frac{p(\theta^{l+1})}{p(\theta^{l})}, \tag{4.33}$$

where θ^{l} is a current value of parameter vector at iteration l. In (4.32) $\zeta(\theta^{l+1}, step | \theta^{l}, g)$ is the probability that θ^{l+1} will be proposed by selecting a certain step given the set of the chain θ^{l} and the observations g. Finally, step^{-1} denotes the inverse of a step, e.g. birth^{-1} = death. The proposal ratio in (4.32) also consists of two terms:

$$\frac{\zeta(\theta^{l}, step^{-1} | \theta^{l+1}, g)}{\zeta(\theta^{l+1}, step | \theta^{l}, g)} = \frac{q(\theta^{l} | \theta^{l+1}, g, step^{-1})}{q(\theta^{l+1} | \theta^{l}, g, step)} \frac{p_s(step^{-1} | \theta^{l+1})}{p_s(step | \theta^{l})}, \tag{4.34}$$

where $q\left(\theta^{l} \mid \theta^{l+1}, g, step^{-1}\right)$ is the proposal distribution from which new parameters are sampled and $p_s\left(step \mid \theta^{l}\right)$ is the probability that, out of five possible types of steps, a particular one will be chosen given the current state of the chain.

All steps are equi-probable and the following exceptions apply: (a) if the current number of sources on θ^{l} is zero, only a birth step is possible, (b) if the current number of sources in θ^{l} is one, a merge step is not possible, (c) if the current number of sources in θ^{l} is equal to some predefined maximum number, birth and split steps are not possible.

Any choice of proposal distributions q will produce samples from the desired target distribution, but the convergence time of the chain will not be the same for every choice.

To create a new source in the birth step, the location, diameter and amplitude parameters are sampled independently:

$$q_b(\omega \mid \theta, g) = q_c\left(c_x, c_y \mid \theta, g, birth\right) q_a\left(a \mid \theta, g, birth\right) q_a\left(r \mid \theta, g, birth\right) \tag{4.35}$$

where $\omega = \begin{bmatrix} c_x & c_y & a & r \end{bmatrix}$ are parameters describing a new source. Location parameters was sampled from the distribution that is proportional to the current residual as proposed in [19]:

$$q_c\left(c_x, c_y \mid \theta^{l}, g, birth\right) \propto \frac{1}{N} \sum_{j=1}^{N} \left(g_j^{(a)}\left(c_x, c_y\right) - g_j^{(c)}\left(c_x, c_y\right) - s\left(c_x, c_y \mid \theta^{l}\right)\right)^2 I_{\left(c_x, c_y, \theta^{l}\right)} \tag{4.36}$$

where $I_{\left(c_x, c_y, \theta^{l}\right)}$ is an indicator function equal to zero if location $\left(c_x, c_y\right)$ is already a center of a source defined by θ^{l} and $s\left(c_x, c_y \mid \theta^{l}\right)$ is the value of the signal defined by θ^{l} at location $\left(c_x, c_y\right)$. Diameter r and amplitude a are sampled from proposal distributions equal to their prior distributions. In the death step each source has an equal chance to be proposed for deletion:

$$q\left(\omega \mid \begin{bmatrix} \theta_u^{l} & \omega \end{bmatrix}, g, death\right) = \frac{1}{K^{l}}, \tag{4.37}$$

where ω is a parameter vector of a source to delete, θ_u^{l} are the other parameters not to be changed and K^{l} is the number of sources at iteration l. For both birth and death steps the determinant of the Jacobian is equal to 1.

In split steps, one source is replaced with two new sources, located near the old one. In merge steps, two sources are replaced with one source located between the two old sources.

The update step makes no change in the parameter space dimensionality. Its purpose is to improve the current estimate of the parameters. The parameters are updated one by one,

dividing the update step in the number of sub-steps equal to the total current of parameters to update, $4K$. At each of these sub-steps an update is proposed for only one parameter and the change is accepted according to the acceptance ratio defined in (4.32). To update a location we sampled again from the distribution proportional to the residual but we restricted the possible choices only to the neighborhood of the current value:

$$q_c\left(c_x, c_y \mid \theta^l, g, update\right) \propto \frac{1}{N}\sum_{j=1}^{N}\left(g_j^{(a)}\left(c_x, c_y\right) - g_j^{(c)}\left(c_x, c_y\right) - s\left(c_x, c_y \mid \theta^l\right)\right)^2 I_{\left(c_x, c_y, c_x^l, c_y^l, c\right)} \qquad (4.38)$$

where $I_{\left(c_x, c_y, \theta^l\right)}$ is an indicator function equal to one if location $\left(c_x, c_y\right)$ is in the neighborhood of the location $\left(c_x^l, c_y^l\right)$ being updated, ε is the parameter defining the neighborhood and i is the index of the source for which the parameters are being updated. The proposed value for the update of the diameter and amplitude are sampled from their respective prior distributions centred around the current value of the parameter being updated.

Once the samples are generated the one that has the maximum posterior probability is chosen.

5 Relevance Vector Machine Analysis of functional neuroimages

5.1 Signal Model

In this chapter we propose the use of the Relevant Vector Machine (RVM) model in detecting activations in functional neuroimages. We consider a two-state neuroimaging study, with one set of images in the activation state denoted by $g^{(a)}$ and another set of images in the control state denoted $g^{(c)}$ which are modeled as:

$$\begin{aligned} g^{(a)}(x) &= b(x) + n^{(a)}(x) \\ g^{(a)}(x) &= b(x) + s(x;w) + n^{(c)}(x) \end{aligned} \tag{5.1}$$

where $b(x)$ is the value of the baseline image in the pixel x, $n^{(a)}(x)$ and $n^{(c)}(x)$ respectively are the noise components in the activation and control states and $s(x;w)$ is a parametric representation of the activation signal. The images in the activation and control state differ only in the activation signal $s(x;w)$.

The noise is assumed to be independent between the images and zero mean Gaussian distributed:

$$n \sim N(0, C_n). \tag{5.2}$$

Furthermore, the noise variance is assumed proportional to the baseline image, and neighboring pixels are assumed correlated. The noise covariance matrix C_n is assumed known. This noise model is proposed in [20].

A simple estimate of the activation signal can be obtained by computing the mean difference between the images in the control and activation states:

$$\hat{s}(x) = \frac{1}{N}\left(g^{(a)}(x) - g^{(a)}(x)\right) \tag{5.3}$$

where N is the total number of images.

This simple estimate works on each pixel independently and does not consider any dependencies of the activation signal between neighboring pixels. Instead, we consider modeling the activation signal using a RVM model. This corresponds to the superposition of kernel functions, one centered at each pixel x_i of the image, given by:

$$s(x;w)=\sum_{i=1}^{M^2} w_i K(x,x_i), \tag{5.4}$$

where the number of pixels in each image is M^2, $K(x,x_0)$ is some kernel function and w_i is the amplitude of the kernel centered at x_i. The weights w_i are assumed to be zero-mean Gaussian distributed:

$$w_i \sim N(0,\alpha_i), \tag{5.5}$$

and their variances α_i are assumed Gamma distributed:

$$\alpha_i \sim Gamma(a,b). \tag{5.6}$$

This prior formulation that is suggested by the RVM model leads to sparse representations of the activation signal, meaning that most of the weights should be set to zero after the training process. Ideally, if the shape of the kernel functions is identical to the shape of the activations, we would expect as many non-zero weights as the activation sources are. However, even if the shape of some activation is significantly different from the shape of the kernel functions, it can be effectively approximated using a combination of kernel functions. Though any kernel function can be used, in practice we assume the commonly used Gaussian kernel functions, of the form:

$$K(x,x_0)=e^{-\frac{1}{2q}\|x-x_0\|^2}. \tag{5.7}$$

The value of the width q of the kernel functions can be specified to adjust the smoothness of the model. Selecting the appropriate value for the width is significant in order to achieve good performance. Cross validation is a method that can be used to select the values of the width parameter that provide the best performance. Alternatively, the width parameter can be optimized during the training process, but this requires significant additional computations.

5.2 Bayesian Inference

Following the Bayesian inference procedure described by the RVM, we compute the posterior weight distribution:

$$p(w\mid t,a)=\frac{p(t\mid w)\,p(w\mid a)}{p(t\mid a)}=N(\mu,\Sigma), \tag{5.8}$$

where

$$\Sigma = \left(\Phi^T C_n \Phi + A\right)^{-1}$$
$$\mu = \Sigma \Phi^T C_n t \tag{5.9}$$

and the marginal likelihood:

$$p(t \mid a) = \int p(t \mid w) p(w \mid a)\, dw = N(0, C) \tag{5.10}$$

with

$$C = C_n + \Phi A^{-1} \Phi^T \tag{5.11}$$

Maximization of the marginal likelihood is achieved either by initially assuming a full model and iteratively pruning the basis functions that are irrelevant, or by initially assuming an empty model and iteratively adding basis functions. Both methods achieve similar results, but the second requires significantly less computations and therefore less run-time.

Ideally, if the shape of the activation is effectively approximated by the shape of the basis functions, we would expect the RVM algorithm to prune all basis functions except those located on an activated region. However, practically it is possible for some basis functions on a non-activated region to be maintained in the model but the associated weight will be set to a small value. Although it is not possible to find the type of kernel functions that gives the best performance, there are methods such as cross-validation [2][1], that give a measure of the performance of a given model, and thus provide a way to compare the performance of a set of different basis functions and choose the best of these.

5.3 Results

To evaluate the method we used a simple phantom whose properties were derived from a positron emission tomography (PET) neuroimaging study, but are also representative of whole-brain, blood-oxygenation-level-dependent (BOLD) functional magnetic resonance imaging (fMRI) studies that have been spatially smoothed [20]. Figure 1 (a) shows the average of ten simulated activation patterns. The location and amplitude of the activation were varied randomly from image to image to represent physiological variability between subjects or scans. Figure 1 (b) shows the average of ten simulated "activated" images. Figures 1 (c) and (d) show the activation pattern estimated by the RVM method for different values of the hyperparameters.

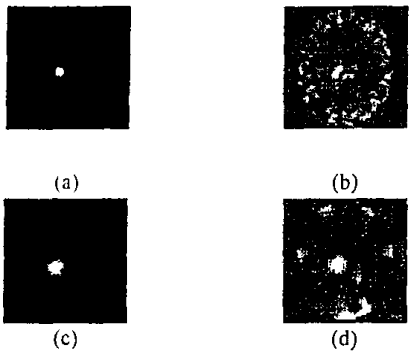


Figure 1. (a) - the average of 10 simulated activation patterns, (b) - the average of 10 "activated" images, (c) - the activation pattern estimated by RVM with α=1, β=0, (d) – the activation pattern estimated by RVM with $\alpha=10^{-2}$, β=0

Two types of activation studies were performed, one where activation was actually present and one where there was no activation. The purpose of this was to simulate both null and alternative hypothesis conditions for purposes of measuring a ROC curve. Both types of activation studies were simulated 50 times to get a good statistical estimate of the ROC curve.

A statistical parametric map (SPM) was generated for each activation study and then the value of the pixel where the activation was sometimes present was thresholded to decide between the two hypotheses. The SPM was generated by computing the value of the likelihood ratio for the two hypotheses at each pixel of the image

$$\frac{p\left(g^{(s)}(x,y)\,|\,\hat{s}(x,y),H_1\right)}{p\left(g^{(s)}(x,y)\,|\,\hat{s}(x,y),H_0\right)} = \frac{\hat{s}(x,y)}{\sigma(x,y)/\sqrt{N}} \tag{5.12}$$

where $\hat{s}(x,y)$ is the estimate of the activation signal, and $\sigma(x,y)$ is the standard deviation of the imaging noise at location (x,y).

To evaluate and compare performance, we used the area under the ROC curve for false positive fraction between 0.0 and 0.1 because this is the most useful range of operating points. A comparison is shown in with various methods listed in order of performance. These methods are described in detail in Table I. The RVM provided the second best performance in the simple case we considered and was only slightly worse than the RJMCMC approach in [13]. However, RJMCMC being a random sampling method is notoriously time consuming. Furthermore, it is difficult to establish when the chain has converged in order to stop sampling.

Method	Area under the ROC curve*
RJMCMC	0.0818
RVM	0.0732
SVD thresholding, column centering	0.0624
t-test, pooled variance estimate	0.0439
SVD thresholding, Fisher, row centering	0.0387
SVD thresholding, Fisher, column centering	0.0318
SVD thresholding, Fisher, double centering	0.0311
SVD thresholding, row centering	0.0252
t-test, single-pixel variance estimates	0.0242
SVD thresholding, double centering	0.0160

Table I. Comparison of performances

In order to find an appropriate value for the width parameter of the Gaussian kernels we used the leave one out cross validation criterion:

$$\hat{\sigma}^2_{LOO} = \frac{\hat{y}^T P(diag(P))^{-2} P\hat{y}}{N} \tag{5.13}$$

where $P = I_N - \Phi\Sigma\Phi^T$ is known as the projection matrix, and Φ is the design matrix consisting of the kernel functions evaluated at each point of the training set. Using cross-validation to select the appropriate kernel width for each experiment, lead to slightly increasing the measured ROC area, compared to using the same width for all experiments.

We also performed a comparison study between the initial iterative algorithm and the improved incremental algorithm. Both algorithms gave similar results in detection performance but the incremental algorithm converged significantly faster. Table II shows the runtime required by each algorithm to compute an estimate of the activation signal on a 30×30 image. The important advantage of the incremental algorithm is that it can be run on larger images, for example 60×60 images, in reasonable time.

Iterative RVM (30x30)	12 sec
Incremental RVM (30x30)	5 sec
Incremental RVM (60x60)	196 sec

Table II. Comparison of the iterative and incremental algorithms.

In the experimental results shown here, C_n was assumed known. We also found that the best detection performance occurred when using a completely uninformative prior which is defined by setting the hyperparameters $\alpha = \beta = 0$. However the resulting RVM fit is quite stable over a large range of values for the hyperparameters, see for example Fig. 1c and 1d.

6 Conclusions

The Relevance Vector Machine (RVM) is a flexible model used for classification and regression problems. The output of the model is a sparse linear combination of some basis functions that are assumed fixed and that depend only on the input of the model. Inference is achieved using a Bayesian framework and sparseness of the model is established by assigning appropriate prior distributions to the parameters of the linear combination. These priors are a type of automatic relevance determination priors (ARD) [15] and automatically prune basis functions that are irrelevant.

We studied the use of the RVM model to estimate the activation signal in functional neuroimages. An artificial phantom was used to evaluate the method and compare it with existing approaches. Two sets of images were constructed, one containing some activated regions and one without activations. Then, the RVM algorithm was used to estimate the activation signal in each image, and a hypothesis test was formulated to decide whether a pixel was activated or not. An ROC curve was constructed based on the percentage of correctly and falsely identified activations and used for evaluation of the method. Results show that the proposed algorithm outperforms all methods reported in [12] and yields very close performance to the RJMCMC based method reported in [13]. However unlike RJMCMC where the number of activations is determined by searching, RVM provides a method for their incremental determination, therefore is extremely faster.

Appendix A. Computation of Marginal Likelihood and Posterior Weight Probability Distribution

The marginal likelihood $p(t|\alpha,\sigma^2)$ can be analytically computed by computing the integral:

$$p(t|\alpha,\sigma^2)=\int p(t|w,\sigma^2)p(w|\alpha)dw,$$

which can be computed since it is the convolution of two Gaussian functions.

Then the posterior weight distribution $p(w|t,\alpha,\sigma^2)$ can be computed from:

$$p(w|t,\alpha,\sigma^2)=\frac{p(t|w,\sigma^2)p(w|\alpha)}{p(t|\alpha,\sigma^2)}.$$

However, it is easier to compute both these quantities from Bayes's rule, as follows:

$$p(w|t,\alpha,\sigma^2)p(t|\alpha,\sigma^2)=p(t|w,\sigma^2)p(w|\alpha)$$

$$=(2\pi\sigma^2)^{-N/2}\exp\left\{-\frac{1}{2\sigma^2}\|t-\Phi w\|^2\right\}(2\pi)^{-(N+1)/2}|A|^{\frac{1}{2}}\exp\left\{-\frac{1}{2}w^TAw\right\}$$

$$=(2\pi)^{-(N+1/2)}(\sigma^2)^{-N/2}|A|^{\frac{1}{2}}\exp\left\{-\frac{1}{2}\left[\sigma^{-2}(t-\Phi w)^T(t-\Phi w)+w^TAw\right]\right\}$$

$$=(2\pi)^{-(N+1/2)}(\sigma^2)^{-N/2}|A|^{\frac{1}{2}}\exp\left\{-\frac{1}{2}\left[\sigma^{-2}(\Phi w)^T(\Phi w)-\sigma^{-2}t^T(\Phi w)-\sigma^{-2}(\Phi w)^Tt+\sigma^{-2}t^Tt+w^TAw\right]\right\}$$

$$=(2\pi)^{-(N+1/2)}(\sigma^2)^{-N/2}|A|^{\frac{1}{2}}\exp\left\{-\frac{1}{2}\left[w^T\left(A+\sigma^{-2}\Phi^T\Phi\right)w-\left(\sigma^{-2}t^T\Phi\right)w-w^T\left(\sigma^{-2}\Phi^Tt\right)+\sigma^{-2}t^Tt\right]\right\}$$

$$=(2\pi)^{-(N+1/2)}(\sigma^2)^{-N/2}|A|^{\frac{1}{2}}\exp\left\{-\frac{1}{2}\left[w^T\Sigma^{-1}w-\left(\sigma^{-2}t^T\Phi\right)\Sigma\Sigma^{-1}w-w^T\Sigma^{-1}\Sigma\left(\sigma^{-2}\Phi^Tt\right)+\sigma^{-2}t^Tt\right]\right\}$$

with $\Sigma=\left(A+\sigma^{-2}\Phi^T\Phi\right)^{-1}$.

$$p(w|t,\alpha,\sigma^2)p(t|\alpha,\sigma^2)=$$

$$=(2\pi)^{-(N+1/2)}(\sigma^2)^{-N/2}|A|^{\frac{1}{2}}$$

$$\exp\left\{-\frac{1}{2}\left[\left(w^T-\sigma^{-2}t^T\Phi\Sigma\right)\Sigma^{-1}w-\left(w^T-\sigma^{-2}t^T\Phi\Sigma+\sigma^{-2}t^T\Phi\Sigma\right)\Sigma^{-1}\Sigma\left(\sigma^{-2}\Phi^Tt\right)+\sigma^{-2}t^Tt\right]\right\}$$

$$=(2\pi)^{-(N+1/2)}(\sigma^2)^{-N/2}|A|^{\frac{1}{2}}$$

$$\exp\left\{-\frac{1}{2}\left[\left(w^T-\left(\sigma^{-2}\Sigma\Phi^Tt\right)^T\right)\Sigma^{-1}\left(w-\left(\sigma^{-2}\Sigma\Phi^Tt\right)\right)-\left(\sigma^{-2}\right)^2t^T\Phi\Sigma\Phi^Tt+\sigma^{-2}t^Tt\right]\right\}$$

Defining $\mu = \sigma^{-2}\Sigma\Phi^T t$ we have:

$$p\left(w \mid t,\alpha,\sigma^2\right) p\left(t \mid \alpha,\sigma^2\right) =$$

$$= (2\pi)^{-(N+1/2)} \left(\sigma^2\right)^{-N/2} |A|^{\frac{1}{2}}$$

$$\exp\left\{-\frac{1}{2}\left[\left(w^T-\mu^T\right)\Sigma^{-1}\left(w-\mu\right) - t^T\left(\left(\sigma^{-2}\right)^2 \Phi\left(\sigma^{-2}\Phi^T\Phi + A\right)^{-1}\Phi^T + \sigma^{-2}I\right)t\right]\right\}$$

$$= (2\pi)^{-(N+1/2)} \left(\sigma^2\right)^{-N/2} |A|^{\frac{1}{2}}$$

$$\exp\left\{-\frac{1}{2}\left[\left(w^T-\mu^T\right)\Sigma^{-1}\left(w-\mu\right) - t^T\left(\left(\sigma^{-2}\right)^2 \left(\sigma^{-2}I + \left(\Phi^T\right)^{-1} A\Phi^{-1}\right)^{-1} - \sigma^{-2}I\right)t\right]\right\}$$

$$= (2\pi)^{-(N+1/2)} \left(\sigma^2\right)^{-N/2} |A|^{\frac{1}{2}}$$

$$\exp\left\{-\frac{1}{2}\left[\left(w^T-\mu^T\right)\Sigma^{-1}\left(w-\mu\right) - t^T\left(\left(\sigma^{-2}\right)^2 \left(\sigma^2 I - \sigma^4\left(\Phi^T\right)^{-1}\left(\sigma^2\Phi^{-1}\left(\Phi^T\right)^{-1} + A^{-1}\right)^{-1}\Phi^{-1}\right) - \sigma^{-2}I\right)t\right]\right\}$$

$$= (2\pi)^{-(N+1/2)} \left(\sigma^2\right)^{-N/2} |A|^{\frac{1}{2}} \exp\left\{-\frac{1}{2}\left[\left(w^T-\mu^T\right)\Sigma^{-1}\left(w-\mu\right) + t^T\left(\sigma^2 I + \Phi A^{-1}\Phi^T\right)^{-1} t\right]\right\}$$

$$= (2\pi)^{-(N+1)/2} |\Sigma|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}\left(w^T-\mu^T\right)\Sigma^{-1}\left(w-\mu\right)\right\}$$

$$(2\pi)^{-N/2} \left|\sigma^2 I + \Phi A^{-1}\Phi^T\right|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2} t^T\left(\sigma^2 I + \Phi A^{-1}\Phi^T\right)^{-1} t\right\}$$

since $|\Sigma|^{-\frac{1}{2}} \left|\sigma^2 I + \Phi A^{-1}\Phi^T\right|^{-\frac{1}{2}} = \left|\sigma^{-2}\Phi^T\Phi + A\right|^{\frac{1}{2}} \left|\sigma^2 I + \Phi A^{-1}\Phi^T\right|^{-\frac{1}{2}}$

$$= \left|\sigma^{-2}A\right|^{\frac{1}{2}} \left|\Phi^T A^{-1}\Phi + \sigma^2 I\right|^{\frac{1}{2}} \left|\sigma^2 I + \Phi A^{-1}\Phi^T\right|^{-\frac{1}{2}}$$

$$= \left|\sigma^{-2}I\right|^{\frac{1}{2}} |A|^{\frac{1}{2}} = \left(\sigma^2\right)^{-\frac{N}{2}} |A|^{\frac{1}{2}}$$

Thus,

$$p\left(w \mid t,\alpha,\sigma^2\right) = (2\pi)^{-(N+1)/2} |\Sigma|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}\left(w^T-\mu^T\right)\Sigma^{-1}\left(w-\mu\right)\right\},$$

and

$$p\left(t \mid \alpha,\sigma^2\right) = (2\pi)^{-N/2} \left|\sigma^2 I + \Phi A^{-1}\Phi^T\right|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2} t^T\left(\sigma^2 I + \Phi A^{-1}\Phi^T\right)^{-1} t\right\}.$$

Appendix B. Computation of Marginal Likelihood Derivatives

Optimization of the marginal likelihood is equivalent to minimizing its negative logarithm.

$$L(\alpha)=\frac{1}{2}\left[\log|\Sigma|+N\log\sigma^2+\log|A|-\sigma^2 t^T\left(t-\Phi\mu\right)\right]+\sum_{i=0}^{N}\left(a\log\alpha_i-b\alpha_i\right)+c\log\sigma^2-d\sigma^2$$

Then, the saddle points of $L(\alpha)$ are the zero-crossings of its derivative. For ease of calculations we consider the derivative with respect to the logarithm of the hyperparameters α_i.

$$\log|A|=\log\prod_{i=1}^{N}\alpha_i=\sum_{i=1}^{N}\log\alpha_i$$

$$\frac{\partial\log|A|}{\partial\log\alpha_i}=1$$

$$\frac{\partial\Sigma}{\partial\log\alpha_i}=-\Sigma\frac{\partial\Sigma^{-1}}{\partial\log\alpha_i}\Sigma=-\Sigma\frac{\partial\left(\sigma^{-2}\Phi^T\Phi+A\right)}{\partial\log\alpha_i}\Sigma=\begin{bmatrix}0&0&0\\0&-\alpha_i\Sigma_{ii}^2&0\\0&0&0\end{bmatrix}$$

$$\frac{\partial\mu}{\partial\log\alpha_i}=\frac{\partial\sigma^{-2}\Sigma\Phi^T t}{\partial\log\alpha_i}=\sigma^{-2}\frac{\partial\Sigma}{\partial\log\alpha_i}\Phi^T t=\begin{bmatrix}0&0&0\\0&-\alpha_i\sigma^{-2}\Sigma_{ii}^2\Phi^T t&0\\0&0&0\end{bmatrix}$$

$$\frac{\partial\log|\Sigma|}{\partial\log\alpha_i}=tr\left[\frac{\partial\log|\Sigma|}{\partial\Sigma}\left(\frac{\partial\Sigma}{\partial\log\alpha_i}\right)^T\right]=tr\left(\left(\frac{1}{|\Sigma|}|\Sigma|\Sigma^{-1}\right)\left(-\alpha_i\Sigma_{ii}^2\right)^T\right)=-\alpha_i\Sigma_{ii}$$

$$\frac{\partial\sigma^2 t^T\left(t-\Phi\mu\right)}{\partial\log\alpha_i}=\sigma^2\frac{\partial t^T\Phi\mu}{\partial\log\alpha_i}=\sigma^2 t^T\Phi\frac{\partial\mu}{\partial\log\alpha_i}=-\alpha_i\mu_i^2$$

$$\frac{\partial L}{\partial\log\alpha_i}=\frac{\partial\left[\frac{1}{2}\left[\log|\Sigma|+\log|A|-\sigma^2 t^T\left(t-\Phi\mu\right)\right]+\sum_{i=0}^{N}\left(a\log\alpha_i-b\alpha_i\right)\right]}{\partial\log\alpha_i}$$

$$=-\frac{1}{2}\left(-\alpha_i\Sigma_{ii}+1+\alpha_i\mu_i^2+2a-2b\alpha_i\right)$$

Setting $\frac{\partial L}{\partial\log\alpha_i}=0$, we obtain the following update formula:

$$\alpha_i=\frac{1+2a}{\Sigma_{ii}-\mu_i^2+2b}.$$

References

[1] Berger J.O., "Statistical desicion theory and Bayesian analysis", Springer, second edition, 1985.

[2] Bowman A., Hall P. and Prvan T. "Bandwidth selection for the smoothing of distribution functions", Biometrika, vol 85, issue 4, pp. 799-808, 1998

[3] Candela J. Q., Hansen L. K., "Time Series Prediction Based On The Relevance Vector Machine With Adaptive Kernels",

[4] Faul, A., Tipping M. E., "A variational approach to robust regression", *Proceedings of ICANN'01*, pp. 95–102, Springer, 2001.

[5] Friston K. J., Frith C.D., Liddle P.E. Frackowiak R.S., "Comparing Functional (PET) Images: The Assesment of Significant Change", *Journal of Cerebral Blood Flow and Metabolism*, vol. 11, pp. 690-669, 1991.

[6] Friston K. J., Holmes A. P., Worsley K. J., Frith J.P., Frackowiak R. S. J., "Statistical Parametric Maps in Functional Imaging: A General Linear Approach", *Human Brain Mapping*, vol 2, pp 189-210.

[7] Geweke J., "Bayesian inference in econometric models using Monte Carlo integration", *Econometrica*, vol 24, pp. 1317-1399.

[8] Hooke R., Jeeves T. A., ""direct search" solution of numerical and statistical problems", *J. Assoc. Comput.*, pp. 212-229, March 1961.

[9] Jackson J. E., Users Guide to Principal Components. New York: John-Wiley & Sons, 1991.

[10] Kay S.M., Fundamentals of Statistical Signal Processing: Detection Theory, vol.2: Prentice-Hall, 1998.

[11] Kustra R., Strother S.C., "Penalized Discriminant Analysis of [15O] Water PET Brain Images with Prediction Error Selection of Smoothing and Regularization Hyperparameters", *IEEE Transactions on Medical Imaging*, pp.376-387, 2001.

[12] Lukic A. S., Wernick M. N., Strother S. C. "An Evaluation of Methods for Detection of Brain Activations from PET or fMRI images", Artificial Intelligence in Medicine, vol. 25, pp. 69-88, 2002.

[13] Lukic A. S., Wernick M. N., Galatsanos N. P., Yang Y., Strother S. C. "A Reversible Jump Markov Chain Monte Carlo Algorithm for Analysis of Functional Neuroimages", *IEEE International Conference on Image Processing*, Rochester, NY, September 2002.

[14] MacKay D. J. C., "Bayesian interpolation", Neural Computation, vol 4, issue 3, pp415-447, 1992.

[15] MacKay D. J. C., "Bayesian methods for backpropagation networks", Models of Neural Networks III, chapter 6, pages 211-254, Springer (1994).

[16] Mardia K. V., Kent J. T., Bibby J.M., Multivariate Analysis. London: Academic Press, 1979.

[17] Scholkopf B., Burges C. J. C., Smola A. J., Advances in Kernel Methods: Support Vector Learning, MIT Press, 1999.

[18] Scholkopf B., Smola A., Learning with Kernels, MIT Press, Cambridge, MA, 2002.

[19] Stawinski G., Doucet A., Duvaut P., "Reversible Jump Markov Chain Monte Carlo for Bayesian Deconvolution of Point Sources", *Proceedings of SPIE, Bayesian Inference for Inverse Problems*, vol. 3459, pp. 179-190, 1998.

[20] Strother S. C., Wernick M. N. "Technical report: Deducing the statistical properties of brain activation from real data for use in constructing phantoms", www.ipl.iit.edu/IPL_papers/ipl_iit_101.pdf.

[21] Talairach T, Tornoux J. A. "Stereotactic Coplanar Atlas of the Human Brain", Stutgard: Thieme, 1988.

[22] Tzikas D., Likas A., Galatsanos N. P., Lukic A. S.., Wernick M. N, "Relevance Vector Machine Analysis of Functional Neuroimages", *IEEE International Symposium on Biomedical Imaging*, April 2004.

[23] Tzikas D., Likas A., Galatsanos N. P., Lukic A. S.., Wernick M. N, "Bayesian Regression of Functional Neuroimages", *European Signal Processing Conference*, September 2004.

[24] Tipping M. E. "Sparse Bayesian learning and the relevance vector machine", Journal of Machine Learning Research 1, 211-244, 2001.

[25] Tipping, M. E., "Sparse kernel principal component analysis", *Advances in Neural Information Processing Systems 13*. MIT Press, 2001.

[26] Tipping, M. E. and A. Faul., "Fast marginal likelihood maximisation for sparse Bayesian models", In Proc of Artificial Intelligence and Statistics '03, 2003

[27] Tipping, M. E., Bishop C. M, "Bayesian Image Super-resolution", *Advances in Neural Information Processing Systems* 15. MIT Press, 2003.

[28] Vapnick V.N., Statistical Learning Theory, Wiley, New York, 1998.

[29] Worsley K.J., Evans A.C, Marett S., Neelin P., "A Three Dimensional Statistical Analysis for CBF Activation Studies in Human Brain", *Journal of Cerebral Blood Flow and Metabolism*, vol. 12, pp. 900-918, 1992.