UNIVERSITY OF SOUTHAMPTON

Interpretable Modelling with Sparse Kernels

by

Jasvinder S. Kandola

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the
Faculty of Engineering and Applied Science
Department of Electronics and Computer Science

June 1, 2001

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND APPLIED SCIENCE
DEPARTMENT OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Jasvinder S. Kandola

A drawback of many statistical modelling techniques, commonly used in machine learning, is that the resulting model is difficult to interpret. The principal focus of this thesis is the development of advanced non-linear interpretable models. Interpretable modelling offers us a powerful tool with which to understand the structure of a model constructed from data, allowing model validation and assisting in model selection. Gibbs (1997) observes the easiest way to introduce model interpretability is to use models where the parameters and the related hyperparameters have clearly interpretable meanings. The Bayesian methodology of Automatic Relevance Determination (ARD) (MacKay, 1994; Neal, 1995) is one such approach. In this thesis Laplace approximations, variational learning and Markov Chain Monte Carlo (MCMC) methods for hyperparameter determination are assessed within a Bayesian neural network. Empirical results highlight the numerical instability of the Laplace and variational methods with convergence to a local rather than global minima. Kernel methods have become a popular modelling approach (Vapnik, 1998; Smola, 1998; Williams, 1998). In this thesis the constructed kernel models are equipped with hyperparameters that allow: the ability to *select* important input variables, the ability to *visualise* the model structure and the ability to incorporate prior or expert knowledge. Ideas from the Bayesian and the signal processing communities together with the representational advantage of a sparse ANOVA decomposition have been merged. Interpretability is introduced by using two forms of regularisation: a 1-norm based structural regulariser to enforce interpretability, and a 2-norm based regulariser to control smoothness. The model structure can be visualised showing the overall effects of different inputs, their interactions, and the strength of the interactions. The performance of these interpretable learning algorithms is demonstrated on both synthetic and "real" data, notably the AMPG dataset, the Boston house price dataset, and the problem of predicting the mechanical property proof stress of a metal based on its chemical composition. Results from these different approaches are compared in terms of their interpretabilty by exploiting prior knowledge of the problem, and show the potential of interpretable data models.

Contents

List of Figures

List of Tables

Nomenclature

α	hyperparameter associated with network weights
β	noise variance hyperparameter
$\boldsymbol{C}$	covariance matrix
$\mathcal{D}$	dataset consisting of input-output pairs
d	VC dimension
$E[\cdot]$	expectation of a random variable
$E_{\mathcal{D}}$	empirical risk
$E_{\boldsymbol{w}}$	regularisation term over parameters
F	input space dimension
$f(\cdot)$	model or regression function
$\mathbf{G}$	Gram matrix
$\mathcal{H}$	hypothesis or model space
$\boldsymbol{A}$	Hessian matrix
$H[f]$	regularised (penalised) cost functional
η	learning rate
$\mathbf{I}$	identity matrix
$K(\cdot,\cdot)$	kernel function
$\boldsymbol{K}$	Kernel matrix
$\mathcal{K}$	Kernel space
λ	regularisation parameter
λ_i	ith eigenvalue
$\mathbf{\Lambda}$	hyperparameter masking matrix
$\mathcal{L}$	loss function
$_{MP}$	refers to the maximum a posteriori estimate
N	total number of input-output data pairs
$P(\cdot)$	probability density function
$p(\cdot)$	probability
$\mathbf{\Sigma}$	covariance matrix
supp	support of a distribution
$\sigma^{\mathbf{2}}$	variance
$\mathbb{R}^d$	d-dimensional Euclidean space
^{T}	transpose operator

W	total number of model parameters
$\boldsymbol{w}$	parameter vector
$\boldsymbol{x}$	input vector
$\mathcal{X}$	input space
y	output
$\boldsymbol{y}$	vector of target observations
ε	noise process
$\mu_{\mathbf{w}}$	prior mean for parameters
σ_y^2	predictive variance
$\phi(\boldsymbol{x})$	basis function
$\boldsymbol{\Phi}$	basis function matrix
$\langle \cdot, \cdot \rangle$	inner product
$\Vert \cdot \Vert$	Euclidean norm
$\Vert \cdot \Vert_n$	n-norm
$\Vert \cdot \Vert_{\boldsymbol{K}}$	norm with respect to matrix $\boldsymbol{K}$
$\hat{}$	estimate
$\bar{}$	mean
$\mathbf{1}$	vector of ones
$\mathbf{0}$	vector of zeros
$\boldsymbol{\zeta}$	vector of slack variables

Acknowledgements

During the period of my Ph.D., I have had cause to be grateful for the advice, support and understanding of many people. In particular I would like to express my sincere appreciation and gratitude to my supervisor Dr. Steve Gunn for his continuous moral and technical support, as well as his unflagging enthusiasm. This work has also benefited greatly from much advice, and many useful discussions with Drs Junbin Gao, Adam Prügel-Bennett and John Manslow. Thanks also to Mike Grant for allowing me use of the barmaids.

As well as the people mentioned above, a mention must also go to the many friends in ISIS who made it such an interesting place to work, and to the inhabitants of RT-Bristol (current, former and hanger's on) who provided a useful distraction whilst I was writing up. This thesis is dedicated to my parents.

Chapter 1

Introduction

"What's the next number in the sequence: 2, 3, 5 . . . ?"

To be able to forecast future events, science infers general laws and principles from particular instances. Many recent approaches to developing models from data have been inspired by the learning capabilities of biological systems and in particular, those of humans. There has always been an appeal to build systems/learning algorithms that imitate human (or animal) brains. In the mid 1980s this led to great enthusiasm about the so called (artificial) neural networks. However, although many neural network models and applications have little in common with biological systems, the biological terminology still remains. The techniques that have been developed in this thesis for learning from data are based on the principles of mathematics and statistics rather than biology, hence the use of any biological analogy or terminology is avoided.

Broadly stated, the goal of research in machine learning is the understanding of complex learning processes and the construction of effective computer based learning systems. Traditionally, the main goal of data modelling systems has been to construct models for accurate predictions of future outputs from the (known) input values. The construction of effective computer learning systems has proven to be exceedingly difficult. The enormity of the task has compelled researchers to focus on developing complex learning algorithms (Rasmussen, 1996). Workers in the machine learning community have embraced ever more complicated models and it is not unusual to find applications with very computationally intensive models containing hundreds or thousands of parameters.

1.1 Problem Statement

Recent approaches to developing methods that learn from examples have been inspired by the need to develop effective solutions that can discover the implicit and non-trivial relationships that exist in data. This thesis is concerned with the problem of model interpretability. Whilst a predictive model is often the ultimate goal of modelling, it is often desirable and often even essential to be able to interpret the final model structure. This is especially true in medical domains, where *black-box* models (Ljung, 1987), for example traditional neural networks and kernel based methods, are viewed with great suspicion (Wyatt, 1995; Plate, 1999). The need for understanding large, complex, information rich datasets has become common to virtually all fields of business, science and engineering. The mathematical formulation of the learning problem often gives the impression that learning algorithms do not require modeller intervention. In practice issues such as selection of the 'relevant' input and output variables, data encoding/representations, as well as incorporating *a priori* domain knowledge into the design of the learning system must be considered.

In response to the lack of model interpretability, statisticians (who traditionally have been great advocates of interpretable models) and other researchers often revert to using simpler, but more interpretable modelling methods, for example multivariate linear regression or logistic regression (Plate, 1999). A notable disadvantage in using such simple models is that they typically suffer from the problem of model mismatch, and hence they may fail to discover an important relationship in the data because they lack the flexibility to model it. This thesis is concerned with trying to address this shortcoming by deploying interpretable modelling methods in complex non-linear environments.

This thesis focuses on the construction of sparse data driven models that can be easily interpreted. As observed by Gibbs (1997) the easiest way to introduce model interpretability is to use models where the parameters and the related hyperparameters have clearly interpretable meanings. In the case of regression models, the hyperparameters quite often indicate more about the underlying physical mechanism that gave rise to the data, than the parameters of the model. Given the recent interest in Bayesian methods for learning, modelling with interpretable parameters and hyperparameters also simplifies the expression of probability distributions. In response to the lack of model interpretability the constructed models are equipped with hyperparameters that allow: the ability to *select* important input variables via principled data driven approaches (these are discussed in Chapter 3 and Chapter 5), the ability to *visualise* the model structure and the ability to incorporate prior or expert knowledge.

1.2 Motivation and Challenges

Many of the flexible modelling approaches that exist in the literature have not been designed with particular learning tasks in mind (Cherkassky and Mulier, 1998). This then introduces the problem of how to choose the best technique for a particular task. All of these general purpose methods rely on various assumptions and approximations, and in many cases it is difficult to know how well these are met in particular applications, and how severe the consequences of breaking them are.

Empirical assessment of these modelling methods seems to be the most appealing way of choosing between them. If one method has been shown to out perform another on a series of learning problems that are deemed to be representative of the applications we are interested in that should be enough to resolve the dispute (Rasmussen, 1996). However, empirical assessment of different learning methods is usually only based on predictive performance. Interpretable modelling algorithms are attractive from a number of viewpoints. Incorporation of interpretability into a model provides an additional means for model selection and model validation, allowing a model to be selected not only on predictive performance but also through prior knowledge.

The problem under investigation can be stated as follows: given a dataset consisting of inputs and an output, how can interpretable data models be developed that are capable of recovering the underlying model structure. A number of researchers (MacKay, 1994; Neal, 1995; Plate, 1999) have tried to address this problem by determining which inputs are relevant in predicting the output. In addition to determining relevant input variables, this research takes a more general approach to model interpretability by trying to visualise the underlying model structure with the aim of evaluating the predicted model trends using prior knowledge. This information can then be used to provide additional information useful for model validation and model selection. In contrast to many other approaches, that are reviewed in Chapter 2, the approaches developed in this thesis try to avoid problems with convergence to local minima.

When describing interpretable modelling methods, many researchers have been concerned with the following performance criteria summarised below (Cherkassky and Mulier, 1998):

- *Robustness* - Does the method tolerate a reasonable amount of noise in the dataset, and how does interpretability degrade in high noise applications.
- *Efficiency* - Machine learning typically requires that an enormous space of alternatives be considered. How much time and memory are required to search this space?
- *Accuracy* - Is the method capable of providing an accurate description of the underlying system.

Introducing interpretability into a learning method is not a straightforward task. There are a few basic problems which afflict every algorithm (Gunn and Kandola, 2000):

- *False Recognition* - Due to correlation effects, inputs which are in fact erroneous to the output are selected as being relevant.

- *Over complex models* - Even highly interpretable methods lose their interpretability as the models become too complex. As a result model interpretation is inherently limited by the models complexity regardless of the method used. As a consequence, methods for penalising model complexity or methods for obtaining sparse models are desirable.

- *Incomplete Features* - The dataset may not contain enough information for a meaningful analysis to be made. This could be because many important attributes of the problem cannot be measured or are not available.

This research develops interpretable learning methods that take these problems into account. The problem of false recognition can be assessed by evaluating the stability with which inputs are selected after multiple random parameter initialisations, and multiple random dataset partitions are employed. Random parameter initialisations and data partitions are desirable since they allows exploration of different regions of the model space. The approach considered in this thesis attempts to resolve the second problem by employing two different approaches. In the first two regularisers are used: a 1-norm based structural regulariser to enforce interpretability, and a 2-norm based regulariser to enforce smoothness. In the second approach, a Bayesian approach is developed in which probability distributions are used to encode prior beliefs of model interpretability and model sparsity. Unfortunately, other than collecting more data very little can be done to resolve the missing features problem.

1.3 Relationship to Existing Approaches

Kernel based methods and Support Vector Machines (SVMs) (Vapnik, 1998; Smola, 1998) in particular are a class of learning methods that can be used for non-linear regression estimation. They have often achieved state of the art performance in many areas where they have been applied (Tipping, 2000b). The class of functions they choose from is determined by a kernel function. The form of this function is of central importance to kernel based methods. Its choice determines the class of functions the model can draw its solution from, and hence the accuracy of the solution.

In this work we aim to develop a kernel based technique capable of providing interpretable modelling. Moreover, the datasets considered can contain irrelevant features. This study encompasses a particular ANalysis Of VAriance (ANOVA) decomposition kernel and is applied to both artificial and real world datasets. Despite the growing interest in SVM research, there has been little effort to explore the underlying model structure.

A number of new concepts and algorithms have been introduced by researchers trying to introduce interpretability into learning algorithms (Breiman et al., 1984; MacKay, 1994;

Neal, 1995). Statisticians have often used an ANOVA kernel when needing to interpret the interactions between attributes of the data (Allen, 1974; Wahba, 1985). Methods for determining low order interactions are considered since they provide a less complex description of the data.

In order to obtain sparse interpretable models it is difficult to rely on a single technology to achieve our goals. This implies that an integration of techniques is required. The proposed method exploits the merits of: (1) Multiple regularisers enforcing interpretability and smoothness, (2) Bayesian learning for hyperparameter determination, (3) ANOVA decomposition kernels, (4) deployment within a kernel based method.

ANOVA kernels have previously been deployed within an SVM with promising performance (Stitson et al., 1999). However, the difference in this thesis has been to develop a technique that will select a sparse ANOVA kernel producing strong interpretability. Within the neural networks community, the method of Automatic Relevance Determination (ARD) (MacKay, 1994; Neal, 1995) for input selection has been introduced. The proposed method uses an approach similar to ARD in an ANOVA kernel. A number of other methods have been introduced where the parameters or the hyperparameters can be given a physical meaning, for example the hierarchical mixtures of experts (Waterhouse and Robinson, 1997), graphical models (Whittaker, 1990) as well as rule based architectures (Brown and Harris, 1994).

Part of the work described in this thesis is dedicated to the empirical study of prior work in model interpretability. The closely related method of ARD in a Bayesian neural network and within a Gaussian process model were successfully implemented to scrutinise their performance with respect to the problem in hand. This exercise provided insight and direction to the work described in this thesis. This is also useful in that it then highlights similarities and differences between various algorithms to allow an analytical comparison. Furthermore, it is important to benchmark the performance of our algorithm against those of existing methods. Overall it is the aim of this thesis to demonstrate the effectiveness of sparse interpretable learning methods in a wide variety of situations.

1.4 Contributions

The main contributions of this work are to introduce interpretable modelling as a means for model selection and model validation; the introduction of two novel algorithms relying on kernel based learning, and a demonstration of the empirical assessment of interpretable kernel based methods with other interpretable modelling methods. The primary contribution of this research has been to sensibly combine existing ideas along with new ones to provide a systematic paradigm for model interpretability and model validation.

- Detailed discussion of the use of interpretable data modelling for model understanding. Additive models are attractive for a number of reasons and their use in interpretable data modelling is highlighted.

- Bayesian inference in learning algorithms has received a renewed research effort. The Bayesian methodology of automatic relevance determination (ARD) is limited in that high dimensional integrals are involved in the inference step. A number of approximation methods have been proposed to overcome this. A discussion as to their advantages and limitations are highlighted with reference to an artificial problem.

- Using ideas from statistics, Bayesian inference and functional analysis, Chapter 4 provides a discussion of different aspects of approximation by kernel based methods that are used for function approximation. Splines are attractive for data modelling and these are discussed with reference to data modelling problems.

- Two popular methods that use multiple regularisers or hyperparameters, have been proposed to obtain sparse solutions. Multiple regularisers have been used extensively in the signal processing community, and their usefulness in obtaining sparse interpretable modelling methods is investigated. Bayesian hyperparameter based methods are also deployed resulting in new kernel based learning algorithms.

- Chapter 6 describes the results of applying the sparse interpretable methods that have been developed to a range of datasets highlighting the potential of the described methods. The sparse interpretable approaches developed in this thesis were applied to a commercial dataset with great success, resulting in the company modifying their process as a direct result of the structural information obtained from interpretable modelling methods.

The work in this thesis has contributed in part or full to the following publications:

• Gunn S.R. and J.S. Kandola (2001). Structural Modelling with Sparse Kernels, Machine Learning: Special Issue on Model Selection and Model Combination. Eds: Y. Bengio and D. Schuurmans. In Press.

• Gao J.B., S.R. Gunn and J.S. Kandola (2000). A Variational Approach for Adapting Kernels in Support Vector Regression, Advances in Neural Information Processing Systems (NIPS13) Kernel Workshop. Denver, USA, December 2000.

• Kandola J.S. and S.R. Gunn (2000). Assessing the Stability of Advanced Transparent Modelling Techniques, CRM Workshop on Combining and Selecting Models using Machine Learning Algorithms. Montreal Canada, April 2000.

• Kandola J.S. and S.R. Gunn (1999). Data Driven Knowledge Extraction of Materials Properties, IEEE IPMM 99 USA, pp. 380-388.

• Kandola J.S. and S.R. Gunn (1999). Understanding Complex Datasets, Young Statisticians Meeting (YSM'99), Royal Statistical Society, University of Bristol, UK, March 1999.

• Kandola J.S. and M. Brown (1998). Statistical Modelling of Complex Processes, EC Bayesian Signal Processing Summer School, Isaac Newton Institute for Mathematical Sciences, University of Cambridge, UK, July 1998.

1.5 Outline of Thesis

This prelude has introduced the idea of interpretable modelling, and has highlighted some of the difficulties before interpretable learning algorithms can be applied to a wide variety of situations.

Chapter 2 provides the basic setting for regression analysis. It introduces the problem of learning from data. This provides the motivation for much of the rest of the thesis. The central theme behind all of the methods described in this thesis is that of knowledge representation. The basic ideas behind converting ill-posed problems to well-posed problems is introduced, and provides a reference point for the ideas described later in this thesis. The remainder of the chapter motivates the necessity for being able to examine the model structure. This is considered to be an important part of the model building process, both for model selection and for model validation. A review of other interpretable modelling algorithms is provided and a description of their associated limitations.

Chapter 3 introduces the concept of Bayesian inference. Bayesian learning for data modelling has received much interest recently since this method provides a consistent theory of learning at all levels. Penalisation of over-complex models can be given a natural interpretation in a Bayesian framework using probability distributions. For regression problems, error bars can be assigned to the predictions generated by a model, and in the case of classification class conditional probabilities can be obtained. The Bayesian learning approach also allows a large number of hyperparameters to be used in the inference step, and offers principled methods for their determination. This chapter provides an introduction to the method. The applicability of Bayesian learning to neural network models is introduced, paying considerable attention to the method of automatic relevance determination for input selection, and hence model interpretability. A number of algorithms have been developed in the neural network community for evaluating the typically high dimensional integrals that occur in the presence of the so-called hyperparameters. This chapter provides a comprehensive discussion as to the advantages and disadvantages of these methods. Much of this discussion centres around the use of Bayesian learning in a neural network to solve a well known artificial problem that has been proposed in the literature.

Chapter 4 describes the use of kernel based methods for learning. Kernel methods have received an immense amount of interest recently due to a number of attractive features and excellent empirical performance. In some applications they have been reported to have given state of the art results. The Bayesian formalism described in Chapter 3 for neural networks, is extended to kernel based learning methods. The idea of defining a prior distribution over a set of functions is introduced and leads to the Gaussian Process formulation. This is then related to the use of the structural risk minimisation principle in the related Support Vector Machine method. Central to both of these methods is the use of a kernel function. This chapter provides a description of the necessary requirements for a function to be an acceptable kernel function. The selection of the kernel function is non-trivial and a review of the current methods for selecting kernels is provided. Spline kernels are extensively used throughout this thesis because they are able to model a wide range of functions. Computational cost is also considered in this chapter, which is a major issue in kernel based learning since evaluation typically involves storage, evaluation and inversion of a kernel matrix, which for large datasets can be prohibitive. However, there are methods to address this and they are discussed.

Chapter 5 describes how an alternative method of interpretability can be introduced into a kernel based method using ANOVA kernels. The solution of a kernel based method is given by a weighted sum of kernel functions, as a consequence the solution is opaque. Within a Gaussian process method, Williams and Rasmussen (1996) have addressed this problem by deploying an automatic relevance determination based kernel function which is capable of selecting which inputs are relevant in predicting the output and hence introducing interpretability into this method. This thesis considers an alternative approach that is based on deploying ANOVA spline kernels within a kernel based method. ANOVA decomposition is a statistical idea of analysing the variances between different variables, and finding dependencies on subsets of the variables. This idea is converted to kernels by ignoring the idea of analysing variances, and considering all possible subsets of variables up to a certain size. This has the associated advantage that relationships between different combinations of inputs can be assessed automatically without explicitly forming additional data inputs. Every ANOVA subterm is a valid kernel in its own right, and hence a sparse selection of terms allows the subsequent deployment in a kernel machine. Each term can also be visualised providing structural information about the underlying model structure. Two popular wavelet methods that can result in a sparse representation are reviewed, and are used to obtain a sparse representation within the proposed framework. Bayesian methods have been extensively used for hyperparameter determination. These methods are also deployed within a kernel based method to allow a sparse representation. The performance of these methods are illustrated on some toy examples.

Chapter 6 is concerned with measuring and comparing the predictive performance of sparse kernel methods on a number of real world datasets. These datasets are illustrative of the types of problems that are typically encountered when using commercial datasets, viz. sparse data, highly correlated input variables and high noise variance. Rasmussen (1996) has used the DELVE environment to rigorously compare many modelling methods in a statistically meaningful sense. While several comparisons are made in this thesis using specific datasets, these are done to illustrate how sparse interpretable models can be useful, and are not meant as definitive comparisons between kernel methods and other methods.

Chapter 7 provides a review of the overall contributions of this thesis, and provides some ideas for future work.

Chapter 2

Function Approximation and Interpretable Modelling

2.1 Learning from Data

Learning, like intelligence, is difficult to define precisely. A dictionary definition includes phrases such as 'to gain knowledge or understanding of a subject by studying, instruction or experience', in addition to 'modification of existing behavioural tendency by experience'. The task of inductive inference is to find laws or regularities underlying some given set of data. The aim of finding laws underlying the data is usually cast in terms of finding a good *model* for the data. The process of finding such a model is called *statistical inference*. Many machine learning algorithms deal with the problem of predictive learning, i.e., estimating an unknown dependency from known observations. In the case of noiseless observations we have the related problem of estimating the value of an unknown function at a new point, given the values of this function at a set of sample points. Statistical methods for dealing with these problems can be considered instances of machine learning because the decision and estimation rules depend on a corpus of samples drawn from the problem environment.

A number of new concepts and algorithms (Bengio et al., 1994; Cherkassky and Mulier, 1998) have been introduced that depart from traditional statistical data analysis in several ways: they use sophisticated models that can learn complicated nonlinear dependencies from large datasets, and rather than using traditional statistical tests to evaluate how "good" a model is, evaluation is instead based on predictive or *generalisation* performance using new and independent test data. Before a model of the system can be built, an understanding of the system must exist. The former is a *learning* problem, i.e. the process of making deductions given a limited set of observations, whilst the latter is an *inference* problem. All modelling tasks involve learning and inference. However, the emphasis of both processes vary among learning algorithms which are tightly coupled to the problem domain. Many techniques for improving a models predictive ability have been inspired by the well-known principle of Occam's razor[1], which is interpreted as stating that the simplest possible model that accurately represents the data is the most desirable. From this perspective, the data modelling problem should take into account some measure of the *sparsity* of the solution in addition to predictive ability.

Bayesian learning is an example of a technique that has seen a renewed research interest, particularly because of its use of probability to express all forms of uncertainty. Chapter 3 provides a more indepth discussion of the advantages of Bayesian learning and some of the limitations associated with this method.

Throughout this thesis, the problem of regression is considered. However, all of the ideas and algorithms described can be applied to the classification scenario. The problem of regression is to approximate an unknown function from the observation of a limited sequence of (typically) noise corrupted input/output data pairs. More formally, consider a dataset $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^{N}$, drawn from an unknown probability distribution, where $\boldsymbol{x}_i \in \mathbb{R}^F$ represents a set of inputs, $y_i \in \mathbb{R}$ represents a single output, and N represents the number of training examples. The empirical modelling problem is to discover an underlying mapping $\boldsymbol{x} \rightarrow y$ that is consistent with the dataset $\mathcal{D}$. The regression function is learnt from a training set, and its performance can be measured using an independent test set.

The finite number of training samples in the dataset, $\mathcal{D}$, implies that any estimate of an unknown function is always inaccurate (biased). The problem of approximating a function from sparse data is inherently *ill-posed* (Hadamard, 1923; Poggio and Girosi, 1989). The solution that minimises the empirical risk is not unique, since there are an infinite number of functions, from the class of continuous functions, that can interpolate the data points giving a local solution. This situation is illustrated in Figure 2.1.

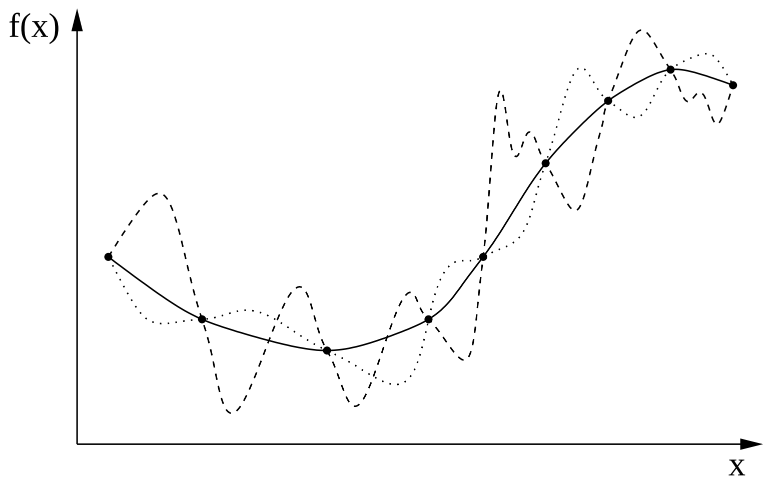


FIGURE 2.1: There are an infinite number models (given by the dashed lines) that can fit the data (●) making the problem ill-posed.

[1]William of Occam (1285-1349): "Causes should not be multiplied beyond necessity".

Definition 2.1 (Well-posed Problems). The problem of determining the solution $\boldsymbol{y} = S(\boldsymbol{x})$ in the space of $\mathcal{Y}$ from the initial data $\boldsymbol{x} \in \mathcal{X}$ is said to be well posed on the spaces $(\mathcal{Y}, \mathcal{X})$ if the following conditions are satisfied:

1) For every element $\boldsymbol{x} \in \mathcal{X}$ there exists a solution $\boldsymbol{y}$ in the space $\mathcal{Y}$.

2) The solution $\boldsymbol{y} = S(\boldsymbol{x})$ is unique.

3) The solution varies continuously with the data.

A problem that is not well posed, i.e. it fails to satisfy one or more of the conditions (1-3) is called *ill-posed*. Some examples of ill-posed problems include: numerical differentiation of noisy and noiseless data, non-parametric smoothing of curves defined using sparse data, image reconstruction, multivariate approximation by radial basis functions and training of neural networks. The survey articles by Engl (1993) and Groetsch (1977) contain many pertinent references. Hence, techniques that can convert ill-posed to well posed problems are attractive.

2.1.1 Model Regularisation

Given the ill-posed nature of the learning problem, in order to choose one particular solution some prior knowledge about the class of functions must be used. All learning methods use *a priori* knowledge in the form of the (given) class of approximating functions of a learning machine. For example, parametric methods use a very restricted set of approximating functions of prespecified parametric form, so only a fixed number of parameters have to be determined from the data. Adaptive methods however use a wide set of functions capable of approximating any continuous mapping.

The most common form of prior knowledge consists in assuming that the function is *smooth*. Smoothness is referred to as a measure of the 'lack of oscillatory' behaviour of a function, i.e., two similar inputs will correspond to two similar outputs if a function is smooth. Therefore, within a class of differentiable functions, one function will be smoother than another if it oscillates less (Girosi and Poggio, 1990).

The imposition of a smoothness constraint as part of the learning process essentially defines possible function behaviour in local neighbourhoods of the input space. For most learning problems, the smoothness constraints describe how individual samples in the training data are combined by the learning method in order to form a function estimate. A consequence of this is that the accuracy of function estimation then depends on having enough samples within the neighbourhood specified by smoothness constraints. However, as the number of dimensions increase the number of samples required to give the same density increases exponentially. This could be offset by increasing the number of data samples falling within the neighbourhood, but this is at the expense of imposing stronger (and possibly incorrect) constraints. This problem is often referred to as the *curse of dimensionality* (Cherkassky and Mulier, 1998).

Techniques that control the capacity of a model can result in an improvement in a model's generalisation performance by trying to convert the problem to one that is well posed (Burges, 1998). To control the capacity the method of regularisation can be incorporated in the learning process. In the case of neural networks, penalising the number of adaptive parameters in a network can vary the model complexity. The main idea underlying regularisation theory[2] is that the solution of an ill-posed problem can be obtained from a variational principle, which contains both the data and prior smoothness information. Smoothness is taken into account by defining a *smoothness functional* $\phi[f]$ in such a way that the lower values of the functional correspond to the smoother functions. Ideally, a solution that is simultaneously close to the data and also smooth is sought, and hence a solution that minimises the following functional is desired,

$$H[f] = \sum_{i=1}^{N} (y_i - f(\boldsymbol{x}_i; \boldsymbol{w}))^2 + \lambda \phi[f] \tag{2.1}$$

where λ is a positive number that is referred to as the regularisation parameter, $\boldsymbol{w}$ are the parameters of the model, and $f(\boldsymbol{x}_i; \boldsymbol{w})$ is the regression function. Model selection is the task of choosing a model of optimal complexity for the given (finite) data. Under the above penalisation formulation, the best penalty functional $\phi[f]$ should reflect (known a priori) properties of a target function so that the penalty is small when the predicted model is close to the target function, and large otherwise. The first term is enforces proximity to the data, and the second term smoothness of the function, while the regularisation parameter controls the balance between the two. To make the learning machine more data driven and flexible, the observed data can be used to select the regularisation parameter via cross-validation techniques (Allen, 1974; Wahba, 1985), whilst the penalty functional is user defined.

A simple method of regularisation is to use the squared norm of the parameter vector,

$$\phi[f] = \|\boldsymbol{w}\|_2^2 = \sum_{i=1}^{W} w_i^2 \tag{2.2}$$

where $\phi[f]$ represents the regularisation functional, W represents the total number of parameters, $\boldsymbol{w}$, in the model. This technique is referred to as zeroth order regularisation, or ridge regression. In a regularised cost minimisation problem, the use of zeroth order regularisation favours models with low weights thereby controlling the capacity of the network. The regularised cost minimisation problem (Bellman, 1961) is a trade-off

[2]In the probabilistic interpretation of regularisation, the different classes of basis functions correspond to different classes of prior probabilities on the approximating function spaces, and therefore to different types of smoothness assumptions.

between a fit to the data, and constraining the model to stay in a small subset of possible models. This can be seen by re-writing Equation 2.1 in the form,

$$\min \sum_{i=1}^{N} (y_i - f(\boldsymbol{x}_i; \boldsymbol{w}))^2 \quad \text{subject to} \quad \phi[f] < \frac{1}{\lambda} \tag{2.3}$$

In the absence of any regularisation or model capacity control, the parameters of a model result in poor generalisation by virtue of their high likelihood of taking on completely arbitrary values, or causing the model to overfit the data in order to produce a slight reduction in the training error (Hush and Horne, 1993). The use of regularisation encourages the excess parameters to assume values close to zero and thereby improve generalisation performance.

More complicated forms of model regularisation than that described by Equation 2.2 have been proposed in the neural network literature. Weigend et al. (1991) proposed a regulariser of the form,

$$\phi[f] = \sum_{i=1}^{W} \frac{(w_i/w_0)^2}{1 + (w_i/w_0)^2} \tag{2.4}$$

where w_0 is a preassigned parameter. When $|w_i| \gg |w_0|$, the penalty for that weight approaches unity. The implication of this condition is that the ith parameter of the model is unreliable and should therefore be eliminated from the model. However, when $|w_i| \ll |w_0|$, the penalty for that parameter approaches unity, and hence is important for the learning process. Hence, the penalty term given in Equation 2.4 identifies the parameters that are of significant influence.

Other approaches that have been proposed have used information from second order derivatives of the error surface in order to implement a trade off between network complexity and training error performance. The starting point in the construction of such a model is the local approximation of the loss function using a Taylor series expansion around an operating point,

$$S(\boldsymbol{w} + \Delta\boldsymbol{w}) = S(\boldsymbol{w}) + g(\boldsymbol{w})^T \Delta\boldsymbol{w} + \frac{1}{2}\Delta\boldsymbol{w}^T \boldsymbol{A} \Delta\boldsymbol{w} \tag{2.5}$$

where $\Delta\boldsymbol{w}$ is a perturbation applied to the parameter $\boldsymbol{w}$, and $g(\boldsymbol{w})$, with Hessian matrix $\boldsymbol{A}$. A number of implementation issues arise when evaluation of the Hessian matrix, and these are discussed at length in Chapter 3. The optimal brain damage procedure (LeCun et al., 1990) simplifies the computations by making the assumption that the Hessian matrix $\boldsymbol{A}$ is a diagonal matrix. However, in the optimal brain surgeon procedure (Hassibi et al., 1992) no such assumption is made. The goal of this method is to set the model parameters to zero by minimising the quadratic form, $\frac{1}{2}\Delta\boldsymbol{w}^T \boldsymbol{A} \Delta\boldsymbol{w}$, with respect to the incremental change in the parameter vector, $\Delta\boldsymbol{w}$, subject to the constraint that $\mathbf{1}^T \Delta\boldsymbol{w} + w_i$ is zero, and then minimise the result with respect to the index i. In their paper, Hassibi et al. (1992) report that on some benchmark problems the optimal brain surgeon procedure resulted in smaller networks than those using the weight decay regulariser, however the computational expense was much higher.

2.1.2 Structural Regularisation

The methods described above prevent model overfitting by imposing a penalty constraint on the set of allowable functions which penalises the models parametric form or penalises global smoothness properties. The smoothness constraints described essentially define possible function behaviour in local neighbourhoods of the input space hence, the regulariser can be seen as imposing an ordering on the hypothesis space. However, when no prior knowledge is available about the data generating function, a large function space needs to be chosen so as to ensure that the approximation error will be small. As a consequence, imposing an order on this space is a difficult task.

In learning theory there is also a need for sparse models, in which the smallest number of functions possible are used to approximate a function $f(\boldsymbol{x})$. In addition to a term that penalises the model parameters, an additional term to enforce sparseness of the model solution is introduced to act as a regulariser on the model structure. Both of these facets have been inspired by the well known principle of Occam's razor. From this formulation of the learning problem, this principle suggests that the design should take into account some measure of the simplicity, or parsimony, of the solution in addition to performance on the training set. A possible approach to obtaining a sparse model has been to build a large model, overspecialised to the training set and attempt to reverse some of the training by retaining only the vital model structure (Rao et al., Feb 1999). This latter approach is adopted in methods such as optimal brain surgeon described above. Sparsity of the model's representation is discussed further in Chapter 5.

2.2 Knowledge Representation

The learning of a regression function can be posed as an optimisation problem, enabling expert or prior knowledge to be incorporated into the modelling process. This knowledge refers to information which might be used to supplement the training data that developed the solution, and which is additional to that provided by the training data. Examples of prior knowledge include information that is known about variable influence, the data generating process as well as the data gathering process. Prior knowledge can either be incorporated directly into the model structure itself, for example by aiding in the position, number and shape of basis functions, or indirectly as an aid to model interpretation and model validation.

The ability to visualise complex nonlinear relationships has provided an important tool for knowledge extraction (Plate, 1999; Gunn and Kandola, 2000). Expert or prior knowledge can be used to increase the robustness of the modelling process. This knowledge can be incorporated into the modelling process, for example by defining smoothness constraints (described in Section 2.1.1), or by introducing a prior distribution over the possible models (described in Chapter 3). Alternatively, expert knowledge can be incorporated as an aid to model interpretation and model validation. Model structure can be interpreted by assessment of parameter or hyperparameter values, as well as by assessing the trends between inputs and output.

Qualitative data visualisation algorithms have been proposed in the literature, see for example (Bishop and Tipping, 1996). However, the majority of these techniques are limited because they are based on a projection of the data onto a low-dimensional visualisation space. Whilst such plots can reveal the structure of simple datasets, they are severely limited for nonlinear datasets with a large numbers of variables. Projection and visualisation techniques which are capable of revealing the underlying data structure in these cases are in demand as complex interactions are commonplace in commercial applications. To this end the use of graphical representations has started to play an increasing role in data modelling (Whittaker, 1990).

A number of algorithms that utilise the idea of structural regularisation have been proposed in the machine learning literature. Graphical models are a marriage between probability theory and graph theory (Jordan, 1999). These models provide a powerful tool for visualising the complex interactions and dependencies between different data variables via a graphical representation. Let X be a k-dimensional vector of random variables. A conditional independence graph, $G = (V, E)$, describes the association structure of X by means of a graph, specified by the vertex set V and the edge set E (Whittaker, 1990). To construct a graphical Gaussian model it is necessary to test for the presence or otherwise of dependencies between the variables. There is a directed edge between vertices i and j if the set E contains the ordered pair (i, j); vertex i is a parent of vertex j, and vertex j is a child of vertex i. An edge can be used to indicate *relevance* or *influence* between data variables.

The notion of independence and conditional independence are a fundamental component of probability theory. Detailed studies of conditional independence properties can be found in (Dawid, 1979a,b) or (Lauritzen, 1995). Given an independence graph G, and a k-dimensional random vector X, a graphical Gaussian model is a family of normal distributions for X constrained to satisfy the pairwise conditional independence restrictions inherent in the independence graph. A graphical Gaussian model is obtained when only continuous random variables are considered. The conditional independence constraints are equivalent to specifying zeros in the inverse variance parameter corresponding to the absence of an edge in G (Whittaker, 1990).

The deviance statistic can be used to assess the overall goodness of fit of a graphical model. A symmetric deviance matrix can be computed using,

$$\text{dev}(X_b \perp\!\!\!\perp X_c|X_a) = -N \ln(1 - \text{corr}_N^2(X_b, X_c|X_a) \qquad (2.6)$$

This test statistic has an asymptotic χ^2 distribution with one degree of freedom. Elements in this deviance matrix, determine the significance of dependencies in the graphical model.

Recent work on Bayesian networks (also known as belief networks) has allowed the modelling of joint probability distributions in a number of systems. A Bayesian network is a graphical model that can be used to encode expert knowledge amongst a set of variables (Heckerman, 1999). A Bayesian network consists of two components. The first is a directed acyclic graph (DAG) in which each vertex corresponds to a random variable. In a manner similar to the graphical Gaussian model, this graph describes conditional independence properties of the represented distribution. The second component is a collection of conditional probability distributions that describe the conditional probability of each variable given its parents in the graph. Together, these two components can be shown to represent a unique probability distribution (Pearl, 1988). Bayesian networks have the advantage that they can be built from prior knowledge alone, although as Heckerman (1999) observes this is only realistic for problems consisting of a few variables and where definite prior knowledge exists. In recent years there has been a growing interest in learning Bayesian networks from data; see for example the work of Buntine (1991) and Heckerman et al. (1995). The majority of this research has focused on learning the global structure, which corresponds to the edges of the DAG, of the network. Once a Bayesian network has been constructed, to be able to determine various probabilities of interest probabilistic inference is required. Although conditional independence is used in a Bayesian network to simplify probabilistic inference, exact inference in an arbitrary Bayesian network for discrete variables is NP-hard (Cooper1990). Even approximate inference (for example by the use of Monte Carlo methods) is NP-hard (Dagum and Luby, 1993).

The use of tree-based classification and regression has been widely used in the machine learning community. Popular methods for decision-tree induction are ID3 (Quinlan, 1986), C4.5 and CART (Classification And Regression Trees) (Breiman et al., 1984). To construct an appropriate decision tree, CART first grows a decision tree by determining a succession of splits (decision boundaries) that partition the training data into disjoint subsets. Starting from the root node that contains all the training data, an exhaustive search is performed to find the split that best reduces some minimum cost-complexity principle. The overall result of this continual process is a sequence of trees of various sizes; the final tree selected is the tree that performs best when an independent test set is presented. Thus, the CART algorithm can be considered to consist of two stages: tree growing and tree pruning. Interpretability can be introduced into this method simply by reading off which inputs are incorporated into the final tree structure.

The Additive Spline Modelling of Observational Data (ASMOD) algorithm has been employed for finding interesting trends in data (Kavli and Weyer, 1995). In the ASMOD approach a set of piecewise polynomial basis functions are defined by a series of knots. The introduction of additional knots within the basis functions enables increasingly complex functions to be approximated, whilst an increase in the order allows potentially smoother functions to be obtained. The resulting model is a multidimensional polynomial surface which can be decomposed as a series of local, low order polynomials, which can be considered as a set of local kth order Taylor series approximation to the system. The model is constructed using a forward selection, backwards elimination algorithm that updates the model iteratively by selecting the best refinement from a set of possible refinements. These refinements can include: knot insertion, knot deletion, subnetwork deletion, as well as decreasing or increasing the order of the B-spline. At each stage in the model construction process an MSE based statistical significance measure (Gunn et al., 1997) can be used to select the optimal model refinement. A limitation of a number of these methods is their convergence to local rather than global minima.

Graphical models, Bayesian network and algorithms such as CART have been described as *space partitioning* algorithms (Rao et al., Feb 1999). This approach is derived from the observation that despite the large "volume" of the data input space, it is often the case that the data is localised to a few relatively dense "clusters". A natural extension of this idea is to divide the input space into regions of different sizes and shapes and to use suitable local regression models in each region. As observed by (Rao et al., Feb 1999) this obviates the need to use the entire training data to obtain a regression estimate. For example, in the case of the graphical model, the dependencies are evaluated on a pairwise basis. This has the associated advantage that the computational complexity is reduced. A limitation of this approach is the input space partition and the local models must be designed carefully.

The CART algorithm divides the feature space into a sequence of nested regions, and uses simple local averaging models in each of the regions. The ASMOD algorithm, and the related Multivariate Adaptive Regression using Splines (MARS) approach of Friedman (1991), are similar to CART but with local averaging based on splines which makes the algorithm more flexible and also more complex. The use of splines is attractive because it allows smoothness (as described earlier) of regression functions across region boundaries.

An important drawback of the CART and MARS approaches is that the shapes of the regions over which local averaging is performed are highly restrictive. In most CART and MARS implementations, the regions are constrained to be hyper-rectangles with sides parallel to the co-ordinate axes (Breiman et al., 1984; Friedman, 1991). This hence restricts the space of models that can be considered. Another serious drawback of these approaches is the greedy nature of the learning algorithm. In the basic design approach, the partitions of the input space are designed in a hierarchical fashion. However, the upper levels in the hierarchy cannot be re-optimised as more regions are introduced resulting in convergence to local rather than global minima.

2.3 Feature Extraction and Data Pre-processing

In real world concept learning problems, the representation of data often uses many features, only a few of which may be related to the target variable. As a result, the learning problem is compounded if the dataset contains a large quantity of redundant information. In this situation feature selection is important both to speed up learning, potentially improve a models predictive ability and to introduce model interpretability. Feature selection can formally be defined as the problem of choosing a small subset of features that ideally is necessary and sufficient to describe the target variable. The notion of feature selection is inherently linked to the idea of structural regularisation described earlier.

A large number of feature selection algorithms have been proposed in the neural network literature. Many of the approaches described involve an exhaustive search over all subsets of a given feature set. However, an exhaustive search of the feature space is intractable. Devijver and Kittler (1982) review heuristic methods for reducing the search space, but they are often found to be suboptimal. It is always possible for the methods to miss relevant features since convergence is typically to a local rather than global minimum.

Pre-processing of data can allow a more efficient and/or meaningful data analysis to be performed by extracting and transforming features from a set of data. Preprocessing typically involves transformation of the raw data to a new set of data, such that the *salient information* within the dataset is retained or enhanced, and additionally the conditioning of the following numerical process can often be improved.

Rather than representing the entire transformation from the set of input variables to the set of output variables by a single function, there is often great benefit in breaking down the mapping into an initial pre-processing stage. This is attractive because this additional step can greatly improve the performance of a learning system (Bishop, 1995).

Feature selection in its basic form consists of eliminating as many features in a given problem as possible, without sacrificing model accuracy. A model with fewer inputs has fewer adaptive parameters to be determined, and these are more likely to be properly constrained by a dataset of limited size. In addition, a model with fewer parameters may be faster to train (Bishop, 1995).

Having a minimal number of features often leads to better generalisation performance, and simpler models which may be more easily interpreted (Bradley et al., 1998).

Any procedure for feature selection must be based on two criteria. Firstly, a criterion must be defined by which it is possible to judge whether one subset of features is better than another. Secondly, a systematic procedure must be found for searching through candidate subsets of features. In a practical application, a non-exhaustive search procedure is needed in order to limit the computational complexity of the search process.

A common problem encountered when using large datasets is the curse of dimensionality (Bellman, 1961). This refers to an exponential growth in complexity, of the learning problem, as a result of an increase in dimensionality of the inputs. Hence, one of the most important forms of pre-processing may be a reduction in the dimensionality of the input data. At the simplest level this could involve using prior knowledge to discard a subset of the inputs or to form input combinations. This knowledge may then be used to simplify the model's representation, and improving the conditioning of the learning problem. A model with fewer inputs typically has fewer adaptive parameters to be determined, and these are more likely to be properly constrained by a dataset of limited size, reducing the possibility of overfitting. However, an important consideration with all pre-processing techniques is that care should be taken to ensure that salient information is not lost from the dataset when they are used.

2.4 Additive Models and Interpretability

Additive models are well known within the statistics community (see for example Hastie and Tibshirani (1990)), and can be considered as a generalisation of linear models. They are appealing because being essentially a superposition of one-dimensional functions, they have a low complexity and they share with linear models the feature that the effects of the different variables can be examined separately. A simple additive model has the form,

$$f(\boldsymbol{x}) = \sum_{i=1}^{F} f_i(x^i) \tag{2.7}$$

where x^i is the i-th component of the F dimensional input vector $\boldsymbol{x}$ and f_i are univariate functions. There has been surprise at the relative success of additive modelling methods in the machine learning community (Rasmussen, 1996). Much of this interest has centered around the predefined nature of the additive model and whether such models can capture the fundamental properties of the physical world.

Girosi et al. (1995) have considered this question by considering the problem of object recognition, or model control. We can recognise almost any object from any of many smaller subsets of its features that are both visual and non-visual. We can perform many motor actions in several different ways. In most situations, our sensory and motor worlds are redundant. In terms of generalised regularisation networks this means that instead of high dimensional centres, any of the several lower dimensional centres are often sufficient to perform a given task. Splitting the recognisable world into additive parts may well be preferable to reconstructing it in its full multidimensionality, because a system composed of several independently accessible parts is inherently more robust than a whole simultaneously model dependent on each of its parts. The small loss in uniqueness of recognition is easily offset by the gain against noise and occlusion. Girosi et al. (1995) also propose the possible meta-argument that humans would not be able to understand the world if it were not additive because of the very large number of necessary examples, for example the high dimensionality of any sensory input such as an image. As such we may be tempted to conjecture that our sensory world is biased towards an additive structure. Additive models and interpretability are discussed further in Chapters 4 and 5.

2.4.1 Interpretability as a means for Model Selection

Selecting a model from a large class of plausible models is an important problem in machine learning and statistics. A classic example is the selection of variables problem in linear regression analysis. Choosing suitable transformations of the predictor and/or the response variable in linear regression is another major instance of model selection. A standard definition involves *good* models being able to make predictions close to what has been observed for an identical experiment.

Among the several criteria that have been proposed for model selection, the Akaike information criterion (AIC) (Akaike, 1974) and Schwarz's Bayes information criterion (BIC) (Schwarz, 1978) are widely used, however they are not well-posed. An inherent problem with these criteria is that they do not allow prior knowledge for model choice. Moreover, their definitions rely on asymptotic considerations that may be inappropriate for finite problems. However, the complete Bayesian approach to model selection requires the specification of prior probabilities over the class of models under consideration, and the specification of priors for the parameters of each model. In selecting between two models, often we can reasonably carry out these specifications, and use Bayes factors or posterior model probabilities to make the final model choice. With a large number of models, the fully Bayesian solution is difficult to implement. In this thesis, model interpretability is used to address the problem of model selection. Here, emphasis is placed on interpreting the model structure rather than the model parameters. From a Bayesian perspective this is a sensible strategy since the set of model parameters do not contain very much physical meaning, and they are free of any asymptotic definitions allowing the ready incorporation of prior knowledge.

2.4.2 Interpretability and Model Validation

A number of questions arise when considering the model generated by a system: in which situations can a particular model be justifiably applied? How well does the model represent the underlying data generating distribution? What are the inherent limitations in the model's construction and how can they be controlled? For model validation to be useful it must involve a broad range of activities, for example the theoretical analysis of a models assumptions, empirical experiments on data, as well as comparisons between model results and prior knowledge.

In data modelling, the term 'validation' is often associated with 'establishing the agreement between predictions and observations', as opposed to 'verification' which tends to be used in the sense of 'checking the models implementation and construction'. Use of this terminology is by no means consistent, nor is it accepted in other fields.

The ability to visualise, and hence assess the model structure, together with a theoretical analysis of the modelling assumptions can be considered to be the key features of model validation. It is necessary to explicitly identify model assumptions and approximations, and to state the general conditions under which these are valid. Visualisation of the trends discovered between inputs and outputs also enables assessment of model accuracy, with respect to prior knowledge.

2.5 Comparison and Critique of Prior Work

A common problem in learning theory occurs when insufficient data exists for making accurate inferences. Many modelling methods will as a result pick a model that is 'too simple'. It is important to realise that this does *not* mean that 'simpler models are *a priori* more likely to be true', or that 'nature prefers simplicity', but rather the rationale behind picking only simple models when few data are available is rather that the dataset is too small to identify a complex model with any certainty. The fact that modelling methods may select an overly simple model, when only small datasets exist, raises the question whether the overly simple model could be used to provisionally make reasonable predictions. Once we acknowledge that our models always have a chance of being partially wrong, a new difficulty arises. This raises the important question of what can and what cannot be reliably inferred from a statistical model of the data: what exactly does a model say about the modelled situation?

Recently there has been a renewed interest for kernel based methods to solve inference problems. This has in part been due to the success of the *Support Vector Machine* (SVM) approach (Cortes and Vapnik, 1995; Vapnik, 1998). Kernel based learning methods represent the function value to be learned with a linear combination of terms of the form $K(\boldsymbol{x}, \boldsymbol{x}_i)$, where $\boldsymbol{x}_i$ is generally the input vector associated with one of the training examples and K is a symmetric positive definite kernel function (Aronszajn, 1950).

Sparsity of the representation is an important issue for both the computational efficiency, and for its influence on model interpretability. However, the sparsity of the solutions found by the SVM algorithm is difficult to control, and often the solutions found are not very sparse.

For reasons that are typically associated with their architectures and their learning algorithms, some learning systems are limited in their capability to handle large datasets and, although not considered in this thesis, to perform online learning. This is in part due to the computational cost associated with inverting matrices the size of the training dataset, and where the number of basis functions grows proportionally to the amount of training data, as such this makes sparse representations desirable. Within the kernel community, a number of computational approaches have been developed that rely on partitions of the large dataset into smaller subsets. Learning then takes place by training algorithms on these smaller subsets and then combining the overall estimate. These approaches are reviewed and discussed in Section 4.7.

A number of algorithms have been developed primarily in the signal processing community that are capable of resulting in a sparse solution. There are many interesting links with the research on kernel based learning algorithms developed in the machine learning community. The wavelet algorithms, matching pursuit and basis pursuit denoising, and their role in obtaining sparse solutions are reviewed extensively in Chapter 5. Connections between wavelet algorithms and SVMs have already been reported in (Poggio and Girosi, 1998). More recently, (Smola and Schölkopf, 2000) have shown connections between matching pursuit, kernel-PCA (Scholkopf et al., 1998), sparse kernel feature analysis, and how this *greedy* algorithm can be used to compress the kernel matrix in kernel methods to allow the handling of large datasets.

In a large number of supervised learning problems feature selection is important for a variety of reasons: generalisation performance, computational running time requirements, as well as interpretability issues imposed by the problem itself (Weston et al., 2000). Previous work on feature selection for SVMs have been limited to linear kernels or linear probabilistic models (Jebera and Jaakkola, 2000).

The feature selection problem as described in Section 2.3 can be addressed in the following two ways: (1) Find the smallest set of features that minimises the expected generalisation error; or (2) Find the smallest set of features that gives an expected generalisation error not more than a constant factor worse than the minimum expected generalisation error. In both of these problems the expected generalisation error is of course unknown and must be estimated.

In the machine learning literature it is necessary to distinguish between two types of method to solve this problem, the so-called *filter* and *wrapper* methods (Weston et al., 2000). Filter methods are defined as a preprocessing step to induction that can remove irrelevant data attributes before induction occurs, and hence we wish them to be valid for any set of functions. An example of a popular filter method is the use of Pearson correlation coefficients. The wrapper method is defined as a search through the space of feature subsets using the estimated accuracy from an induction algorithm as a measure of goodness of a particular feature subset. Weston et al. (2000) propose that wrapper methods can provide more accurate solutions than filter methods, but in general are more computationally expensive since the induction algorithm must be evaluated over each feature set considered, typically using cross validation techniques as a measure of goodness of fit. In their work Weston et al. (2000) suggest a feature selection algorithm for SVMs that takes advantage of the performance increase of wrapper methods whilst avoiding their computational complexity. Their approach avoids the combinatorial nature of the feature selection problem by using a greedy search method that adds or removes features in a forwards regression/backwards elimination approach, similar to that used in the ASMOD (Kavli and Weyer, 1995) training algorithm; however convergence is to a local minima.

In kernel based methods, large datasets can pose significant problems since the number of basis functions required for an optimal solution often equals the number of samples. Smola and Schölkopf (2000) and Smola and Bartlett (2000) have proposed a sparse greedy approximation method, in a manner similar to that of Weston et al. (2000), that constructs a sparse compressed representation of the kernel matrix. Their approach is based upon the minimisation of an L_0 norm on a set of hyperparameters in a manner similar to that deployed in the wavelet based method of matching pursuit. This algorithm is discussed in more detail in Sections 4.7 and 5.1. An inherent limitation of such greedy learning algorithms is their convergence to local rather than global minima.

As well as providing a good fit to the data and plausible predictions based on the data, an important requirement is that our model gives us information about the underlying probability distribution from whence the data came. A popular method for obtaining sparse and interpretable solutions has centered on the introduction of hyperparameters. For example, Automatic Relevance Determination (ARD) (MacKay, 1994; Neal, 1995), the process of determining whether individual inputs cause significant variation in the outputs, can be performed by specifying different regularisation classes for weights connected to each input. This allows a sparse representation of the model structure by removing redundant input variables. The Relevance Vector Machine (RVM) has recently been introduced by (Tipping, 2000b). In this model a separate regularisation parameter is introduced for each parameter in the model, and the objective is to drive as many of these parameters to zero reflecting a sparse representation. These methods are reviewed in the context of sparse interpretable methods in Chapters 3 and 4.

2.6 Summary

A central objective of machine learning research is to develop algorithms that learn predictive relationships from data. This is a central component of data mining and knowledge discovery tasks. However, this is a difficult task because inferring a predictive function from data is in fact an ill-posed problem. To make this problem well-posed one needs to somehow *calibrate* the complexity of the proposed function class to the amount and quality of available sample data. A classical approach is to perform model selection where one imposes a preference structure over function classes, and then optimises a combined objective of class preference and data fit. In doing so, however, it would be useful to have an accurate estimate of the expected generalization performance at each preference level; one could then pick the function class that obtained the lowest expected error, or combine functions from the functions classes with the lowest expected error, and so on.

This chapter has surveyed the main ideas behind recent research in sparse interpretable modelling methods. Many approaches have been proposed in the past for this purpose, both in the statistics and the machine learning communities. Recently in machine learning there has been significant interest in new techniques for evaluating generalisation error, for optimising generalisation error, and for combining and selecting models. A number of methods are applicable only to a highly restricted class of datasets. Invariably these overly restrictive representations embody strong assumptions about the modelling scenario, and consequently they are only suitable for solving a restricted set of modelling tasks that fit these assumptions.

Throughout this thesis, the main focus is on model interpretability, in which knowledge of a model is provided by an explicit model of the underlying data generating mechanism. Recognition is accomplished by finding a correspondence between certain features of the data and comparable features of the model. The two most important issues that a method must address are what constitutes important features, and how a correspondence may be found between image and model features. These new approaches suggest that better generalisation performance can be obtained using new, broadly applicable procedures. Progress in this area has not only been important for improved understanding of how machine learning algorithms generalise, it has already been demonstrated to be very useful for practical applications of machine learning and data analysis.

Chapter 3

Bayesian Inference and Learning Algorithms

3.1 The Advantages and Disadvantages of being Bayesian

Bayesian inference can be regarded as a powerful tool with which to tackle the problem of data modelling. The Bayesian learning approach is based upon the expression of this knowledge in terms of a probability. In general, these probabilities can be interpreted as expression of our degrees of belief in the various possibilities (Neal, 1995). A model is designed for a particular system and when the data arrives from the system the model is adapted in light of the data. The model then provides a representation of our prior beliefs about the system and the information derived from the data (Bishop, 1995).

The statistical methodology of Bayesian learning for inductive inference has been the focus of an intensified research effort over the last decade. This has in part been due to the work of MacKay (1991), Neal (1995) and Gull (1989). The concepts and methods described by the framework are general and have been applied to many data modelling problems (Sykacek et al., 2000). The practical benefits of the Bayesian approach include principled methods for model regularisation, feature selection, hyperparameter determination, model selection, active learning and the calculation of error bars (Bishop, 1995; Dybowski and Roberts, 2000).

Despite the apparent advantages surrounding the use of Bayesian methods, their application is not always straightforward. Two principle areas of difficulty exist. The first one centres around the mathematical complexity that often occurs in Bayesian approaches. Approximations often have to be made to avoid the intractable high dimensional integrals that typically exist. This point is discussed further in Section 3.7. Another potential problem with the Bayesian approach is the choice of prior probability distribution that captures our degrees of belief. Classical statisticians (or *frequentists*) would argue that the Bayesian prior is subjective, and hence the Bayesian approach is of questionable worth. In contrast the frequentist approach to statistical inference does not have this problem, since the objective here is to only express the long-run frequencies of the outcome of repeatable experiments. A frequentist strategy for learning takes the form of a point estimator of unknown quantities. However, as Gibbs (1997) argues this is not the case. Any prior placed on an event is determined by our prior knowledge of that event. The only requirement for a prior to be objective is that two people having exactly the same prior knowledge about an event choose the same prior probability distribution.

As observed by Gibbs (1997) and Jaynes (1994), the problem with priors lies not with subjectivity but with the difficult process of assigning probabilities. How do we decide which numbers truly reflect what we believe? Priors also cause problems when the prior belief is very hard to express in terms of the models that are being used. The temptation then arises to adapt the priors to fit the model, viz. choose mathematically convenient priors for a given problem that may not accurately reflect our prior beliefs.

This debate over Bayesian versus frequentist statistics has been an ongoing argument between both camps, see for example the work of Jaynes (1994). It is not the aim of this thesis to contribute to this argument, which at its heart is a philosophical argument about the meaning of a probability, but rather to show how the Bayesian approach can be used to introduce interpretability into advanced nonlinear modelling methods.

3.2 Uncertainty and Probability Distributions

The problem of regression is to approximate an unknown function from the observation of a limited sequence of (typically) noise corrupted input/output data pairs. Consider a dataset $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^N$, drawn from an unknown probability distribution, where $\boldsymbol{x}_i \in \mathbb{R}^F$ represents a set of inputs and $y_i \in \mathbb{R}$ represents a single output. The empirical modelling problem is to discover an underlying mapping $\boldsymbol{x} \rightarrow y$ that is consistent with the dataset $\mathcal{D}$.

The Bayesian inference problems can be posed as follows. Consider a set of models $\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_L$ competing to explain the data. Our initial beliefs about the plausibility of these models are described by a set of probabilities $p(\mathcal{H}_1), p(\mathcal{H}_2), \ldots, p(\mathcal{H}_L)$. Each model, $\mathcal{H}_i$, makes predictions about how likely the different datasets $\mathcal{D}$ are. These predictions are described by a probability distribution $P(\mathcal{D}|\mathcal{H})$. When the data are observed, Bayes' theorem (Equation 3.1) can be used to describe how to infer the parameters, $\boldsymbol{w}$, of the model given the data

$$P(\boldsymbol{w}|\mathcal{D}, \mathcal{H}) = \frac{P(\mathcal{D}|\boldsymbol{w}, \mathcal{H})P(\boldsymbol{w}|\mathcal{H})}{P(\mathcal{D}|\mathcal{H})} \tag{3.1}$$

The fundamental concept of Bayesian analysis is that the plausibilities of alternative hypotheses are represented by probabilities, then the process of inference is performed by evaluating these probabilities. The quantity $P(\boldsymbol{w}|\mathcal{H})$ represents a prior probability for the parameters $\boldsymbol{w}$ of a model $\mathcal{H}$. If there is no particular reason to prefer one model over another, equal prior probabilities can be assigned to all of the models. Since the denominator of Equation 3.1 does not depend upon the model, the different models, $\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_L$, can be compared by evaluating the likelihood term, $p(\mathcal{H}_i|\mathcal{D})$, which is called the *evidence* for model $\mathcal{H}_i$ (MacKay, 1994).

3.3 Traditional Neural Networks

One of the most straightforward approaches to data modelling is to represent the underlying model structure in parametric form. The values of the parameters can then be optimised to give the best fit to the data. In contrast, non-parametric methods do not assume a particular functional form, but allow the form of the model structure to be determined entirely by the data. These methods typically suffer from the problem that the number of parameters in the model grows with the dataset size.

Neural networks have received an immense research interest and have been applied in many areas. The books by Bishop (1995) and Haykin (1999) provide good descriptions. As Bishop (1995) observes neural networks can be seen as providing a framework for representing non-linear functional mappings between a set of input variables and a set of output variables. This is achieved by representing the non-linear function of many variables in terms of compositions of non-linear functions of a single variable often referred to as *activation functions.* A number of neural network architectures have been proposed, in this work we only consider a neural network with a single hidden layer, with a single output node. The network output can be represented as,

$$f(\boldsymbol{x}) = \sum_i a_i S\left(\sum_j w_{ij}\boldsymbol{x}_j\right) \tag{3.2}$$

where the a_i denote weights between the hidden layer and the output node, w_{ij} are the weights between the inputs and the hidden layer, $S(z) = \tanh(z)$ is the hidden node activation function (i.e. bounded, smooth, and strictly monotonically increasing), and the input vector $\boldsymbol{x}$ is taken to be augmented with a fixed input $x_0 = 1$, to provide an adjustable bias for the model; a similar augmentation is used in the hidden layer.

Neural networks have received an immense research interest and have been applied in many areas. The books by Bishop (1995) and Haykin (1999) provide good descriptions. Neural networks can be considered to be semi-parametric modelling methods, that allow a very general class of models to be considered in which the number of parameters can be controlled in a systematic way to build flexible models. The total number of parameters in the model can be varied independently from the size of the dataset. Neural networks can be used to define probabilistic models for regression and classification tasks by using the network outputs to define the conditional distribution for one or more target variables given the various possible values for the input vector $\boldsymbol{x}$. This is discussed further in Section 3.4.

3.4 Bayesian Neural Networks

Bayesian techniques have been widely and successfully used in the neural networks and statistics communities. In a neural network, the Bayesian framework can be used to learn the parameters $\boldsymbol{w}$ of the network from the dataset. The Bayesian approach considers a probability distribution function over parameter (or weight) space, representing the relative degrees of belief in different values for the parameter vector. Since, in general we have little idea of what the parameter values should be, the prior distribution may express some rather general properties such as the smoothness of the network function. Once the data has been observed, this prior distribution can be converted to a posterior distribution using Bayes' theorem given by Equation 3.1. As (Bishop, 1995) observes the posterior distribution will be more compact, expressing the fact that we have learnt something about the extent to which the different network parameter values are consistent with the data. However, in order to evaluate the posterior; specific functional forms for the prior distribution over weights, $P(\boldsymbol{w})$, and for the likelihood $P(\mathcal{D}|\boldsymbol{w})$ need to be chosen.

MacKay (1994) has considered the simple case of using a Gaussian distribution for both the prior and likelihood. This can then be written as,

$$P(\boldsymbol{w}) = \frac{1}{Z_{\boldsymbol{w}}(\alpha)} \exp\left\{-\alpha E_{\boldsymbol{w}}(\boldsymbol{w})\right\} \tag{3.3}$$

where $Z_{\boldsymbol{w}}(\alpha)$ is a normalisation factor given by,

$$Z_{\boldsymbol{w}}(\alpha) = \int \exp\left\{-\alpha E_{\boldsymbol{w}}(\boldsymbol{w})\right\} d\boldsymbol{w} \tag{3.4}$$

where $E_{\boldsymbol{w}}$ is a model regularisation term used to enforce smoothness as described in Chapter 2.

Since the parameter α controls the distribution of the neural network parameters it is referred to as a *hyperparameter* (MacKay, 1994; Bishop, 1995; Neal, 1995). The likelihood function can be written in the form (MacKay, 1994),

$$P(\mathcal{D}|\boldsymbol{w}) = \frac{1}{Z_{\mathcal{D}}(\beta)} \exp\left\{-\beta E_{\mathcal{D}}(\boldsymbol{w})\right\} \tag{3.5}$$

where $E_{\mathcal{D}}$ is an error function, and β is a hyperparameter controlling the variance of the noise present in the data. The function $Z_{\mathcal{D}}$ is a normalisation factor given by,

$$Z_{\mathcal{D}}(\beta) = \int \exp\left\{-\beta E_{\mathcal{D}}(\boldsymbol{w})\right\} d\mathcal{D} \tag{3.6}$$

Assuming that the target data is generated from a smooth function with additive zero-mean Gaussian noise, the probability of observing a data value y for a given input vector $\boldsymbol{x}$ is,

$$P(y|\boldsymbol{x}, \boldsymbol{w}) \propto \exp\left\{-\frac{\beta}{2}(y - f(\boldsymbol{x}; \boldsymbol{w}))^2\right\} \tag{3.7}$$

where $f(\boldsymbol{x}; \boldsymbol{w})$ is the output of the Bayesian neural network model. Provided that data points are drawn independently from this distribution,

$$\begin{aligned} P(\mathcal{D}|\boldsymbol{w}) &= \prod_{n=1}^{N} P(y_n|\boldsymbol{x}_n, \boldsymbol{w}) \\ &= \frac{1}{Z_{\mathcal{D}}(\beta)} \exp\left\{-\frac{\beta}{2}\sum_{n=1}^{N}(y_n - f(\boldsymbol{x}_n; \boldsymbol{w})^2\right\} \end{aligned} \tag{3.8}$$

Using the general expressions for the prior and likelihood Bayesian learning simplifies to finding the parameter vector that minimises the costfunction,

$$S(\boldsymbol{w}) = \frac{\beta}{2}\sum_{n=1}^{N}(y_n - f(\boldsymbol{x}_n; \boldsymbol{w}))^2 + \frac{\alpha}{2}\sum_{i=1}^{W} w_i^2 \tag{3.9}$$

where N represents the number of data points in the dataset $\mathcal{D}$, and W represents the number of parameters in the neural network model. Apart from an overall multiplicative term, this is a sum of squares error function with an L_2 norm, or zeroth order regularisation term (Bishop, 1995).

The Bayesian formalism for a neural network model is described in terms of the posterior probability distribution of parameters. If a new input vector is presented to the network, then as (Bishop, 1995) observes the distribution of parameters gives rise to a distribution of network outputs. The distribution of outputs for a given input vector $\boldsymbol{x}$ can then be written as,

$$P(y|\boldsymbol{x}, \mathcal{D}) = \int P(y|\boldsymbol{x}, \boldsymbol{w}) P(\boldsymbol{w}|\mathcal{D})\, d\boldsymbol{w} \tag{3.10}$$

where $P(\boldsymbol{w}|\mathcal{D})$ is the posterior distribution of the model parameters. Making a Gaussian approximation for the posterior distribution together with Equation 3.9 gives,

$$P(y|\boldsymbol{x}, \mathcal{D}) \propto \int \exp\left\{-\frac{\beta}{2}(y - f(\boldsymbol{x}; \boldsymbol{w})^2)\right\} \exp\left\{-\frac{1}{2}\Delta\boldsymbol{w}^T \boldsymbol{A} \Delta\boldsymbol{w}\right\} d\boldsymbol{w} \tag{3.11}$$

Approximating the network output $y(\boldsymbol{x}; \boldsymbol{w})$ by a linear expansion around the MAP estimate of the parameter vector (MacKay, 1994; Bishop, 1995) gives a Gaussian distribution of the form,

$$P(y|\boldsymbol{x},\mathcal{D}) = \frac{1}{\sqrt{2\pi}\sigma_y}\exp\left\{-\frac{(y-y_{MP})}{2\sigma_y^2}\right\}, \tag{3.12}$$

which has a mean y_{MP} and variance $\sigma_y^2 = \frac{1}{\beta}\Delta\boldsymbol{w}^T\boldsymbol{A}^{-1}\Delta\boldsymbol{w}$.

3.5 Hyperparameters can Introduce Interpretability

A problem associated with a traditional neural network model is that learning is compounded because learning often takes place in the presence of input variables that have no effect on the output. As described in Section 2.3, feature selection methods are attractive to speed up learning, improve a models predictive ability and to introduce interpretability.

Bayesian learning within a neural network addresses this problem by introducing Automatic Relevance Determination (ARD)(MacKay, 1994; Neal, 1995). This methodology attempts to reject redundant inputs, thereby increasing generalisation performance, it also has the added advantage of introducing interpretability allowing an expert to assess the final model structure by looking at which inputs were selected as being relevant in determining the output.

In an ARD model, each input variable has an associated hyperparameter that controls the magnitudes of the weights on connections to the input unit. Figure 3.1 shows a scematic diagram of a traditional neural network with hyperparameters. The figure shows a network with three inputs ($\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3$), and three associated hyperparameters ($\alpha_1, \alpha_2, \alpha_3$) controlling the parameters (represented by three different dashed lines) from the inputs to the hidden layer. If the hyperparameter associated with an input indicates a small standard deviation for weights out of that input, these weights are likely to be small, and that input will have little effect on the output; if the hyperparameter specifies a large standard deviation the effect of that input will be significant (Neal, 1995). From the work of MacKay (1991) and Neal (1995), together with the empirical evaluation of the ARD hyperparameter determination methods in section 3.8 the ARD hyperparameters for relevant inputs converge to a value of zero, whilst those for irrelevant inputs converge to infinity. Interpretation of these hyperparameter values enables an inputs influence on the network to be assessed, providing a method for knowledge extraction. In addition to hyperparameters that control the input to hidden layer weights, there are three additional hyperparameters in the network that control the hidden nodes bias, output bias, and the hidden to output layer weights.

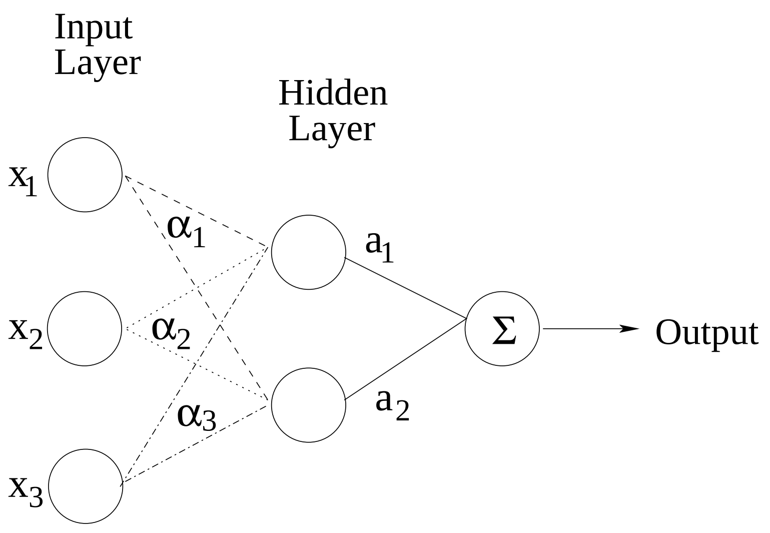


FIGURE 3.1: Illustration of a Bayesian neural network with ARD hyperparameters. The network consists of three inputs, three associated hyperparameters, two hidden nodes and a single output. The different parameters connecting the different inputs to the hidden nodes are represented by different dashed lines.

3.6 Prior Distributions for Parameters and Hyperparameters

As Neal (1995) observes at first sight the Bayesian framework may not appear to be suitable for use with neural networks. The first stage of Bayesian inference starts with a prior distribution for the model parameters, which is supposed to embody our prior beliefs about the problem. In a neural network, the parameters connect the input variables to the hidden nodes, whose relationship to anything that we might know about the problem seems obscure. MacKay (1991) has used a Gaussian prior distribution for the weights and biases of the neural network. In this work the variance of the Gaussian prior is a hyperparameter that allows the model to adapt to the degree of smoothness indicated by the data. In a Bayesian model of this type, the role of the hyperparameters controlling the priors for weights is roughly analogous to the role of a weight decay constant in conventional training.

In their work Buntine (1991) discuss several possible schemes for prior distributions, such as priors that favour networks that produce high or low entropy predictions, or that compute smooth functions. The degree of preference imposed can be controlled by a hyperparameter. This work links the choice of prior for the network weights to the actual effects of these weights on the function computed by the network which is clearly necessary if a prior is to be chosen that represents our beliefs about this function. A limitation of this work however is that it can only be applied to simple network models.

Likelihood	Prior
Binomial	Beta
Poisson	Gamma
Normal	Normal
Exponential	Gamma

TABLE 3.1: Some common choices for conjugate priors.

3.6.1 The Notion of Conjugate Priors

The conditional distribution of the predicted target, y, given the dataset, $\mathcal{D}$, involves evaluating the following integral,

$$P(y|\boldsymbol{x}, \mathcal{D}) = \int P(\mathcal{D}, \boldsymbol{w})P(\boldsymbol{w}|\mathcal{D})\, d\boldsymbol{w} \tag{3.13}$$

Typically, the evaluation of such integrals over the entire parameter space is a very complex undertaking given the high dimensionality of the parameter space. As observed by (Bernardo and Smith, 1996) such evaluation is only analytically feasible for the class of model functions for which the posterior distribution given by Equation 3.13 has the same functional form as the prior. For a given choice of likelihood, a prior distribution that gives rise to a posterior having the same functional form is said to be a *conjugate prior*. An example of a conjugate prior analysis is an iterative Bayesian prior distribution which is the update of the prior distribution for a set of parameters $P(\boldsymbol{w})$ using a succession of data points, with the resulting posterior at each stage that is used as the prior in the next stage. This is a conjugate prior because the distribution would retain the same functional form throughout. Duda et al. (2000) refer to such distributions as *reproducing densities*, and include the Gaussian distribution as the most commonly encountered example. Table 3.1 lists some common choices for conjugate prior given a particular form of the likelihood.

3.7 Hyperparameter Determination Methods

In the Bayesian neural network method and ARD method described in Section 3.4, we have assumed that the values of the hyperparameters $\boldsymbol{\alpha}$ and β are known. However, for most applications we will have little idea of suitable values for $\boldsymbol{\alpha}$, although we may have an idea of the noise variance β. In a truly Bayesian framework, parameters and hyperparameters whose values are unknown should be integrated out by a process referred to as marginalisation. Three approaches to the treatment of hyperparameters have been discussed in the literature. Two of these are based on approximations and are discussed in Sections 3.7.1 and 3.7.8, whilst the last performs integrals over $\boldsymbol{\alpha}$ and β analytically and is discussed in Section 3.7.2.

3.7.1 Laplace Approximations

A standard approach for parametric approximation is the Laplace approximation. A variation of this approach has been introduced to the neural networks community by MacKay (1994) under the title of the evidence framework. This is computationally equivalent to the *type-II* maximum likelihood method of Gull (1989). This approach thus finds values for the hyperparameters which maximise the posterior probability and then perform the remaining calculations with the hyperparameters set to these values.

As MacKay (1994) observes, the posterior distribution of the network parameters is given by,

$$P(\boldsymbol{w}|\mathcal{D}) = \iint P(\boldsymbol{w}|\boldsymbol{\alpha}, \beta, \mathcal{D}) P(\boldsymbol{\alpha}, \beta|\mathcal{D})\, d\boldsymbol{\alpha}\, d\beta \tag{3.14}$$

where the dependency on the hyperparameters $\boldsymbol{\alpha}$ and β on the posterior has been made explicit. Assuming that the posterior probability distribution $P(\boldsymbol{\alpha}, \beta|\mathcal{D})$ for the hyper-parameters given in Equation 3.14 is sharply peaked around their most probable values $\boldsymbol{\alpha}_{MP}$ and β_{MP}. The maximum a posteriori (MAP) values for these hyperparameters can then be found by maximising the likelihood $P(\mathcal{D}|\boldsymbol{\alpha}, \beta)$ a term called the *evidence* for $\boldsymbol{\alpha}$ and β (MacKay, 1994; Bishop, 1995).

Using a Gaussian approximation for the posterior distribution of the network parameters, the log of the evidence can be given by,

$$\begin{aligned} \log P(\mathcal{D}|\boldsymbol{\alpha}, \beta) &= \sum_i -\frac{\alpha_i}{2} \boldsymbol{w}_{MP}^T \boldsymbol{\Lambda}_i \boldsymbol{w}_{MP} - \frac{\beta}{2} E_{\mathcal{D}}\left(\boldsymbol{w}_{MP}\right) - \frac{1}{2}\log|\boldsymbol{A}| \\ &+ \frac{\mathbf{1}^T \boldsymbol{\Lambda}_i \mathbf{1}}{2} \log \alpha_i + \frac{N}{2}\log\beta - \frac{N}{2}\log(2\pi) \end{aligned} \tag{3.15}$$

where $\boldsymbol{\Lambda}_i$ is a diagonal masking matrix which is used to extract the weights associated with hyperparameter α_i. In order to maximise the log evidence with respect to α_i requires evaluation the trace of the matrix $\boldsymbol{A}^{-1}$, where $\boldsymbol{A} = \boldsymbol{H} + \sum_i \alpha_i \boldsymbol{\Lambda}_i$ and $\boldsymbol{H}$ is the Hessian matrix. Maximisation of the log evidence with respect to $\boldsymbol{\alpha}$ and β then results in the following re-estimation following formulae,

$$\begin{aligned} \alpha_i^{new} &= \frac{\gamma}{\boldsymbol{w}_{MP}^T \boldsymbol{\Lambda}_i \boldsymbol{w}_{MP}} \\ \beta^{new} &= \frac{N-\gamma}{2E_{\mathcal{D}}(\boldsymbol{w}_{MP})} \end{aligned} \tag{3.16}$$

where the quantity γ is defined by,

$$\gamma \equiv \sum_{i=1}^{W} \frac{\lambda_i}{\lambda_i + \alpha} \tag{3.17}$$

To deal with the numerical instability of the hyperparameters discussed in Section 3.7.1.1, a modified form of hyperparameter re-estimation formulae were obtained based on the principle of gradient descent,

$$\alpha_i^{new} = (1-\eta)\alpha_i^{old} + \eta \frac{\gamma}{\boldsymbol{w}_{MP}^T \boldsymbol{\Lambda}_i \boldsymbol{w}_{MP}} \qquad \eta \in [0,1] \tag{3.18}$$

Setting $\eta = 1$ gives a solution that is equivalent to that described in Section 3.7.1, Other values correspond to performing gradient descent on the hyperparameters with a step size controlled by η.

The following pseudo-code outlines the implementation of a Bayesian neural network using the evidence framework,

Algorithm 1: Bayesian Neural Network - Evidence Framework

```
Input       Dataset $\mathcal{D} = (\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$

Create      Construct Bayesian neural network with specified number of
            inputs, hidden nodes, and outputs

Initialise  Draw $\boldsymbol{w}$ from a Gaussian distribution
            Initialise ARD values, e.g. $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_d) = \mathbf{0.01}$
            Initialise noise variance, e.g. $\beta = 100$;
            Set alphatol to a constant value e.g. 0.001

Algorithm   Generate prior distributions for hyperparameters
            do{
                Use gradient descent algorithm to optimise $\boldsymbol{w}_{MP}$
                Update hyperparameters using Equation 3.17
                Generate Hessian Matrix
                Get Predictions and Variances
            }while($\frac{1}{F}\sum_i^F |\frac{1}{\alpha_i} - \frac{1}{\alpha_i^{old}}| > \delta$)

Output      $\boldsymbol{\alpha}_{MP}, \beta_{MP}, y$.
```

3.7.1.1 Limitations

The evidence framework can be considered to be an attractive method. It shares some very direct similarities with straight forward regularised regression, in the way it tries to infer the optimal hyperparameter/regularisation level. Furthermore, the evidence allows several levels of Bayesian inference to be performed. In the first level, among a number of models, the best one is chosen by maximising the evidence of the model. Despite these apparent benefits, Penny and Roberts (1998) observe that the evidence framework has only been applied to a small selection of problems; assessing the fat content of carcasses (Thodberg, 1995), vowel recognition and classification of thyroid disease (Gutjahr and Nautze, 1997), prediction of energy consumption (MacKay, 1994), classification of car number plates (Oldfield, 1995), and the classification of EEG data (Sykacek et al., 1997).

A major cause for concern is that the evidence framework relies on a number of assumptions. The posterior distribution for the parameters of a neural network is typically very complex. The Gaussian approximation to the posterior distribution, for given values of the hyperparameters is central to MacKay's Bayesian evidence framework (MacKay, 1994). Walker (1969) has shown that in the limit of an infinite training set the posterior distribution does become Gaussian. However, with a finite number of samples the assumption fails, hence the evidence cannot be computed accurately.

A number of the limitations inherent to MacKay's evidence framework, can be attributed to not evaluating the Hessian matrix explicitly, which is avoided because of its computational expense. Algorithms such as scaled conjugate gradient (which was used to minimise the Bayesian costfunction) do not evaluate the Hessian matrix explicitly, but instead compute it using a finite difference approximation to the dot product of the Hessian and the search direction.

However, when computing the ARD hyperparameters the evidence framework is constructed using only the positive eigenvalues. This is because at a local minimum, some eigenvalues of the Hessian may be negative implying that the posterior variance in some directions of weight space is negative (Bishop, 1995; Kandola et al., 1999). This situation is handled by ignoring these problematic directions and just considering the distribution in the non-negative directions. In networks which have redundant weights, that occur because they contain irrelevant features, it is possible that the Hessian matrix will be singular or nearly singular. In these cases, eigenvalues will be of machine precision order, hence the calculation of the determinant (which uses products of eigenvalues) will be unreliable. Before the hyperparameters can be re-estimated they need to be set to some initial values. The choice of initial starting values is entirely arbitrary and as shown in Figure 3.2 an incorrect choice affects the quality of the approximation.

Two alternative approaches have been proposed to overcome this problem. The first suggested by Nabney (1999), uses relatively small values of α since they correspond to a small weight decay, allowing greater model flexibility during the early stages of training. As shown in Figure 3.2, this is a sensible strategy. However, MacKay (1999b) has suggested that this problem can be overcome by using Markov Chain Monte Carlo (MCMC) methods (Geman et al., 1992; Neal, 1995) methods, which use correct Bayesian sampling of the hyperparameters and parameters, and may result in more stable behaviour.

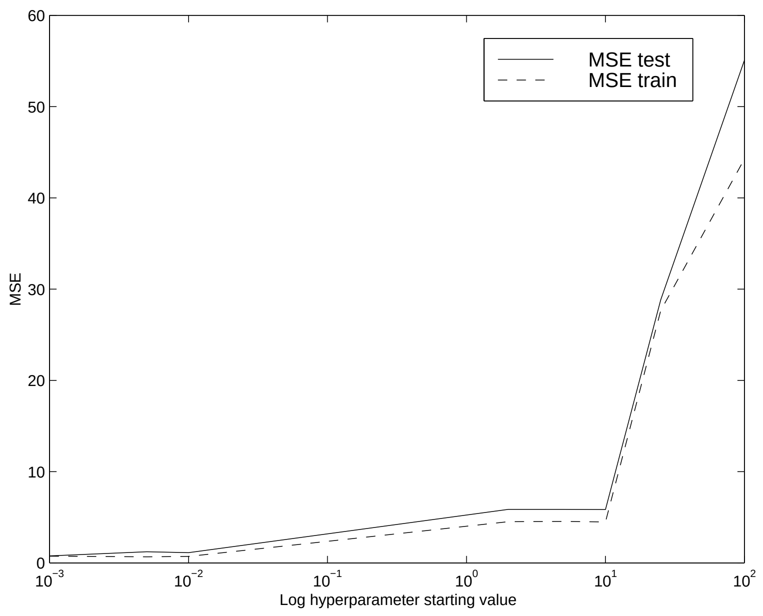


FIGURE 3.2: Variation of approximation error with choice of starting ARD hyperparameter with using the evidence framework.

3.7.2 Markov Chain Monte Carlo Sampling

Bayesian learning results in integrals over multi-dimensional spaces. Many standard numerical integration techniques, which can be used successfully for integration over a small number of dimensions, are unsuitable for evaluating the complex integrals that occur in Bayesian inference that typically involve integration over spaces of hundreds or thousands of parameters.

MCMC methods have been used extensively to solve problems in statistical physics. The Monte-Carlo method provides approximate solutions to a variety of mathematical problems by performing statistical sampling experiments. Among all numerical methods that rely on N-point evaluations in M-dimensional space to produce an approximate solution, the Monte Carlo method has absolute error of estimate that decreases as $N^{-1/2}$ whereas, in the absence of exploitable special structure all others have errors that decrease as $N^{-1/M}$ at best. An important advantage of MCMC methods is that they make no assumptions concerning the form of the underlying distribution, such as whether it can be approximated by a Gaussian, which is inherent to the evidence framework described above.

Bayesian statistics often requires integration over possibly high dimensional probability distributions to make inferences about model parameters or to make predictions. Monte Carlo integration draws samples from the required distribution, and then forms sample averages to approximate expectations. Markov Chain Monte Carlo (MCMC) sampling draws these samples using a Markov chain over a substantial period of time. For tutorial introductions see MacKay (1999a) and Besag (2000).

Once the probability density functions are known, the Monte Carlo simulation can proceed by random sampling. Many simulations are then performed and the desired result is taken as an average over the number of observations (which may be a single observation or perhaps millions of observations). Assuming that the evolution of the physical system can be described by probability density functions, then the Monte Carlo simulation can proceed by sampling from these probability distribution functions, which necessitates a fast and effective way to generate random numbers uniformly distributed on the interval [0,1].

Formally, Monte-Carlo methods attempt to solve one or both of the following problems (MacKay, 1999a):

Problem 1: to generate samples $\{x_r\}_{r=1}^{R}$ from a given probability distribution $P(x)$.[1]

Problem 2: to estimate expectations of functions under this distribution,

$$\Phi = \int \phi(x)\, dP(x) \tag{3.19}$$

Problem one is more general since the second can be solved by random samples to give the estimator,

$$\hat{\Phi} = \frac{1}{R} \sum_{r=1}^{R} \phi(x_r) \tag{3.20}$$

where $\{x_1, \ldots, x_R\}$ are generated by a process that results in each of them having a distribution defined by the posterior probability density for the parameters. This is one of the important properties of Monte Carlo methods. The accuracy of the estimate (Equation 3.20) is independent of the dimensionality of the space sampled. In simple Monte Carlo methods, the x_r are independent. When the posterior distribution is a complicated distribution generating such independent values is often infeasible. To use the MCMC method to estimate an expectation with respect to some posterior distribution a Markov chain which is *ergodic* must be constructed, which has the posterior distribution as its equilibrium distribution to which it converges from any initial state.

3.7.3 Markov Chains

Suppose that a sequence of random variables, $X_0, X_1, X_2, \ldots$ is generated, such that at time $t \geq 0$, the next state X_{t+1} is sampled from a distribution $P(X_{t+1}|X_t)$ which depends only on the current state of the chain, X_t. That is, *given* X_t, the next state X_{t+1} does not depend further on the history of the chain $X_0, X_1, \ldots, X_{t-1}$. This sequence is called a Markov chain, and $P(\cdot|\cdot)$ is called the transition kernel of the chain (Gilks et al., 1996). An important issue when using Markov chains is to understand how the starting state X_0 effects the current state X_t. This question concerns the distribution

[1]Here the term sample refers to a single realisation x whose probability distribution is $P(x)$

of X_t given X_0 which is denoted by $P^{(t)}(X_t|X_0)$. Subject to regularity conditions, the chain will gradually 'forget' its initial state and $P^{(t)}(\cdot|X_0)$ will eventually converge to a unique *stationary* distribution, $\phi(\cdot)$, which does not depend on t or X_0. Thus as t increases the sampled points X_t will look increasingly like dependent samples from $\phi(\cdot)$. Therefore after a sufficiently long *burn in* period of m iterations, points $X_t; t = 1, \ldots, R$ will be samples drawn from $\phi(\cdot)$. Therefore the output from the Markov chain can be used to estimate the expectation $E[f(X)]$, where X has a distribution given by $\phi(\cdot)$. Burn in samples are normally discarded for this calculation giving an estimator,

$$f = \frac{1}{R-m} \sum_{t=m+1}^{R} f(X_t) \tag{3.21}$$

3.7.4 The Metropolis-Hastings algorithm

The Metropolis algorithm has been widely used in optimisation theory and is the basis for the widely used optimisation method of *simulated annealing* (Kirkpatrick et al., 1983; Kandola et al., 1998). In the Markov chain defined by the Metropolis algorithm, a new state $\theta^{(t+1)}$ is generated from the previous state $\theta^{(t)}$, by first generating a candidate state using a specified proposal distribution, and then deciding whether or not to accept the candidate state, based on its probability density relative to that of the old state. If the candidate state is accepted, it becomes the next state of the Markov chain; if the candidate state is instead rejected, the new state is the same as the old state, and is included again in any averages to estimate expectations.

This algorithm only considers symmetric proposals, having the form $P(Y|X) = P(X|Y)$ for all X and Y. For example, when X is continuous, $P(\cdot|X)$ might be a Gaussian distribution with mean X and constant covariance matrix. When choosing a proposal distribution, its scale may need to be chosen carefully. A cautious proposal distribution generating small steps $(Y|X_t)$ will generally have a high acceptance rate. A bold proposal distribution generating large steps will often propose moves from the body to the tails of the distribution resulting in a low probability of acceptance. Ideally, the proposal distribution should be scaled to avoid both of these extremes (Kandola et al., 1998).

3.7.5 Gibbs sampling

Gibbs sampling, also known as the *heatbath* method in the physics and chemistry literature, is a simple case of the Metropolis-Hastings algorithm. It is applicable when sampling from a multi-dimensional parameter distribution. The posterior probability distribution is defined in terms of *conditional* distributions of the joint distribution $P(x)$. It is assumed that whilst $P(x)$ is too complex to draw samples from directly, its conditional distributions is tractable to work with. For the Gibbs sampler, the proposal distribution for updating the i^{th} component of X is,

$$P_i(Y_i|X_i, X_{-i}) = R(Y_i|X_{-i}) \tag{3.22}$$

where $R(Y_i|X_{-i})$ is the full conditional distribution. Thus, Gibbs sampling consists purely in sampling from full conditional distributions, i.e. Gibbs sampler candidates will always be accepted. The usefulness of Gibbs sampling for Bayesian inference is dependent on whether the posterior distribution of one parameter conditional on given values for the other parameters can be easily sampled from.

3.7.6 Hybrid Monte Carlo for network parameters

The implementation of Bayesian learning used in this thesis is based on the hybrid Monte Carlo algorithm of Duane et al. (1987) introduced to the neural networks community by Neal (1995). The advantage of this method is that it avoids the random walk behaviour of the Metropolis algorithm which typically increases sampling time. A Markov chain that explores the entire posterior distribution can be obtained by alternating Gibbs sampling updates for the hyperparameters, with hybrid Monte Carlo updates for the network parameters.

The hybrid Monte Carlo algorithm is expressed in terms of sampling from a Boltzmann distribution for the state of a fictitious physical system, which is defined in terms of an energy function. However, the algorithm can be used to sample from any distribution for a set of real valued variables for which the derivatives of the probability density can be computed. The idea is to introduce conjugate variables p_i and define the Hamiltonian energy (the sum of potential and kinetic $K(p)$) energy),

$$H(\boldsymbol{w}, \boldsymbol{p}) = U(\boldsymbol{w}) + K(\boldsymbol{p}) = U(\boldsymbol{w}) + \sum_i \frac{p_i^2}{2m_i} \tag{3.23}$$

The parameters m_i correspond to the masses of particles in a fictitious physical system, the weights w_i to their coordinates, and the conjugate variables p_i to their momenta. The masses are free parameters that can, in principle at least, be chosen arbitrarily. In practice they control the step-widths of the numerical integration scheme and therefore play a crucial role in preventing numerical instabilities. A heuristic algorithm for setting their values is discussed in Neal (1995). The potential energy $U(\boldsymbol{w})$ is derived from the log of the prior and the log of the likelihood due to the training cases as follows,

$$U(\boldsymbol{w}) = F(\gamma) - \log p(\boldsymbol{w}|\gamma) - \sum_{i=1}^{N} \log P(y_i|x_i, \boldsymbol{w}, \gamma) \tag{3.24}$$

where $F(\gamma)$ is any function of the hyperparameters that is convenient (Neal, 1995). The Boltzmann distribution for this energy function will then produce the posterior probability density for $\boldsymbol{w}$ given γ.

Solving the Hamiltonian equations of motion,

$$\frac{\partial w_i}{\partial t} = \frac{\partial H}{\partial p_i}, \quad \frac{\partial p_i}{\partial t} = -\frac{\partial H}{\partial w_i} \tag{3.25}$$

leads to a trajectory on a hypershell of constant energy. The algorithm for sampling thus reduces to a numerical integration of the Hamiltonian equations of motion, and a regular resampling of the momentum variables from a Boltzmann distribution. However, inaccuracies in the numerical integration can lead to systematic errors in the sampling process. This is avoided when simulating the movement of the energy hypershell (that is between two consecutive resamplings of the momentum), the final configuration w_2 is compared with the initial configuration w_1 and accepted only with the probability,

$$p_{accept}(w_2|w_1) = \min\{1, \exp[H(w_1) - H(w_2)]\} \tag{3.26}$$

It is seen that in this way unstable algorithms with $H(w_2) \gg H(w_1)$ lead to high rejection rates.

3.7.7 Gibbs sampling for the Hyperparameters

The network prediction is given by marginalisation.

$$\begin{aligned} P(y|\boldsymbol{x}, D) &= \iint P(y, \boldsymbol{w}, \boldsymbol{\alpha}|\boldsymbol{x}, D)\, d\boldsymbol{w}\, d\boldsymbol{\alpha} \\ &= \iint P(y|\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{x}, D) P(\boldsymbol{w}, \boldsymbol{\alpha}|\boldsymbol{x}, D)\, d\boldsymbol{w}\, d\boldsymbol{\alpha} \\ &= \iint P(y|\boldsymbol{w}, \boldsymbol{x}) P(\boldsymbol{w}, \boldsymbol{\alpha}|D)\, d\boldsymbol{w}\, d\boldsymbol{\alpha} \end{aligned} \tag{3.27}$$

The last step follows from the fact that the prediction model is solely determined by the network weights $\boldsymbol{w}$ and that learning is supervised Husmeier (1999). Sampling from the posterior distribution $P(\boldsymbol{w}, \boldsymbol{\alpha}|D)$ now follows an iterative process known as Gibbs sampling. In a first step, the hyperparameters $\boldsymbol{\alpha}$ are kept constant and $\boldsymbol{w}$ is sampled from $P(\boldsymbol{w}|\boldsymbol{\alpha}, D)$ using the hybrid Monte Carlo algorithm. In a second step, the parameters $\boldsymbol{w}$ are kept constant and the hyperparameters are sampled from a conjugate prior. This can be considered to be a generalisation of the Expectation-Maximization (EM) algorithm (Ghahramani and Jordan, 1997). A natural choice for the conjugate prior is the gamma distribution, the use of a gamma distribution for Bayesian learning is further discussed in Chapters 4 and 5. Small values of the so-called *shape parameter* correspond to vague prior distributions, whereas for larger values the distribution becomes more localised and thus more informative.

The following pseudo-code outlines the implementation of a Bayesian neural network using MCMC sampling,

```
Algorithm 2: Bayesian Neural Network - MCMC Sampling

Input       Dataset $\mathcal{D} = (\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$

Create      Construct Bayesian neural network with specified number of
            inputs, hidden nodes, and outputs

Initialise  Draw $\boldsymbol{w}$ from a Gaussian distribution
            Initialise ARD values, e.g. $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_d) = \mathbf{0.01}$
            Initialise noise variance, e.g. $\beta = 100$;

Algorithm   Generate prior distributions for hyperparameters
            For $k = 1$ to $M$
               Use hybrid Monte Carlo to optimise $\boldsymbol{w}_{MP}$
               Optimise hyperparameters using Gibbs sampling
               Get Predictions
            end.

Output      $\boldsymbol{\alpha}_{MP}, \beta_{MP}, y$.
```

3.7.7.1 Limitations

A number of issues arise when using MCMC sampling. An excellent description of these issues is provided by Kass et al. (1999) and Gilks et al. (1996). A popular feature of MCMC methods is that any proposal distribution will ultimately deliver samples from the underlying target distribution. However, the rate of convergence to the underlying stationary distribution will depend crucially on the relationship between the proposed distribution and the target distribution. As the models become more complex, it becomes increasingly likely that the discrepancy between the distributions will result in the Markov chain not moving rapidly throughout the support of the target distribution, a phenomena referred to as *mixing*.

In high dimensional problems with little symmetry, it is often necessary to perform exploratory analysis to determine approximately the shape and orientation of the underlying distribution. As Gilks et al. (1996) observe such analysis often serves to help choose the proposal distribution. However, even though such analysis is useful, often the proposal distribution is chosen for computational efficiency so that it can be easily sampled from and evaluated.

One of the practical problems associated with using MCMC methods is that to reduce the possibility of inferential bias caused by the effect of starting values, iterations within an initial transient phase or *burn in* period are discarded. One of the most difficult implementation problems is that of determining the length of the required burn in, since rates of convergence of different algorithms on different target distributions may vary considerably. Ideally, we would like to compute analytically, or to estimate, convergence and then take sufficient iterations for any particular desired accuracy. For example, given geometric ergodicity, in which case the t-step transition distribution $P'(x, \cdot)$ is such that,

$$P'(x, \cdot) - R(\cdot) \leq M(x)\rho \tag{3.28}$$

for some $M, \rho \in \mathbb{R}$, we might stop the chain once $P'(x, \cdot) \leq \varepsilon$, for some $\varepsilon \geq 0$, in which case the length of the burn in period is given by,

$$t^* = \frac{log(\varepsilon/M(x))}{log(\rho)} \tag{3.29}$$

However, in general, it is extremely difficult to prove even the existence of a geometric rate of convergence to stationarity (Brooks, 1998), and there are many commonly used algorithms which fail to converge geometrically at all. Roberts and Polson (1994) have provided qualitative geometric convergence results for quite general target densities. Although these results have theoretical importance in ensuring the existence of central limit theorems, they do not offer explicit bounds on the rate of convergence, which could be used to determine MCMC sample lengths. It is possible however to construct less general bounds, applicable to only a small class of samplers.

Since in general, practical bounds on the convergence rate are rarely available, a large amount of work has been performed on the statistical analysis of the sample output to determine, either *a posteriori* or during run time, whether the chain converges during a particular sample run. In practice, such diagnostics tend to be of limited use since, for example, Asmussen et al. (1992) showed that no one diagnostic method will work for all problems.

One of the most contentious issues associated with the implementation of MCMC algorithms is in choosing whether to run one long chain or several shorter chains in parallel. The argument for taking a single long run is that the chain will be 'closer' to the target distribution at the end than it would be at the end of any number of shorter runs, and that several shorter runs may be wasteful, in that the initial burn in periods for each smaller chain will have to be discarded. In contrast, proponents of the 'many replications' approach argue that, although a single run will eventually cover the entire sample space, by taking a number of parallel replications we can guard against a single chain leaving a 'significant proportion' of the sample space unexplored. By taking several chains, each started in different states, it is also possible to monitor the sample paths to determine how well the chains have *mixed*, i.e. to what extent the outputs from the different chains are indistinguishable. In effect, multiple replications protect against bias by attempting to ensure that the sampler output covers the entire sample space, whereas one long run provides less variable estimates.

Another important issue is the determination of suitable starting points. Although, any inference gained via MCMC sampling will be independent of the starting values, since observations are only used after the chain has achieved equilibrium, and hence lost all dependence on those values. However, the choice of starting values may affect the performance of the chain, and in particular the speed and ease of detection of convergence. Many users adopt *ad hoc* methods for selecting starting values. Approaches include simplifying the model, by setting hyperparameters to fixed values, or ignoring missing data, for example. Alternatively, maximum likelihood estimates may provide starting points for the chain or, in the case where *informative* priors exist, the prior might also be used to select suitable starting values (Neal, 1995).

3.7.8 Variational Learning

Variational learning (also known as *ensemble* learning) has been proposed by a number of researchers as being a method of approximating high dimensional integrals. Variational methods are an important technique for the approximation of complicated probability distributions deriving originally from statistical physics. A well known method for approximating a complex distribution in a physical system is *mean field theory*. Mean field theory is a special case of a *variational free energy* approach described by Feynman (1972). Examples of these variational approximations can be found in (Saul et al., 1996; Ghahramani, 1994; Jaakkola, 1997; Ghahramani and Jordan, 1997). The variational method is often the technique of choice for studying complex physical systems such as multi-electron atoms and molecules.

In statistical physics Ising models are important models of magnetism and other systems that show phase transitions. An Ising model is an array of spins (e.g. atoms that take on ± 1 states) that are magnetically coupled to each other. If one spin is in the $+1$ state then it is energetically favourable for its immediate neighbours to be in the same state, in the case of a ferromagnetic model, and in the opposite state for an antiferromagnet. Hence, probability distributions of the form,

$$P(\boldsymbol{x}|\beta, J) = \frac{1}{Z(\beta, J)} \exp[-\beta E(\boldsymbol{x}; J)], \tag{3.30}$$

are commonplace where for example the state vector is $\boldsymbol{x} \in [-1, +1]^N$, and $E(\boldsymbol{x}, J)$ is some energy function such as,

$$E(\boldsymbol{x}, J) = -\frac{1}{2} \sum_{m,n} J_{mn} x_m x_n - \sum_n h_n x_n \tag{3.31}$$

where m and n are neighbours, J_{mn} is a coupling constant between spins m and n, and $\beta = 1/k_B T$, where k_B is the Boltzmann constant at some temperature T. The normalisation constant (partition function) is,

$$Z(\beta, J) \equiv \sum_x \exp[-\beta E(\boldsymbol{x}; J)] \tag{3.32}$$

An evaluation of the normalisation constant is desirable because then all of the thermodynamic properties of the system can be derived. Variational free energy minimisation is a method for *approximating* the complex distribution $P(\boldsymbol{x})$ by a simpler ensemble $Q(\boldsymbol{x}; \theta)$ that is parameterised by adjustable parameters θ. A by-product of this approximation is that a lower bound on $Z(\beta, J)$ can be derived. The cost function chosen to measure the quality of the approximation is the variational free energy originally introduced by Feynman (1972),

$$\beta \hat{F}(\theta) = \sum_x Q(\boldsymbol{x}; \theta) \log \frac{Q(\boldsymbol{x}; \theta)}{\exp[-\beta E(\boldsymbol{x}; J)]} \tag{3.33}$$

Using the definition of $p(\boldsymbol{x}|\beta, J)$ (see Equation 3.30) the variational free energy can be written as,

$$\begin{aligned} \beta \hat{F}(\theta) &= \sum_x Q(\boldsymbol{x}; \theta) \log \frac{Q(\boldsymbol{x}; \theta)}{p(\boldsymbol{x}|\beta, J)} - \log Z(\beta, J) \\ &= D_{KL}(Q\|P) + \beta F \end{aligned} \tag{3.34}$$

where F is the true free energy defined by,

$$\beta F \equiv -\log Z(\beta, J) \tag{3.35}$$

and $D_{KL}(Q\|P)$ is the Kullback-Leibler divergence measuring the (asymmetric) difference between the approximating distribution $Q(\boldsymbol{x}; \theta)$ and the true distribution $P(\boldsymbol{x}|\beta, J)$. Hence, by Jensens' inequality the variational free energy $\hat{F}$ is bounded from below by F and only attains this value when $Q(\boldsymbol{x}; \theta) = P(\boldsymbol{x}|\beta, J)$. The strategy that is adopted is to vary θ in such a way that $\beta \hat{F}(\theta)$ is minimised. The approximating distribution then gives a simplified approximation to the true distribution that may be useful and the value of $\beta \hat{F}(\theta)$ will be an upper bound for βF. The Kullback-Leibler divergence satisfies $D_{KL}(Q\|P) \geq 0$ (Jensens' inequality) with equality only if $Q = P$. However, in general $D_{KL}(Q\|P) \neq D_{KL}(P\|Q)$.

Given these ideas it is possible to work in terms of an approximating *ensemble* $Q(\boldsymbol{w}; \theta)$, which is a probability distribution over the parameters, and optimise the ensemble by varying its own parameters θ so that it approximates the posterior distribution of the parameters $P(\boldsymbol{w}|\mathcal{D}, H)$ well. This approach was first proposed for use in a single hidden layer neural network by Hinton and van Camp (1993) using the restriction that $Q(\boldsymbol{w}; \theta)$ is Gaussian. It has since been applied to various other models with hidden states and no restrictions on the approximating distribution other than the assumption that they factor in some way (Waterhouse et al., 1995; MacKay, 1997; Bishop, 1999; Attias, 1999; Ghahramani and Beal, 2000).

As Penny and Roberts (2000) observe variational learning may be considered as two distinct steps,

1. Step 1: Approximate E-step With model parameters fixed at $\boldsymbol{w}_{t-1}$, update the variational parameters θ to maximise $\hat{F}(\theta)$.

2. Step 2: M-step With variational parameters fixed at θ_t update the models parameters $\boldsymbol{w}$ to maximise $F(\theta)$.

These steps are iterated as necessary and are analogous to the Expectation (E) and Maximisation (M) steps of the EM algorithm. In the approximate E-step the parameters of the approximating density (the variational parameters) are updated and in the M-step the parameters of the probabilistic model are updated. The M-steps are identical, however in variational learning the exact E-step is replaced with an approximate E-step where the Kullback-Leibler divergence is *minimised*.

A cost function which may be used to measure the quality of the approximation is the variational free energy,

$$\hat{F}(\theta) = \int Q(\boldsymbol{w};\theta) \log \frac{Q(\boldsymbol{w};\theta)}{P(\mathcal{D}|\boldsymbol{w},\mathcal{H})P(\boldsymbol{w}|\mathcal{H})}\, d^k\boldsymbol{w} \tag{3.36}$$

The denominator of Equation 3.36 is, to within a multiplicative constant, equal to the posterior probability distribution $P(\boldsymbol{w}|\mathcal{D},\mathcal{H})$. Hence, the variational free energy $\hat{F}(\theta)$ can be viewed as the sum of $-\log P(\mathcal{D}|\mathcal{H})$ and the relative entropy between $Q(\boldsymbol{w};\theta)$ and $P(\boldsymbol{w}|\mathcal{D},\mathcal{H})$. As (MacKay, 1995) observes for certain models and certain approximating distributions this free energy and its derivatives with respect to the ensemble's parameters can be evaluated.

The approximation of posterior probability distributions using variational free energy minimization provides a useful approach to approximating Bayesian inference in a number of fields. The application of this approach may be characterised according to what form of approximating distribution is used and what probabilistic model is involved (Penny and Roberts, 2000). The choice of approximating distribution is of central importance to the usefulness of the method; a distribution must be chosen which can make a good approximation to the true posterior but which at the same time is sufficiently simple to enable the variational free energy to be computed. This can be compared to the MCMC approach where the choice of starting values can affect the rate of convergence of the Markov chain.

The use of separable probability distributions, for approximating the posterior distribution in machine learning, was first suggested by MacKay (1994). The following description is based on a simple extension of the ideas proposed in this paper to the case of ARD with unknown noise variance. The statistical model $\mathcal{H}$ for the case of a Bayesian neural network trained with a dataset $\mathcal{D}$, with ARD can be described by,

$$P(\mathcal{D},\boldsymbol{w},\boldsymbol{\alpha},\beta|\mathcal{H}) = P(\mathcal{D}|\boldsymbol{w},\beta,\mathcal{H})P(\boldsymbol{w}|\boldsymbol{\alpha},\mathcal{H})P(\boldsymbol{\alpha},\beta|\mathcal{H}) \tag{3.37}$$

The likelihood function $P(\mathcal{D}|\boldsymbol{w},\beta,\mathcal{H})$ describes the assumed noise process, parameterized by a noise variance term $1/\beta$. The simplest representation of this is a Gaussian given by,

$$P(\mathcal{D}|\boldsymbol{w},\beta,\mathcal{H}) = \frac{1}{Z_{\mathcal{D}}(\beta)}\exp\{-\beta E_{\mathcal{D}}(\boldsymbol{w})\} \tag{3.38}$$

The prior probability of the parameters, $P(\boldsymbol{w}|\boldsymbol{\alpha},\mathcal{H})$, embodies assumptions about the spatial correlations and smoothness that the true function is expected to have, parameterised by a vector of hyperparameters $\boldsymbol{\alpha}$ one for each input variable. This in the simplest case is represented by a spherical Gaussian,

$$P(\boldsymbol{w}|\boldsymbol{\alpha},\mathcal{H}) = \prod_i \frac{1}{Z_{\boldsymbol{w}}(\alpha_i)}\exp(-\alpha_i\frac{1}{2}\boldsymbol{w}^T\boldsymbol{\Lambda}_i\boldsymbol{w}) \tag{3.39}$$

A gamma distribution prior is assumed for the hyperparameters $\boldsymbol{\alpha}$ and β. As described in Section 3.6.1 the gamma distribution is also the distribution of choice since it is the conjugate prior to the Gaussian distribution.

$$\Gamma(\boldsymbol{\alpha};\boldsymbol{b},\boldsymbol{c}) = \prod_i \frac{1}{\Gamma(c_i)}\frac{\alpha_i^{c_i-1}}{b_i^{c_i}}\exp\left(-\frac{\alpha_i}{b_i}\right) \tag{3.40}$$

A single assumption is made in the variational formulation which is that the approximate distribution, $Q(\boldsymbol{w},\boldsymbol{\alpha},\beta)$, to the posterior, $P(\boldsymbol{w},\boldsymbol{\alpha},\beta|\mathcal{D},\mathcal{H})$, is separable into the form $Q(\boldsymbol{w},\boldsymbol{\alpha},\beta) = Q_{\boldsymbol{w}}(\boldsymbol{w})\prod_i Q_{\boldsymbol{\alpha}}(\alpha_i)Q_\beta(\beta)$. Importantly no functional form for these distributions is assumed. The advantage of this separability is that the notion of a conjugate prior, described in Section 3.6.1, can be introduced for the parameters $\boldsymbol{w}$ and hyperparameters $\boldsymbol{\alpha}$ and β, allowing the evidence or marginal likelihood to be computed. The variational free energy can then be written as, (where $\mathcal{H}$ has been dropped for clarity),

$$F(Q) = -\iiint Q_{\boldsymbol{w}}(\boldsymbol{w})Q_{\boldsymbol{\alpha}}(\boldsymbol{\alpha})Q_\beta(\beta)\log\frac{P(\mathcal{D}|\boldsymbol{w},\beta)P(\boldsymbol{w}|\boldsymbol{\alpha})P(\boldsymbol{\alpha})P(\beta)}{Q_{\boldsymbol{w}}(\boldsymbol{w})Q_{\boldsymbol{\alpha}}(\boldsymbol{\alpha})Q_\beta(\beta)}\,d\beta\,d\boldsymbol{\alpha}\,d\boldsymbol{w} \tag{3.41}$$

The optimal separable distribution Q can be found by separate optimisation of $Q_{\boldsymbol{w}}(\boldsymbol{w})$, $Q_{\boldsymbol{\alpha}}(\boldsymbol{\alpha})$, and $Q_\beta(\beta)$.

Optimisation of $Q_{\boldsymbol{w}}(\boldsymbol{w})$

As a functional of $Q_{\boldsymbol{w}}(\boldsymbol{w})$, F can be written as (ignoring constant terms),

$$-\int Q_{\boldsymbol{w}}(\boldsymbol{w}) \int Q_{\boldsymbol{\alpha}}(\boldsymbol{\alpha}) \int Q_\beta(\beta) \Big(\log P(\mathcal{D}|\boldsymbol{w}, \beta) + \log P(\boldsymbol{w}|\boldsymbol{\alpha}) - \log Q_{\boldsymbol{w}}(\boldsymbol{w}) \Big)\, d\beta\, d\boldsymbol{\alpha}\, d\boldsymbol{w}.$$

Ignoring constant terms this can be rewritten as,

$$\begin{aligned} &\int Q_{\boldsymbol{w}}(\boldsymbol{w}) \int Q_{\boldsymbol{\alpha}}(\boldsymbol{\alpha}) \int Q_\beta(\beta) \Big(\beta E_D(\boldsymbol{w}) + \tfrac{1}{2} \sum_i \alpha_i \boldsymbol{w}^T \boldsymbol{\Lambda}_i \boldsymbol{w} + \log Q_{\boldsymbol{w}}(\boldsymbol{w}) \Big)\, d\beta\, d\boldsymbol{\alpha}\, d\boldsymbol{w} \\ &= \int Q_{\boldsymbol{w}}(\boldsymbol{w}) \int Q_{\boldsymbol{\alpha}}(\boldsymbol{\alpha}) \Big(\bar{\beta} E_D(\boldsymbol{w}) + \tfrac{1}{2} \sum_i \alpha_i \boldsymbol{w}^T \boldsymbol{\Lambda}_i \boldsymbol{w} + \log Q_{\boldsymbol{w}}(\boldsymbol{w}) \Big)\, d\boldsymbol{\alpha}\, d\boldsymbol{w} \\ &= \int Q_{\boldsymbol{w}}(\boldsymbol{w}) \Big(\bar{\beta} E_D(\boldsymbol{w}) + \tfrac{1}{2} \sum_i \bar{\alpha}_i \boldsymbol{w}^T \boldsymbol{\Lambda}_i \boldsymbol{w} + \log Q_{\boldsymbol{w}}(\boldsymbol{w}) \Big)\, d\boldsymbol{w}. \end{aligned} \tag{3.42}$$

As MacKay (1995) observes the $\boldsymbol{w}$-dependent terms are the log of the posterior distribution, and using the theorem that a divergence $\int Q \log(Q/P)$ is minimised by setting $Q = P$, the optimising distribution $Q^{opt}_{\boldsymbol{w}}(\boldsymbol{w})$ is Gaussian and is identical to the posterior distribution for a particular value of α,

$$Q^{opt}_{\boldsymbol{w}}(\boldsymbol{w}) = P(\boldsymbol{w}|D, \bar{\boldsymbol{\alpha}}, \bar{\beta}) = \mathcal{N}(\boldsymbol{w}_{MP|\bar{\boldsymbol{\alpha}},\bar{\beta}}, \boldsymbol{\Sigma}_{\bar{\boldsymbol{\alpha}},\bar{\beta}}). \tag{3.43}$$

Optimisation of $Q_{\boldsymbol{\alpha}}(\boldsymbol{\alpha})$

As a functional of $Q_{\boldsymbol{\alpha}}(\boldsymbol{\alpha})$, F can be written as (ignoring constant terms),

$$\begin{aligned} &-\int Q_{\boldsymbol{\alpha}}(\boldsymbol{\alpha}) \int Q_{\boldsymbol{w}}(\boldsymbol{w}) \int Q_\beta(\beta) \Big(\log P(\boldsymbol{w}|\boldsymbol{\alpha}) + \log P(\boldsymbol{\alpha}) - \log Q_{\boldsymbol{\alpha}}(\boldsymbol{\alpha}) \Big)\, d\beta\, d\boldsymbol{w}\, d\boldsymbol{\alpha} \\ &= \int Q_{\boldsymbol{\alpha}}(\boldsymbol{\alpha}) \int Q_{\boldsymbol{w}}(\boldsymbol{w}) \Big(-\log P(\boldsymbol{w}|\boldsymbol{\alpha}) - \log P(\boldsymbol{\alpha}) + \log Q_{\boldsymbol{\alpha}}(\boldsymbol{\alpha}) \Big)\, d\boldsymbol{w}\, d\boldsymbol{\alpha} \end{aligned}$$

Ignoring constant terms this can be rewritten as,

$$\begin{aligned} &\int Q_{\boldsymbol{\alpha}}(\boldsymbol{\alpha}) \sum_i \bigg(\frac{\alpha_i}{2} \int Q_{\boldsymbol{w}}(\boldsymbol{w}) \boldsymbol{w}^T \boldsymbol{\Lambda}_i \boldsymbol{w}\, d\boldsymbol{w} - \Big(\frac{\mathbf{1}^T \boldsymbol{\Lambda}_i \mathbf{1}}{2} + c_i - 1 \Big) \log \alpha_i + \frac{\alpha_i}{b_i} \bigg) + \log Q_{\boldsymbol{\alpha}}(\boldsymbol{\alpha})\, d\boldsymbol{\alpha} \\ &= \int Q_{\boldsymbol{\alpha}}(\boldsymbol{\alpha}) \sum_i \bigg[\alpha_i \Big(\tfrac{1}{2} \boldsymbol{w}^T_{MP|\bar{\boldsymbol{\alpha}},\bar{\beta}} \boldsymbol{\Lambda}_i \boldsymbol{w}_{MP|\bar{\boldsymbol{\alpha}},\bar{\beta}} + \tfrac{1}{2} \text{trace} \boldsymbol{\Lambda}_i \boldsymbol{\Sigma}_{\bar{\boldsymbol{\alpha}},\bar{\beta}} + \frac{1}{b_i} \Big) \\ &\qquad - \Big(\frac{\mathbf{1}^T \boldsymbol{\Lambda}_i \mathbf{1}}{2} + c_i - 1 \Big) \log \alpha_i \bigg] + \log Q_{\boldsymbol{\alpha}}(\boldsymbol{\alpha})\, d\boldsymbol{\alpha} \end{aligned} \tag{3.44}$$

Again using the divergence theorem gives the following expression for the optimal distribution that minimises F for fixed $Q_{\boldsymbol{w}}(\boldsymbol{w})$,

$$Q^{opt}_{\boldsymbol{\alpha}}(\boldsymbol{\alpha}) = \Gamma(\boldsymbol{\alpha}; \boldsymbol{b}, \boldsymbol{c}), \tag{3.45}$$

where,

$$\begin{aligned}\frac{1}{b_i^{new}} &= \frac{1}{b} + \tfrac{1}{2}\boldsymbol{w}^T_{MP|\bar{\boldsymbol{\alpha}},\bar{\beta}}\boldsymbol{\Lambda}_i\boldsymbol{w}_{MP|\bar{\boldsymbol{\alpha}},\bar{\beta}} + \tfrac{1}{2}\mathrm{trace}\boldsymbol{\Lambda}_i\boldsymbol{\Sigma}_{\bar{\boldsymbol{\alpha}},\bar{\beta}} \\ c_i^{new} &= \frac{\mathbf{1}^T\boldsymbol{\Lambda}_i\mathbf{1}}{2} + c_i\end{aligned} \tag{3.46}$$

Due to numerical instability associated with the hyperparameter convergence rate, a modified hyperparameter re-estimation formulae was used based on the principle of gradient descent as used in the Laplace approximation.

Optimisation of $Q_\beta(\beta)$

As a functional of $Q_\beta(\beta)$, F can be written as (ignoring constant terms),

$$\begin{aligned}&-\int Q_\beta(\beta)\int Q_{\boldsymbol{w}}(\boldsymbol{w})\int Q_{\boldsymbol{\alpha}}(\boldsymbol{\alpha})\Big(\log P(\mathcal{D}|\boldsymbol{w},\beta) + \log P(\beta) - \log Q_\beta(\beta)\Big)\,d\boldsymbol{\alpha}\,d\boldsymbol{w}\,d\beta \\ &= \int Q_\beta(\beta)\int Q_{\boldsymbol{w}}(\boldsymbol{w})\Big(-\log P(\mathcal{D}|\boldsymbol{w},\beta) - \log P(\beta) + \log Q_\beta(\beta)\Big)\,d\boldsymbol{w}\,d\beta\end{aligned} \tag{3.47}$$

Ignoring constant terms this can be rewritten as,

$$\int Q_\beta(\beta)\left(\beta\int Q_{\boldsymbol{w}}(\boldsymbol{w})E_D(\boldsymbol{w})\,d\boldsymbol{w} - \frac{N}{2}\log\beta - (c-1)\log\beta + \frac{\beta}{b} + \log Q_\beta(\beta)\right)d\beta \tag{3.48}$$

Again using the above theorem gives the following expression as the optimal distribution that minimises F for fixed $Q_{\boldsymbol{w}}(\boldsymbol{w})$,

$$Q_\beta^{opt}(\beta) = \Gamma(\beta; b, c) \tag{3.49}$$

where,

$$\begin{aligned}\frac{1}{b^{new}} &= \frac{1}{b} + \frac{1}{2}\int Q_{\boldsymbol{w}}(\boldsymbol{w})E_D(\boldsymbol{w})\,d\boldsymbol{w} \\ c^{new} &= \frac{N}{2} + c\end{aligned} \tag{3.50}$$

Although the integral over $Q_{\boldsymbol{w}}(\boldsymbol{w})E_D(\boldsymbol{w})$ is analytically intractable, two alternative approaches have been proposed in the literature to deal with this (MacKay, 1994; Barber and Bishop, 1998). In the approach of MacKay (1994) the noise variance is set to a constant value, and is not re-estimated, however Barber and Bishop (1998) propose reduction of this integral to a one dimensional integration by transforming the parameter system to a new isotropic basis and then differentiating with respect to the elements of the covariance matrix. In this work, the approch of MacKay (1994) is adopted where the noise variance is set to that value obtained from the Laplace approximation.

The following pseudo-code outlines the implementation of a Bayesian neural network using variational learning,

Algorithm 3: Bayesian Neural Network - Variational Learning

```
Input       Dataset $\mathcal{D} = (\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$

Create      Construct Bayesian neural network with specified number of
            inputs, hidden nodes, and outputs

Initialise  Draw $\boldsymbol{w}$ from a Gaussian distribution
            $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_d) = \mathbf{0.01}$; (or some other constant value).
            Fix noise variance e.g. $\beta = 100$;
            Set alphatol to a constant value e.g. 0.001

Algorithm   Generate prior distributions for hyperparameters
            do{
               Use gradient descent algorithm to optimise $\boldsymbol{w}_{MP}$
               Optimise ARD hyperparameters using Equation 3.46
               Get Predictions
            }while($\frac{1}{F}\sum_i^F |\frac{1}{\alpha_i} - \frac{1}{\alpha_i^{old}}| > \delta$)

Output      $\boldsymbol{\alpha}_{MP}, y$.
```

Besides variational bounds, there are other integration techniques that are inspired by mean-field statistical physics. As (Minka, 2001) observes most of these are extensions of the TAP (Thouless et al., 1977) approach. Notable examples include the cavity method (Opper and Winther, 2000), and Bethe approximations (Yedidia, 2000).

3.7.8.1 Limitations

For a Bayesian neural network with ARD, the posterior distribution is composed of both parameters and hyperparameters, $P(\boldsymbol{w}, \boldsymbol{\alpha}|D)$. The only assumption made in the variational learning approach is based upon the factorisation of the posterior approximating distribution $Q(\boldsymbol{w}, \boldsymbol{\alpha}|D)$. By factoring the approximating distribution into $Q(\boldsymbol{w})Q(\boldsymbol{\alpha})$, the inherent assumption is that the parameters $\boldsymbol{w}$ and hyperparameters $\boldsymbol{\alpha}$ are independent. In the ARD formulation the hyperparameters act to control the magnitudes of the weights out of each input, hence the assumption is clearly flawed. The Relevance Vector Machine (RVM) has recently been proposed by (Tipping, 2000b). The model formulation is identical to that of the Bayesian neural network but with a separate hyperparameter for *every* parameter in the model. The aim then is to set as many of these parameters to zero as possible. When performing variational inference on an RVM (Tipping, 2000a), the invalidity of the factorisation assumption is even more pronounced. This could provide a possible explanation for the marginal improvement seen in the variational RVM compared to the original RVM which was optimised in a fashion similar to the evidence framework.

Hinton (2000) has conjectured that the marginal improvement in performance of the variational RVM is a result of the factorisation assumption limiting the class of allowable models to only selecting the best model from the set of all *bad* models. The proposed argument is illustrated in Figure 3.3. Consider a model hypothesis space, $\mathcal{H}$, consisting of two distinct sets of models representing the set of all good models ($\bullet$) and bad models ($\times$).

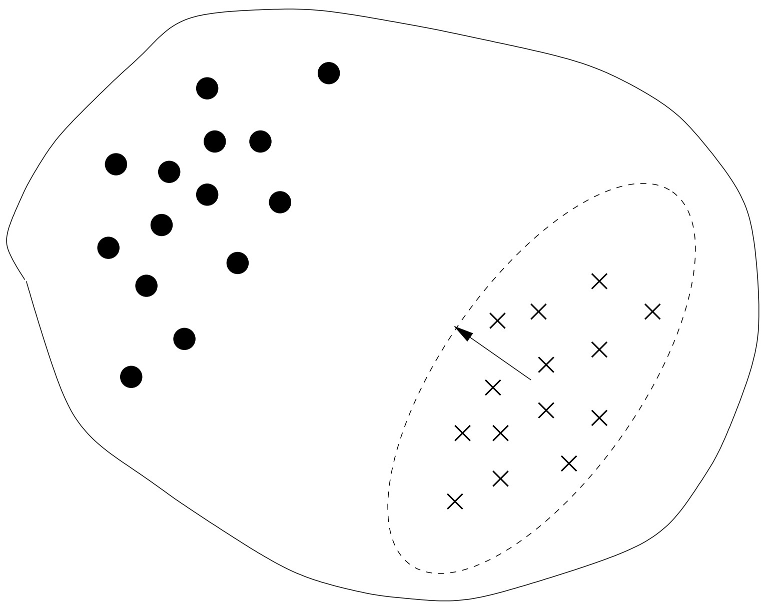


FIGURE 3.3: Variational learning in a model hypothesis space.

Given that the factorisation assumption for a model consisting of parameters and hyperparameters is flawed, the class of allowable models is restricted to the *bad* set of models where the factorisation assumption for these class of models is a good assumption. However, given that the factorisation assumption allows you to define separate prior distributions on both the parameters and the hyperparameters, allows the model that can be obtained to be the *best* of the set of *bad* models, which is represented by the dashed line in Figure 3.3.

3.8 Demonstration of Hyperparameter Re-estimation

To evaluate the performance of the different hyperparameter re-estimation approaches, two artificial modelling problems proposed by (Bishop, 1995) and Friedman (1991) were used. These datasets were considered because they are reflective of many real world datasets in that learning takes place in the presence of highly correlated inputs variables, and in the presence of irrelevant input variables. The objective is to evaluate how well each of the Bayesian hyperparameter determination techniques deal with such data.

The first is a synthetic dataset involving three input variables: x_1 is sampled uniformly from the range $(0, 1)$ and has a low level of added Gaussian noise, x_2 is a copy of x_1 with a higher level of added noise, and x_3 is sampled randomly from a Gaussian distribution. The single target variable is given by $t = \sin(2\pi x_1)$ with additive Gaussian noise. Hence, x_1 is very relevant for determining the target value, x_2 is of some relevance, while x_3 should in principle be irrelevant. The second model is a ten input function, which contains five redundant inputs given by,

$$f(\mathbf{x}) = 10\sin(\pi x_1 x_2) + 20\left(x_3 - \frac{1}{2}\right)^2 + 10x_4 + 5x_5 + \eta \tag{3.51}$$

where η is zero mean, unit variance, additive Gaussian noise, corresponding to approximately 20% noise, and the inputs were generated independently and randomly from a uniform Gaussian distribution in the interval [0,1]. The experiments were performed using 200 examples, 180 for training and 20 for estimating the generalisation performance. To reduce the effects of data partitioning on the generalisation estimate, the modelling algorithms were evaluated for ten different (random) partitions of the data. A single hidden layer Bayesian neural network with a varying numbers of hidden nodes was used to model the relationship between the inputs and the output. The network weights were initially randomised from a Gaussian distribution. A linear activation function was used on the output node.

3.8.1 Bishop's Dataset: Evaluation of the evidence framework

A Bayesian neural network with ARD was trained using the evidence framework described in Section 3.7.1. A hidden node network structure and 500 iterations of the scaled conjugate gradient algorithm was used to optimise the network structure. These values are in keeping with those used by (Bishop, 1995) in their evaluation. All of the hyperparameters were set to an initial value of 0.01 before re-estimation. Table 3.3 gives the mean and the associated standard deviation (std) for the hyperparameter associated with each input variable across each of the ten data partitions, whilst Table 3.2 gives the ranked importance of the input variables across the different data partitions.

	x_1	x_2	x_3
α^{-1}	3.49	3.25×10^{-5}	1.67×10^{-11}
$\sigma_{\alpha^{-1}}$	0.065	3.08×10^{-5}	1.44×10^{-11}

Table 3.2: Bishop's Dataset: Mean and associated standard deviation for the ARD hyperparameter values for a two hidden node Bayesian neural network trained with the evidence framework.

From 3.2 the evidence framework is able to distinguish between the relevant and irrelevant inputs with a high degree of accuracy. In all data partitions input x_1 is chosen as being the most important variable in predicting the output, a fact that is in keeping

	Datasets									
	1	2	3	4	5	6	7	8	9	10
x_1	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}
x_2	2^{rd}	2^{nd}	2^{rd}	2^{nd}	3^{rd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}
x_3	3^{nd}	3^{rd}	3^{nd}	3^{rd}	2^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}

TABLE 3.3: Bishop's Dataset: Ranked Importance of the input variables using a Bayesian neural network trained with the evidence framework.

with our prior knowledge of the problem. The evidence framework gives less influence to the irrelevant inputs, x_2 and x_3, however in some instances input x_3, which is made up entirely of random numbers, is given a higher relevance than input x_2 which is actually a copy of x_1 only with a higher level of noise. This point is discussed further with reference to Friedman's toy problem 3.8.2 where a similar effect is observed.

3.8.2 Friedman's Dataset: Evaluation of the evidence framework

A Bayesian neural network with ARD was trained using the evidence framework described in Section 3.7.1. A plot of the mean MSE for both the training and the test data sets is shown in Figure 3.4 for an increasing number of hidden nodes when using the evidence framework. The optimal network structure was determined to be six hidden nodes since this corresponds to the lowest error on the test set. The scaled conjugate

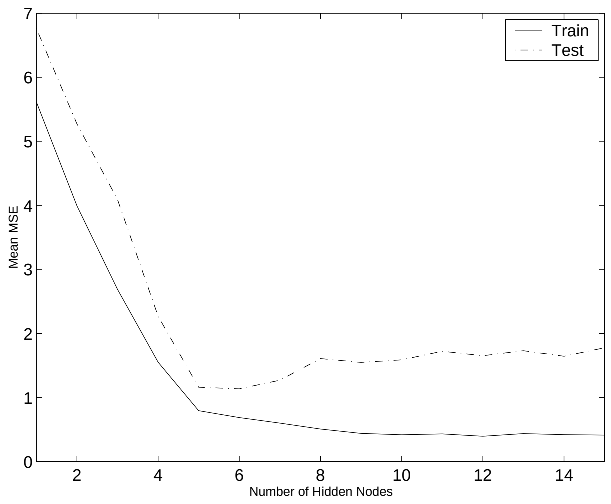


FIGURE 3.4: Variation of mean training and test MSE for a Bayesian Neural Network trained with varying numbers of hidden nodes using the evidence framework.

gradient training algorithm was run for 5000 iterations to optimise the parameters $\boldsymbol{w}$ of the neural network, and the re-estimation formulae derived by (MacKay, 1995) given in Equation 3.17 were used to optimise the values of the hyperparameters. All of the hyperparameters were set to an initial value of 0.01 before re-estimation. Table 3.4 gives the mean and the associated standard deviation (std) for the hyperparameter associated with each input variable across each of the ten data partitions, whilst Table 3.5 gives the ranked importance of the input variables across the different data partitions.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
α^{-1}	1.06	1.08	0.685	0.020	0.004	δ	δ	δ	0.009	δ
$\sigma_{\alpha^{-1}}$	0.20	0.30	0.11	0.009	0.0001	0.001	δ	δ	0.008	δ

TABLE 3.4: Friedman's Dataset: Mean and associated standard deviation for the ARD hyperparameter values for a six hidden node Bayesian neural network trained with the evidence framework. $\delta < 1 \times 10^{-5}$.

	Input variables									
Dataset	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
1	2^{nd}	1^{st}	3^{rd}	4^{th}	5^{th}	-	-	-	-	-
2	1^{st}	3^{rd}	2^{nd}	4^{th}	5^{th}	-	-	-	-	-
3	2^{nd}	1^{st}	3^{rd}	4^{th}	5^{th}	-	-	-	-	-
4	1^{st}	2^{nd}	3^{rd}	4^{th}	6^{th}	-	-	-	5^{th}	-
5	2^{nd}	1^{st}	3^{rd}	4^{th}	5^{th}	-	-	-	-	-
6	1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}	-	-	-	-	-
7	1^{st}	2^{nd}	3^{rd}	6^{th}	-	-	-	-	5^{th}	-
8	1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}	-	-	-	6^{th}	-
9	2^{nd}	1^{st}	3^{rd}	4^{th}	5^{th}	-	-	-	-	-
10	2^{nd}	1^{st}	3^{rd}	4^{th}	5^{th}	-	-	-	-	-

TABLE 3.5: Friedman's Dataset: Ranked importance of input variables when using evidence framework over the ten random data partitions.

Overall, the network does broadly appear to give the highest influence to the five relevant input variables ($x_1 \ldots x_5$), and place less influence on the irrelevant inputs. However, the ranked importance of each input variable across each of the ten data partitions reveals a number of interesting trends. Given that the first term of Friedman's toy problem, given in Equation 3.51, is a symmetric function of x_1 and x_2, there is no reason for the network to favour one input variable over the other. The ranked importance for these two input variables shows an even distribution of relevance. The importance of the remaining input variables is somewhat variable. In 90% of the datasets, the network gives a disproportionate amount of influence to the irrelevant inputs often at the expense of the relevant inputs. This occurrence can be related back to the limitation of the evidence framework described in Section 3.7.1.1.

Given that the solution to the inference problem is highly dependent upon the choice of starting values, the non-convex nature of the error surface leads to the model converging to *bad* regions of the model space. Correlation effects between the different input variables also plays a role in the false recognition of the input variables, and this may be more pronounced in poor models of the data.

In a Hinton diagram the area of the shaded box is proportional to the magnitude of the parameter vector. The colour of the shading indicates whether the parameter is positive (grey) or negative (black). A Hinton diagram for the evidence framework is given in Figure 3.5, showing that the ARD hyperparameters typically reduce the size of the network parameters for the irrelevant inputs and increase them for the relevant input variables.

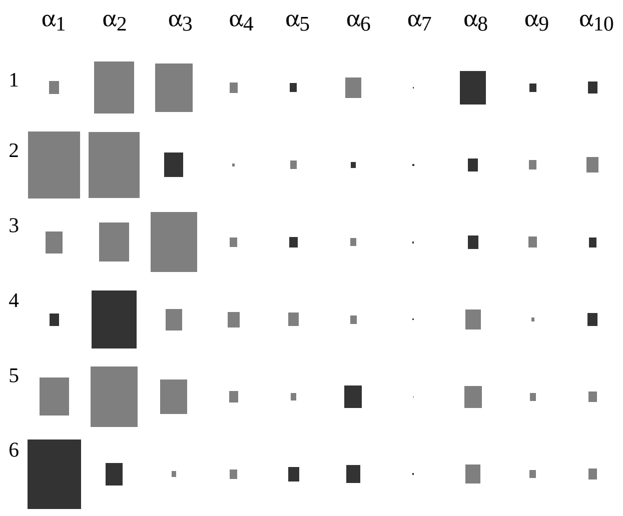


FIGURE 3.5: Hinton diagram showing variation of network weights when using the evidence framework for hyperparameter re-estimation.

3.8.3 Bishop's Dataset: Evaluation of MCMC sampling

A Bayesian neural network was trained in a manner similar to that used in the evidence framework. Hybrid Monte Carlo updates were used to optimise the parameters of the network, and Gibb's sampling for the hyperparameters. Table 3.6 gives the mean and the associated standard deviation (std) for the hyperparameter associated with each input variable across each of the ten data partitions, whilst Table 3.7 gives the ranked importance of the input variables across the different data partitions.

	x_1	x_2	x_3
α^{-1}	3.04	0.023	0.021
$\sigma_{\alpha^{-1}}$	2.56	0.01	0.02

TABLE 3.6: Bishop's Dataset: Mean and associated standard deviation for the ARD hyperparameter values for a two hidden node Bayesian neural network trained with MCMC sampling.

	Datasets									
	1	2	3	4	5	6	7	8	9	10
x_1	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}
x_2	3^{rd}	2^{nd}	3^{rd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}
x_3	2^{nd}	3^{rd}	2^{nd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}

TABLE 3.7: Bishop's Dataset: Ranked Importance of the input variables using a Bayesian neural network trained with MCMC sampling.

From Table 3.7 the MCMC sampling approach is able to distinguish between the relevant and irrelevant inputs with a high degree of accuracy. In all data partitions input x_1 is chosen as being the most important variable in predicting the output. However, as with the evidence framework the network gives some influence to the noisy input x_3. This could be a consequence of the network not having reached equilibrium, a point that is discussed further with reference to Friedman's toy problem where a similar effect is observed.

3.8.4 Friedman's Dataset: Evaluation of MCMC Sampling

The evaluation of the Bayesian ARD neural network with the MCMC algorithm was undertaken in the same manner as for the evidence framework. Hybrid Monte Carlo updates for the network parameters were alternated with Gibbs sampling updates for the hyperparameters. Table 3.8 gives the mean and the associated standard deviation values for the ARD hyperparameters associated with each input variable. In compar-

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
α^{-1}	1.347	0.943	0.711	0.087	0.066	0.001	0.00	0.01	0.01	0.00
$\sigma_{\alpha^{-1}}$	0.57	0.39	0.26	0.06	0.05	0.01	0.00	0.01	0.01	0.01

TABLE 3.8: Friedman's Dataset: Mean and associated standard deviation ARD hyperparameter values for six hidden nodes when using MCMC sampling.

	Input variables									
Dataset	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
1	1st	2nd	3rd	4th	5th	-	-	-	-	-
2	1st	2nd	3rd	4th	5th	6th	-	-	-	-
3	1st	2nd	3rd	4th	5th	-	-	6th	-	-
4	2nd	1st	3rd	4th	5th	-	-	6th	-	-
5	1st	2nd	3rd	4th	5th	-	-	-		-
6	1st	2nd	3rd	4th	6th	-	-	-		5th
7	1st	3rd	2nd	5th	4th	-	-	-	-	-
8	3rd	1st	2nd	4th	5th	-	-	-	-	-
9	1st	2nd	3rd	4th	5th	-	-	-	6th	-
10	1st	2nd	3rd	4th	5th	-	-	-	-	-

TABLE 3.9: Friedman's Dataset: Ranked importance of input variables when using MCMC resampling.

ison to the values associated with the evidence framework the hyperparameters for the irrelevant inputs have been driven to much smaller values, and much higher values are recovered for the relevant inputs, showing that the MCMC approach has been able to distinguish between relevant and irrelevant inputs much more readily. Table 3.9 gives the ranked importance of the input variables when using MCMC sampling. The network gives noticeably less influence to the irrelevant inputs, and can distinguish the relevance of the important inputs more readily than the evidence framework approach. The Hinton diagram, given in Figure 3.6, shows that the weights for the irrelevant inputs are negligible.

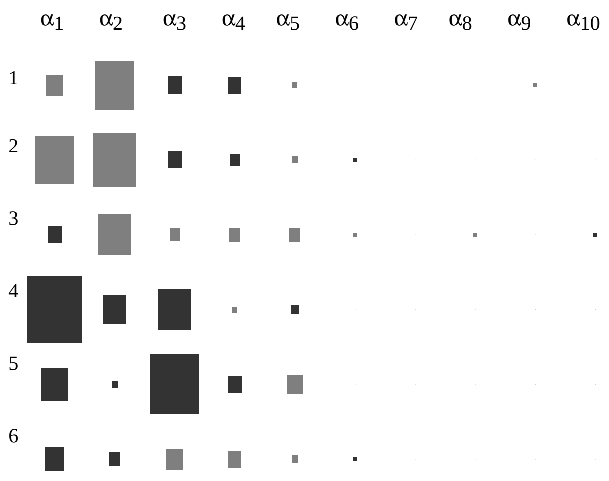


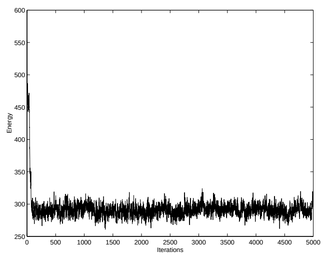
FIGURE 3.6: Hinton diagram showing variation of network weights when using the MCMC sampling for hyperparameter re-estimation.

As described in Section 3.7.7.1, one of the major limitations of the MCMC sampling approach is that convergence cannot be guaranteed. This is illustrated by the convergence plots, shown in Figure 3.7, in which distinct phase transitions are evident. Given that the original MCMC problem is formulated in a statistical physics framework, the phase transitions are indicative of movement from a high energy state to a low energy greater stability state. From an inference perspective, this corresponds to the network with a set of parameters and hyperparameters that are more consistent with the data. Using a fewer training iterations could result in the network not converging to the best possible solution, since the burn-in phase may not been passed.

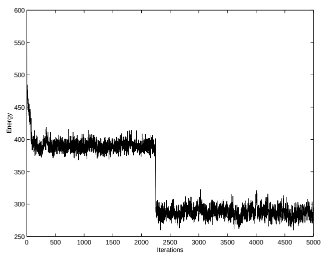
3.8.5 Bishop's Dataset: Evaluation of Variational Learning

A Bayesian neural network was trained in a manner similar to that used in the evidence framework. The scaled conjugate gradient algorithm was used to optimise the network parameters, whilst the variational re-estimation formulae given in Section 3.7.8 where used to optimise the ARD hyperparameters. Table 3.10 gives the mean and the associated standard deviation (std) for the hyperparameter associated with each input variable across each of the ten data partitions, whilst Table 3.11 gives the ranked importance of the input variables across the different data partitions.

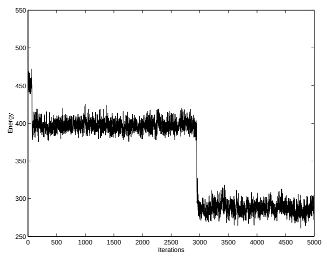
From the ranked importance of the input variables, given in Table 3.11, the variational framework is able to distinguish between the relevant and irrelevant inputs variables correctly. In contrast to both the evidence framework and MCMC sampling the ranked



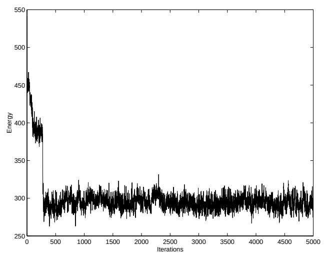
(a) Friedman's Dataset 1



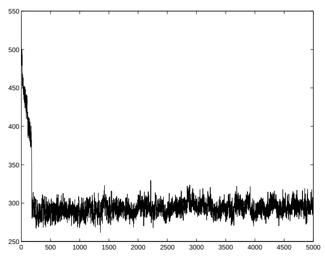
(b) Friedman's Dataset 2



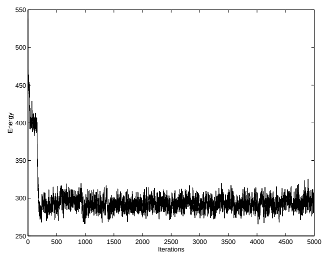
(c) Friedman's Dataset 3



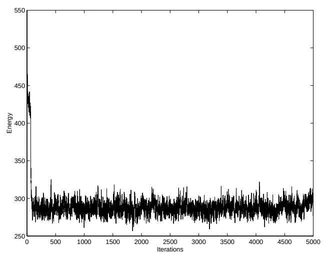
(d) Friedman's Dataset 4



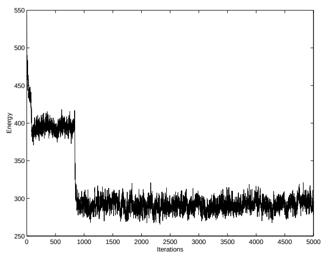
(e) Friedman's Dataset 5



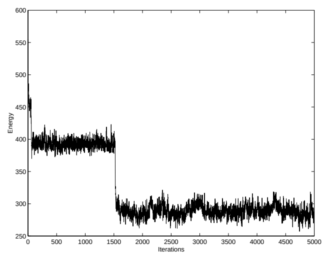
(f) Friedman's Dataset 6



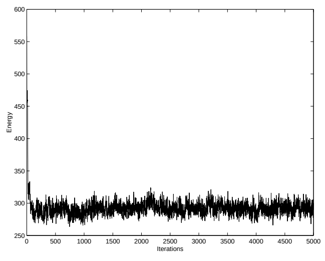
(g) Friedman's Dataset 7



(h) Friedman's Dataset 8



(i) Friedman's Dataset 9



(j) Friedman's Dataset 10

FIGURE 3.7: Friedman's Dataset: Energy Convergence for a Bayesian Neural Network trained using MCMC sampling on the randomly partitioned Friedman datasets.

importance of the influence of the network is in conjunction with our prior knowledge of the problem. A consequence of this is that the factorisation assumption, which is integral to the variational framework, is correct. This observation is discussed further with reference to Friedman's dataset in Section 3.8.6.

3.8.6 Friedman's Dataset: Evaluation of Variational Learning

A Bayesian neural network was optimised using the variational learning parameter and hyperparameter updates described in Section 3.7.8. Table 3.12 gives the mean and the

	x_1	x_2	x_3
α^{-1}	3.45	3.89×10^{-4}	1.70×10^{-4}
$\sigma_{\alpha^{-1}}$	0.04	2.62×10^{-6}	2.47×10^{-7}

TABLE 3.10: Bishop's Dataset: Mean and associated standard deviation for the ARD hyperparameter values for a two hidden node Bayesian neural network trained with variational learning.

	Datasets									
	1	2	3	4	5	6	7	8	9	10
x_1	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}
x_2	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}
x_3	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}

TABLE 3.11: Bishop's Dataset: Ranked Importance of the input variables using a Bayesian neural network trained with the variational learning framework.

associated standard deviations for each of the input variables, and Table 3.13 gives the ranked importance of the input variables. Although the network gives less influence

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
α^{-1}	0.692	0.704	0.334	0.006	0.002	δ	δ	δ	δ	δ
$\sigma_{\alpha^{-1}}$	0.02	0.01	0.01	0.003	0.001	0.003	0.001	0.003	0.004	0.001

TABLE 3.12: Friedman's Dataset: Mean and associated standard deviation ARD hyperparameter values for six hidden nodes when using variational learning, where $\delta < 1 \times 10^{-4}$.

	Input variables									
Dataset	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
1	1^{st}	2^{nd}	3^{rd}	6^{th}	5^{th}	-	-	-	-	-
2	1^{st}	3^{rd}	2^{nd}	7^{th}	6^{th}	-	-	-	-	-
3	2^{nd}	1^{st}	3^{rd}	4^{th}	5^{th}	-	-	-	-	-
4	2^{nd}	1^{st}	3^{rd}	4^{th}	5^{th}	-	-	-	-	-
5	1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}	-	-	-	-	-
6	1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}	-	-	-	-	-
7	2^{nd}	1^{st}	3^{rd}	4^{th}	5^{th}	-	-	-	-	-
8	1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}	-	-	-	-	-
9	2^{nd}	1^{st}	3^{rd}	4^{th}	5^{th}	-	-	-	-	-
10	2^{nd}	1^{st}	3^{rd}	4^{th}	5^{th}	-	-	-	-	-

TABLE 3.13: Ranked importance of input variables when using variational learning.

to the irrelevant inputs compared to the evidence framework, it does not achieve the consistency of the MCMC approach. Variational learning has difficulties ranking the importance of the input variables it selects, and often assigns higher importance to inputs which are irrelevant rather than inputs which contribute to the target. This could be a direct consequence of the limitations described in Section 3.7.8.1. The Hinton diagram shown in Figure 3.8 shows the effect of the hyperparameters on the irrelevant inputs.

The weights for these inputs are large and have not been reduced to reflect their lack of influence on the target.

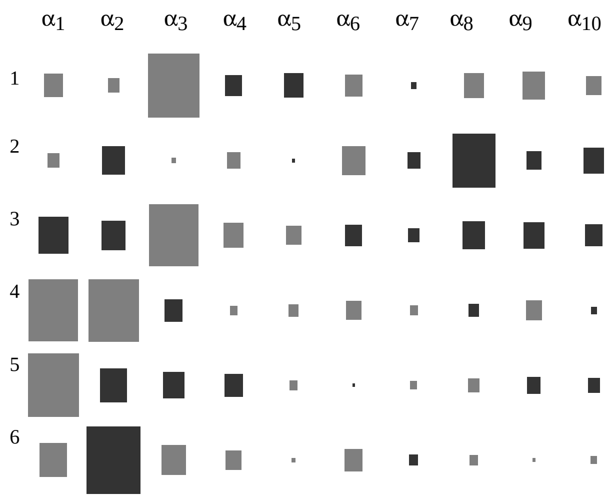


FIGURE 3.8: Hinton diagram showing variation of network weights when using Variational Bayesian Learning.

3.9 Computational Complexity of Bayesian Inference

In Section 3.8, three Bayesian hyperparameter determination algorithms were evaluated on an artificial problem. The main conclusion from these simulations is that the Bayesian learning approach can be applied to moderate size problems (with tens of inputs, and hundreds of training patterns) with little computational expense. A discussion of Bayesian learning would be incomplete without some consideration being given to computational complexity. The computational cost of Bayesian learning is of paramount interest for the technique to be widely applicable.

The principle disadvantage of the evidence framework is in the computational complexity of the training phase. A key component to this framework is the evaluation and storage of the Hessian matrix. The Hessian is required for the computation of error bars on the network predictions, and also as part of the hyperparameter determination approach. To repeatedly compute and invert the Hessian matrix requires $\mathcal{O}(N^2)$ and $\mathcal{O}(N^3)$ complexity respectively. For large datasets this makes training considerably slow. In such cases algorithms such as scaled conjugate gradient (which was used to minimise the Bayesian costfunction) do not evaluate the Hessian matrix explicitly, but instead compute it using a finite difference approximation to the dot product of the Hessian and the search direction.

The MCMC approach consists of choosing N samples in an M-dimensional space resulting in an error term that decreases as $N^{-1/2}$. However, as (MacKay, 1999a) observes the

MCMC approach can be considered to be the most computationally demanding hyper-parameter determination method. As part of an MCMC implementation, it is important to determine how long the simulations should be run for, and to discard a number of initial 'burn-in' iterations (Gilks et al., 1996). Saving all simulations from an MCMC run can consume a large amount of storage, especially when consecutive iterations are highly correlated necessitating a long simulation. Raftery and Lewis (1996) have proposed an alternative method whereby they only save every k^{th} iteration ($k > 1$), a process they refer to as *thinning* the chain. The advantage of this approach is that it reduces the amount of data often saved from an MCMC run. However, a limitation of this approach is that it requires the value of k to be chosen in advance, as such for chains that do not 'mix' well (see Section 3.7.7.1) this approach may not alleviate the computational expense.

The computational complexity of variational learning can be considered to be a hybrid of both the evidence and the MCMC framework. An analogy can be drawn between the number of samples required to minimise the KL-divergence, and those needed to converge to an acceptable solution in the MCMC framework. Minka (2001) and Lawrence (2000) argue that the evaluation of Jensen's inequality in variational Bayesian learning is also computationally demanding.

For approximating an integral, we thus find ourselves in the following position. There exist methods that work well for simple functions in low dimensions (evidence and variational learning frameworks), and complex functions in high dimensions (MCMC), methods that are simple and fast, although sometimes inaccurate exist(evidence), methods even exist that apply in special cases (TAP, Bethe approximations). Recent work by (Minka, 2001) has tried to address this problem by developing an Expectation-Propagation (EP) algorithm that unifies and generalises two previous techniques. The EP algorithm exploits an extension of the Kalman filter (Harvey, 1989), and loopy belief propagation (Murphy et al., 1999) an extension of belief propagation in Bayesian networks (Heckerman et al., 1995). In this work Minka (2001) shows how both of these algorithms can be viewed as approximating the true posterior distribution with a simpler distribution which is close in the KL-divergence sense. This framework has been demonstrated in a variety of statistical models using synthetic and real-world data, and is shown to have good generalisation performance for equivalent or less computation.

3.10 Defining Priors over Functions

In Chapter 3, the uncertainty in the inference problem was described through a probability distribution over the weights. As a result of the work of (Neal, 1995) on prior distributions for neural networks with an infinite number of hidden nodes, it is also possible to deal directly with uncertainty with respect to the function values at the points of interest. Williams (1998) refers to this as the stochastic process or *function-space* view of the problem. Gaussian processes are a subset of stochastic processes that can be specified by only giving the mean vector and covariance matrix for any finite subset of points. This topic is discussed further in Chapter 4.

3.11 Summary

Bayesian learning can be considered to consist of two fundamental characteristics. Bayesian learning starts with a prior probability distribution for model parameters and hyperparameters that represents prior beliefs about the problem. Typically these beliefs are derived from background or expert knowledge. Secondly, Bayesian predictions are not based on a single estimate for the model parameters, but rather are found by integrating the model's predictions with respect to the posterior parameter distribution that is obtained when the prior distribution is updated to take account of the data.

As Neal (1995) observes for neural network models, both of the aspects described above present major difficulties, in particular, integration over the posterior distribution is typically computationally intractable. A number of approximations have been suggested, however these are often limited because their assumptions are inappropriate, they converge to local minima, or they are computationally demanding. The use of hyperparameters in addition to model parameters has allowed the technique of ARD to be developed. This method allows the data to determine which inputs should influence the model's predictions, and as shown in this chapter can be successfully be used to introduce interpretability into an otherwise non-interpretable model.

Chapter 4

Kernel Based Methods

4.1 Motivations from Neural Networks

Many neural network architectures, including those described in Chapter 3, deal with the problem of predictive learning. One of the attractions of such models is their flexibility, i.e. their ability to model a wide variety of functions. However, this flexibility comes at a cost in that a large number of parameters may need to be determined from the data. As described in Chapter 2, regularisation methods attempt to penalise a models parametric and structural form, hence avoiding "overfitting" of the data and restoring the well-posedness of the learning problem. As Cherkassky and Mulier (1998) observe, no single universally accepted theoretical framework for predictive learning currently exists.

One of the strengths of kernel based methods is that they are *non-parametric* modelling techniques, where it is not necessary to specify for example the number of basis functions beforehand. Whilst this is advantageous in general, parametric models are useful in their own right. This is particularly so if prior knowledge about the problem exists, for example it may be that the vast majority of the data's properties are described by a small set of linearly independent basis functions $\{\phi_1(\cdot), \dots, \phi_n(\cdot)\}$. It also plays a part when an interpretable model is required without sacrificing generalisation performance. This may be some motivation for the construction of *semi-parametric* models (Smola et al., 1998), which due to the parametric part are easily interpreted and perform well due to the nonparametric term.

In the Bayesian interpretation of neural network modelling, a non-linear function $y(\boldsymbol{x})$ parameterised by parameters $\boldsymbol{w}$ is assumed to underline the dataset $\mathcal{D} = (\boldsymbol{x}, y)_{n=1}^{N}$, and the adaptation of the model to the data corresponds to an inference of the function given the data. The Bayesian approach to neural networks can be considered to be an example of the parameter or weight space approach to learning, viz. the learning machine is parameterised by a weight vector and the task of learning is to find optimal values of these weights.

An alternative perspective on learning can be achieved by taking a function space approach. Instead of considering priors over weights, the Bayesian approach to neural networks can be shown to induce a prior distribution over functions (Neal, 1995; Williams and Rasmussen, 1996). The basis of *Gaussian process* modelling is without parameterising $y(\boldsymbol{x})$, to place a prior distribution, $P(y(\boldsymbol{x}))$, directly on the space of functions. The simplest type of prior over functions is called a Gaussian process (Williams and Rasmussen, 1996; Rasmussen, 1996; Gibbs, 1997). It can be thought of as a generalisation of a Gaussian distribution over a finite vector space to a function space of potentially infinite dimension (Rasmussen, 1996).

As (Gibbs, 1997) observes just as a Gaussian distribution is fully specified by its mean and covariance matrix, a Gaussian process is fully specified by a mean and a covariance function. Here, the mean is a function of $\boldsymbol{x}$, which as (MacKay, 1995) observes is often the zero function, and the covariance is a function $C(\boldsymbol{x}, \boldsymbol{x}')$ that expresses the expected covariance between the function at the points $\boldsymbol{x}$ and $\boldsymbol{x}'$.

An important property of the covariance function was given by (Lamperti, 1977) which summarises the above description,

Theorem 4.1 (Lamperti's Theorem). *A function C defined on a parameter set $\mathcal{X}$, $C : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is the covariance of some process $y(\boldsymbol{x})$ iff C is positive semi-definite.*

An important corollary follows from Lamperti's theorem,

Corollary 4.2 (Lamperti 1977). *Assume that we have some function C that is positive semi-definite. For each finite set $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathcal{X}$ the matrix $C(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is then symmetric and positive semi-definite. There is accordingly, a unique Gaussian distribution defined on $\mathbb{R}^d$ with mean zero having $C(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for its covariance matrix. It can then be shown that there is a real valued Gaussian process having mean zero and given function C for its covariance.*

The current interest in Gaussian processes as kernel based methods has been initiated by the work of (Neal, 1995) on priors for infinite networks. This work showed that the prior over functions defined by a neural network with one hidden layer converges to a Gaussian process as the number of hidden nodes tends to infinity for certain priors on the weights.

Having defined a Gaussian process, Williams and Rasmussen (1996) have shown how this can be useful in making predictions with unseen data. Given the set of inputs $\boldsymbol{x}$ with mean $\mu_{\boldsymbol{x}}$ and covariance matrix C, the ijth element of C is given by the covariance function $C(\boldsymbol{x}_i, \boldsymbol{x}_j)$ which is dependent on the inputs associated with the ith and jth observations respectively. The predicted output y for some new test data $\boldsymbol{x}^*$ allows the definition of a new Gaussian process model based on the new data and the assumption that the new predicted output has a mean μ_y. Following the work of Williams (1998) the new Gaussian process will have mean $\boldsymbol{\mu}_{\boldsymbol{x}}\mu_y$ and partitioned covariance matrix,

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{C} & \boldsymbol{c} \\ \boldsymbol{c}^T & c \end{bmatrix}$$

where $c = c(\boldsymbol{x}^*, \boldsymbol{x}^*)$ and $\boldsymbol{c} = [c(\boldsymbol{x}^*, \boldsymbol{x}_1), \ldots, c(\boldsymbol{x}^*, \boldsymbol{x}_N)]^T$. The random variables $\boldsymbol{x}$ and $y' = y$ are statistically independent where the mean and variance of y' are given by,

$$\mu_{y'} = \mu_y - \boldsymbol{c}^T \boldsymbol{C}^{-1} \boldsymbol{\mu}_{\boldsymbol{x}} \quad \text{and} \quad \sigma_y^2 = c - \boldsymbol{c}^T \boldsymbol{C}^{-1} \boldsymbol{c} \tag{4.1}$$

The study of Gaussian processes for regression is far from new. Within the geostatistics field, Matheron (1963) proposed a framework for regression using optimal linear estimators which he termed 'kriging'. This framework is identical to the Gaussian process approach to regression. Kriging has been developed considerably over the past thirty years (Cressie, 1993), including several Bayesian treatments (Omre, 1987; Kitanidis, 1986). However, the geostatistics approach to the Gaussian process model has concentrated mainly on low-dimensional input spaces, and has largely ignored any probabilistic interpretation of the model, and any interpretation of the individual parameters of the covariance function (Gibbs, 1997).

The Gaussian process framework encompasses a wide range of different regression models. O'Hagan (1978) introduced an approach that is essentially similar to Gaussian processes. Generalised radial basis functions (Poggio and Girosi, 1989), autoregressive moving average (ARMA) models (Wahba, 1990a), and variable metric kernel methods (Lowe, 1995) are all closely related to Gaussian processes. The Bayesian interpretation of Gaussian processes was extended in (Williams and Rasmussen, 1996) and (Williams and Barber, 1998), and a comparison of Gaussian processes with other methods such as neural networks and MARS was carried out by (Rasmussen, 1996).

Another class of kernel methods, namely Support Vector Machines (SVMs) have been developed from a very different viewpoint (Vapnik, 1998). The proponents of SVMs have been concerned with the question under what conditions the ill-posed learning problem of determining the probabilistic dependence between an input and an output can actually be solved uniquely. As well as what learning paradigm should be proposed to construct learning algorithms for this task. This has given rise to the discipline of *statistical learning theory* (SLT) (Cortes and Vapnik, 1995; Vapnik, 1998; Smola, 1998). The basis of this approach is the *structural risk minimisation* (SRM) principle. This problem can formally be defined by choosing a loss function and a model hypothesis space, both choices are guided by our prior belief into the nature of the underlying data generating mechanism and noise model for the process to be learned. A function is then sought that minimises the *expected loss* or *expected risk*.

Since the probability distribution, $p(\boldsymbol{x}, y)$, from which the data is draw is typically unknown, the expected loss cannot be computed directly. Hence, a stochastic approximation termed the *empirical risk* is introduced. The law of large numbers guaranteed that the empirical risk converges to the expected risk, and as such a common approach consists in minimising the empirical risk rather than the expected risk. The empirical risk minimisation (ERM) principle states that if the empirical risk converges to the expected risk, then the minimum of the empirical risk may converge to the minimum of the expected risk (Vapnik, 1998). If this principle does not hold the ERM principle does not allow us to make any inference based on the dataset and is therefore inconsistent.

Central to the SLT principle is the notion of the VC-dimension (Cortes and Vapnik, 1995), however in terms of support vector regression the problem is that the VC-dimension does not contain any *scale* information and is therefore too conservative in most cases (Smola, 1998; Vapnik, 1998). Hence, it is essential to use *scale dependent* quantities such as the (level) fat shattering VC dimension ($\text{fat}_{\mathcal{F}}$) or quantities like entropy or covering numbers. Definitions of these quantities can be found in Vapnik (1998), Smola (1998) , or Cristianini and Shawe-Taylor (2000). A discussion of these quantities is beyond the scope of this thesis.

4.2 Support Vector Regression

Regression is one of the most common data modelling problems, and numerous methods exist for tackling it. An elegant feature of the support vector machine approach, and other kernel based methods, derives from the fact the problem is reduced to a linear setting. Consider the inference problem described in Chapter 3, of approximating a set of data, $\mathcal{D}$, consisting of inputs, $\boldsymbol{x}$, and an output y, with a linear function f of the form,

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b \quad \text{with} \quad \boldsymbol{w} \in \mathbb{R}^d, b \in \mathbb{R} \tag{4.2}$$

The problem of minimising the empirical risk is generally an ill-posed problem (as defined in Chapter 2) except in very restrained model classes and leads to model overfitting (Vapnik, 1998). The problem can be converted to one that is well posed, by restricting the set of functions from which f is chosen to some *compact* set $\mathcal{F}$. As observed by Smola (1998), restriction of a function to a compact set that is sufficiently well behaved, for example it has a finite covering number, the empirical risk will converge to the expected risk for increasing sample size. In practice this is achieved by imposing a convex penalty term to the cost function that is being minimised.

4.3 Loss Functions

In order to construct algorithms for minimising the empirical risk it is necessary to specify which loss function to use. This thesis only considers convex loss functions, the practical reason being that to solve the problems described above can be proven to have a unique minimum (Smola, 1998). As Smola (1998) observes for many nonconvex loss functions, the attempt to solve the corresponding risk minimisation problem results in combinatorial optimisation settings that are NP-hard as they exhibit many local minima. Although the original formulation for minimising the empirical risk was specific to classification, in terms of a margin, to generalise the support vector algorithm to regression estimation (Vapnik, 1998) an analogue of the margin is constructed in the space of the target values t where $t \in \mathbb{R}$.

For the conventional quadratic loss, the SVM framework can be written as,

$$\max_{\alpha,\alpha^*} W(\alpha,\alpha^*) = \max_{\alpha,\alpha^*} \begin{cases} -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*) \\ \langle \boldsymbol{x}_i, \boldsymbol{x}_j\rangle + \sum_{i=1}^{N}(\alpha_i-\alpha_i^*)y_i \\ -\frac{1}{2C}\sum_{i=1}^{N}(\alpha_i^2+\alpha_i^{*2}) \end{cases} \tag{4.3}$$

As observed by Smola (1998), the corresponding optimisation problem can be simplified by exploiting the Karush-Kuhn-Tucker conditions, and noting that these imply $\beta_i^* = |\beta_i|$. The resultant optimisation problem is then given by,

$$\min_{\beta} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\beta_i\beta_j\langle \boldsymbol{x}_i, \boldsymbol{x}_j\rangle - \sum_{i=1}^{N}\beta_i y_i + \frac{1}{2C}\sum_{i=1}^{N}\beta_i^2 \tag{4.4}$$

with the constraints,

$$\sum_{i=1}^{N}\beta_i = 0, \tag{4.5}$$

where α_i, β_i are Lagrange multipliers and N is the number of samples in the dataset $\mathcal{D}$.

In ϵ-insensitive support vector regression Vapnik (1995, 1998), the goal is to find a function $f(\boldsymbol{x})$ that has at most ϵ deviation from the target values of the training data, and is as smooth as possible[1]. That is to say errors are acceptable provided they are less than ϵ. It follows from the work of Huber (1981), that this loss function is robust in the class of uniform densities. In cases where a function f does not exist or situations where errors greater than ϵ need to be allowed, slack variables ξ_i, ξ_i^* can be introduced to cope with otherwise infeasible optimisation constraints (Cortes and Vapnik, 1995).

[1]Smoothness was defined in Chapter 2

This then leads to the formulation stated in(Vapnik, 1998),

$$\begin{array}{rrcl} \text{minimise} & \frac{1}{2}\|w\|^2 & + & C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \\ \text{subject to} & ((\boldsymbol{w}\cdot\boldsymbol{x}_i)+b)-y_i & \leq & \epsilon+\xi_i \\ & y_i-((\boldsymbol{w}\cdot\boldsymbol{x}_i)+b) & \leq & \epsilon+\xi_i^* \\ & \xi_i,\xi_i^* & \geq & 0 \end{array} \tag{4.6}$$

Constructing a Lagrange function from the objective function and the corresponding constraints gives,

$$\begin{aligned} L = \frac{1}{2}\|w\|^2 \quad &+ \quad C\sum_{i=1}^{N}(\xi_i+\xi_i^*) - \sum_{i=1}^{N}\alpha_i(\epsilon+\xi_i-y_i+\langle w,x_i\rangle+b) \\ &- \quad \sum_{i=1}^{N}\alpha_i^*(\epsilon+\xi_i^*+y_i-\langle w,x_i\rangle-b) - \sum_{i=1}^{N}(\eta_i\xi_i+\eta_i^*\xi_i^*) \end{aligned} \tag{4.7}$$

Given that the *dual* variables, $\alpha_i, \alpha_i^*, \eta_i, \eta_i^*$ need to satisfy positivity constraints the solution is then given by,

$$\text{maximise} \quad \left\{ \begin{array}{l} -\frac{1}{2}\sum_{i,j=1}^{N}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)\langle x_i,x_j\rangle \\ -\epsilon\sum_{i=1}^{N}(\alpha_i+\alpha_i^*)+\sum_{i=1}^{N}(\alpha_i+\alpha_i^*) \end{array} \right.$$

$$\text{subject to} \quad \left\{ \begin{array}{c} \sum_{i=1}^{N}(\alpha_i-\alpha_i^*)=0 \\ \alpha_i,\alpha_i^*\in[0,C] \end{array} \right. \tag{4.8}$$

After some algebraic manipulation, the SVM solution is given by,

$$f(\boldsymbol{x}) = \sum_{i=1}^{N}(\alpha_i-\alpha_i^*)\langle \boldsymbol{x}_i,\boldsymbol{x}\rangle + b \tag{4.9}$$

The computation of the constant b is obtained by exploiting the Karush-Kuhn-Tucker (KKT) conditions (Smola, 1998),

$$\begin{array}{rcllc} b & = & y_i-\langle \boldsymbol{w},x_i\rangle-\epsilon & \text{for} & \alpha_i\in(0,C) \\ b & = & y_i-\langle \boldsymbol{w},x_i\rangle+\epsilon & \text{for} & \alpha_i^*\in(0,C) \end{array} \tag{4.10}$$

A number of other loss functions also exist. In the limit as $\epsilon \rightarrow 0$ we get the Laplacian loss function which leads to median type estimators. Quadratic loss which has been shown to be optimal for Gaussian additive noise.

A cautionary remark is necessary regarding the use of loss functions other than the ϵ-insensitive. Unless $\epsilon > 0$ all the advantages associated with a sparse parameter decomposition will be lost. This may be acceptable for small datasets, but will render the prediction step computationally expensive. As a result a trade off between the potential loss in predictive accuracy and speed of predictions may have to be made. The Laplace, Gaussian, Huber's robust, and the ϵ-insensitive loss function lead to quadratic programming problems, which can be solved using readily available quadratic programming optimisers (Mészáros, 1998). Other loss function may lead to convex programming problems that are more difficult to optimise.

4.4 Kernel Functions

SVMs make use of reproducing kernels which are functions that provide an elegant approach to dealing with nonlinear algorithms by reducing them to linear ones in some feature space F nonlinearly related to the input space. The only way in which the data appears in the training problem is in the form of dot products, $\boldsymbol{x}_i \cdot \boldsymbol{x}_j$. If the data is mapped to some other (possibly infinite dimensional) Hilbert space H[2].

Definition 4.3 (Kernel Function). A kernel is a function K such that for all $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$, where $\mathcal{X}$ is a data input space,

$$K(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}) \cdot \phi(\boldsymbol{x}') \rangle \tag{4.11}$$

where ϕ is a mapping from $\mathcal{X}$ to an (inner product) feature space $\mathcal{F}$.

As observed by Vapnik (1998) and Smola (1998), an important consequence of the SVM formulation is that the dimensionality of the feature space need not affect the computation. Since the feature vectors are not represented explicitly, the number of operations required to compute the inner product by evaluating the kernel function is not necessarily proportional to the number of features. The computational complexity of kernel based learners is discussed further in Section 4.7.

The use of a kernel function enables operations to be performed in the input space rather than the potentially high dimensional feature space, by taking a dot product there,

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \Phi(\boldsymbol{x}_i) \cdot \Phi(\boldsymbol{x}_j) \tag{4.12}$$

The solution of SVMs is then a weighted linear summation of kernels,

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \beta_i K(\boldsymbol{x}_i, \boldsymbol{x}_j), \tag{4.13}$$

where these kernels are *centered* on the data points. The elegance of using kernels lies in the fact that one can deal explicitly with spaces f of arbitrary dimensionality without

[2] A Hilbert space is any linear space, with an inner product defined but which is also complete with respect to the Euclidean norm

having to compute the map Φ explicitly. Despite these attractive features, there is a need to consider under which conditions a symmetric kernel $K(\boldsymbol{x}, \boldsymbol{x}')$ corresponds to a dot product in some feature space.

4.4.1 Mercer's Theorem

Theorem 4.4 (Mercer's theorem). *There exists a mapping Φ and an expansion,*

$$K(\boldsymbol{x}, \boldsymbol{x}') = \sum_i \Phi(\boldsymbol{x}_i)\Phi(\boldsymbol{x}'_i) \tag{4.14}$$

and

$$\int K(x, x')g(x)g(x')dxdx' \geq 0 \tag{4.15}$$

if and only if,

$$\int g(x)^2 dx < \infty \tag{4.16}$$

A consequence of Mercer's theorem is that any positive definite function in L_2 can be chosen and we have a valid kernel function without even having to construct Φ. However, defining Φ implicitly through K also creates some serious problems. Mostly, this map and many of its properties are unknown. Even worse, this method does not generate any rule about which kernel should be used, or why mapping into a very high dimensional space often provides good results, seemingly defying the curse of dimensionality Bellman (1961). This dilemma can be resolved by showing that kernels $K(x, x')$ correspond to regularisation operators Smola (1998). Prior knowledge can be incorporated in SVMs by careful choice of kernel function. Particular kernels may be more appropriate in certain circumstances given the data distribution.

A key issue in every learning problem concerns the input (and output) data representation (Fukunga, 1990). As Evgeniou and Pontil (1999) observe, in practical problems the choice of the regressors is often much more important than the choice of learning algorithms or technique. The choice of an appropriate input representation typically depends on prior knowledge about the particular learning problem. Jaakkola and Haussler (1999) have considered the case in which prior information is available in terms of a parametric probabilistic model $p(\boldsymbol{x}, y)$ of the process generating the data.

A number of interesting observations can be made that are relevant to the kernel based methods described above. As observed by Vapnik (1998), the choice of the kernel K is equivalent to choosing features, $\phi_i(\boldsymbol{x})$, related to the original inputs $\boldsymbol{x}$, where the basis functions are defined by $K(\boldsymbol{x}, \boldsymbol{x}_i) \equiv \sum_{i=1}^{N} \phi_i(\boldsymbol{x})\phi_i(\boldsymbol{x}_i)$. Following Smola (1998), if we assume that K is given and that the input representation is changed through a vector function $\boldsymbol{h}(\boldsymbol{x})$ mapping the original input $\boldsymbol{x}$ onto the new feature vector $\boldsymbol{h}$, this can be considered to be equivalent to using a new kernel $K^{'}$ defined in terms of the *composite* features $\phi_i(\boldsymbol{h}(\boldsymbol{x}))$ as $K^{'}(\boldsymbol{x},\boldsymbol{x}_i) \equiv \sum_{i=1}^{N} \lambda_i \phi_i(\boldsymbol{h}(\boldsymbol{x}))\phi_i(\boldsymbol{h}(\boldsymbol{x}_i))$.

The work of Aronszajn (1950) describes several ways to construct positive definite kernels, and thereby the associated reproducing kernel Hilbert space[3] (RKHS). This work showed that there are several symmetric positive definite kernels and a number of ways to construct new kernels from existing ones, by operating on them with operations such as addition and convolution(Evgeniou and Pontil, 1999). For example, if K_1 and K_2 are Mercer kernels, then $K_1 + K_2$ is also a Mercer kernel, as is K_1K_2. These ideas are exploited in the work on ANOVA spline kernels described in Section 5.2.1 and Chapter 5.

4.5 How do you choose the Kernel?

Whilst Mercer's condition provides a formal definition for a kernel function, it gives no information about how to construct Φ and hence what $\mathcal{H}$ is. The most popular strategy to date, for choosing the kernel, has been based on prior knowledge (Schölkopf et al., 1998; Burges, 1999), Cristianini and Shawe-Taylor (2000) have proposed an alternative method for selecting a suitable kernel, and that is to start from the features and to then evaluate the corresponding inner product. An advantage of this approach is that there is no need to check whether the kernel satisfies Mercer's condition, since this will follow automatically from the definition of an inner product. As (Vapnik, 1998) observes, in a classification scenario, the upper bound on the VC dimension is a potential avenue to provide a means for comparing kernels. However, this requires the estimation of the radius of the hyperplane enclosing the data in the nonlinear feature space. As a final caution, even if a strong theoretical method for selecting a kernel is developed unless this can be validated using an independent test set on a large number of problems, methods such as bootstrapping and cross validation will remain the preferred method for kernel selection.

Even though arguments about the choice of kernel function continues which at its heart is data and problem dependent, work on the USPS dataset, that despite the choice of kernel function similar sets of support vectors and similar classification results can be obtained. Despite these proposed methods, the best choice of kernel for a given problem still remains an active research problem.

4.6 Bayesian Learning in Kernel Methods

The Gaussian process model described in Section 4.1 can be considered to be inherently Bayesian given its probabilistic derivation. The solution to an SVM has been shown to

[3]The basic ideas are outlined in Appendix A

be a weighted sum of kernel functions however, as Tipping (2000b) observes the support vector methodology does exhibit a number of disadvantages. In a regression scenario, the predictions are not *probabilistic*, the SVM outputs a point estimate whereas ideally the conditional distribution $p(t|\boldsymbol{x})$ is needed to capture the uncertainty in the prediction. In regression this may take the form of error bars, whilst in the classification class conditional probabilities are required. It is also necessary to determine the hyperparameters C (the capacity control term) and ϵ (the insensitivity parameter). Typical methods for determining these are based on a cross validation procedure, that can be considered to be both wasteful of data and computation. Although relatively sparse, SVMs make liberal use of kernel functions, the requisite number of which grows linearly with the size of the training set.

Recent work by (Sollich, 1999a,b, 2000) and (Gao et al., 2000) has addressed the non-probabilistic nature of support vector classification (SVC) and support vector regression (SVR) respectively. Sollich showed that SVC can be can be interpreted as a MAP solution to the inference problem based on Gaussian process priors, and an appropriate likelihood function based on a probabilistic interpretation. This has allowed class conditional probabilities and error bars to be obtained.

Standard regularization theory formulates the learning problem as a functional variational problem of finding the function $y(\boldsymbol{x})$ that minimises a functional of the form

$$R_{\text{emp}}[y(\boldsymbol{x})] = \sum_{i=1}^{N} L(t_i, y(\boldsymbol{x}_i)) + \frac{\lambda}{2}\|y(\boldsymbol{x})\|_K^2 \tag{4.17}$$

where $L(\cdot,\cdot)$ is a loss function and $\|\cdot\|$ is a norm in a Reproducing Kernel Hilbert Space $\mathcal{H}$ with a kernel K, (Wahba, 1990a; Evgeniou et al., 2000). The solution to equation (4.17) has the following representation

$$y(\boldsymbol{x}) = \sum_{i=1}^{N} w_i K(\boldsymbol{x}, \boldsymbol{x}_i) + w_0 \tag{4.18}$$

where w_0 can be assumed to be zero, for an explanation see (Evgeniou et al., 2000).

It is well known that a variational principle of the type of equation (4.17) can be derived not only in the context of functional analysis, but also in a probabilistic framework (Wahba, 1990a; Girosi et al., 1995; Evgeniou et al., 2000; Sollich, 1999b). Let $\mathcal{D}$ be the training dataset as defined above, and define $P(y(\boldsymbol{x})|\lambda) \propto \exp\{-\frac{\lambda}{2}\|y(\cdot)\|_K^2\}$ as the prior probability of the random field $y(\boldsymbol{x})$ and $P[\mathcal{D}|y(\cdot)] \propto \exp\{-C\sum_{i=1}^{N} L(t_i, y(\boldsymbol{x}_i))\}$ as the conditional probability of the data given the field $y(\boldsymbol{x})$. i.e., the likelihood. Then the posterior distribution $P(y(\boldsymbol{x})|\mathcal{D})$ can now be computed by using Bayes' rule as:

$$P(y(\boldsymbol{x})|\mathcal{D},\lambda) = \frac{P(\mathcal{D}|y(\boldsymbol{x}))P(y(\boldsymbol{x})|\lambda)}{P(\mathcal{D}|\lambda)} \propto \exp\{-\sum_{i=1}^{N} L(t_i, y(\boldsymbol{x}_i)) - \frac{\lambda}{2}\|y(\cdot)\|_K^2\} \tag{4.19}$$

Hence the Maximum A Posterior (MAP) estimate of the probability distribution (4.19), which maximizes the a posterior probability $P(y(\boldsymbol{x})|\mathcal{D})$, is the minimiser of the functional (4.17). The MAP estimate depends on the knowledge of λ and σ_t^2. The above framework is general enough to cover both the Gaussian Processes and the Support Vector Machines as well as classification (Vapnik, 1998; Evgeniou et al., 2000)

Recently Tipping (2000b) has formulated the Relevance Vector Machine (RVM), which is a probabilistic sparse kernel model identical in functional form to the SVM. However, a Bayesian approach to the inference problem is adopted from the outset, where a prior over the weights governed by a set of hyperparameters is introduced. In contrast to the Bayesian neural network model described in Chapter 3, where a separate hyperparameter was introduced for each input vector, the RVM uses a separate hyperparameter for every weight in the network[4]. The most probable values for these hyperparameters are iteratively estimated from the data in an evidence type framework (see Section 4.2). Sparsity in an RVM is achieved because the posterior distributions of many of the weights are sharply peaked around zero. The most compelling feature of the RVM is that it achieves comparable generalisation performance to an equivalent SVM, and it typically uses dramatically fewer kernel functions.

Given the problems associated with the evidence framework described in Section 3.7.1.1 Bishop and Tipping (2000) have extended the original RVM formulation to a more rigourous Bayesian setting using variational learning as described in Section 3.7.8. This has in part been motivated by the observation made by Tipping (2000b) that the principal disadvantage of the RVM is in the complexity of the training phase, since it is necessary to repeatedly compute and invert the Hessian matrix, requiring $\mathcal{O}(N)^2$ and $\mathcal{O}(N)^3$ computations. For large datasets, this makes training considerably slower than the SVM.

As observed by (Vapnik, 1998) in a classification setting the non-zero Lagrange multipliers, or the so called *support vectors* are the data points that lie closest to the decision boundary. However, for an RVM the non-zero weights are *not* associated with examples close to the decision boundary, but appear to represent what Tipping (2000b) describes as *prototypical* examples of the classes. However, it can be argued that in a classification scenario the support vectors that are chosen inherently contain information about the decision boundary. As such, if the data points *not* corresponding to the support vectors are removed, and the SVM is again trained with the data points that are support vectors the same decision boundary will be obtained. However, if this same procedure were to be repeated using the relevance vectors a *different* decision boundary would be obtained.

As described in Section 4.3 support vector machine regression measures the goodness of fit of a model not only by the usual quadratic loss function, but also by the ϵ-insensitive

[4]The introduction of multiple hyperparameters for the parameters of a model is exploited in the algorithm described in Chapter 5

loss function which is similar to the robust loss functions introduced by Huber (1981). The quadratic loss function is well justified under the assumption of additive Gaussian noise. However, the noise model underlying the ϵ-insensitive loss function is less clear. Work by Pontil et al. (1998) has used Bayesian learning to show that the noise model is equivalent to a model of additive Gaussian noise, where the mean and variance are random variables. In this work the traditional assumption that noise variables all have identical probability distributions is dropped. Different data points may have been collected at different times, under different conditions, so it is more realistic to assume that the noise variables $\delta(i)$ have probability distributions P_i which are not necessarily identical. Hence.

$$P(g|f) = \prod_{i=1}^{N} P_i \delta_i \tag{4.20}$$

Instead, assume that the noise distributions P_i are actually Gaussian but they do not necessarily have zero mean.

$$P_i \delta_i \propto \exp^{-\beta_i(\delta_i - t_i)^2} \tag{4.21}$$

Hence, we can write,

$$P_i(\delta_i|\beta_i, t_i) \propto \exp^{\beta_i(\delta_i - t_i)^2} \tag{4.22}$$

and therefore the marginal distribution integrating over β and t gives,

$$\begin{aligned}
P(f|g) &\propto P(g|f)P(f) \\
P(f|g,\beta,t) &\propto \prod_{i=1}^{N} P(\delta_i|\beta_i, t_i)P(f) \\
P(f|g) &\propto \int\int \prod_{i=1}^{N} P_i(\delta_i|\beta_i, t_i)P(f)P(\beta, t)d\beta dt
\end{aligned} \tag{4.23}$$

Since the values of β and t are not known we want to marginalise over them. The function that minimises Equation 4.23 is the one that minimises,

$$H[f] = \sum_{i=1}^{N} V(f(\boldsymbol{x}_i) - y_i) + \alpha\Psi[f]$$

where,

$$V(x) = -\log \int_0^\infty d\beta \int_{-\infty}^{\infty} dt \sqrt{\beta} \exp^{-\beta(x-t)^2} P(\beta, t)$$

and

$$P(\beta, t) = \mu(\beta)\lambda(t) \tag{4.24}$$

with the following priors on β and λ,

$$\begin{aligned} \mu(\beta) &= \beta^{-2} \exp^{-\frac{1}{4\beta}} \\ \lambda_\epsilon(t) &= \frac{1}{2(\epsilon+2)} \left(\chi_{[-\epsilon,\epsilon]}(t) + \delta(t-\epsilon) + (t+\epsilon)\right) \end{aligned} \tag{4.25}$$

4.7 Computational Complexity when using Kernel Based Learners

Computational cost is a major issue in learning methods. Kernel methods require the evaluation of an $N \times N$ positive definite matrix, where N denotes the number of samples in the dataset $\mathcal{D}$. Gaussian processes, which were discussed in Section 4.1, allow exact Bayesian analysis with simple matrix manipulations. However, obtaining the MAP estimate requires the computation, storage and inversion of the *full* covariance matrix which is an $\mathcal{O}(N)^3$ process. Gibbs and MacKay (1997) have addressed this problem by considering a matrix condensation method proposed by Gull (1989) which reduces the computational cost of finding a solution to $\mathcal{O}(kN^2)$ rather than $\mathcal{O}(N^3)$, when the covariance matrix contains a significant number of small eigenvalues. Smola and Bartlett (2000) have proposed a sparse greedy method that approximates the MAP solution of a Gaussian process that involves a computational requirement of $\mathcal{O}(m^2N)$, a matrix storage requirement of $\mathcal{O}(mN)$, and a cost of m for prediction where $m \ll N$. Their method exploits the ideas proposed in the context of wavelets, that are reviewed and utilised in the algorithms described in Chapter 5. In addition to the traditional cost function that is being minimised, an L_0 penalty is imposed from which a greedy algorithm similar to matching pursuit (5.1.2) is obtained. The algorithm starts from an empty structure to which basis functions are added or deleted in a greedy fashion. The inherent limitation of this approach is its convergence to local minima (see Chapter 7 for a further discussion of this limitation).

Williams and Seeger (2000) have proposed an alternative method to speed up the computation of kernel methods that is based on an approximation to the eigendecomposition of the positive definite kernel matrix using the Nyström method. Given that the kernel, $K(\boldsymbol{x}, \boldsymbol{x}')$, can be written in the form,

$$K(\boldsymbol{x}, \boldsymbol{x}') = \sum_{i=1}^{N} \lambda_i \phi_i(\boldsymbol{x}) \phi_i(\boldsymbol{x}') \tag{4.26}$$

where $N \leq \infty, \lambda_1, \geq \lambda_2, \geq \cdots \geq 0$ denote the eigenvalues, and $\phi_1, \phi_2, \ldots$ denote the eigenfunctions of the kernel K. The Nyström approximation to the i^{th} eigenfunction can then be computed as,

$$\phi_i(\boldsymbol{x}') \approx \frac{\sqrt{q}}{\lambda_i^{(q)}} \sum_{k=1}^{q} K(\boldsymbol{x}', \boldsymbol{x}_k) \mathcal{U}_{k,i}^{(q)} \tag{4.27}$$

where q is the number of i.i.d samples drawn from the probability density of the input vector $\boldsymbol{x}$. The time required for this computation is of order $\mathcal{O}(m^2 N)$.

A different approach for dealing with large datasets has been suggested by Tresp (2000) for Gaussian process regression. This method is based on splitting the dataset into smaller subsets and training individual Gaussian process predictors on each of them. The final prediction is obtained by a specific weighting of the individual predictors.

Within the SVM community, methods for reducing computation time have been proposed that are based on active constraints (Cristianini and Shawe-Taylor, 2000). Given that the solution to an SVM optimisation problem is based on inequality constraints, if it were possible to know in advance which constraints were active, it would be possible to discard all the inactive constraints and simplify the problem. The simplest iterative heuristics that builds up an *active dataset* is known as *chunking* (Cristianini and Shawe-Taylor, 2000). The algorithm starts with an arbitrary subset or 'chunk' of the data, and trains an SVM on that portion of the data. The algorithm then retrains using only the data points that are support vectors from the chunk whilst discarding the other points, and then it uses the hypothesis found to test the points in the remaining part of the data. The points that most violate the KKT conditions are added to the support vectors of the previous system to form a new chunk. This procedure is iterated until some stopping criteria is satisfied.

Despite the attractiveness of the chunking method, this approach can fail when the kernel matrix for the set of support vectors does not fit into a computers memory, or when the problem under consideration is not sparse. A further complication arises when the dataset size is so large that the set of support vectors is still too large to be dealt with by a generic optimisation routine (Cristianini and Shawe-Taylor, 2000). The *decomposition* algorithm of Osuna et al. (1996) has been proposed to overcome these limitations by only updating a fixed subset of multipliers whilst keeping the others constant. Hence, each time a new point is added to the active set, another point has to be removed. In this algorithm, the goal is not to identify all of the active constraints in order to run the optimiser on all of them, but rather to optimise the global problem by only acting on a small subset of data at a time. The sequential minimal optimisation (SMO) algorithm of (Platt, 1998) is derived by taking the idea of the decomposition method to its extreme and optimising a minimal subset of just two points at each iteration. Given that the optimisation problem is based only on two data points, the solution can be found analytically obviating the need for a quadratic programming routine.

4.8 Summary

Kernel methods, that incorporate a number of attractive features, have become a popular method for regression estimation. Although the two principle methods, Gaussian processes and support vector machines, have been developed in different ways their close correspondence has led to considerable synergetic efforts. Work on Gaussian processes stemmed from the observation by Neal (1995) that when an appropriate prior probability distribution is used, it is not necessary to limit the complexity of a neural network model based on the amount of training data available. The SVM approach (Vapnik, 1998) is an implementation of the structural risk minimisation principle. Kernel and spline smoothing can be considered to be powerful non-parametric statistical models that, while free from unreasonable parametric restrictions, allow the specification of prior knowledge about an unknown relation between data variables in a simple and convenient way. From the work of Kimeldorf and Wahba (1971), it is easy to show how the general spline smoothing model serves as a common basis for Gaussian processes and support vector machines.

Given that both the Gaussian process and the support vector machine can be interpreted probabilistically, the Bayesian probabilistic viewpoint results in a natural and elegant to choose values for the free parameters or to integrate them out. The framework also provides a clear interpretation of the kernel that may assist in its choice. Considering the kernel learning problem as an optimisation within a reproducing kernel Hilbert space (RKHS) allows the ideas and techniques from functional analysis, operator theory and approximation theory to be used. Given that the solution to these kernel based methods is a weighted sum of kernel functions the resulting solution is opaque. Methods for constructing interpretable models are described in the next chapter, where the solution is still given as the weighted sum of kernel functions but is not opaque.

Chapter 5

Interpretable Sparse Kernels

The interpretation of complex models has started to receive some attention within the machine learning community (Plate, 1999; Gibbs, 1997). Given that the majority of data modelling studies are performed in a particular application domain the ability to understand a final model structure can be regarded as being an attractive feature of the data modelling process. The ability to visualise the overall effects of different inputs, their interactions, and the strength of their interactions can aid in the model understanding issue. The incorporation of interpretability into the model building process can aid in model understanding, model validation and selection and indirectly model performance.

Additive models as discussed in Section 2.4, are one framework that enables more extensive understanding of a model structure over simple input selection. Such models are hence particularly attractive for model interpretability. Methods for enforcing or formulating additivity in various families of flexible models have been investigated by a number of researchers. Girosi et al. (1995), considering generalised functions only, show that additive models can be formulated as regularisation networks, thereby allowing additive regularisers to be constructed. Moody and Rögnvaldsson (1996) discuss various smoothing terms for feedforward neural networks that penalise higher order derivatives with respect to the inputs; incorporation of a regularisation term pushes the model towards an additive structure Plate (1999). Other notable additive models include the Smoothing-Spline ANOVA (SS-ANOVA) model of (Wahba et al., 1994). This method is based on a Gaussian process model with a particular covariance function, and an additive structure.

5.1 Methods for Sparse Approximation

A considerable focus of activity in the signal processing community has been on the development of signal representations that use overcomplete and hence non-orthogonal bases. Linear superpositions of a small number of basis functions selected from a large, redundant set of basis functions commonly referred to as a *dictionary* are being advocated (Mallat and Zhang, 1993). Using the terminology introduced by Mallat and Zhang (1993), a dictionary is a collection of parameterised waveforms, $q = (\phi_\gamma : \gamma \in \Gamma)$. In the signal processing community, the waveforms, ϕ_γ, are discrete time signals of length n called atoms. As Mallat and Zhang (1993) observe, dictionaries can be complete or overcomplete, in which case they contain exactly n atoms or more than n atoms. For the work described in this thesis the waveforms can be considered to correspond to sets of basis functions, and it is the goal of this work to select a sparse basis from an overcomplete set of basis functions.

Given a dictionary of J basis functions $\phi_1(\boldsymbol{x}), \ldots, \phi_J(\boldsymbol{x})$, where J is very large or possibly infinite, sparse approximation techniques seek a function $f(\boldsymbol{x})$ of the form,

$$f(\boldsymbol{x}) = \sum_{j=1}^{J} c_j \phi_j(\boldsymbol{x}) \tag{5.1}$$

That is a linear combination of the smallest number of elements of the dictionary, with the smallest number of non-zero coefficients c_i. Formally the problem can be formulated as minimising the following cost function,

$$E[\boldsymbol{c}] = \mathcal{L}\left(f(\boldsymbol{x}), \sum_{j=1}^{J} c_j \phi_j(\boldsymbol{x}) \right) + \lambda \|\boldsymbol{c}\|_0 \tag{5.2}$$

where $\mathcal{L}$ is a loss function. The L_0 norm of a vector counts the number of elements that are different from zero (this is a technique referred to as atomic decomposition (Chen et al., 1999)), and λ is a parameter that controls the trade off between sparsity and approximation.

In order to minimise $E[\boldsymbol{c}]$ (given in Equation 5.2), values of the learning function f at all points $\boldsymbol{x}$ needs to be evaluated. In the learning paradigm, in the particular case that,

$$\mathcal{L}\left(f(\boldsymbol{x}), \sum_{j=1}^{J} c_j \phi_j(\boldsymbol{x}_j) \right) = \left\| f(\boldsymbol{x}) - \sum_{j=1}^{J} c_j \phi_j(\boldsymbol{x}) \right\|_2^2 \tag{5.3}$$

the first term in Equation 5.2 is replaced by an empirical one giving,

$$\frac{1}{N} \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{J} c_j \phi_j(\boldsymbol{x}_i) \right)^2 + \lambda \|\boldsymbol{c}\|_0 \tag{5.4}$$

Minimising Equation 5.4 can be used to find sparse approximations in the case that the function f is generated by a function f_0 corrupted by additive noise. In this case the problem can be formulated as finding a solution $\boldsymbol{c}$ to, $f = \Phi \boldsymbol{c} + \eta$, with the smallest number of non-zero elements, where Φ is the matrix with columns that correspond to the elements of the dictionary, and η is the additive noise.

If a probabilistic approach to this learning problem is taken, and the additive noise is assumed to be Gaussian, minimisation of the following function is required,

$$E[\boldsymbol{c}] = \left\| f(\boldsymbol{x}) - \sum_{j=1}^{J} c_j \phi_j(\boldsymbol{x}) \right\|_2^2 + \lambda \|\boldsymbol{c}\|_0 \tag{5.5}$$

However, it can be shown that minimising Equation 5.2 and Equation 5.5 is NP-hard because of the L_0 norm (Chen, 1995). In order to overcome this shortcoming, approximated versions of the loss function (Equation 5.5) have been proposed particularly within the wavelet community.

5.1.1 Wavelet Based Methods

The method of frames (MOF) (or ridge regression)(Daubechies, 1992) selects among all solutions one whose coefficients have minimum L_2 norm. This method is has the attractive feature that it is computationally very attractive. It leads to a quadratic optimisation problem with linear equality constraints, and is the solution of a system of linear equations. However, there are a number of problems with this approach, most notably the method does *not* preserve sparsity. If the underlying probability distribution has a very sparse representation in terms of the dictionary, then the co-efficients found by the MOF are likely to be less sparse.

5.1.2 Matching Pursuit

The matching pursuit algorithm can be described as a general, greedy, sparse function approximation scheme with the squared error loss, that iteratively adds new functions (i.e. basis functions) to the model. Matching pursuit and its variants were developed primarily in the signal processing and wavelets community, but there are many interesting links with the research on kernel based learning algorithms developed in the machine learning community. Connections between a related algorithm (basis pursuit de-noising (Chen, 1995)), which is described in Section 5.1.3), and SVMs have already been reported in Poggio and Girosi (1998). More recently Smola and Schölkopf (2000) have shown connections between matching pursuit, kernel-PCA, sparse kernel feature analysis, and how this type of greedy algorithm can be used to compress the kernel matrix in SVMs thereby allowing modelling of extremely large datasets.

The original matching pursuit algorithm was introduced into the signal processing community as an algorithm that is capable of *“decomposing any signal into a linear expansion* *of waveforms that are selected from a redundant dictionary of functions"* (Mallat and Zhang, 1993). In machine learning matching pursuit algorithms learn a function that is a weighted sum of basis functions, by sequentially appending functions to an initially empty basis, to approximate a target function in the least squares sense. From the definition provided by Mallat and Zhang (1993) if the dictionary of functions is given by functions of the form $K(x, x_i)$, then the expansion has the same form as a kernel based method.

Given a dataset $\mathcal{D}$, consisting of inputs $\boldsymbol{x} \in \mathbb{R}^F$ and output $t \in \mathbb{R}$, and a finite dictionary, $\Phi = \{\phi_1, \ldots, \phi_M\}$, of functions in a Hilbert space $\mathcal{H}$, we are interested in sparse approximations that are of the following form,

$$\hat{f}_J = \sum_{i=1}^{J} \alpha_i g_i \tag{5.6}$$

where $\{\alpha_1, \ldots, \alpha_J\} \in \mathbb{R}^N$, and $\{g_1, \ldots, g_J\} \subset \mathcal{D}$ are chosen to minimise the squared norm of the error. The set $\{g_1, \ldots, g_N\}$ is referred to as the *basis*, and J the number of *basis functions* in the expansion. In general, finding the optimal basis set for a given number, J, of allowed basis functions is in general an NP-complete problem hence, the matching pursuit algorithm proceeds in a greedy iterative manner.

In the signal processing literature, the algorithm is usually stopped when what is termed the *reconstruction error* ($\|R\|^2$) goes below a predefined given threshold. In the machine learning community, the error estimated on an independent test set is used to decide when to stop the algorithm. As Vincent and Bengio (2000) observe, J can be regarded as the primary capacity control parameter of the algorithm. The pseudo-code for the matching pursuit algorithm is given below,

As Vincent and Bengio (2000) observe in the basic version of the matching pursuit algorithm, the set of basis functions that are obtained at every step of the iterative procedure and the associated coefficients $\alpha_{1,\ldots,J}$ are suboptimal. This can often be corrected in a step called *back-fitting* or *back-projection*, and the resulting algorithm is known as Orthogonal Matching Pursuit (OMP) Pati et al. (1993); Mallat and Zhang (1993). Whilst still choosing the optimal set of basis functions as in the basic matching pursuit algorithm, the optimal set of coefficients are recomputed at each step using,

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} \left\| \left(\sum_{k=1}^{J+1} \alpha_k \boldsymbol{\Phi}_{\cdot,\gamma_k} \right) - f \right\|^2 \tag{5.7}$$

As (Vincent and Bengio, 2000) observe this can be interpreted as being similar to linear regression with parameters $\boldsymbol{\alpha}$.

Algorithm 4: Matching Pursuit

```
Input        Dataset $\mathcal{D} = (\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$

Initialise   Construct dictionary of functions $\Phi(\boldsymbol{x}) = \{\phi_1(\boldsymbol{x}), \ldots, \phi_M(\boldsymbol{x})\}$

             Initialise residue vector $R$ and dictionary matrix $\boldsymbol{\Phi}$

             $R \leftarrow \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix}$ and $\boldsymbol{\Phi} \leftarrow \begin{pmatrix} \phi_1(\boldsymbol{x}_1) & \cdots & \phi_M(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ \phi_1(\boldsymbol{x}_N) & \cdots & \phi_M(\boldsymbol{x}_N) \end{pmatrix}$

             Select either:
               a) a desired number of basis functions, $J$, in the expansion
               b) a test dataset to determine algorithm termination

Algorithm    for $j = 1$ to $J$
                 $\gamma_j \leftarrow \arg\max_{k=1,\ldots,N} \left| \frac{\langle \boldsymbol{\Phi}_{\cdot,k}, R\rangle}{\|\boldsymbol{\Phi}_{\cdot,k}\|} \right|$
                 $\alpha_j \leftarrow \frac{\langle \boldsymbol{\Phi}_{\cdot,\gamma_j}, R\rangle}{\|\boldsymbol{\Phi}_{\cdot,\gamma_j}\|^2}$
                 $R \leftarrow R - \alpha_j \boldsymbol{\Phi}_{\cdot,\gamma_j}$
             end

Output       $\boldsymbol{\alpha}, R, \hat{f}_N = \sum_{j=1}^{J} \alpha_j \phi_{\gamma_j}(\boldsymbol{x})$
```

5.1.3 Basis Pursuit Denoising

Mallat and Zhang (1993) have proposed a general method for approximate decomposition, that attempts to avoid the problems associated with the MOF, and addresses the sparsity issue directly. The matching pursuit algorithm starts from an initial approximation, and build up a sequence of sparse approximations iteratively. A similar algorithm has also been proposed by Qian and Chen (1994). An intrinsic feature of the algorithm is that when stopped after a few steps, it yields an approximation using only a few basis functions from the complete dictionary representation. Empirical work by Chen et al. (1995) has shown that if the algorithm chooses an incorrect set of basis functions in the early stages of training, it ends up spending the remaining iterations trying to correct for these mistakes.

To overcome these problems, Chen et al. (1995) have used an L_1 norm as an approximation to the L_0 norm, obtaining an approximation scheme that they refer to as *Basis*

Pursuit De-noising (BPDN) that consists of minimising the following loss function.

$$E[\boldsymbol{c}] = \left\| f(\boldsymbol{x}) - \sum_{j=1}^{J} c_j \phi_j(\boldsymbol{x}) \right\|_2^2 + \lambda \|\boldsymbol{c}\|_1 \tag{5.8}$$

The principle of basis pursuit is to find a representation of the function whose co-efficients have minimal L_1 norm. (Poggio and Girosi, 1998) have recently drawn an interesting parallel between basis pursuit and SVMs. Whereas, the MOF could be solved simply, in contrast basis pursuit requires the solution of a convex optimisation problem, which involves considerably more effort and sophistication.

The linear program can be described as a constrained optimisation problem defined in terms of a variable $\boldsymbol{x} \in \mathbb{R}^M$ by,

$$\min \boldsymbol{c}^T \boldsymbol{x} \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0} \tag{5.9}$$

where $\boldsymbol{c}^T\boldsymbol{x}$ is the objective function, $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ is a collection of equality constraints and $\boldsymbol{x} \geq 0$ is a set of bounds. $\boldsymbol{A}$ is an $M \times N$ where typically $M > N$.

Chen (1995) observes that the basis pursuit problem can be equivalently reformulated as a linear program consistent with Equation 5.9, by making the following translations. Let $\boldsymbol{u}$ and $\boldsymbol{v}$ be N-dimensional vectors. Consider the constrained optimisation problem defined in terms of $\boldsymbol{u}$ and $\boldsymbol{v}$ by,

$$\min \boldsymbol{1}^T\boldsymbol{u} + \boldsymbol{1}^T\boldsymbol{v} \quad \text{subject to} \quad \Phi(\boldsymbol{u} - \boldsymbol{v}) = \boldsymbol{s}, \boldsymbol{u}, \boldsymbol{v} \geq \boldsymbol{0} \tag{5.10}$$

Equation 5.10 can be written as a linear program in the standard form by making the following translations,

$$m \Leftrightarrow 2N, \;\; \boldsymbol{x} \Leftrightarrow (\boldsymbol{u}, \boldsymbol{v}), \;\; c \Leftrightarrow (1, 1), \;\; A \Leftrightarrow (\Phi, -\Phi), \;\; \boldsymbol{b} \Leftrightarrow \boldsymbol{s} \tag{5.11}$$

The non-zero co-efficient of $\boldsymbol{c}$ are associated with N columns of $\boldsymbol{A}$, and hence these columns make up a basis of $\mathbb{R}^d$. Once the basis is identified, the solution is uniquely dictated by the basis. As a result, finding a solution to the linear program is identical to finding the optimal basis.

The connection of basis pursuit with linear programming is useful in several ways (Chen, 1995). For a linear programming problem involving an $M \times N$ matrix, with $M > N$, in the nondegenerate case there will always be exactly N nonzeros. In this case, the nonzero coefficients are associated with N columns of this matrix and these columns make up a basis. The identity of the columns in this optimal basis is not, in general, known in advance. Hence, finding a solution a solution to the linear program is identical to finding this basis, since once the basis is identified the solution is uniquely dictated by the basis.

Despite the linear programming approach, a basis pursuit is computationally expensive to perform because it minimises a global cost function over all dictionary vectors. The matching pursuit algorithm, described in Section 5.1.2, reduces the computational complexity with a greedy strategy.

5.2 Spline Kernels

The use of splines for non-parametric regression, has been a topic of intense research interest within the statistics community for over three decades. Wegman and Wright (1983) observe splines to be an evolution of classical parametric inference, and are seen to bridge the gap between parametric and non-parametric modelling methods. Whilst splines are not parametric in a function form, in many cases they can be written as a linear combination of basis functions that usually have a polynomial representation. The attractiveness of splines stems from their providing a natural and flexible approach to density estimation which have been shown to cope well with data that is sparsely represented.

Interpolating splines (Ahlbery et al., 1968; Amos and Slater, 1969; Anselone and Laurent, 1968) have received a large amount of interest for smoothing noise free data, as a result they have a limited use in a "real world" statistical setting. A number of different approaches to spline fitting methods exist corresponding to different perspectives on how to deal with the noise inherently present in the data. The most popular method (of recent times) parallels the approach described in Chapter 3, of minimising a criterion that depends on a least squares like term plus a regularisation term enforcing smoothness. As Silverman (1985) observes, a major conceptual problem to statisticians with curve estimates like the *spline smoother* is that they are defined implicitly as the solution to a minimisation problem rather than as an explicit formula involving data values. This difficulty can be resolved, at least approximately, by considering how the estimate behaves on large datasets.

As in the Bayesian case a parameter exists that controls the amount of smoothing. Changing the smoothing parameter changes the amount of smoothing applied generally, as such its correct determination is of central importance. Several methods have been proposed for selecting the smoothing parameter in splines. The most notable perhaps is the use of cross validation (Stone, 1974). The basic principle of cross validation is to leave the data points out one at a time, and to choose that value of the regularisation parameter under which the missing data points are best predicted by the remainder of the data. Craven and Wahba (1979) suggest the use of a related criterion termed *generalised* cross validation which uses a weighted least squares cross validation function, where the weights are chosen to reflect unequally spaced data and other data dependent effects.

The idea of viewing non-parametric density estimation in a Bayesian context has been proposed by Whittle (1958) and Silverman (1982). It is perhaps natural to look at the problem in a Bayesian framework, since the decision of how much to smooth corresponds to some sort of prior information. Bayesian models previously suggested in connection with non-parametric smoothing (Kimeldorf and Wahba, 1971; Good and Gaskins, 1971) have involved inference in infinite dimensional spaces. Apart from causing conceptual difficulties, the use of an infinite dimensional formulation leads to paradoxes such as the one alluded to by Wahba (1983); although the intention is to choose among curves for which the regularisation term is finite, the posterior distribution is entirely concentrated *outside* the space of such smooth curves.

A natural extension of the spline smoothing approach, that is of central importance to kernel based methods, is to devise a robust version of the procedure, by replacing the least squares error criterion by a different function of the errors. The new function would be a convex function which is less rapidly increasing than x^2. Minimising this loss function and associated regularisation term, would give a smoothing spline that is robust against or resistant to outliers in the data. This idea has been discussed by Lenth (1977), Huber (1981) and Cox (1983). Huber (1981) observes that minimising a convex loss function may be carried out in practice by an iterative scheme, where a sequence of functionals defined by a weighted sum of squares are minimised successively for weights and data points which are modified at each stage. The basic ideas of iteratively weighted linear regression (Green, 1984) carry over to the spline smoothing case and have been explored and developed by O'Sullivan (1983).

An attractive property of kernel-based approaches is that many functions commonly employed within modelling have kernels that satisfy Mercer's theorem (see Section 4.4.1). Gaussian Radial Basis Function kernels have been successfully deployed in kernel methods. However, whilst they have some attractive properties from a regularisation perspective they are poor at modelling functions with different degrees of smoothness, and require the determination of an additional smoothing parameter. Multi-Layer Perceptron (MLP) kernels, using a set of sigmoidal functions, have also been used. However, the MLP kernel is only positive definite for particular values of its two controlling parameters, making deployment more difficult. Polynomial kernels have often been used and are cheap to compute. Their disadvantage is that in an ANOVA framework a high order polynomial will be required to model arbitrary functions. Splines are an attractive choice for modelling (Wahba, 1990a) due to their ability to approximate arbitrary functions. Many types of splines have kernel representations, such as odd order B-splines and infinite splines. B-splines have been used in other modelling approaches and are favourable when a rule-base interpretation is desired (Brown and Harris, 1994). However, whilst they can have some computational advantages, the regularisation operator corresponding to a B-spline kernel representation has some weaknesses (Smola, 1998). This has been observed experimentally by the production of models with a tendency to oscillation (Gunn, 1998). An infinite spline incorporates the flexibility of a spline approach without the oscillation problem associated with B-splines, and this motivates it use within an ANOVA framework. Another advantage of the infinite spline kernel is that is has no scale, and therefore no associated scale parameter to determine. This is of great advantage in any interpretable kernel technique, since the ANOVA decomposition would introduce a multitude of such parameters which would need to be determined. The first order infinite spline kernel, which passes through the origin, is defined on the interval $[0, \infty)$ by,

$$k_{spline}(u, v) = \int_0^\infty (u - \tau)_+ (v - \tau)_+ d\tau, \tag{5.12}$$

where $(x)_+$ is equal to the positive part of x. The solution has the form of a piece-wise cubic polynomial,

$$k_{spline}(u, v) = uv + \tfrac{1}{2}(u + v)\min(u, v) - \tfrac{1}{6}(\min(u, v))^3, \tag{5.13}$$

and therefore the form of the SVM solution is a piecewise cubic with knots located at a subset of the data points. Multivariate spline kernels obtained from (5.13) will produce a lattice of piecewise multi-cubic functions.

5.2.1 ANOVA Decomposition Kernels

A number of approximation and learning techniques can be studied in the framework of regularisation theory and RKHS. For example, starting from a reproducing kernel, kernels can be constructed that correspond to tensor products of the original RKHS. It is also possible to construct the additive sum of several RKHS in terms of a Mercer kernel.

Consider the case of tensor product splines, in which the form of the kernel is given by,

$$K(\boldsymbol{u}, \boldsymbol{v}) = \prod_{j=1}^{d} k(u^j, v^j) \tag{5.14}$$

where x^j is the jth co-ordinate of the vector $\boldsymbol{x}$, and k is a positive definite function with one dimensional input vectors. The solution to the learning problem then becomes,

$$f(\boldsymbol{x}) = \sum_i w_i \prod_{j=1}^{d} k(x_i^j, x^j) \tag{5.15}$$

Hence, tensor product splines can be obtained by choosing kernels of the form given by Equation 5.14. In the particular case that the kernel is of an additive form given by,

$$K(\boldsymbol{u}, \boldsymbol{v}) = \sum_{j=1}^{d} k(u^j, v^j) \tag{5.16}$$

as in the case for additive spline models, the solution to the learning problem then becomes,

$$f(\boldsymbol{x}) = \sum_i w_i \left(\sum_{j=1}^{d} k(x_i^j, x^j) \right) = \sum_{j=1}^{d} \left(\sum_i w_i k(x_i^j, x^j) \right) = \sum_{j=1}^{d} f_j(x^j) \tag{5.17}$$

Hence, a set of additive approximations of the form,

$$f(\boldsymbol{x}) = \sum_{j=1}^{d} f_j(x^j) \tag{5.18}$$

can be obtained.

The motivation for an additive model representation was provided in Chapter 2, and is stimulated by the work of Stone (1985) and Buja et al. (1989). The additive model methodology models the N-dimensional regression function f as the sum of lower dimensional functions,

$$f(\boldsymbol{x}) = g_1(\boldsymbol{x}) + g_2(\boldsymbol{x}) + \cdots + g_p(\boldsymbol{x}) \tag{5.19}$$

where the g_j have dimensionality less than N. The aim of the additive model methodology is to bypass the difficulty associated with the curse of dimensionality (Bellman, 1961). The methodology achieves its goal by reducing the dimensionality of the regression function. The majority of attention has focused on the following special case of model 5.19,

$$f(x_1, \ldots, x_N) = f_0 + f_1(x_1) + \cdots + f_N(x_N) \tag{5.20}$$

where f_0 is a constant, and the f_j are univariate functions satisfying,

$$f_j(0) = 0 \quad \text{for} \quad 1 \leq j \leq N \tag{5.21}$$

Although the above additive model, given by Equation 5.20, overcomes successfully the difficulty caused by the curse of dimensionality, it suffers approximation errors in using an additive function to model the N-dimensional function f. As Chen (1993) observes it would seem that not enough attention has been directed to more general cases of Equation 5.19, where the dimensionality of the terms in the sum is larger than one. As such an interactive model can be considered.

Consider f to be an arbitrary two-dimensional integrable function. Let,

$$\begin{aligned} f_0 &= f(0,0) \\ f_1(x_1) &= \int f(x_1,x_2)dx_2 - f_0 \\ f_2(x_2) &= \int f(x_1,x_2)dx_1 - f_0 \\ f_{12}(x_1,x_2) = f(x_1,x_2) &- \int f(x_1,x_2)dx_2 - \int f(x_1,x_2)dx_1 + f_0 \end{aligned}$$

then

$$f(x_1,x_2) = f_0 + f_1(x_1) + f_2(x_2) + f_{12}(x_1,x_2)$$

In general, a N-dimensional function f can be decomposed as,

$$f(x_1,\ldots,x_N) = f_0 + \sum_{i=1}^{N} f_i(x_i) + \sum_{i<j} f_{ij}(x_i,x_j) + \cdots + f_{1,\ldots,N}(x_1,\ldots,x_N) \tag{5.22}$$

where the term f_0 is a constant and the other components are integrated to zero with respect to any one of their arguments. The decomposition above can be viewed as a functional version of the statistical methodology ANalysis Of VAriance (ANOVA). A problem arises when the order of the interactions increases, the difficulty associated with the curse of dimensionality resurfaces. As such models of lower order interactions are desirable. Much of the recent work on ANOVA splines has focused on interactive functions which are finite sums of products of appropriately smooth univariate functions in a certain function space. In this thesis, ANOVA spline kernels are deployed since these interactive models are considerably more flexible than traditional additive models whilst retaining all the advantages of additive models, namely the ability to overcome the curse of dimensionality and their interpretability. The extra feature also is that it has no scale, and therefore no associated parameter to determine.

5.3 Interpretable Sparse Approximations

In this work we introduce interpretability, or *transparency*, by producing a parsimonious model, which has a sparse structural representation, but is flexible enough to avoid problems of model mismatch. The transparency is beneficial in that it enables the model to be validated and interpreted. Features that aid transparency are input selection and ways of decomposing the model into smaller more interpretable pieces that can be easily visualised. To address this issue a modified kernel model of the form is introduced,

$$f(x) = \sum_{i=1}^{N} \alpha_i \sum_j c_j K_j(x^i, x), \quad c_j \geq 0, \tag{5.23}$$

where the kernel is replaced by a weighted, c_j, linear sum of kernels, K_j. Transparency can then be introduced by a careful choice of the additive kernels, K_j and by making their weighting coefficients, c_j, sparse. In this thesis the integration of an ANOVA (ANalysis Of Variance) representation to provide a transparent approach to modelling is focused upon. ANOVA kernels (Stitson et al., 1999) have previously been used with SVMs, with promising performance. However, the difference here is to develop a technique that will select a sparse ANOVA kernel producing strong transparency. The ANOVA representation is motivated by the decomposition of a function into additive components, with the goal of representing the function by a subset of the terms from this expansion. A function may be decomposed into

$$f(x) = f_0 + \sum_{i}^{d} f_i(x_i) + \sum_{i<j}^{d} f_{i\otimes j}(x_i, x_j) + \cdots + f_{1\otimes 2\otimes \ldots \otimes d}(x), \tag{5.24}$$

where d is the number of inputs, f_0 represents the bias and the other terms represent the univariate, bivariate, etc., components. The notation x_i denotes the scalar value of input i. The basis functions are semi-local and are similar to the approaches used by Friedman (Friedman, 1991) in the Multivariate Adaptive Regression Splines (MARS) technique and in the Adaptive Spline Modelling of Observational Data (ASMOD) technique (Kavli and Weyer, 1995). The additive representation is advantageous when the higher order terms can be ignored, so that the resulting model is represented by a small subset of the ANOVA terms, which may be easily visualised. This produces a transparent model, in contrast to the majority of neural network models, providing the modeller with structural knowledge that can be used for both validation and model interpretation. Due to the curse of dimensionality (Bellman, 1961), an exhaustive search of the possible model structures is demanding. Even in the highly restrictive scenario, that the solution is a weighted linear combination of *fixed* basis functions, the parameter space has size 2^d. Extension to flexible basis functions, which is required for typical modelling, will only compound this dimension. Accordingly, greedy methods are typically used. ASMOD employs an evolutionary strategy to search the model space using a forward selection/backward elimination algorithm to select suitable refinements to a model. The MARS algorithm employs a recursive partitioning procedure to search the model space for an appropriate model. The drawback with both approaches is that they can become entrapped by local minima, due to the greedy nature of their search algorithms. A problem with deploying additive models in advanced flexible non-linear modelling methods is that they cannot provide a transparent model if the phenomenon being modelled contains high dimensional interactions. One possibility is to enforce transparency by constraining the order of possible interactions (e.g. restriction to univariate and bivariate terms only), providing a coarse, but interpretable structure, at the expense of structural integrity.

Using the ideas of sparse function approximation described above, the following section is concerned with describing their deployment in ANOVA spline kernels, within kernel based methods to obtain sparse and interpretable solutions without sacrificing generalisation performance (Gunn and Kandola, 2000). This is achieved by combining the ideas described in Section 2.1.1 and Section 2.1.2 to employ two forms of regularisation: a L_1-norm based structural regulariser as described in Section 5.7, and a L_2 norm weight decay regulariser to control smoothness. To address some of the difficulties associated with the preceding methods, such as model mismatch, poor interpretability and poor generalisation a new additive sparse kernel method is proposed. An additive sparse kernel model extends a standard kernel model by replacing the kernel with a weighted linear sum of kernels,

$$f(x) = \sum_{i=1}^{N} \alpha_i \sum_{j=1}^{m} c_j K_j(x^i, x), \quad c_j \geq 0, \tag{5.25}$$

where K_j are positive definite functions and where the positivity constraints on the kernel coefficients, c_j, ensure that the complete kernel function is positive definite. Here, the term sparse refers to sparseness in the kernel coefficients c_j rather than the usual sparseness in the multipliers, α_i; sparseness in these multipliers can still be obtained by employing an appropriate loss function. A conventional kernel model regulariser will not enforce sparsity in the kernel coefficients and hence a more complex regulariser is required. The goal in selecting a sparse representation is to minimise the number of non-zero coefficients, c_i. This can be achieved with a p-norm on the kernel coefficients. As p increases the solution becomes less sparse and the computational complexity of the resulting optimisation problem is relaxed. Ideally a value of $p = 0$, which counts the number of terms in the expansion is attractive. This case is employed in the atomic decomposition of (Chen, 1995), but it results in a computationally hard combinatorial optimisation problem. Alternatively choosing a value of $p = 2$ produces a straightforward optimisation problem. This case is referred to as the method of frames or ridge regression, but crucially the sparseness within the expansion is now lost. A good compromise occurs when $p = 1$ producing a sparse solution, with a practical implementation. This penalty function has successfully been used in basis-pursuit de-noising (Chen, 1995). To enforce sparsity in the kernel expansion we consider a regularised cost functional of the form

$$\Phi(\boldsymbol{\alpha}, \boldsymbol{c}) = L(y, K(\boldsymbol{c})\boldsymbol{\alpha}) + \lambda_\alpha \|\boldsymbol{\alpha}\|^2_{K(\boldsymbol{c})} + \lambda_c \|\boldsymbol{c}\|_1, \quad c_i \geq 0, \lambda_\alpha, \lambda_c > 0 \tag{5.26}$$

where L is the loss function, and λ_α, λ_c are regularisation parameters controlling the smoothness and sparsity of the kernel expansion respectively.

The direct solution of this problem is non-trivial, so an iterative method is introduced, whereby we solve two separate sub-problems: $\min_\alpha \Phi$ with $\boldsymbol{c}$ fixed; $\min_{\boldsymbol{c}} \Phi$ with $\boldsymbol{\alpha}$ fixed.

The solution for a quadratic loss, $L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = (\boldsymbol{y} - \hat{\boldsymbol{y}})^T(\boldsymbol{y} - \hat{\boldsymbol{y}})$, is given by

$$\begin{aligned}
\Phi(\boldsymbol{\alpha}, \boldsymbol{c}) &= \left\| \boldsymbol{y} - \sum_i c_i \boldsymbol{K}_i \boldsymbol{\alpha} \right\|_2^2 + \lambda_\alpha \sum_i c_i \boldsymbol{\alpha}^T \boldsymbol{K}_i \boldsymbol{\alpha} + \lambda_c \sum_i c_i, \quad \forall_p\, c_p \geq 0. \\
\boldsymbol{\alpha}^* &= \arg\min_{\boldsymbol{\alpha}} \; \boldsymbol{\alpha}^T (\sum_i \sum_j c_i c_j \boldsymbol{K}_i \boldsymbol{K}_j + \lambda_\alpha \sum_k c_k \boldsymbol{K}_k) \boldsymbol{\alpha} \\
&\quad -(2\boldsymbol{y}^T \sum_l c_l \boldsymbol{K}_l) \boldsymbol{\alpha} \\
\boldsymbol{c}^* &= \arg\min_{\boldsymbol{c}} \; \sum_i \sum_j c_i c_j (\boldsymbol{\alpha}^T \boldsymbol{K}_i \boldsymbol{K}_j \boldsymbol{\alpha}) \\
&\quad + \sum_k c_k (\lambda_\alpha \boldsymbol{\alpha}^T \boldsymbol{K}_k \boldsymbol{\alpha} + \lambda_c - 2\boldsymbol{y}^T \boldsymbol{K}_k \boldsymbol{\alpha}), \quad \forall_p\, c_p \geq 0,
\end{aligned}$$

where y and $\hat{y}$ are vectors of target and predicted values respectively. The solution for an ϵ-Insensitive Loss, $L(y, \hat{y}) = \sum_i \max(0, |y_i - \hat{y}_i| - \epsilon)$ by,

$$\begin{aligned}
\Phi(\boldsymbol{\alpha}, \boldsymbol{c}) &= \left\| y - \sum_i c_i \boldsymbol{K}_i \boldsymbol{\alpha} \right\|_{1,\epsilon} + \lambda_\alpha \sum_i c_i \boldsymbol{\alpha}^T \boldsymbol{K}_i \boldsymbol{\alpha} + \lambda_c \sum_i c_i, \quad \forall_p\, c_p \geq 0. \\
\boldsymbol{\alpha}^* &= \arg \min_{\boldsymbol{\alpha}=\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-} \; (\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-)^T (\lambda_\alpha \sum_k c_k \boldsymbol{K}_k)(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-) \\
&\quad - \sum_i (\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-) y_i + \sum_i (\boldsymbol{\alpha}^+ + \boldsymbol{\alpha}^-)\epsilon, \\
&\quad \forall_i\, 0 \leq \boldsymbol{\alpha}_i^+, \boldsymbol{\alpha}_i^- \leq \frac{1}{2\lambda_\alpha}. \\
\boldsymbol{c}^* &= \arg \min_{\boldsymbol{c}, \boldsymbol{\zeta}^+, \boldsymbol{\zeta}^-} \; \sum_i (\boldsymbol{\zeta}_i^+ + \boldsymbol{\zeta}_i^-) + \sum_j c_j (\lambda_\alpha \boldsymbol{\alpha}^T \boldsymbol{K}_j \boldsymbol{\alpha} + \lambda_c), \\
&\qquad \forall_{i,j}\, c_j \geq 0, \zeta_i^+, \zeta_i^- \geq 0, -\boldsymbol{\zeta}^- - \boldsymbol{\eta} \leq \sum_k c_k \boldsymbol{K}_k \boldsymbol{\alpha} \leq \boldsymbol{\zeta}^+ + \boldsymbol{\eta}.
\end{aligned}$$

where ζ_i^+ and ζ_i^- are slack variables. An attraction of this iterative technique is that it decomposes the problem into two simple convex optimisation problems. In the quadratic loss case the solution for $\boldsymbol{\alpha}^*$ is given by simple matrix inversion, and for $\boldsymbol{c}^*$ by a bound constrained quadratic program. In the ϵ-insensitive case the solution for $\boldsymbol{\alpha}^*$ is given by a box constrained quadratic program, and for $\boldsymbol{c}^*$ by a bound constrained linear program with linear constraints. Consequently, they can all be solved readily using a standard quadratic programming optimiser (Mészáros, 1998). A similarity can be drawn between this approach and Bayesian methods (MacKay, 1995) that employ a two stage iterative procedure, a parameter update and 'hyperparameter' update. However, unlike most Bayesian methods, the update stages consist of convex optimisation problems.

If λ_α and λ_c are known the solution can be obtained by,

$$\begin{array}{ll} \text{Initialise:} & \boldsymbol{\alpha}_0^* = \arg\min_{\boldsymbol{\alpha}} \Phi(\boldsymbol{\alpha}, \boldsymbol{c}_0^*), \quad \boldsymbol{c}_0^* = \mathbf{1} \\ \text{Iteration:} & \begin{array}{ll} \text{(a)} & \boldsymbol{c}_{i+1}^* = \arg\min_{\boldsymbol{c}} \Phi(\boldsymbol{\alpha}_i^*, \boldsymbol{c}) \\ \text{(b)} & \boldsymbol{\alpha}_{i+1}^* = \arg\min_{\boldsymbol{\alpha}} \Phi(\boldsymbol{\alpha}, \boldsymbol{c}_{i+1}^*). \end{array} \end{array}$$

In the quadratic case the second order partial derivatives with respect to $\boldsymbol{\alpha}$ and $\boldsymbol{c}$ are always positive ensuring that every slice is convex. This fact combined with the knowledge that the solution is finite in $\boldsymbol{\alpha}$ and $\boldsymbol{c}$ should ensure convergence to the global minimum. A similar result should be obtainable for the ϵ-insensitive loss function. The convergence properties of this algorithm will be studied in future work. In practice the situation is more complicated since λ_α and λ_c will not be known but will need to be estimated. Intuitively, both λ_c and λ_α should initially be set large and reduced gradually; reducing λ_α too quickly will over smooth the space making the sparse selection harder; reducing λ_c too quickly will tend to produce an over-sparse model. To provide a workable solution the method used in this thesis uses an initialisation step and one iteration. In the initilisation step and part (b) of the iteration, λ_c does not enter the optimisation and as such does not need to be determined; λ_α can be determined using cross-validation. The difficult part is determining the parameters in part (a) of the iteration. A possible method could fix λ_α at the value used in the initialisation step and select λ_c to obtain a comparable loss to that of the initialisation step. However, the method chosen, which was based on the best empirical performance, was to set $\lambda_\alpha = 0$ and to select λ_c such that the loss was equal to that of the validation error in the initialisation step. Alternative methods for determining these parameters will be investigated in future work. In the next section a particular class of sparse additive kernel model is introduced with some attractive transparency properties.

5.4 SUpport vector Parsimonious ANOVA (SUPANOVA) Technique

The SUPANOVA technique is designed to select a parsimonious model representation by selecting a small set of terms from the complete ANOVA representation (5.24). The technique is an additive kernel model, (5.25) with a particular choice of ANOVA kernel that can be employed in the sparse kernel method described in the previous section. This section considers some possibilities for ANOVA kernel models. The following theory is based upon Reproducing Kernel Hilbert Spaces (RKHS) (Aronszajn, 1950; Wahba, 1990b). If K is a symmetric positive definite function, which satisfies Mercer's Conditions, then the kernel represents a legitimate inner product in feature space and it may be deployed within (5.25). The following two theorems (Aronszajn, 1950) are required in proving that ANOVA kernels satisfy Mercer's Conditions.

Theorem 1: If k_1 and k_2 are both positive definite functions then so is $k_1 + k_2$

Theorem 2: If k_1 and k_2 are both positive definite functions then so is $k_1 \otimes k_2$

It follows from theorem 2 that multidimensional kernels can be obtained by forming tensor products of univariate kernels. A multivariate ANOVA kernel is given by the tensor product of a univariate kernel plus a bias term,

$$\begin{aligned} K_{ANOVA}(u, v) &= \prod_{i=1}^{d} (1 + k(u_i, v_i)) \\ &= 1 + \sum_{i}^{d} k(u_i, v_i) + \sum_{i<j}^{d} k(u_i, v_i) k(u_j, v_j) + \\ &\quad \cdots + \prod_{i=1}^{d} k(u_i, v_i). \end{aligned} \tag{5.27}$$

It follows from theorems 1 and 2 that if k is a valid kernel then so is K_{ANOVA}. Considering (5.27) it is evident that the tensor product produces the ANOVA terms of (5.30), producing a flexible model. Another consequence of theorems 1 and 2 is that each of the additive terms in the expansion (5.27) is also positive definite, and hence a valid kernel in its own right. This enables partial forms of (5.27) to be used as valid kernels, and this is the method employed within the SUPANOVA technique to produce parsimonious kernels. The choice of univariate kernel, k, will control the form of the final model. For simplicity, we shall restrict ourselves to the case where the same kernel is used for each dimension, although different univariate kernels could be deployed.

Using a complete ANOVA kernel (5.27) has drawbacks when it comes to interpretation of the model, due to the large number of terms within the expansion. To introduce enhanced transparency we employ a parsimonious ANOVA kernel. Considering the expansion of (5.27) an additional set of positive coefficients, c_i, are introduced,

$$\begin{aligned} K_{ANOVA}(u, v) &= c_0 + \sum_{i}^{d} c_i k(u_i, u_i) + \sum_{i<j}^{d} c_{i,j} k(u_i, u_i) k(u_j, u_j) + \\ &\quad \cdots + c_{1,2,\ldots,d} \prod_{i=1}^{d} k(u_i, u_i). \end{aligned} \tag{5.28}$$

Consequently the resulting kernel is a weighted linear sum of kernels, and a parsimonious model solution can be obtained by using the method of the previous section.

Since the univariate ANOVA term is constrained to pass through the origin, bivariate and higher order terms will be constrained to be zero along their axes. Consequently the parsimonious model will not simply consist of the single highest order ANOVA term, but will favour low order terms in preference to high order terms. The ANOVA terms in the parsimonious model can be recovered from the final SVM expansion. For example, the univariate terms are given by,

$$f_g(x) = c_g \sum_{i=1}^{N} \alpha_i k(x_g^i, x_g), \tag{5.29}$$

and the bivariate terms are given by,

$$f_{g \otimes h}(x) = c_{g,h} \sum_{i=1}^{N} \alpha_i k(x_g^i, x_g) k(x_h^i, x_h), \tag{5.30}$$

where α_i are the Lagrange multipliers obtained from the complete ANOVA kernel solution, and N is the number of data points in the dataset $\mathcal{D}$. However, the computation required to solve the optimisation problem is extremely demanding due to the combinatorial nature of the problem and the curse of dimensionality (Bellman, 1961) associated with the full ANOVA expansion. To overcome this problem the ANOVA expansion can be truncated to simplify the problem, since if transparency is to be obtained the selected terms should be of low order. This technique contrasts with other parsimonious techniques, such as MARS and ASMOD, in that it aims to find a full model and sub-select the significant terms. The drawback with the MARS and ASMOD approaches is that they are local, and can suffer from entrapment in local minima within the construction process. Additionally, they may not be strictly well-posed. A further attraction of the SUPANOVA technique is that it decomposes the problem into three simple convex optimisation problems. An important issue is the form of solution produced when highly correlated inputs exist. The combination of the regularisers, (5.26) will produce a model that is distributed for two or more identical inputs; if a $\|\boldsymbol{c}\|_0$ regulariser was used the model would not be distributed. In the case when the inputs are only highly correlated, the technique will produce a sparse model, and therefore a simple correlation test could be employed to identify the limiting case.

5.4.1 Bishop's Dataset: Evaluation of the SUPANOVA algorithm

To assess the performance of the SUPANOVA algorithm (Gunn and Kandola, 2000) within a kernel based method, the algorithm was run on Bishop's toy problem described in Chapter 3. The selected inputs across the ten dataset partitions are represented in Table 5.1.

From the selection of the input variables, the SUPANOVA algorithm is able to select the most important input variable x_1 across all ten data partitions. The irrelevant input variables x_2 and x_3 are not selected with any consistency reflecting the potential of the technique to select relevant inputs. Interestingly however, the SUPANOVA method selects the bivariate ANOVA term $x_1 x_2$ as being relevant for all datasets. This appears surprising given that the output of Bishop's toy problem is dependent on input x_1.

	Dataset									
Components	1	2	3	4	5	6	7	8	9	10
x_1	×	×	×	×	×	×	×	×	×	×
x_2	–	–	–	–	–	×	–	–	–	×
x_3	–	–	–	–	–	–	–	–	–	–
$x_1 \otimes x_2$	×	×	×	×	×	×	×	×	×	×
$x_1 \otimes x_3$	–	–	–	–	–	–	–	–	–	–
$x_2 \otimes x_3$	–	–	–	–	–	–	–	–	–	–
$x_1 \otimes x_2 \otimes x_3$	–	–	–	–	–	–	–	–	–	–

Table 5.1: Input Selection via ANOVA decomposition in a Support Vector Machine using Bishop's dataset.

Loss Function		Estimated Generalisation Error					
Training	Testing	Stage I		Stage III		Linear Model	
Quadratic	Quadratic	4.84	(1.20)	2.22	(2.54)	6.53	(3.60)
ϵ-insensitive	ϵ-insensitive	0.93	(0.04)	0.47	(0.11)	1.17	(0.08)
ϵ-insensitive	Quadratic	4.88	(1.59)	2.32	(2.24)	6.61	(3.79)

Table 5.2: SUPANOVA Results for the Additive Data Set ($\epsilon = 1.0$). Quoted values are for the mean (and variance) of the estimated generalisation error.

5.4.2 Friedman's Dataset: Evaluation of the SUPANOVA algorithm

To assess the performance of the SUPANOVA algorithm (Gunn and Kandola, 2000) within a kernel based method, the algorithm was run on Friedman's toy problem described in Chapter 3. The results are presented in Table 5.2 for the random dataset partitions.

From the selection of the input variables, the approach is able to distinguish between the relevant and the irrelevant inputs with a high degree of accuracy across the ten different data partitions. Given the unique representation of the ANOVA spline kernel and in particular how it deals with combinations of inputs, it is able to select the x_1x_2 term that alluded the ARD covariance kernel that is used extensively in the Gaussian process community and the Bayesian neural network ARD scheme. An added advantage of the ANOVA spline representation is that the individual terms that are selected can be visualised based on using all of the data.

Figure 5.1 illustrates one of the 100 models, obtained from the SUPANOVA technique. It can be seen that it has selected 7 interaction terms (bias, five univariates, and one bivariate) from a possible 1024 terms. Table 5.2 demonstrates that the difference in the mean of the estimated generalisation error between a full ANOVA model is twice as high as the error for the parsimonious ANOVA model. These results were corroborated by the results using the $\epsilon-$Insensitive function. Comparing the two different loss functions shows that, for this particular data-set, there is very little performance difference. The

Terms	Quadratic	ϵ-insensitive	"Difference"
bias	50	50	0.00
x_1	50	50	0.00
x_2	50	50	0.00
x_3	34	32	0.16
x_4	50	50	0.00
x_5	50	50	0.00
$x_1 \otimes x_2$	49	50	0.02
$x_3 \otimes x_8$	5	3	0.08
$x_3 \otimes x_9$	4	5	0.10
$x_3 \otimes x_{10}$	0	5	0.10
$x_4 \otimes x_5$	1	0	0.02

TABLE 5.3: SUPANOVA terms selected for Friedman's Additive Data Set. ($\epsilon = 1.0$)

	Dataset									
Components	1	2	3	4	5	6	7	8	9	10
x_1	×	×	×	×	×	×	×	×	×	×
x_2	×	×	×	×	×	×	×	×	×	×
x_3	×	×	–	–	–	×	×	×	×	×
x_4	×	×	×	×	×	×	×	×	×	×
x_5	×	×	×	×	×	×	×	×	×	×
$x_1 \otimes x_2$	×	×	×	×	×	×	–	×	×	×

TABLE 5.4: Input Selection via ANOVA decomposition in a Support Vector Machine using Friedman's dataset.

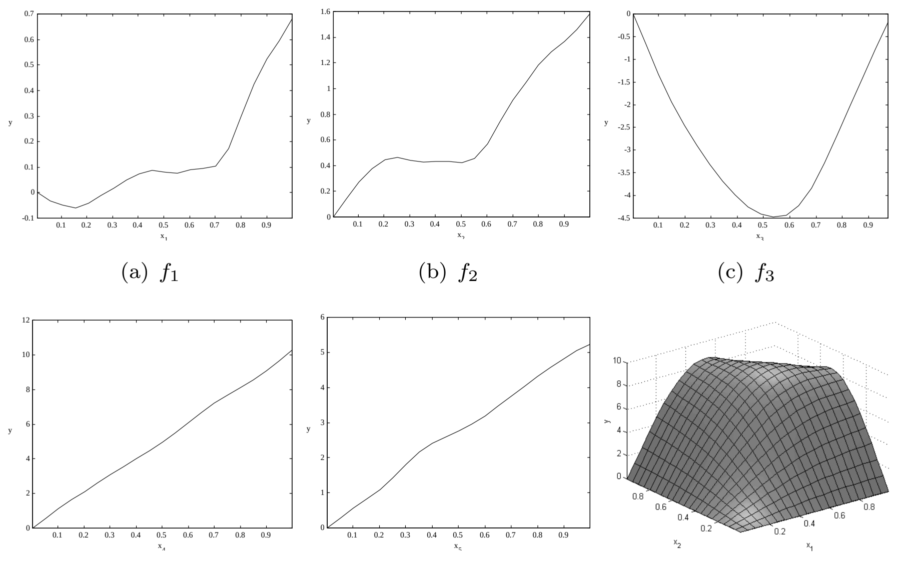
ANOVA terms selected by the 100 models are shown in Table 5.3 showing a high consistency, demonstrating the potential of the technique. One point of interest is brought out by the results. The spline kernel employed will produce ANOVA terms which are zero at the origin, and hence bivariate terms will be zero along both axes, which is illustrated by the $x_1 \otimes x_2$ term in Figure 5.1. Accordingly, the additive model should not require the univariate terms x_1, x_2 to model the data generated by Friedman's equation.

5.4.2.1 Simplified Additive Data Modelling

To investigate the inclusion of the two univariate terms x_1, x_2, further the generating function was simplified to a two input function,

$$f(\mathbf{x}) = 10\sin(\pi x_1 x_2). \tag{5.31}$$

A 15 by 15 grid of points on $[0,1] \times [0,1]$ were used to induce a new model. In this experiment the regularisation parameter λ_α, was controlled manually, and varied over a wide range. The result for a larger value of regularisation is shown in Figure 5.2. It is evident that the technique has modelled the function using both the univariate and


(a) f_1 (b) f_2 (c) f_3

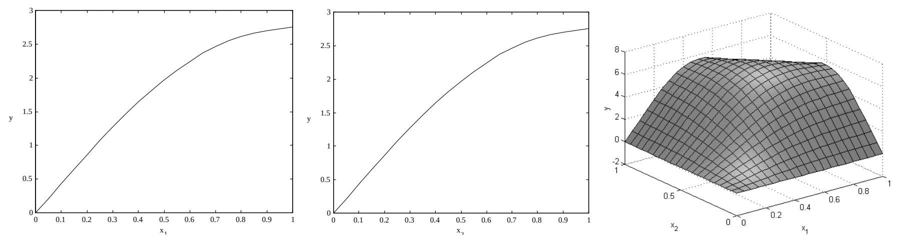
(d) f_4 (e) f_5 (f) $f_{1\otimes 2}$

FIGURE 5.1: Visualisation of the selected ANOVA terms using a Quadratic additive model (1 of 50) when applied to the Additive Dataset.

bivariate terms. This is in contrast to a technique that uses a small amount of regularisation, in which the function is entirely modelled by the bivariate term. This behaviour can be explained by considering the way the regularisation term penalises the spline basis functions. The regularisation term is penalising the square of the amplitude of the basis functions. Hence, as this term becomes more significant the optimisation problem can attain a lower value by decomposing the single bivariate term into a combination of bivariate and univariate ANOVA terms. In the initialisation stage where the ANOVA model space is large, it will be necessary to employ a significant amount of regularisation to control the capacity of the flexible model. Therefore, this behaviour will be common when a ridge regression type regulariser is employed. This problem could be addressed by considering alternative regularisation operators/kernels. It also explains the fact that the quadratic term was extracted less consistently than the other terms, which is evident from Table 5.3. However, its consequence will be to introduce ANOVA terms that are factors of a main effect and as such this is not an overriding problem, since the main effect terms typically have a low dimension. In the case when the main effect term has a high dimension, transparency has already been lost.

5.5 Bayesian Interpretable Sparse Kernel Inference Technique (BISKIT)

Given that SVMs can be interpreted probabilistically (see Chapter 4), the methods described in Chapter 3 can be used to determine the hyperparameters associated with the kernel functions. In an alternative approach to the SUPANOVA algorithm, the



(a) f_1 (b) f_2 (c) $f_{1\otimes 2}$

FIGURE 5.2: Visualisation of ANOVA terms when deploying Regularisation effects ($C = 10$).

problem of selecting the relevant subkernels of the ANOVA spline kernel expansion is cast into a Bayesian learning framework. In an alternative approach, the inference problem is placed in a Bayesian framework with the hyperparameters for the ANOVA expansion, c_i, being optimised using variational learning (Kandola et al., 2000). The method described is closely related to the variational relevance vector machine (Bishop and Tipping, 2000) reviewed in Section 3.7.8. Variational learning, despite the limitations described in Chapter 3, has the advantage that the factorisation assumption allows separate priors to be formulated over the parameters of the kernel model, and the hyperparameters associated with each term of the kernel expansion.

The solution to the kernel based model has been shown to correspond to a weighted sum of kernel functions (Vapnik, 1998; Smola, 1998)

$$y(\boldsymbol{x}) = \sum_{j=1}^{N} w_j K_c(\boldsymbol{x}, \boldsymbol{x}_j) \tag{5.32}$$

where K_c is an ANOVA spline kernel defined by,

$$K_c(\boldsymbol{x}, \boldsymbol{x}') = \sum_{d=1}^{J} c_d K_d(\boldsymbol{x}, \boldsymbol{x}') \tag{5.33}$$

where K_d are the sub-kernels associated with the univariate, bivariate and higher order terms of the ANOVA expansion. Assuming that the inputs $\boldsymbol{x}$ of the dataset, $\mathcal{D}$, are independent and identically distributed, the model likelihood can be written as,

$$P(\mathcal{D}|\boldsymbol{w}, \boldsymbol{c}, \beta) = \frac{\beta^{N/2}}{(2\pi)^{N/2}} \exp\left\{-\frac{\beta}{2}(\boldsymbol{y} - \boldsymbol{K}_c\boldsymbol{w})^T(\boldsymbol{y} - \boldsymbol{K}_c\boldsymbol{w})\right\} \tag{5.34}$$

with unknown noise variance β. The prior over the model weights $\boldsymbol{w}$ is defined by,

$$P(\boldsymbol{w}|\boldsymbol{c}, \lambda) = \frac{\lambda^{N/2}}{(2\pi)^{N/2}|\boldsymbol{K}_c|^{\frac{1}{2}}} \exp\left\{-\frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{K}_c\boldsymbol{w}\right\} \tag{5.35}$$

with model capacity control term λ. The hyperparameters β and λ are given a Gamma prior distribution since this is the associated conjugate prior (see Section 3.6.1),

$$P(\beta) = \Gamma(\beta|a_\beta, b_\beta) \quad \text{and} \quad P(\lambda) = \Gamma(\lambda|a_\lambda, b_\lambda) \tag{5.36}$$

where $a_\beta, b_\beta, a_\lambda$, and b_λ are parameters of the Gamma distribution. To ensure positivity of the ANOVA kernel hyperparameters, c_i, they are given a Gamma prior distribution with mean m_d and variance m_d^2,

$$P(c) = \prod_{d=1}^{J} \frac{1}{m_d} \exp\left\{-\frac{c_d}{m_d}\right\} \tag{5.37}$$

The joint distribution of parameters and hyperparameters is then given by,

$$P(\boldsymbol{w}, \boldsymbol{c}, \beta, \lambda|\mathcal{D}) = P(\boldsymbol{w}, \boldsymbol{c}, \beta|\mathcal{D})P(\boldsymbol{w}|\boldsymbol{c}, \lambda)P(\boldsymbol{c})P(\beta)P(\lambda) \tag{5.38}$$

To achieve a complete Bayesian treatment of this learning problem, the ideas relating to variational methods described in Section 3.7.8 are exploited. A single assumption is made in the variational formulation, and that is that the approximate distribution, $Q(\boldsymbol{w}, \boldsymbol{c}, \beta, \lambda|\mathcal{D})$, to the posterior, $P(\boldsymbol{w}, \boldsymbol{c}, \beta, \lambda|\mathcal{D})$, is separable into the following form,

$$Q(\theta) = Q(\boldsymbol{w}, \boldsymbol{c}, \beta, \lambda|\mathcal{D}) = Q_{\boldsymbol{w}}(\boldsymbol{w}) \prod_{j=1}^{J} Q_c(c_j) Q_\beta(\beta) Q_\lambda(\lambda) \tag{5.39}$$

In the work of MacKay (1995) and Bishop and Tipping (2000), the approximate distribution over the kernel hyperparameters is allowed to be a free form distribution, and its optimal choice is determined by exploiting the divergence theorem (described in Section 3.7.8. However, if this approach is adopted in the Bayesian ANOVA inference problem, a difficult optimisation problem results (Kandola and Gunn, 2000). As such, in the approach considered here, the approximating distribution, $Q_{\boldsymbol{c}}(\boldsymbol{c})$, is restricted to be a Gamma distribution with mean n_d and variance n_d^2,

$$Q_{\boldsymbol{c}}(\boldsymbol{c}) = \prod_{d=1}^{J} \frac{1}{n_d} \exp\left\{-\frac{c_d}{n_d}\right\} \tag{5.40}$$

The variational inequality is given by,

$$\begin{aligned}
F(Q) &= \int Q(\theta) \log \frac{P(\mathcal{D}|\boldsymbol{w}, \boldsymbol{c}, \beta)P(\boldsymbol{w}|\boldsymbol{c}, \lambda)P(\boldsymbol{c})P(\beta)P(\lambda)}{Q(\theta)}\, d\theta \\
&= \int Q(\theta) \left[\frac{N}{2} \log\left(\frac{\beta}{2\pi}\right) - \frac{\beta}{2}(\boldsymbol{y} - \boldsymbol{K}_c\boldsymbol{w})^T(\boldsymbol{y} - \boldsymbol{K}_c\boldsymbol{w}) + \frac{N}{2}\log\left(\frac{\lambda}{2\pi}\right) \frac{1}{2}\log|\boldsymbol{K}_c| \right. \\
&- \left. -\frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{K}_c\boldsymbol{w} - \sum_{d=1}^{J}\left(\log m_d + \frac{c_d}{m_d}\right) + \log P(\beta) + \log P(\lambda) - \log Q(\theta)\right] d\theta
\end{aligned}$$

The variational problem is to minimise F with respect to $Q_{\boldsymbol{w}}(\boldsymbol{w})$, $Q_{\boldsymbol{c}}(\boldsymbol{c})$ $Q_\beta(\beta)$, $Q_\lambda(\lambda)$.

Optimisation of $Q_{\boldsymbol{w}}(\boldsymbol{w})$

As a function of $Q_{\boldsymbol{w}}(\boldsymbol{w})$, F can be written as,

$$\begin{aligned} F &= \int Q_{\boldsymbol{w}}(\boldsymbol{w}) \Big[-\frac{\bar{\beta}}{2} \int Q_{\boldsymbol{c}}(\boldsymbol{c})(\boldsymbol{y} - \boldsymbol{K}_c\boldsymbol{w})^T(\boldsymbol{y} - \boldsymbol{K}_c\boldsymbol{w})\, d\boldsymbol{c} \\ &- \frac{\bar{\lambda}}{2}\boldsymbol{w}^T \int Q_{\boldsymbol{c}}(\boldsymbol{c})\boldsymbol{K}_c\, d\boldsymbol{c}\ \boldsymbol{w} - \log Q_{\boldsymbol{w}}(\boldsymbol{w})\Big]\, d\boldsymbol{w} + \text{const.} \end{aligned} \tag{5.41}$$

where

$$\begin{aligned} \bar{\beta} &= \int \beta Q_\beta(\beta)\, d\beta \\ \bar{\lambda} &= \int \lambda Q_\lambda(\lambda)\, d\lambda \end{aligned} \tag{5.42}$$

and we have

$$\begin{aligned} &\int Q_{\boldsymbol{c}}(\boldsymbol{c})(\boldsymbol{y} - \boldsymbol{K}_c\boldsymbol{w})^T(\boldsymbol{y} - \boldsymbol{K}_c\boldsymbol{w})\, d\boldsymbol{c} \\ &\qquad = \boldsymbol{y}^T\boldsymbol{y} - 2\boldsymbol{y}^T \int Q_{\boldsymbol{c}}(\boldsymbol{c})\boldsymbol{K}_c\, d\boldsymbol{c}\ \boldsymbol{w} + \boldsymbol{w}^T \int Q_{\boldsymbol{c}}(\boldsymbol{c})\boldsymbol{K}_c^T\boldsymbol{K}_c\, d\boldsymbol{c}\ \boldsymbol{w} \\ &\qquad = \boldsymbol{y}^T\boldsymbol{y} - 2\boldsymbol{y}^T\widetilde{\boldsymbol{K}}_{1c}\boldsymbol{w} + \boldsymbol{w}^T\widetilde{\boldsymbol{K}}_{2c}\boldsymbol{w} \end{aligned} \tag{5.43}$$

where $\widetilde{\boldsymbol{K}}_{1c} = \sum_{d=1}^J n_d\boldsymbol{K}_d$ and $\widetilde{\boldsymbol{K}}_{2c} = 2\sum_{d=1}^J n_d^2\boldsymbol{K}_d^T\boldsymbol{K}_d + \sum_{d_1\neq d_2}^J n_{d_1}n_{d_2}\boldsymbol{K}_{d_1}^T\boldsymbol{K}_{d_2}$. The best $Q_{\boldsymbol{w}}(\boldsymbol{w})$ will be given by

$$Q_{\boldsymbol{w}}(\boldsymbol{w}) = \frac{1}{Z(\widetilde{\boldsymbol{K}}_{1c}, \widetilde{\boldsymbol{K}}_{2c}, \bar{\beta}, \bar{\lambda})} \exp\left\{ -\frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}_{MP})^T\boldsymbol{\Sigma}_{MP}^{-1}(\boldsymbol{w} - \boldsymbol{w}_{MP})\right\} \tag{5.44}$$

with

$$\boldsymbol{\Sigma}_{MP} = (\bar{\lambda}\widetilde{\boldsymbol{K}}_{1c} + \bar{\beta}\widetilde{\boldsymbol{K}}_{2c})^{-1} \qquad \boldsymbol{w}_{MP} = \boldsymbol{\Sigma}_{MP}\widetilde{\boldsymbol{K}}_{1c}^T\boldsymbol{y} \tag{5.45}$$

Optimisation of $Q_{\boldsymbol{c}}(\boldsymbol{c})$

As a function of $Q_{\boldsymbol{c}}(\boldsymbol{c})$, F can be written as,

$$\begin{aligned} F &= \int Q_{\boldsymbol{c}}(\boldsymbol{c})\Big[-\frac{\bar{\beta}}{2}\int Q_{\boldsymbol{w}}(\boldsymbol{w})(\boldsymbol{y}-\boldsymbol{K}_c\boldsymbol{w})^T(\boldsymbol{y}-\boldsymbol{K}_c\boldsymbol{w})\,d\boldsymbol{w} \\ &- \frac{\bar{\lambda}}{2}\int Q_{\boldsymbol{w}}(\boldsymbol{w})\boldsymbol{w}^T\boldsymbol{K}_c\boldsymbol{w}\,d\boldsymbol{w} - \sum_{d=1}^{J}\frac{c_d}{m_d} - \log Q_{\boldsymbol{c}}(\boldsymbol{c})\Big]d\boldsymbol{c} + \text{const} \\ &= \int Q_{\boldsymbol{c}}(\boldsymbol{c})\Big[-\frac{\bar{\beta}}{2}\Big((\boldsymbol{y}-\boldsymbol{K}_c\boldsymbol{w}_{MP})^T(\boldsymbol{y}-\boldsymbol{K}_c\boldsymbol{w}_{MP}) + \text{tr}(\boldsymbol{K}_c^T\boldsymbol{K}_c\boldsymbol{\Sigma}_{MP})\Big) \\ &- \frac{\bar{\lambda}}{2}\Big(\boldsymbol{w}_{MP}^T\boldsymbol{K}_c\boldsymbol{w}_{MP} + \text{tr}(\boldsymbol{K}_c\boldsymbol{\Sigma}_{MP})\Big) + \frac{1}{2}\log|\boldsymbol{K}_c|\Big]\,d\boldsymbol{c} - \sum_{d=1}^{J}\frac{n_d}{m_d} + \sum_{d=1}^{J}\log n_d \\ &+ \text{const.} \end{aligned} \tag{5.46}$$

Using the relationship, $\langle f(\boldsymbol{c})\rangle \approx f(\langle\boldsymbol{c}\rangle)$, which is extensively used in the mean field learning community (Opper and Winther, 2000), allows us to write the determinant term as,

$$\begin{aligned} \frac{1}{2}\int Q_{\boldsymbol{c}}(\boldsymbol{c})\log|\boldsymbol{K}_c|\,d\boldsymbol{c} &\approx \frac{1}{2}\log\Big(\int Q_{\boldsymbol{c}}(\boldsymbol{c})|\boldsymbol{K}_c|\,d\boldsymbol{c}\Big) \\ &= \frac{1}{2}\log|\widetilde{\boldsymbol{K}}_{1c}| \end{aligned} \tag{5.47}$$

We should notice that

$$\boldsymbol{K}_c = \sum_{d=1}^{J}c_d\boldsymbol{K}_d \qquad \boldsymbol{K}_c^T\boldsymbol{K}_c = \sum_{d_1=1}^{J}\sum_{d_2=1}^{J}c_{d_1}c_{d_2}\boldsymbol{K}_{d_1}^T\boldsymbol{K}_{d_2} \tag{5.48}$$

From this we can obtain,

$$\begin{aligned} F &= -\frac{\bar{\beta}}{2}\sum_{d=1}^{J}n_d^2\boldsymbol{w}_{MP}^T\boldsymbol{K}_d^T\boldsymbol{K}_d\boldsymbol{w}_{MP} - \frac{\bar{\beta}}{2}\sum_{d_1,d_2=1}^{J}n_{d_1}n_{d_2}\boldsymbol{w}_{MP}^T\boldsymbol{K}_{d_1}^T\boldsymbol{K}_{d_2}\boldsymbol{w}_{MP} \\ &- \frac{\bar{\lambda}}{2}\sum_{d=1}^{J}n_d\boldsymbol{w}_{MP}^T\boldsymbol{K}_d\boldsymbol{w}_{MP} - \frac{\bar{\beta}}{2}\sum_{d=1}^{J}n_d^2\text{trace}(\boldsymbol{K}_d^T\boldsymbol{K}_d\boldsymbol{\Sigma}_{MP}) \\ &- \frac{\bar{\beta}}{2}\sum_{d_1,d_2=1}^{J}n_{d_1}n_{d_2}\text{trace}(\boldsymbol{K}_{d_1}^T\boldsymbol{K}_{d_2}\boldsymbol{\Sigma}_{MP}) - \frac{\bar{\lambda}}{2}\sum_{d=1}^{J}n_d\text{trace}(\boldsymbol{K}_d\boldsymbol{\Sigma}_{MP}) \\ &- \sum_{d=1}^{J}\frac{n_d}{m_d} + \sum_{d=1}^{J}\log n_d + \bar{\beta}\boldsymbol{y}^T\widetilde{\boldsymbol{K}}_{1c}\boldsymbol{w}_{MP} + \text{const} \end{aligned} \tag{5.49}$$

At the solution the partial derivative w.r.t n_d is then zero and hence,

$$\begin{aligned}
\frac{\partial F}{\partial n_d} &= -\bar{\beta} n_d \boldsymbol{w}_{\scriptscriptstyle MP}^T \boldsymbol{K}_d^T \boldsymbol{K}_d \boldsymbol{w}_{\scriptscriptstyle MP} - \bar{\beta} \sum_{c_1=1}^{J} n_{d_1} \boldsymbol{w}_{\scriptscriptstyle MP}^T \boldsymbol{K}_d^T \boldsymbol{K}_{d_1} \boldsymbol{w}_{\scriptscriptstyle MP} - \frac{\bar{\lambda}}{2} \boldsymbol{w}_{\scriptscriptstyle MP}^T \boldsymbol{K}_d \boldsymbol{w}_{\scriptscriptstyle MP} \\
&- \bar{\beta} n_d \text{trace}(\boldsymbol{K}_d^T \boldsymbol{K}_d \boldsymbol{\Sigma}_{\scriptscriptstyle MP}) - \bar{\beta} \sum_{d_1=1}^{J} n_{d_1} \text{trace}(\boldsymbol{K}_d^T \boldsymbol{K}_{d_1} \boldsymbol{\Sigma}_{\scriptscriptstyle MP}) - \frac{\bar{\lambda}}{2} \text{trace}(\boldsymbol{K}_d \boldsymbol{\Sigma}_{\scriptscriptstyle MP}) \\
&+ \bar{\beta} \boldsymbol{y}^T \widetilde{\boldsymbol{K}}_{1c} \boldsymbol{w}_{\scriptscriptstyle MP} - \frac{1}{m_d} + \frac{1}{n_d} + \frac{1}{2} \frac{\partial}{\partial n_d} \log |\widetilde{\boldsymbol{K}}_{1c}| = 0
\end{aligned}$$

A simplification can be made for the $\log |\widetilde{K}_{1c}|$ term by considering the associated partial derivative,

$$\frac{1}{2} \frac{\partial}{\partial c_d} \log |\widetilde{\boldsymbol{K}}_{1c}| = \frac{1}{2} \text{trace} \left((\widetilde{\boldsymbol{K}}_{1c})^{-1} \frac{\partial \widetilde{\boldsymbol{K}}_{1c}}{\partial n_d} \right) \tag{5.50}$$

Using the result that,

$$\frac{\partial \widetilde{\boldsymbol{K}}_{1c}}{\partial n_d} = \boldsymbol{K}_d$$

allows the above term to be re-written as,

$$\frac{1}{2} \frac{\partial}{\partial n_d} \log |\widetilde{\boldsymbol{K}}_{1c}| = \frac{1}{2} \text{trace} \left((\widetilde{\boldsymbol{K}}_{1c})^{-1} \boldsymbol{K}_d \right) \tag{5.51}$$

Optimisation of $Q_\beta(\beta)$

As a function of $Q_\beta(\beta)$, F can be written as (ignoring constant terms),

$$\begin{aligned}
F &= \iiint Q_\beta(\beta) Q_{\boldsymbol{w}}(\boldsymbol{w}) Q_{\boldsymbol{c}}(\boldsymbol{c}) \Big[\log P(\mathcal{D}|\boldsymbol{w}, \boldsymbol{c}, \beta) + \log P(\beta) - \log Q_\beta(\beta) \Big] \, d\beta \, d\boldsymbol{w} \, d\boldsymbol{c} \\
&= \int Q_\beta(\beta) \Big[\frac{N}{2} \log \beta - \frac{\beta}{2} \iint Q_{\boldsymbol{w}}(\boldsymbol{w}) Q_{\boldsymbol{c}}(\boldsymbol{c}) (\boldsymbol{y} - \boldsymbol{K}_c \boldsymbol{w})^T (\boldsymbol{y} - \boldsymbol{K}_c \boldsymbol{w}) \, d\boldsymbol{w} \, d\boldsymbol{c} \\
&+ \log P(\beta) - \log Q_\beta(\beta) \Big] \, d\beta + \text{const} \\
&= \int Q_\beta(\beta) \Big[\frac{N}{2} \log \beta - \frac{\beta}{2} \Big(\boldsymbol{y}^T \boldsymbol{y} - 2 \boldsymbol{y}^T \widetilde{\boldsymbol{K}}_{1c} \boldsymbol{w}_{\scriptscriptstyle MP} + \boldsymbol{w}_{\scriptscriptstyle MP}^T \widetilde{\boldsymbol{K}}_{2c} \boldsymbol{w}_{\scriptscriptstyle MP} + \text{trace}(\widetilde{\boldsymbol{K}}_{2c} \boldsymbol{\Sigma}_{\scriptscriptstyle MP}) \Big) \\
&+ (a_\beta - 1) \log \beta - \frac{\beta}{b_\beta} - \log Q_\beta(\beta) \Big] \, d\beta + \text{const}
\end{aligned} \tag{5.52}$$

Hence the best $Q_\beta(\beta)$ is a Gamma distribution with new parameters

$$\begin{aligned}
\widetilde{a}_\beta &= \frac{N}{2} + a_\beta - 1 \\
\widetilde{b}_\beta &= b_\beta + \frac{1}{2} (\boldsymbol{y}^T \boldsymbol{y} - 2 \boldsymbol{y}^T \widetilde{\boldsymbol{K}}_{1c} \boldsymbol{w}_{\scriptscriptstyle MP} + \boldsymbol{w}_{\scriptscriptstyle MP}^T \widetilde{\boldsymbol{K}}_{2c} \boldsymbol{w}_{\scriptscriptstyle MP} + \text{trace}(\widetilde{\boldsymbol{K}}_{2c} \boldsymbol{\Sigma}_{\scriptscriptstyle MP}))
\end{aligned} \tag{5.53}$$

from which,

$$\bar{\beta} = \frac{\widetilde{a}_\beta}{\widetilde{b}_\beta} \tag{5.54}$$

Optimisation of $Q_\lambda(\lambda)$

As a function of $Q_\lambda(\lambda)$, F can be written as (ignoring constant terms),

$$\begin{aligned}
F &= \iiint Q_\lambda(\lambda) Q_{\boldsymbol{w}}(\boldsymbol{w}) Q_{\boldsymbol{c}}(\boldsymbol{c}) \left[\log P(\boldsymbol{w}|\boldsymbol{c},\lambda) + \log P(\lambda) - \log Q_\lambda(\lambda) \right] d\boldsymbol{w}\, d\boldsymbol{c}\, d\lambda \\
&= \int Q_\lambda(\lambda) \left[\frac{N}{2} \log \lambda - \frac{\lambda}{2} \iint Q(\boldsymbol{w}) Q(\boldsymbol{c}) \boldsymbol{w}^T \boldsymbol{K}_c \boldsymbol{w}\, d\boldsymbol{w}\, d\boldsymbol{c} \right. \\
&+ \left. \log P(\lambda) - \log Q_\lambda(\lambda) \right] d\lambda + \text{const} \\
&= \int Q_\lambda(\lambda) \left[\frac{N}{2} \log \lambda - \frac{\lambda}{2} \left(\boldsymbol{w}_{MP}^T \widetilde{\boldsymbol{K}}_{1c} \boldsymbol{w}_{MP} + \text{trace}(\widetilde{\boldsymbol{K}}_{1c} \boldsymbol{\Sigma}_{MP}) \right) \right. \\
&+ \left. (a_\lambda - 1) \log \lambda - \frac{\lambda}{b_\lambda} - \log Q_\lambda(\lambda) \right] d\lambda + \text{const}
\end{aligned} \tag{5.55}$$

Hence the best $Q_\lambda(\lambda)$ is a Gamma distribution with new parameters,

$$\begin{aligned}
\widetilde{a}_\lambda &= \frac{N}{2} + a_\lambda - 1 \\
\widetilde{b}_\lambda &= b_\lambda + \boldsymbol{w}_{MP}^T \widetilde{\boldsymbol{K}}_{1c} \boldsymbol{w}_{MP} + \text{trace}(\widetilde{\boldsymbol{K}}_{1c} \boldsymbol{\Sigma}_{MP})
\end{aligned} \tag{5.56}$$

from which,

$$\bar{\lambda} = \frac{\widetilde{a}_\lambda}{\widetilde{b}_\lambda} \tag{5.57}$$

As was the case for variational Bayesian learning for a neural network, described in Section 3.7.8, hyperparameter re-estimation formulae can be obtained,

$$\begin{aligned}
\frac{1}{n_d} &= \frac{1}{m_d} - \frac{1}{2}\text{trace}\left((\widetilde{\boldsymbol{K}}_{1c})^{-1} \boldsymbol{K}_d \right) + \frac{\bar{\lambda}}{2} \left(\boldsymbol{w}_{MP}^T \boldsymbol{K}_d \boldsymbol{w}_{MP} + \text{trace}(\boldsymbol{K}_d \boldsymbol{\Sigma}_{MP}) \right) \\
&+ \bar{\beta} \left(n_d (\boldsymbol{w}_{MP}^T \boldsymbol{K}_d^T \boldsymbol{K}_d \boldsymbol{w}_{MP} + \text{trace}(\boldsymbol{K}_d^T \boldsymbol{K}_d \boldsymbol{\Sigma}_{MP})) + \boldsymbol{w}_{MP}^T \boldsymbol{K}_d^T \widetilde{\boldsymbol{K}}_{1c} \boldsymbol{w}_{MP} \right. \\
&+ \left. \text{trace}(\boldsymbol{K}_d^T \widetilde{\boldsymbol{K}}_{1c} \boldsymbol{\Sigma}_{MP}) - \boldsymbol{y}^T \widetilde{\boldsymbol{K}}_{1c} \boldsymbol{w}_{MP} \right)
\end{aligned} \tag{5.58}$$

The variational estimate for the ANOVA hyperparameters can be summarised by the following pseudo-code,

5.5.1 Demonstration on Bishop's Dataset

To evaluate whether the proposed method can reliably select relevant input variables its performance was assessed on a toy problem proposed by Bishop (1995) which was reviewed in Chapter 3. The experiments were performed using 100 examples. To reduce the effects of data partitioning on the generalisation estimate, the modelling algorithms were evaluated for ten different (random) generations of the data. Table 5.5 summarises the ranked importance of the input variables that were deemed to be important.

Algorithm 5: Variational Learning for BISKIT

```
Input        Dataset $\mathcal{D} = (\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$

Initialise   Initialise $\boldsymbol{n}_d = (n_1, n_2, \ldots, n_J)$ to a constant value (e.g. 0.01);

Algorithm    Do{
                 Use a kernel method to optimise $\boldsymbol{w}_{MP}$
                 Update $\bar{\beta}$ using Equation 5.54
                 Update $\bar{\lambda}$ using Equation 5.57
                 Update each $n_d$ using Equation 5.58
             }While($\frac{1}{J}\sum_{i=1}^{J} |n_i - n_i^{old}| > \delta$)

Output       $\boldsymbol{n}, \bar{\beta}, \bar{\lambda}, \boldsymbol{w}_{MP}$.
```

	Dataset									
Components	1	2	3	4	5	6	7	8	9	10
x_1	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}
x_2	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}
x_3	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}
$x_1 \otimes x_2$	4^{th}	4^{th}	–	4^{th}	4^{th}	4^{th}	4^{th}	4^{th}	–	–

Table 5.5: Bishop's Dataset: Ranked importance of the input variables via variational learning in a Support Vector Machine as part of the BISKIT algorithm.

	x_1	x_2	x_3	$x_1 \otimes x_2$
n	0.0146	0.0117	0.0087	0.006
σ_n	0.001	0.0001	0.0001	0.0001

Table 5.6: Bishop's Dataset: Mean and associated standard deviation hyperparameter values using variational learning within a Support Vector Machine as part of the BISKIT algorithm.

Given that derivative information also exists (see Equation 5.50), the alternative methodology of gradient descent was also tested. The update for the hyperparameters is carried out in a manner similar to equation Equation 3.18. In the neural network community, it is well known that the choice of the learning rate can affect the rate of convergence of the algorithm as well as the quality of the final solution obtained.

	x_1	x_2	x_3	$x_1 \otimes x_2$	$x_1 \otimes x_3$
n	29.95	23.98	13.93	4.02	0.88
σ_n	3.02	2.78	2.52	1.60	0.41

Table 5.7: Bishop's Dataset: Mean and associated standard deviation hyperparameter values using gradient descent within a Support Vector Machine as part of the BISKIT algorithm.

	Dataset									
Components	1	2	3	4	5	6	7	8	9	10
x_1	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}
x_2	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}
x_3	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}
$x_1 \otimes x_2$	4^{th}	4^{th}	4^{th}	4^{th}	–	4^{th}	4^{th}	4^{th}	–	4^{th}
$x_1 \otimes x_3$	–	5^{th}	5^{th}	–	–	–	–	–	–	–

Table 5.8: Bishop's Dataset: Ranked importance of the input variables via gradient descent in a Support Vector Machine as part of the BISKIT algorithm.

From the ranked importance of the input variables, the approach described in algorithm 5 is able to distinguish between the relevant and the irrelevant inputs with a high degree of accuracy across the ten different data partitions using both the gradient descent method and using the hyperparameter re-estimation formula given in Equation 5.58. However, both approaches select the irrelevant term $x_1 x_2$ as being important which may be a consequence of the correlation between the data inputs.

5.5.2 Demonstration on Friedman's Dataset

To demonstrate the performance of the BISKIT algorithm the method was run on Friedman's dataset used in assessing the SUPANOVA algorithm (see Section 5.4.2) and used in Chapter 3 to assess the effectiveness of Bayesian learning. The mean and associated standard deviation of the hyperparameters are given in Table 5.9. The ranked importance of the input variables are given in Table 5.10.

	x_1	x_2	x_3	x_4	x_5	$x_1 \otimes x_2$
n	0.017	0.011	0.008	0.006	0.0052	0.0274
$\sigma_n (\times 10^{-4})$	3.05	2.57	1.77	1.42	1.15	1.15

Table 5.9: Friedman's Dataset: Mean and associated standard deviation hyperparameter values using the BISKIT algorithm.

	Dataset									
Components	1	2	3	4	5	6	7	8	9	10
x_1	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}
x_2	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}
x_3	4^{th}	4^{th}	4^{th}	4^{th}	4^{th}	4^{th}	4^{th}	4^{th}	4^{th}	4^{th}
x_4	5^{th}	5^{th}	5^{th}	5^{th}	5^{th}	5^{th}	5^{th}	5^{th}	5^{th}	5^{th}
x_5	6^{th}	6^{th}	6^{th}	6^{th}	6^{th}	6^{th}	6^{th}	6^{th}	6^{th}	6^{th}
$x_1 \otimes x_2$	1^{st}	1^{st}	2^{nd}	1^{st}	1^{st}	1^{st}	–	1^{st}	1^{st}	1^{st}

Table 5.10: Friedman's Dataset: Ranked importance of the input variables when using the BISKIT algorithm on Friedman's dataset.

From the ranked importance of the inputs using Friedman's dataset the BISKIT algorithm is able to distinguish between the relevant and irrelevant input variables with a high degree of certainty.

5.6 Monte-Carlo BISKIT (McBISKIT)

The approach taken here was similar in manner to that described by Neal (1995), and reviewed in Section 3.7.2. A Gamma prior distribution was placed on the ANOVA kernel hyperparameters. The method of rejection sampling was used to generate samples from the prior distribution. Rejection sampling in general is discussed by Devroye (1986). This method produces a sample of independent values from the posterior distribution given the training data. These independent values from the posterior are obtained by generating independent values from the prior and then rejecting some of these with probability proportional to the likelihood due to the training data. However, as Neal (1995) observes the rejection rate with this method can be extremely high. It can be feasibly applied only to very small training sets, with priors carefully chosen to give a high probability to parameter values that are well-matched to the data. As discussed in Section 3.7.2 the initial choice of starting values can greatly affect the time it takes for the system to reach equilibrium as well as the quality of the final approximation.

5.6.1 Demonstration on Bishop's Dataset

To illustrate the performance of the McBISKIT problem, the problem was illustrated on the same artificial datasets, proposed by (Bishop, 1995), as those used for the BISKIT algorithm. The mean and the associated standard deviation values are given in Table 5.11, whilst Table 5.12 provides a ranked importance of the input variables.

	x_1	x_2	x_3	$x_1 \otimes x_2$	$x_1 \otimes x_3$	$x_1 \otimes x_2$	$x_1 \otimes x_2 \otimes x_3$
$n(\times 10^{-4})$	2.14	1.63	74.5	1.92	1.43	92.5	1.18
$\sigma_n(\times 10^{-4})$	4.641	2.65	1.18	3.23	3.00	2.38	1.84

TABLE 5.11: Bishop's Dataset: Mean and associated standard deviation hyperparameter values within a Support Vector Machine as part of the McBISKIT algorithm.

From the ranked importance of the inputs using Friedman's dataset the McBISKIT algorithm is able to distinguish between the relevant and irrelevant input variables with a high degree of certainty.

	Dataset									
Components	1	2	3	4	5	6	7	8	9	10
x_1	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}
x_2	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}
x_3	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}
$x_1 \otimes x_2$	4^{th}	4^{th}	4^{th}	4^{th}	–	4^{th}	4^{th}	4^{th}	–	4^{th}
$x_1 \otimes x_3$	–	5^{th}	5^{th}	–	–	–	–	–	–	–

TABLE 5.12: Bishop's Dataset: Ranked importance of the input variables using the McBISKIT algorithm (initial hyperparameter values $\boldsymbol{\alpha} = \mathbf{0.01}$).

5.6.2 Demonstration on Friedman's Dataset

Given the large number of ANOVA terms that exist in a complete ANOVA expansion, and the observations made by Neal (1995) regarding Gibbs sampling (described in Section 5.6), the hyperparameter values obtained from the SUPANOVA approach are used as starting values for the MCMC sampling algorithm. The mean and associated standard deviation of the hyperparameters are given in Table 5.13. The ranked importance of the input variables are given in Table 5.14.

	x_1	x_2	x_3	x_4	x_5	$x_1 \otimes x_2$
n	0.016	0.011	0.008	0.006	0.005	0.043
$\sigma_n(\times 10^{-4})$	3.05	2.57	1.77	1.42	1.15	1.14

TABLE 5.13: Friedman's Dataset: Mean and associated standard deviation hyperparameter values using the McBISKIT algorithm.

	Dataset									
Components	1	2	3	4	5	6	7	8	9	10
x_1	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}
x_2	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}
x_3	3^{rd}	4^{th}	4^{th}	4^{th}	4^{th}	4^{th}	4^{th}	4^{th}	4^{th}	4^{th}
x_4	3^{rd}	4^{th}	5^{th}	5^{th}	5^{th}	5^{th}	5^{th}	5^{th}	5^{th}	5^{th}
x_5	6^{th}	6^{th}	6^{th}	6^{th}	6^{th}	6^{th}	6^{th}	6^{th}	6^{th}	6^{th}
$x_1 \otimes x_2$	1^{st}	1^{st}	2^{nd}	1^{st}	1^{st}	1^{st}	–	1^{st}	1^{st}	1^{st}

TABLE 5.14: Friedman's Dataset: Ranked importance of the input variables when using the BISKIT algorithm on Friedman's dataset.

5.7 ARD Gaussian Processes

As described in Section 4.4, the function that is chosen must generate a positive definate matrix for any set of input points. From a Bayesian inference perspective, the covariance function should contain our prior beliefs about the structure of the function that is being modelled.

Williams and Rasmussen (1996) use an additive covariance function between the points $x^{(p)}$ and $x^{(q)}$, $p, q = 1, \ldots, N$ of the form,

$$C(x^{(p)}, x^{(q)}) = a_0 + a_1 \sum_{i=1}^{m} x_i^{(p)}, x_i^{(q)} + \upsilon_0 \exp\left(-\frac{1}{2}\sum_{i=1}^{m} \alpha_i (x_i^{(p)} - x_i^{(q)})^2\right) \tag{5.59}$$

This additive covariance function is conceptually made up of two components. The first part involving the a_0 and a_1 parameters controls the scale of the bias and linear contributions to the covariance. The second part of the covariance function expresses the idea that cases with nearby inputs should have highly correlated outputs. The α_i parameters are multiplied by the co-ordinate wise distances in input space and therefore allow for different distance measures for each input dimension. For irrelevant inputs, the corresponding α_i should be small in order for the model to ignore these inputs. As Rasmussen (1996) observes the "characteristic lengths" for input directions are given by $\alpha_i^{-1/2}$. When a α_i parameter becomes large, the resulting function will have a short characteristic length indicating that this input is of high importance. This idea (Williams and Rasmussen, 1996; Rasmussen, 1996) is closely related to the ARD method described in Chapter 3. The MAP estimates for these hyperparameters can be determined using a gradient descent based technique to locate a maximum of the posterior distribution.

5.7.1 Demonstration of Hyperparameter Re-estimation

In order to evaluate the performance of the hyperparameter re-estimation approach when applied to a Gaussian process, the artificial modelling problem proposed by (Friedman, 1991) used in Chapter 3 was used. In order to main comparability between the previous results the same datasets as those used to assess the Bayesian neural network were used. The Gaussian process model was trained using the scaled conjugate gradient algorithm.

The mean and the associated standard deviation for the hyperparameters associated with this covariance function are given in Table 5.15, and the ranked importance of the input variables are given in Table 5.16. The stability with which the Gaussian process model selects the relevant input variables is comparable to that of the Bayesian neural network trained using MCMC described in Chapter 3. The spurious inputs are given little influence showing that the ARD covariance function by its use of width scales is able to set the hyperparameters associated with the irrelevant inputs to zero. However, in common with the Bayesian neural network, the ARD covariance kernel in its standard form is unable to select the combined influence of inputs x_1 and x_2.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
α	3.55	2.67	2.09	1.56	1.52	0.59	0.26	0.06	0.14	0.07
σ_α	1.88	1.63	1.44	1.24	1.32	0.77	0.51	0.24	0.38	0.28

TABLE 5.15: Mean ARD hyperparameter values for the Gaussian Process ARD covariance function.

	Input variables									
Dataset	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
1	1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}	-	-	-	-	-
2	1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}	-	-	-	-	-
3	1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}	6^{th}	-	-	7^{th}	-
4	1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}	-	-	-	-	-
5	1^{st}	2^{nd}	3^{th}	5^{th}	6^{th}	4^{th}	-	-	7^{rd}	-
6	3^{rd}	2^{nd}	4^{th}	5^{th}	1^{st}	-	6^{th}	-	-	-
7	1^{st}	5^{th}	4^{th}	2^{nd}	3^{rd}	-	6^{th}	-	-	-
8	1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}	-	-	-	-	-
9	1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}	-	-	-	-	-
10	1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}	-	-	-	-	-

TABLE 5.16: Ranked importance of input variables when using the Gaussian Process ARD covariance function.

5.8 Summary

The incorporation of interpretability can be achieved by exploiting the ideas of work in different communities. The wavelet community have been concerned with sparse approximations and from the work of Chen (1995), sparse approximations can be achieved be addition of a 1-norm penalty term to the cost function being minimised. Within the Bayesian community, a number of hyperparameter determination methods have been proposed and two have been exploited in the algorithms developed in this chapter. The three interpretable learning algorithms that were developed, were illustrated on two toy problems highlighting the potential of the approach. The next chapter is concerned with illustrating the algorithms that have been developed in this thesis on a range of "real-world" modelling problems.

Chapter 6

Data Analysis

Throughout the course of this thesis a number of different interpretable modelling methods have been developed. To illustrate the effectiveness of these methods, they have been applied to synthetic artificial datasets where prior knowledge of their expected behaviour exists. These studies were useful for algorithm development, highlighting algorithm deficiencies and consequently aiding algorithm validation. To test these methods further, in this chapter they are applied to a number of more realistic situations. Three examples are considered; automobile miles per gallon prediction, prediction of house prices in the Boston area, and using a commercial materials dataset for predicting the strength of a metal used in the manufacture of aeroplane wings. The demonstrations on Friedman's artificial dataset, described in the previous sections, have helped to illustrate the different properties of the models. This knowledge should be of use when applying these models to real problems.

When comparing learning procedures a key point of interest is how the models differ in many respects. This includes the accuracy of the predictions made, the amount of computation required to produce these predictions, the ease with which the problem can be formulated in an appropriate form, and importantly for this thesis how the construction of the predictive model increases our understanding of the nature of the problem. As Neal (1995) observes, only in the context of a real application is it possible to judge the relative importance of these aspects. Traditionally, neural networks and other modelling methods have been compared primarily on the basis of their predictive performance. In the results presented in this chapter model interpretability is also used as a assessment measure of a models capability.

Learning procedures cannot be compared in complete absence of context. For the results of a comparison to be meaningful, it is necessary to somehow distinguish between models that just happen to do well on a particular problem, and those that not only do well, but which also might have been chosen prior to our seeing the test results for various procedures. Which procedures might reasonably have been chosen will depend upon the prior knowledge that is available for the particular problem. Multiple parameter initialisations, multiple runs and multiple partitions of the data into training and test set have been employed here. It may also be possible to choose between models by other means, such as cross validation. Any of these methods may allow the effective model used to be determined to a large degree by the data. If the chosen model performs well, one can argue that such good performance could indeed have been achieved in a real application of a similar nature. This point is discussed further in Chapter 7.

6.1 Automobile Miles Per Gallon Dataset

In this section the results of modelling the automobile miles per gallon of a car are presented. The performance of the different modelling approaches is demonstrated by application to the problem of modelling automobile miles per gallon data Blake and Merz (1998). The AMPG data set contains the miles travelled, per gallon of fuel consumed, for various different cars. The input variables measure six characteristics of a car; the number of cylinders (discrete), displacement, horsepower, weight, acceleration and model year (discrete). The goal is to discover a relationship between the AMPG and the cars' characteristics. After removing a small number of entries with missing values from the original data set, the experiments were performed using 392 examples, 352 for training and validation and 40 for estimating the generalisation performance.

6.1.1 Bayesian Neural Network with Evidence Framework

A Bayesian neural network with ARD was trained using the evidence framework described in Section 3.7.1. A single hidden layer Bayesian neural network with varying numbers of hidden nodes was used to model the relationship between the inputs and the output. The network weights were initially randomised from a Gaussian distribution. A linear activation function was also used on the output node. The optimal network structure was determined to be six hidden nodes since this corresponds to the lowest error on the test set.

The evidence framework of (MacKay, 1994) described in Section 3.7.1 was used for ARD hyperparameter determination. Table 6.1 gives the mean and associated standard deviation for the ARD hyperparameters, whilst Table 6.2 gives the ranked importance of the input variables.

	C	D	H	W	A	Y
α^{-1}	7.29	9.97	20.51	1.24	0.69	3.25
$\sigma_{\alpha^{-1}}$	14.53	6.99	28.78	2.43	0.70	3.37

Table 6.1: AMPG Dataset: Mean and associated standard deviation ARD hyperparameter values for six hidden nodes when using the evidence framework.

	Dataset									
Components	1	2	3	4	5	6	7	8	9	10
C	6th	6th	6th	6th	6th	6th	6th	6th	6th	6th
D	1st	1st	1st	1st	1st	1st	1st	1st	1st	1st
H	5th	5th	5th	5th	5th	5th	5th	5th	5th	5th
W	4th	4th	4th	4th	4th	4th	4th	4th	4th	4th
A	3rd	3rd	3rd	3rd	3rd	3rd	3rd	3rd	3rd	3rd
Y	2nd	2nd	2nd	2nd	2nd	2nd	2nd	2nd	2nd	2nd

Table 6.2: AMPG Dataset: Input Selection when using a Bayesian Neural Network trained using the Evidence Framework. (C-No of Cylinders, D-Displacement, H-Horse Power, W-Weight, A-Acceleration, Y-Year)

From these tables it is evident that the network is giving the most influence to the weight, acceleration year and horsepower.

6.1.2 Bayesian Neural Network with Variational Learning

A Bayesian neural network with ARD was trained using the variational learning framework described in Section 3.7.8. Table 6.3 gives the mean and associated standard deviation for the ARD hyperparameters, whilst Table 6.4 gives the ranked importance of the input variables.

	C	D	H	W	A	Y
α^{-1}	δ	0.002	0.036	0.271	0.940	1.684
$\sigma_{\alpha^{-1}}$	δ	δ	0.001	0.005	0.02	0.033

Table 6.3: AMPG Dataset: Mean and associated standard deviation ARD hyperparameter values for six hidden nodes when using variational learning where $\delta < 1 \times 10^{-4}$.

	Dataset									
Components	1	2	3	4	5	6	7	8	9	10
C	6th	6th	6th	6th	6th	6th	6th	6th	6th	6th
D	5th	5th	5th	5th	5th	5th	5th	5th	5th	5th
H	4th	4th	4th	4th	4th	4th	4th	4th	4th	4th
W	3rd	3rd	3rd	3rd	3rd	3rd	3rd	3rd	3rd	3rd
A	2nd	2nd	2nd	2nd	2nd	2nd	2nd	2nd	2nd	2nd
Y	1st	1st	1st	1st	1st	1st	1st	1st	1st	1st

Table 6.4: AMPG Dataset: Ranked importance of the AMPG input variables when using a Bayesian Neural Network trained using variational learning. (C-No of Cylinders, D-Displacement, H-Horse Power, W-Weight, A-Acceleration, Y-Year)

6.1.3 Bayesian Neural Network with MCMC Sampling

The evaluation of the Bayesian neural network with ARD was trained in a manner similar to that used in the evidence framework. Hybrid Monte Carlo updates for the network parameters were alternated with Gibbs sampling updates for the hyperparameters. Table 3.8 gives the mean and the associated standard deviation values for the ARD hyperparameters associated with each input variable, whilst Table 6.4 gives the ranked importance of the input variables. From these tables it is evident that the network is giving the most influence to the horsepower, weight, year and displacement. Figure 6.1 shows the convergence of energy with increasing number of iterations.

	C	D	H	W	A	Y
α^{-1}	0.44	1.16	2.85	2.57	0.34	1.78
$\sigma_{\alpha^{-1}}$	0.78	1.46	1.89	2.49	0.41	0.88

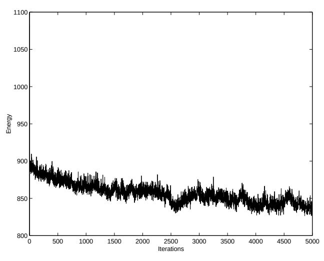
Table 6.5: AMPG Dataset: Mean and associated standard deviation ARD hyperparameter values for six hidden nodes when using MCMC sampling.

	Dataset									
Components	1	2	3	4	5	6	7	8	9	10
C	3^{rd}	6^{th}	–	4^{th}	–	–	–	–	–	–
D	1^{st}	3^{rd}	–	–	–	3^{rd}	3^{rd}	–	3^{rd}	3^{rd}
H	2^{nd}	1^{st}	4^{th}	1^{st}	3^{rd}	$2nd$	1^{st}	1^{st}	2^{nd}	1^{st}
W	6^{th}	4^{th}	1^{st}	3^{rd}	1^{st}	1^{st}	–	2^{nd}	4^{th}	2^{nd}
A	5^{th}	5^{th}	3^{rd}	–	4^{th}	–	4^{th}	–	–	–
Y	4^{th}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	4^{th}	2^{nd}	3^{rd}	1^{st}	4^{th}

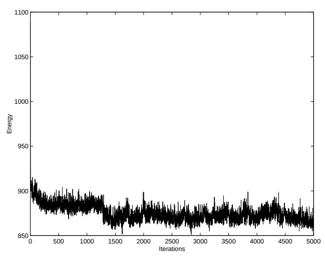
Table 6.6: AMPG Dataset: Input Selection when using a Bayesian Neural Network trained using MCMC Sampling.(C-No of Cylinders,D-Displacement, H-Horse Power, W-Weight, A-Acceleration, Y-Year)

In contrast to the behaviour of the Bayesian neural networks trained on Friedman's additive dataset described in Chapter 3, the selection of input variables indicating relevance is somewhat more variable. Looking at the ranked importance of the input variables shown in tables 6.2 and 6.6 there is a significant degree of variation in the choice of input variables chosen as being important. In fact, no clear input variable stands out as being important, this could be a consequence of the limitations associated with each of the hyperparameter determination methods described in Chapter 3. Given that the evidence framework is based around a number of assumptions a possible explanation is that they are particularly unrealistic for this dataset. It is highly likely that the error surface for this particular problem is highly non-convex and hence the approach is particularly sensitive to the choice of starting values.

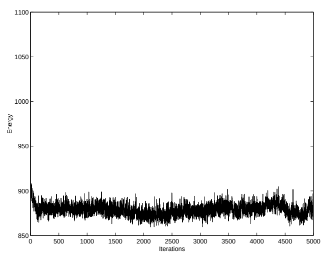
Given that the ARD approach only selects input variables, a useful feature to determine which of the two hyperparameter determination methods is the more accurate visualisation of the trends predicted by each method would be useful. MacKay (MacKay, 1994) has suggested that this could be achieved by constructing an artificial dataset in which all but one input variable is held fixed to its mean value, whilst the one input is allowed to vary between its maximum and minimum values. The predicted behaviour can then be visualised. However, an inherent limitation of this approach is that it only considers



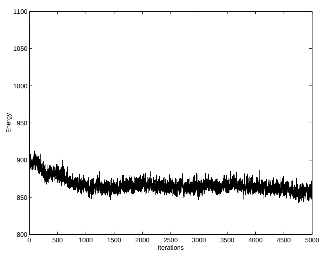
(a) AMPG Dataset 1



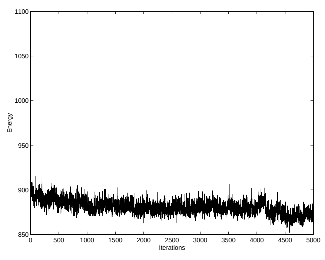
(b) AMPG Dataset 2



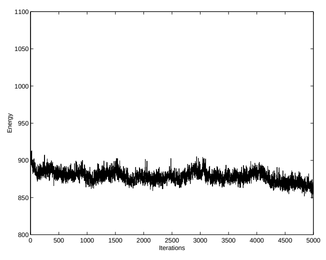
(c) AMPG Dataset 3



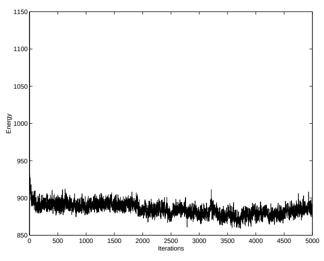
(d) AMPG Dataset 4



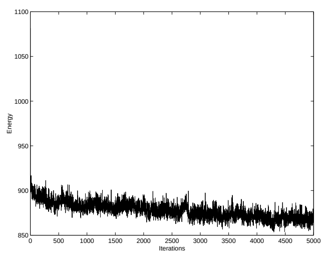
(e) AMPG Dataset 5



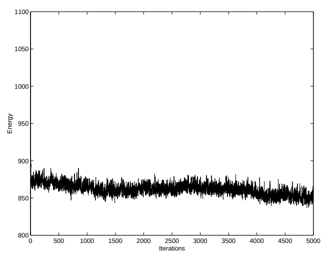
(f) AMPG Dataset 6



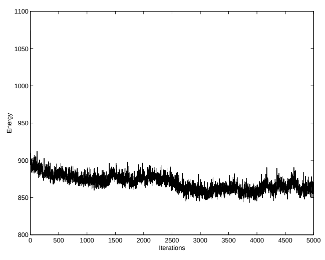
(g) AMPG Dataset 7



(h) AMPG Dataset 8



(i) AMPG Dataset 9



(j) AMPG Dataset 10

FIGURE 6.1: AMPG Dataset: Energy Convergence for a Bayesian Neural Network trained using MCMC sampling on the randomly partitioned AMPG datasets.

a *slice* of the input space and cannot be realistically expected to demonstrate the real trend.

6.1.4 ARD Gaussian Process Model

A Gaussian process model was trained on the AMPG dataset, using the ARD covariance function described in Chapter 4. Table 6.7 gives the mean and the associated standard deviation values for the ARD hyperparameters associated with each input variable, whilst Table 6.8 gives the ranked importance of the input variables. From these tables it is evident that the network is giving the most influence to displacement, number of cylinders, acceleration and year.

	C	D	H	W	A	Y
α	5.92	18.18	1.67	1.00	5.55	4.54
σ_α	2.55	13.79	0.30	0.25	2.49	0.80

Table 6.7: AMPG Dataset: Mean and associated standard deviation ARD hyperparameter values for an ARD Gaussian Process model.

	Dataset									
Components	1	2	3	4	5	6	7	8	9	10
C	1^{st}	2^{nd}	3^{rd}	1^{st}	2^{nd}	2^{nd}	2^{nd}	6^{th}	1^{st}	2^{nd}
D	2^{nd}	1^{st}	1^{st}	2^{nd}	1^{st}	1^{st}	1^{st}	1^{st}	2^{nd}	1^{st}
H	5^{th}	5^{th}	5^{th}	5^{th}	5^{th}	5^{th}	5^{th}	4^{th}	5^{th}	5^{th}
W	6^{th}	6^{th}	6^{th}	6^{th}	6^{th}	6^{th}	6^{th}	5^{th}	6^{th}	6^{th}
A	3^{rd}	3^{rd}	2^{nd}	4^{th}	4^{th}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}
Y	4^{th}	4^{th}	4^{th}	3^{rd}	3^{rd}	4^{th}	4^{th}	2^{nd}	4^{th}	4^{th}

Table 6.8: AMPG Dataset: Ranked importance of the input variables when using a Gaussian process trained using the ARD covariance function. (C-No of Cylinders, D-Displacement, H-Horse Power, W-Weight, A-Acceleration, Y-Year)

6.1.5 Sparse Kernel Support Vector Machines

The kernel based SUPANOVA technique was applied to the miles per gallon dataset using both quadratic and ϵ-insensitive loss functions. Table 6.10 shows that the method has selected eight ANOVA terms from a possible 64 as being relevant in determining the output. These are bias; 3 univariate corresponding to displacement, weight and year; 3 bivariate terms corresponding to a tensor product between displacement and weight, number of cylinders and acceleration and acceleration and year; and one trivariate term composed of a tensor product between horse power, acceleration and year.

Table 6.9 demonstrates that the difference in the mean of the estimated generalisation error between a full ANOVA model and a parsimonious ANOVA model is negligible. However, it also demonstrates that the parsimonious kernel has a lower variance and hence suggests that it is more robust.

Loss Function		Estimated Generalisation Error					
Training	Testing	Stage I		Stage III		Linear Model	
Quadratic	Quadratic	6.97	(7.39)	7.08	(6.19)	11.4	(11.0)
ϵ-insensitive	ϵ-insensitive	0.48	(0.04)	0.49	(0.03)	1.80	(0.11)
ϵ-insensitive	Quadratic	7.07	(6.52)	7.13	(6.04)	11.72	(10.94)

Table 6.9: AMPG Dataset: SUPANOVA Results for the AMPG Data Set ($\epsilon = 1.0$). Quoted values are for the mean (and variance) of the estimated generalisation error.

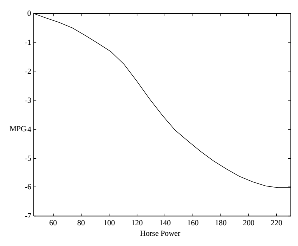
In the ARD approach the selection of the inputs corresponding to displacement, weight and year is highly variable, and it cannot be said with any certainty that the model has reliably selected any of these inputs as being important in predicting the output. Also due to the restricted nature of the ARD approach, in that it cannot look at combinations of inputs, no information can be gathered as to whether there are any higher order interactions.

These results were corroborated by the results using the quadratic loss function. Comparing the two different loss functions shows that, for this particular data-set, there is very little performance difference. Inspection of the ANOVA terms selected by the 100 models (Table 6.10) shows a high consistency, and confirms the robustness of the technique.

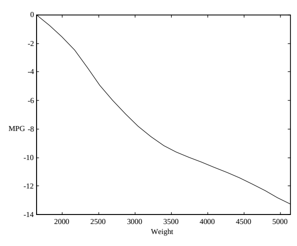
Terms	Quadratic	ϵ-Insensitive	"Difference"
bias	50	50	0.00
C	3	1	0.08
D	35	8	0.66
H	2	20	0.44
W	50	50	0.00
Y	50	50	0.00
$C \otimes D$	9	26	0.54
$C \otimes W$	0	4	0.08
$C \otimes A$	1	11	0.24
$C \otimes Y$	2	18	0.40
$D \otimes W$	35	44	0.38
$C \otimes A$	42	43	0.16
$H \otimes Y$	10	5	0.18
$W \otimes Y$	2	1	0.06
$A \otimes Y$	50	47	0.06
$C \otimes D \otimes W$	0	1	0.02
$C \otimes W \otimes A$	0	1	0.02
$C \otimes W \otimes Y$	0	1	0.02
$C \otimes A \otimes Y$	0	7	0.14
$D \otimes H \otimes W$	1	2	0.06
$H \otimes A \otimes Y$	50	49	0.02
$W \otimes A \otimes Y$	0	4	0.08
$C \otimes D \otimes W \otimes A$	0	1	0.02
$C \otimes D \otimes A \otimes Y$	4	0	0.08

TABLE 6.10: AMPG Dataset: SUPANOVA terms for the AMPG Data Set. ($\epsilon = 2.5$), (C-No of Cylinders, D-Displacement, H-Horse Power, W-Weight, A-Acceleration, Y-Year) (All remaining terms were zero)

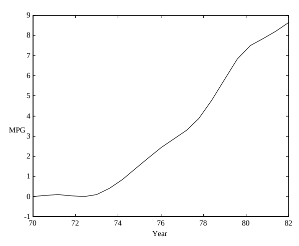
The transparency of the terms is evident from Figure 6.2, although the trivariate is harder to interpret. An example of model validation is demonstrated by the ability to verify the trends in the interaction terms using prior knowledge. There is a decrease in the miles per gallon of a car as both the horse power and the weight of the cars increase. The univariate year term is of particular interest. It can be seen that before 1973 this



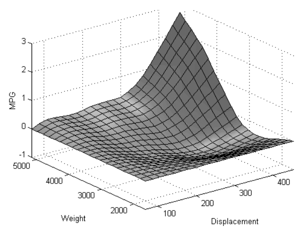
(a) f_H



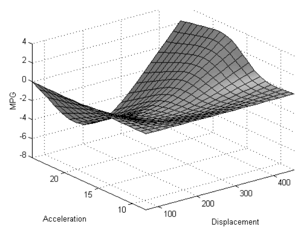
(b) f_W



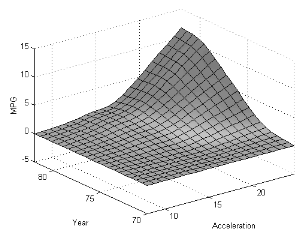
(c) f_Y



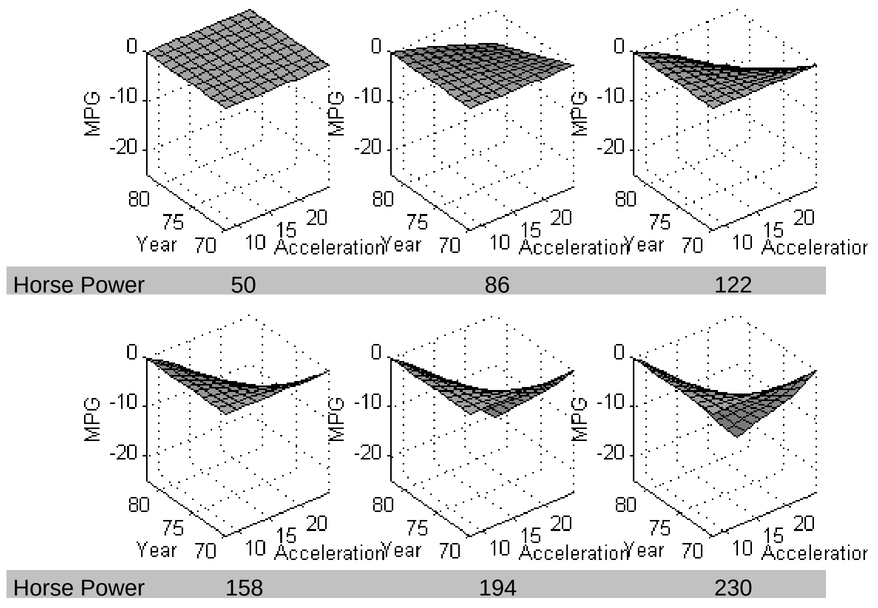
(d) $f_{D\otimes W}$



(e) $f_{D\otimes A}$



(f) $f_{A\otimes Y}$



(g) $f_{H\otimes A\otimes Y}$

FIGURE 6.2: AMPG Dataset: SUPANOVA model using ϵ-insensitive loss function (1 of 50).

term has no effect on the MPG, but after 1973 there is a sharp rise in MPG; this could be a consequence of the oil crisis. This example has acted to show that visualisation of the selected terms can greatly assist in model validation and model selection.

6.1.6 AMPG Dataset: BISKIT Evaluation

To assess the performance of the BISKIT algorithm on a real world modelling problem, the algorithm was run on the AMPG problem described in Section 6.1.5. Given the high computational cost associated with the BISKIT algorithm (in particular the inversion of the kernel matrix as part of the hyperparameter reestimation described in Chapter 5), the final hyperparameter values from the SUPANOVA algorithm were used as starting values for the BISKIT algorithm. Table 6.11 gives the mean and the associated standard deviation for the chosen terms, whilst Table 6.12 gives the ranked importance of the ANOVA terms.

	C	D	H	W	A
$n \times 10^{-3}$	0.009	0.057	0.013	0.150	0.03
$\sigma_n \times 10^{-4}$	0.01	0.23	0.03	0.99	0.04

Table 6.11: AMPG Dataset: Mean and associated standard deviation ARD hyperparameter values for a kernel method trained using the BISKIT algorithm.

	Dataset									
Components	1	2	3	4	5	6	7	8	9	10
D	5^{th}	5^{th}	5^{th}	5^{th}	5^{th}	5^{th}	5^{th}	4^{th}	5^{th}	5^{th}
W	3^{rd}	1^{st}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	2^{nd}	3^{rd}
Y	4^{th}	3^{rd}	4^{th}	4^{th}	4^{th}	4^{th}	4^{th}	5^{th}	4^{th}	4^{th}
$H \otimes A$	1^{st}	4^{th}	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}	1^{st}
$W \otimes Y$	2^{nd}	2^{nd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	2^{nd}

Table 6.12: AMPG Dataset: Ranked importance of the input variables when using the BISKIT algorithm for ANOVA hyperparameter determination. (C-No of Cylinders, D-Displacement, H-Horse Power, W-Weight, A-Acceleration, Y-Year)

Across the ten random data partitions on which the BISKIT algorithm was run, the algorithm is able to distinguish between irrelevant inputs and appears to give the most influence to the bivariate term $H \otimes A$ corresponding to horsepower and acceleration, the univariate term weight and then the bivariate term $W \otimes A$ corresponding to weight and year. These terms are also selected by the SUPANOVA algorithm, reflecting the robustness of the BISKIT algorithm.

6.2 Boston Housing Data

In this section the results of modelling home prices in the Boston area are presented. The Boston housing dataset originates from the work of Harrison and Rubinfield (1978) who were interested in the effect of air pollution on housing prices. The data concerns the median price in 1970 of owner-occupied houses in 506 census tracts within the Boston metropolitan area. Twelve attributes pertaining to each census tract are available for use in predicting the median price. The input variables are: *Crime rate* - per capita crime rate by town, *% Residential land* - proportion of residential land zoned for lots over 25,000 sq.ft., *% Non-retail Business* - proportion of non-retail business acres per town, *Nitric Oxides* - Nitric oxides concentration (parts per 10 million), *Mean no. of rooms* - Average number of rooms per dwelling, *% built pre 1940* - Proportion of owner-occupied units built prior to 1940, *distance to job centre* - weighted distance to five Boston employment centres, *Access to Highways* - index of accessibility to radial highways, *Property Tax* - full value property tax per \$10,000, *Pupil:Teacher Ratio* - Pupil teacher ratio by town ,*% Blacks* - $1000(Blks - 0.63)^2$ where Blks is the proportion of blacks by town, and *% low stat population* which represents the percentage proportion of the population living below the poverty line.

As Neal (1995) observes the data is *messy* is several regards. Some of the attributes are not actually measured on a per-tract basis, but only for larger regions. The median prices for the highest-priced tracts appear to be censored. Censoring is suggested by the fact the highest median price of exactly \$50,000 is reported for sixteen of the tracts, while fifteen tracts are reported to have median prices. Considering these potential problems, it appears unreasonable to expect that the distribution of the target variable, given the input variables, to be Gaussian.

6.2.1 Bayesian Neural Network with Evidence Framework

A Bayesian neural network with ARD was trained using the evidence framework described in Section 3.7.1. A single hidden layer Bayesian neural network with varying numbers of hidden nodes was used to model the relationship between the inputs and the output. The network weights were initially randomised from a Gaussian distribution. A linear activation function was also used on the output node. The optimal network structure was determined to be six hidden nodes since this corresponds to the lowest error on the test set.

The evidence framework of (MacKay, 1994) described in Section 3.7.1 was used for ARD hyperparameter determination. Table 6.13 gives the mean and associated standard deviation for the ARD hyperparameters, whilst Table 6.14 gives the ranked importance of the input variables.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}
α^{-1}	9.12	2.91	8.42	7.12	1.51	0.02	41.30	5.64	19.02	2.35	0.28	2.25
$\sigma_{\alpha^{-1}}$	12.57	6.93	12.77	8.22	1.29	0.03	23.65	6.56	22.02	4.00	0.73	1.39

Table 6.13: Boston Dataset: Mean and associated standard deviation ARD hyperparameter values for six hidden nodes when using the evidence framework.

From these tables, it is evident that the network is giving the most influence to the input variables concerned with: distance to job centre, mean number of rooms, property tax and nitric oxides. Interestingly enough, the network does not give the variable crime a high degree of relevance.

	Input variables											
Dataset	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}
1	4th	7th	2nd	-	5th	-	1st	8th	3rd	-	-	6th
2	5th	10th	3rd	4th	9th	-	2nd	7th	1st	8th	-	6th
3	7th	-	-	-	5th	6th	1st	2nd	4th	-	8th	3rd
4	6th	9th	-	3rd	7th	8th	1st	4th	5th	-	-	2nd
5	6th	-	8th	4th	5th	10th	1st	2nd	3rd	-	9th	7th
6	4th	3rd	5th	6th	-	-	1st	8th	2nd	7th	-	-
7	2nd	-	-	3rd	-	6th	1st	5th	4th	-	-	7th
8	3rd	6th	-	4th	8th	-	2nd	9th	1st	7th	-	5th
9	6th	9th	-	3rd	5th	-	1st	2nd	4th	-	8th	7th
10	7th	-	5th	8th	-	6th	1st	3rd	2nd	4th	-	-

Table 6.14: Boston Dataset: Ranked importance of input variables when using the evidence framework on the Boston house price dataset.

6.2.2 Bayesian Neural Network with Variational Learning

A Bayesian neural network with ARD was trained using the variational learning framework described in Section 3.7.8. Table 6.15 gives the mean and associated standard deviation for the ARD hyperparameters, whilst Table 6.16 gives the ranked importance of the input variables.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}
α^{-1}	1.18	0.005	0.03	2.04	1.44	0.04	44.53	0.95	1.59	0.24	0.36	8.86
$\sigma_{\alpha^{-1}}$	2.56	0.009	0.049	1.47	1.60	0.11	78.77	0.70	1.37	0.25	0.54	7.14

Table 6.15: Boston Dataset: Mean and associated standard deviation ARD hyperparameter values for six hidden nodes when using variational learning. (Values are given up to a constant: $\alpha^{-1} \times 10^{-3}$, $\sigma_{\alpha^{-1}} \times 10^{-7}$)

	Input variables											
Dataset	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}
1	4th	7th	2nd	8th	5th	-	1st	-	3rd	-	-	6th
2	5th	-	3rd	4rd	9th	-	2nd	7th	1st	8th	-	6th
3	7th	-	-	5th	6th	-	1st	2nd	4th	8th	3rd	-
4	6th	-	-	3rd	-	7th	1st	4th	5th	8th	-	2nd
5	-	-	8th	5th	6th	-	1st	2nd	3rd	-	9th	7th
6	4th	3rd	5th	6th	-	-	1st	8th	2nd	7th	-	9th
7	2nd	-	4rd	3rd	-	-	1st	6th	5th	7th	-	8th
8	3rd	6th	-	4th	8th	-	2nd	-	1st	7th	-	5th
9	6th	9th	-	3rd	5th	-	1st	2nd	4th	-	8th	7th
10	7th	-	5th	8th	6th	-	1st	3rd	2nd	4th	-	9th

Table 6.16: Boston Dataset: Ranked importance of input variables when using variational learning on the Boston house price dataset.

6.2.3 Bayesian Neural Network with MCMC Sampling

The evaluation of the Bayesian neural network with ARD was trained in a manner similar to that used in the evidence framework. Hybrid Monte Carlo updates for the network parameters were alternated with Gibbs sampling updates for the hyperparameters. Table 6.17 gives the mean and the associated standard deviation values for the ARD hyperparameters associated with each input variable, whilst Table 6.18 gives the ranked importance of the input variables. From these tables it is evident that the network is giving the most influence to the horsepower, weight, year and displacement. Figure 6.1 shows the convergence of energy with increasing number of iterations.

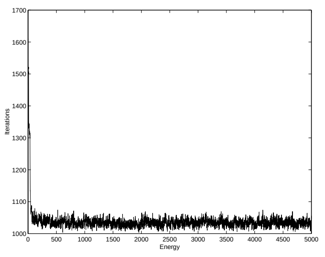
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}
α^{-1}	2.93	1.10	2.76	2.93	1.94	0.69	14.7	3.67	3.36	2.04	0.74	2.72
$\sigma_{\alpha^{-1}}$	1.82	0.78	1.55	1.67	0.83	0.46	6.19	1.80	4.17	3.93	0.29	2.27

TABLE 6.17: Mean and associated standard deviation ARD hyperparameter values for six hidden nodes when using MCMC sampling.

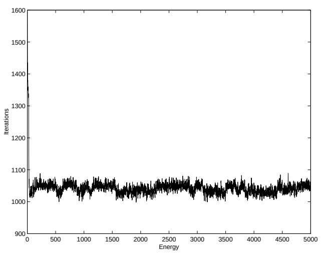
	Input variables											
Dataset	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}
1	4th	-	-	5th	6th	-	1st	3rd	-	-	-	2nd
2	4th	-	-	5th	3rd	-	1st	2nd	-	-	-	-
3	5th	-	3rd	4th	-	-	1st	-	2nd	-	-	-
4	-	-	3rd	-	4th	-	1st	2nd	-	-	-	5th
5	4th	-	5th	-	3rd	-	1st	2nd	-	-	-	-
6	2nd	5th	3rd	4th	-	-	1st	-	-	-	-	6th
7	-	6th	5th	3rd	-	-	1st	2nd	4th	-	-	-
8	4th	-	2nd	6th	-	-	1st	3rd	-	-	-	5th
9	6th	3rd	2nd	-	-	1st	4th	-	-	-	-	5th
10	-	-	-	4th	-	3rd	2nd	1st	-	-	-	-

TABLE 6.18: Ranked importance of input variables when using MCMC sampling on the Boston house price dataset.

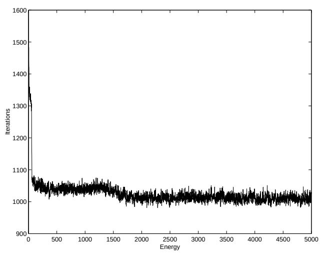
Work carried out by Husmeier (1999) on using an ensemble of Bayesian neural networks trained using an Expectation-Maximisation (EM) algorithm and incorporating automatic relevance determination (ARD) was able to select "relevant" inputs. The input variables rooms, distance to employment centres, access to radial highways, property tax, and the percentage of lower status in the population were selected as being relevant inputs. Husmeier (1999) observes, and this is confirmed in our approach, that the variable crime seems to be an irrelevant input. The remaining input variables show an ambiguous behaviour.



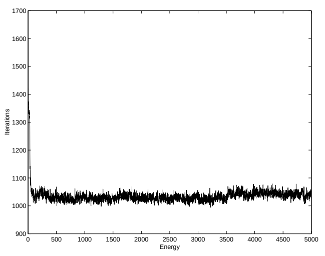
(a) Boston Dataset 1



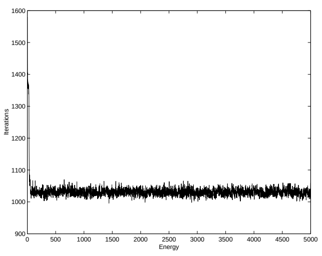
(b) Boston Dataset 2



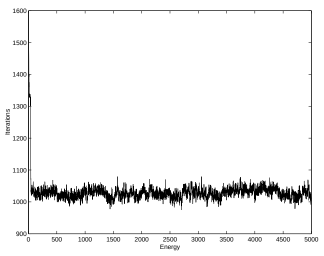
(c) Boston Dataset 3



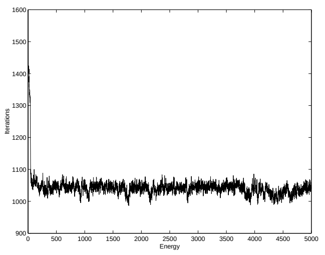
(d) Boston Dataset 4



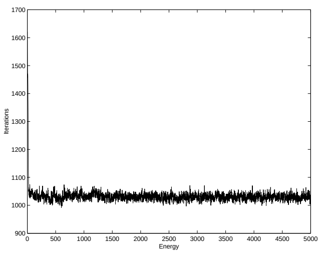
(e) Boston Dataset 5



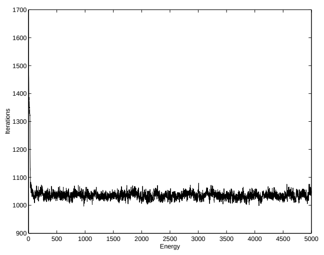
(f) Boston Dataset 6



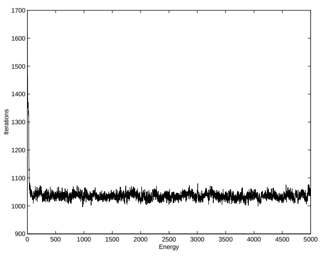
(g) Boston Dataset 7



(h) Boston Dataset 8



(i) Boston Dataset 9



(j) Boston Dataset 10

FIGURE 6.3: Boston Dataset: Energy Convergence for a Bayesian Neural Network trained using MCMC sampling on the randomly partitioned Boston datasets.

6.2.4 ARD Gaussian Process Model

A Gaussian process model was trained on the Boston dataset, using the ARD covariance function described in Chapter 4. Table 6.19 gives the mean and the associated standard deviation values for the ARD hyperparameters associated with each input variable, whilst Table 6.20 gives the ranked importance of the input variables. From these tables the input variables rooms, distance to employment centres, access to radial highways, property tax, and the percentage of lower status in the population were selected as being important inputs.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}
α	7.66	0.05	0.82	14.7	6.22	0.69	30.5	4.40	7.76	0.40	1.82	16.1
σ_α	6.96	0.10	0.70	11.1	1.58	0.32	15.0	2.97	8.51	0.20	1.53	9.53

Table 6.19: Boston Dataset: Mean and associated standard deviation ARD hyperparameter values when using a Gaussian Process.

	Input variables											
Dataset	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}
1	4^{th}	-	-	2^{nd}	5^{th}	-	6^{th}	7^{th}	3^{rd}	-	-	1^{st}
2	-	-	7^{th}	2^{nd}	4^{th}	-	1^{st}	5^{th}	-	-	6^{th}	3^{rd}
3	7^{th}	-	-	4^{th}	5^{th}	-	1^{st}	3^{rd}	-	-	6^{th}	2^{nd}
4	4^{th}	-	-	3^{rd}	5^{th}	-	1^{st}	6^{th}	7^{th}	-	-	2^{nd}
5	5^{th}	-	-	1^{st}	6^{th}	8^{th}	2^{nd}	7^{th}	4^{th}	-	-	3^{rd}
6	-	-	7^{th}	3^{rd}	5^{th}	-	1^{st}	2^{nd}	-	-	6^{th}	4^{th}
7	-	-	-	3^{rd}	4^{th}	-	1^{st}	2^{nd}	-	-	6^{th}	5^{th}
8	4^{th}	-	-	5^{th}	6^{th}	-	3^{rd}	-	1^{st}	-	-	2^{nd}
9	-	-	-	6^{th}	4^{th}	-	1^{st}	3^{rd}	-	-	5^{th}	2^{nd}
10	3^{rd}	-	-	1^{st}	6^{th}	-	5^{th}	-	4^{th}	-	-	2^{nd}

Table 6.20: Boston Dataset: Ranked importance of input variables when using an ARD Gaussian process model.

6.2.5 Sparse Kernel Support Vector Machines

The kernel based SUPANOVA technique was applied to the Boston housing dataset using a quadratic loss function. Fourteen ANOVA terms were selected as being important.

Figure 6.4 illustrates the model obtained from the SUPANOVA technique. Fourteen interaction terms (bias, 4 univariate, 9 bivariate) were selected as being important by the SUPANOVA technique. Inspection of the ANOVA terms selected by the 100 models shows a high consistency, and confirms the robustness of the technique. An example of the terms chosen are shown in Figure 6.4. There is an increase in the house price as the size of the house, as depicted by the mean number of rooms, increases, and a corresponding increase exists with the percentage of homes built before 1940. The bivariate terms given the nature of their surfaces are somewhat harder to interpret, but trends such as an increase in property tax with an increase in the number of rooms can be determined. Overall the trends depicted are broadly consistent with prior knowledge about the problem.

6.2.6 Boston House Price Dataset: BISKIT Evaluation

To assess the performance of the BISKIT algorithm on another real world modelling problem, the algorithm was run on the Boston house price problem described above. As with the AMPG problem, given the high computational cost associated with the

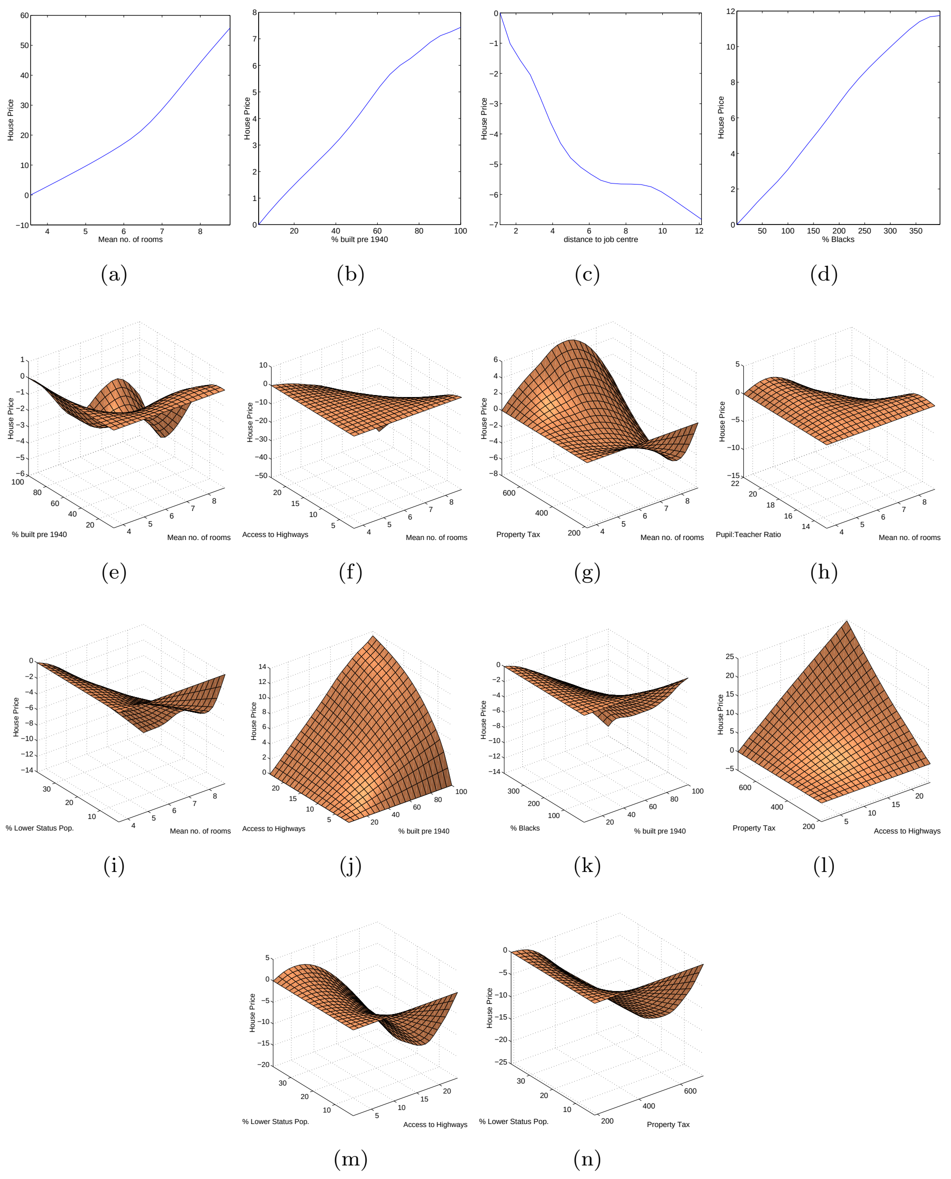


FIGURE 6.4: Visualisation of the ANOVA terms from the Boston House Price Data

BISKIT algorithm (in particular the inversion of the kernel matrix as part of the hyperparameter reestimation described in Chapter 5), the final hyperparameter values from the SUPANOVA algorithm were used as starting values for the BISKIT algorithm. Table 6.21 gives the mean and the associated standard deviation for the chosen terms, whilst Table 6.24 gives the ranked importance of the ANOVA terms.

	x_5	x_6	x_7	x_{11}	$x_5 \otimes x_6$
$n \times 10^{-3}$	0.001	0.01	0.02	0.11	0.01
$\sigma_n \times 10^{-3}$	0.002	0.003	0.03	0.30	0.01

TABLE 6.21: Boston Dataset: Mean and associated standard deviation ARD hyperparameter values for a kernel method trained using the BISKIT algorithm (Part I).

	$x_5 \otimes x_8$	$x_5 \otimes x_9$	$x_5 \otimes x_{10}$	$x_5 \times x_{12}$	$x_6 \otimes x_8$
$n \times 10^{-3}$	0.008	0.008	0.001	0.004	0.015
$\sigma_n \times 10^{-3}$	0.04	0.001	0.004	0.007	0.02

TABLE 6.22: Boston Dataset: Mean and associated standard deviation ARD hyperparameter values for a kernel method trained using the BISKIT algorithm (Part II).

	$x_6 \otimes x_{11}$	$x_8 \otimes x_9$	$x_8 \otimes x_{12}$	$x_9 \otimes x_{12}$
α	0.03	0.04	0.06	0.009
σ_α	0.03	0.04	0.08	0.0002

TABLE 6.23: Boston Dataset: Mean and associated standard deviation ARD hyperparameter values when using a Gaussian Process.

	Dataset									
Components	1	2	3	4	5	6	7	8	9	10
x_5	–	5	–	–	–	–	–	–	–	–
x_6	7th	5th	5th	3rd	6th	6th	3rd	6th	5th	9th
x_7	1st	7th	7th	8th	3rd	7th	8th	2nd	3rd	8th
x_{11}	5th	2nd	1st	4th	5th	1st	4th	3rd	2nd	6th
$x_5 \otimes x_6$	10th	–	9th	10th	–	10th	9th	10th	–	–
$x_5 \otimes x_8$	–	–	3rd	–	9th	9th	5th	9th	7th	5th
$x_5 \otimes x_9$	3rd	8th	–	6th	7th	8th	8th	7th	8th	7th
$x_5 \otimes x_{10}$	–	–	–	–	–	–	–	–	–	–
$x_5 \otimes x_{12}$	8th	–	–	–	10th	–	10th	–	10th	10th
$x_6 \otimes x_8$	4th	6th	8th	7th	1st	3rd	6th	5th	9th	3rd
$x_6 \otimes x_{11}$	9th	4th	4th	5th	2nd	5th	7th	4th	6th	4th
$x_8 \otimes x_9$	6th	1st	2nd	2nd	8th	4th	1st	8th	1st	1st
$x_8 \otimes x_{12}$	2nd	3rd	6th	1st	4th	2nd	2nd	1st	4th	2nd
$x_9 \otimes x_{12}$	–	5th	–	–	–	–	–	–	–	–

TABLE 6.24: Boston Dataset: Ranked importance of the input variables when using the BISKIT algorithm for ANOVA hyperparameter determination.

Across the ten random data partitions on which the BISKIT algorithm was run, the algorithm appears to be able to distinguish between relevant and irrelevant inputs. Many of the terms selected by the SUPANOVA algorithm are also selected by the BISKIT algorithm reflecting the robustness of this approach.

6.3 Commercial Dataset

A commercial processing-properties dataset for DC cast aluminium plate, used extensively in the manufacture of aircraft wings, is considered, concentrating on prediction of the mechanical property 0.2% proof stress. Proof stress is a mechanical property that is related to the strength of the metal. This dataset is illustrative of the problems and challenges that arise in real world modelling; sparsely distributed data and highly correlated inputs. The raw dataset consists of ten input variables and 290 data pairs covering alloy composition and thermomechanical processing information. The ten input variables were; final gauge (FG), Cu, Fe, Mg, Mn, Si (all in weight percent), cast slab length (SL), solution treatment time (STT), percentage stretch (%st.) and reduction-ratio (RR).

6.3.1 Bayesian Neural Network with Evidence Framework

The Bayesian MLP network was trained in the same manner as it was for the artificial dataset. Figure 6.5 shows the variation of training and test set errors for increasing numbers of hidden nodes. The optimal network structure was determined to have seven hidden nodes since this corresponds to the lowest error on the test set.

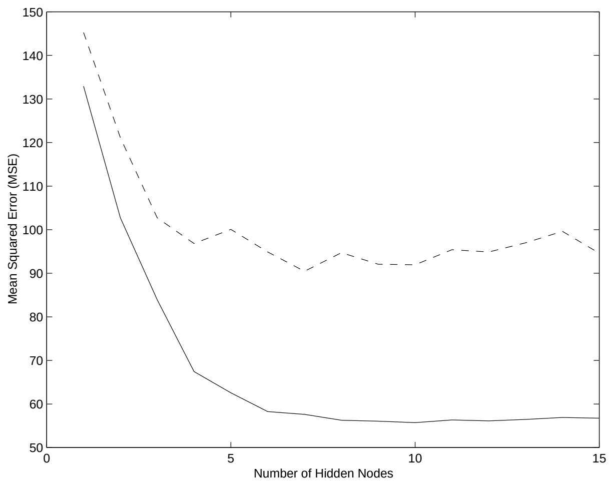


FIGURE 6.5: Variation of mean training and test MSE for a Bayesian MLP trained with varying numbers of hidden nodes.

Table 6.25 shows the mean ARD hyperparameter values (and associated standard deviation over the ten datasets) indicating the influence of each variable on the output for the optimal model structure, and Table 6.26 shows the ranked selection of each input variable for each of the ten models trained. From the values quoted final gauge (FG), silicon (Si), percentage stretch (%st.) and slab length (SL) exhibit the largest values.

	FG	Cu	Fe	Mg	Mn	Si	SL	STT	%st.	RR
α^{-1}	0.006	0.34	0.38	0.27	0.44	1.25	0.22	5.02	0.12	0.11
$\sigma_{\alpha^{-1}}$	0.009	0.37	0.32	0.79	0.85	0.86	0.48	6.15	0.15	0.18

TABLE 6.25: Commercial Dataset: Mean ARD Hyperparameter values for seven hidden nodes using the evidence framework.

6.3.2 Bayesian Neural Network with Variational Learning

A Bayesian neural network with ARD was trained using the variational learning framework described in Section 3.7.8. Table 6.27 gives the mean and associated standard

	Input variables									
Dataset	FG	Cu	Fe	Mg	Mn	Si	SL	STT	%st.	RR
1	4th	-	-	-	-	1st	2nd	-	3rd	-
2	2nd	-	-	-	-	1st	4th	5th	3rd	6th
3	5th	-	-	3rd	6th	1st	2nd	-	4th	-
4	-	-	-	1st	4th	2nd	3rd	-	5th	6th
5	-	-	3rd	-	-	1st	-	-	2nd	4th
6	-	-	-	-	-	1st	2nd	-	3rd	4th
7	5th	-	-	4th	-	2nd	1st	-	3rd	-
8	3rd	-	-	-	-	2nd	1st	-	-	-
9	5th	-	-	4th	-	1st	2nd	-	3rd	-
10	2nd	5th	7th	-	-	1st	4th	-	3rd	6th

TABLE 6.26: Commercial Dataset: Ranked importance of the input variables.

deviation for the ARD hyperparameters, whilst Table 6.28 gives the ranked importance of the input variables.

	FG	Cu	Fe	Mg	Mn	Si	SL	STT	%st.	RR
$\alpha^{-1}(\times 10^{-4})$	1.63	0.99	0.98	0.96	0.09	4.43	0.001	0.001	0.04	0.009
$\sigma_{\alpha^{-1}}(\times 10^{-3})$	0.03	0.009	0.007	0.002	0.001	0.68	0.003	0.004	0.008	0.002

TABLE 6.27: Commercial Dataset: Mean and associated standard deviation ARD Hyperparameter values for seven hidden nodes using variational leaning.

	Input variables									
Dataset	FG	Cu	Fe	Mg	Mn	Si	SL	STT	%st.	RR
1	1st	-	-	-	5th	3rd	-	4th	2nd	-
2	2nd	-	6th	-	-	3rd	-	4th	1st	5th
3	2nd	-	-	-	-	3rd	-	4th	1st	5th
4	2nd	-	5th	-	6th	3rd	-	4th	1st	-
5	1st	-	-	-	-	3rd	-	4th	2th	5th
6	1st	-	5th	-	7th	3rd	-	4th	2nd	6th
7	1st	-	6th	-	-	3rd	-	4th	2nd	5th
8	2nd	-	5th	-	-	3rd	-	4th	1st	6th
9	2nd	-	5th	-	-	3rd	-	-	1st	4th
10	1st	-	3rd	-	-	-	-	4th	2nd	-

TABLE 6.28: Commercial Dataset: Ranked importance of the input variables.

6.3.3 Bayesian Neural Network with MCMC Sampling

The evaluation of the Bayesian neural network with ARD was trained in a manner similar to that used in the evidence framework. Hybrid Monte Carlo updates for the network parameters were alternated with Gibbs sampling updates for the hyperparameters. Table 6.29 gives the mean and the associated standard deviation values for the ARD hyperparameters associated with each input variable, whilst Table 6.30 gives the ranked importance of the input variables. From these tables it is evident that the network is giving the most influence to silicon (Si) concentration, final gauge (FG) and the iron (Fe) concentration. Figure 6.1 shows the convergence of energy with increasing number of iterations.

	FG	Cu	Fe	Mg	Mn	Si	SL	STT	%st.	RR
α^{-1}	34.07	1.28	25.74	10.78	11.80	26.81	0.26	0.52	6.80	2.55
$\sigma_{\alpha^{-1}}$	54.65	2.92	33.05	28.12	24.45	35.23	0.35	1.39	9.35	6.73

Table 6.29: Commercial Dataset: Mean and associated standard deviation ARD Hyperparameter values for seven hidden nodes when using MCMC sampling.

	Input variables									
Dataset	FG	Cu	Fe	Mg	Mn	Si	SL	STT	%st.	RR
1	1^{st}	-	-	2^{nd}	-	5^{th}	-	4^{th}	-	3^{rd}
2	1^{st}	-	2^{nd}	-	3^{rd}	5^{th}	-	-	4^{th}	-
3	6^{th}	5^{th}	3^{rd}	-	-	1^{st}	-	-	4^{th}	2^{nd}
4	4^{th}	-	1^{st}	-	-	2^{nd}	-	-	3^{rd}	-
5	1^{st}	-	2^{nd}	6^{th}	-	-	-	4^{th}	3^{rd}	5^{th}
6	4^{th}	-	3^{rd}	-	2^{nd}	1^{st}	-	-	5^{th}	-
7	2^{nd}	-	1^{st}	-	4^{th}	-	-	-	-	3^{rd}
8	-	4^{th}	5^{th}	2^{nd}	3^{rd}	1^{st}	-	-	-	6^{th}
9	3^{rd}	-	-	-	4^{th}	1^{st}	-	-	2^{nd}	-
10	2^{nd}	-	-	4^{th}	-	1^{st}	-	-	3^{rd}	-

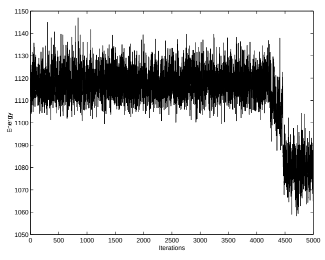
Table 6.30: Commercial Dataset: Ranked importance of the input variables for a Bayesian Neural Network trained using MCMC sampling on the randomly partitioned commercial datasets.

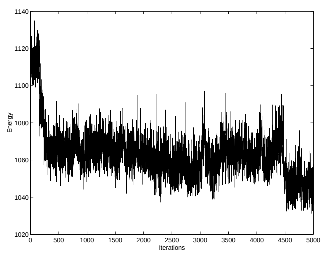
6.3.4 ARD Gaussian Process Model

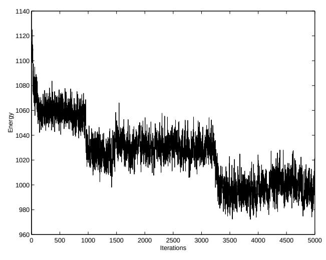
A Gaussian process model was trained on the commercial dataset, using the ARD covariance function described in Chapter 4. Table 6.31 gives the mean and the associated standard deviation values for the ARD hyperparameters associated with each input variable, whilst Table 6.32 gives the ranked importance of the input variables. From these tables the input variables rooms, distance to employment centres, access to radial highways, property tax, and the percentage of lower status in the population were selected as being important inputs.

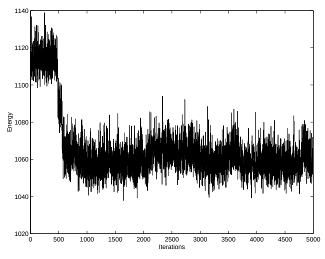
	FG	Cu	Fe	Mg	Mn	Si	SL	STT	%st.	RR
α	1.89	2.39	3.34	10.27	1.55	36.37	0.15	22.44	1.67	12.35
σ_α	0.88	2.79	1.95	6.80	1.07	12.43	0.14	15.80	0.42	5.41

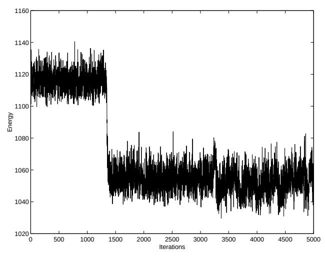
Table 6.31: Commerical Dataset: Mean and associated standard deviation using ARD Gaussian process model.

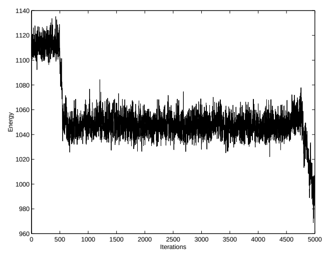

(a) Commercial Dataset 1

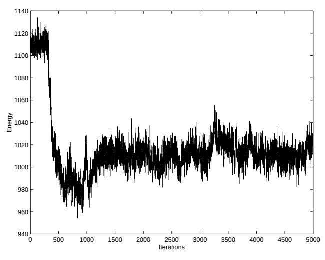

(b) Commercial Dataset 2

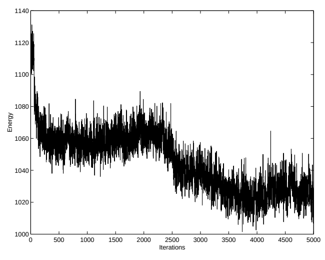

(c) Commercial Dataset 3

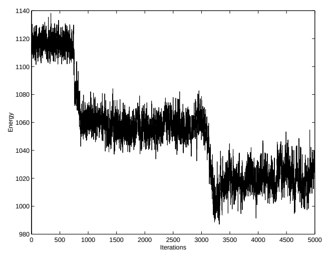

(d) Commercial Dataset 4


(e) Commercial Dataset 5


(f) Commercial Dataset 6


(g) Commercial Dataset 7


(h) Commercial Dataset 8


(i) Commercial Dataset 9

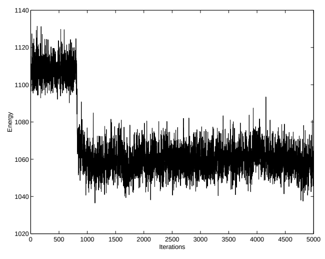

(j) Commercial Dataset 10

FIGURE 6.6: Commercial Dataset: Energy Convergence for a Bayesian Neural Network trained using MCMC sampling on the randomly partitioned commercial datasets.

6.3.5 Sparse Kernel Support Vector Machines

The SVM based SUPANOVA technique was applied to the ten input materials dataset, of the possible 1024 different terms in the full ANOVA expansion, only 12 terms were chosen as being significant. The full selection of terms is given in Table 6.33. The univariate terms selected were the bias, Mg, Si, STT, %st., the bivariate terms were FG$\otimes$Mg, FG$\otimes$RR, Cu$\otimes$Si, Fe$\otimes$Si, Mn$\otimes$SL, Si$\otimes$RR, and the trivariates terms FG$\otimes$Cu$\otimes$Si and Fe$\otimes$Si$\otimes$%.st. Examples of these are illustrated in Figure 6.7. Table 6.33 shows the stability of these terms across the ten different data partitions.

	Input variables									
Dataset	FG	Cu	Fe	Mg	Mn	Si	SL	STT	%st.	RR
1	-	-	5^{th}	3^{rd}	-	1^{st}	-	2^{nd}	-	4^{th}
2	-	-	5^{th}	4^{th}	-	1^{st}	-	2^{nd}	6^{th}	3^{rd}
3	-	4^{th}	-	2^{nd}	-	3^{rd}	-	1^{st}	-	5^{th}
4	-	-	5^{th}	4^{th}	-	1^{st}	-	3^{rd}	-	2^{nd}
5	7^{th}	6^{th}	5^{th}	4^{th}	-	1^{st}	-	2^{nd}	-	3^{rd}
6	-	-	5^{th}	4^{th}	-	1^{st}	6^{th}	3^{rd}	-	2^{nd}
7	6^{th}	-	5^{th}	4^{th}	-	1^{st}	-	3^{rd}	7^{th}	2^{nd}
8	6^{th}	-	5^{th}	4^{th}	-	1^{st}	-	3^{rd}	-	2^{nd}
9	7^{th}	6^{th}	5^{th}	4^{th}	-	1^{st}	-	2^{nd}	-	3^{rd}
10	6^{th}	5^{th}	-	4^{th}	-	2^{nd}	-	1^{st}	7^{th}	3^{rd}

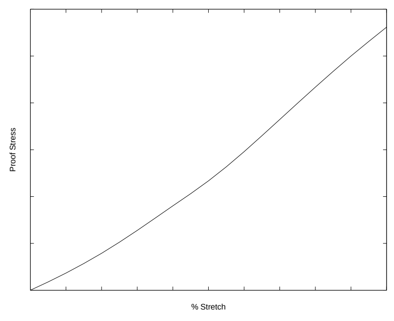
Table 6.32: Commercial Dataset: Ranked importance of the input variables when using an ARD Gaussian process model.

	Dataset									
Components	1	2	3	4	5	6	7	8	9	10
Cu	$-$	$\times$	$\times$	$-$	$-$	$-$	$-$	$-$	$-$	$\times$
Mg	$\times$	$-$	$-$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$-$
Si	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$
STT	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$
%st.	$\times$	$\times$	$-$	$\times$	$\times$	$-$	$\times$	$\times$	$\times$	$\times$
FG⊗Mg	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$
FG⊗%st.	$-$	$\times$	$\times$	$\times$	$\times$	$-$	$\times$	$\times$	$\times$	$\times$
FG⊗RR	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$
Cu⊗Si	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$
Fe⊗Si	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$-$
Mn⊗SL	$\times$	$\times$	$-$	$\times$	$\times$	$-$	$\times$	$\times$	$\times$	$-$
Si⊗RR	$\times$	$\times$	$-$	$\times$	$\times$	$-$	$\times$	$\times$	$\times$	$-$
FG⊗Cu⊗Si	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$
FG⊗Mg⊗%st.	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$\times$	$-$
Cu⊗Mg⊗%st.	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$\times$	$-$
Fe⊗Si⊗%st.	$\times$	$\times$	$-$	$\times$	$\times$	$-$	$\times$	$\times$	$\times$	$\times$
Fe⊗Si⊗RR	$-$	$-$	$\times$	$-$	$\times$	$\times$	$-$	$-$	$-$	$\times$
Fe⊗SL⊗RR	$-$	$\times$	$-$	$-$	$-$	$-$	$-$	$-$	$\times$	$-$
Fe⊗Si⊗SL⊗RR	$-$	$-$	$-$	$-$	$-$	$-$	$\times$	$-$	$-$	$-$

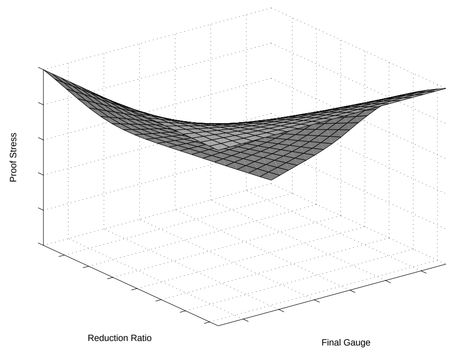
Table 6.33: Commercial Dataset: Input Selection via ANOVA decomposition in a Support Vector Machine.

6.3.6 Commercial Dataset: BISKIT Evaluation

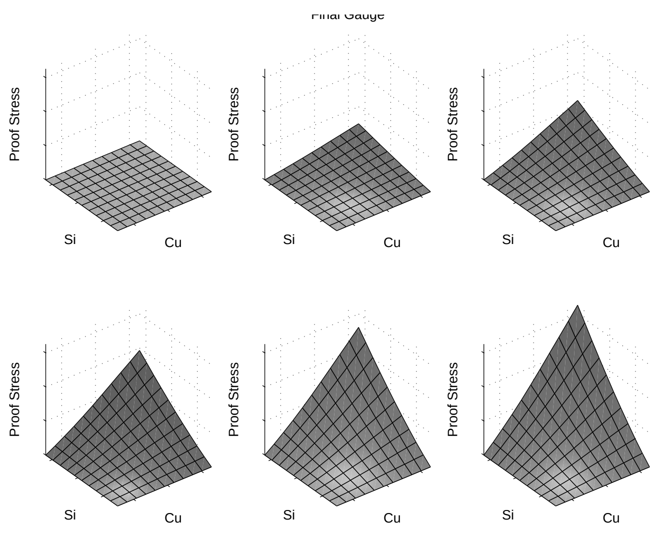
To assess the performance of the BISKIT algorithm on another real world modelling problem, the algorithm was run on the commercial dataset. As with the AMPG and Boston house price problems, given the high computational cost associated with the BISKIT algorithm (in particular the inversion of the kernel matrix as part of the hyper-parameter reestimation described in Chapter 5), the final hyperparameter values from



(a) Univariate interaction term



(b) Bivariate interaction term



(c) Trivariate interaction term

FIGURE 6.7: Commecial Dataset: Examples of univariate, bivariate and trivariate interaction terms obtained from SUPANOVA applied to the commerical materials dataset.

the SUPANOVA algorithm were used as starting values for the BISKIT algorithm. Table 6.34 gives the mean and the associated standard deviation for the chosen terms, whilst Table 6.35 gives the ranked importance of the ANOVA terms.

	Si	RR	$FG \otimes Mg$	$FG \otimes Si$	$Cu \otimes Si$	$Cu \otimes STT$	$Si \otimes RR$
n	0.35	2.15	1.58	0.43	0.42	1.00	1.39
σ_n	0.15	2.31	1.51	0.09	0.08	0.007	0.73

TABLE 6.34: Commerical Dataset: Mean and associated standard deviation using the BISKIT algorithm.

Across the ten random data partitions on which the BISKIT algorithm was run, the algorithm is able to distinguish between relevant and irrelevant inputs.

	Dataset									
Components	1	2	3	4	5	6	7	8	9	10
Si	7^{th}	6^{th}	7^{th}	6^{th}	7^{th}	–	6^{th}	2^{nd}	6^{th}	2^{nd}
RR	2^{nd}	1^{st}	1^{st}	2^{nd}	2^{nd}	1^{st}	1^{st}	1^{st}	2^{nd}	1^{st}
$FG \otimes Mg$	3^{rd}	3^{rd}	3^{rd}	5^{th}	3^{rd}	2^{nd}	4^{th}	3^{rd}	4^{th}	3^{rd}
$FG \otimes Si$	5^{th}	5^{th}	6^{th}	4^{th}	5^{th}	6^{th}	7^{th}	4^{th}	7^{th}	4^{th}
$Cu \otimes Si$	6^{th}	7^{th}	5^{th}	7^{th}	6^{th}	5^{th}	5^{th}	6^{th}	3^{rd}	6^{th}
$Cu \otimes STT$	1^{st}	2^{nd}	2^{nd}	3^{rd}	3^{rd}	3^{rd}	3^{rd}	5^{th}	5^{th}	5^{th}
$Si \otimes RR$	4^{th}	4^{th}	4^{th}	1^{st}	1^{st}	4^{th}	2^{nd}	7^{th}	1^{st}	7^{th}

Table 6.35: Commerical Dataset: Ranked importance of the input variables when using the BISKIT algorithm for ANOVA hyperparameter determination.

6.4 Computational Complexity of Leaning Real Data

A number of interpretable learning algorithms have been applied to ”real-world” datasets. A discussion of the computational complexity of these methods is of paramount interest for these techniques to be widely applicable. The principle disadvantage of the evidence framework is in the computational complexity of the training phase. A key component to this framework is the evaluation and storage of the Hessian matrix. The Hessian is required for the computation of error bars on the network predictions, and also as part of the hyperparameter determination approach. To repeatedly compute and invert the Hessian matrix requires $\mathcal{O}(N^2)$ and $\mathcal{O}(N^3)$ complexity respectively. For the largest “real-world” dataset considered in this chapter, the Boston house price dataset (that consisted of twelve input variables), this makes training considerably slow. The computational effort for the AMPG dataset, that only contained six input variables, was lower.

The McBISKIT algorithm applies MCMC to determine the ARD hyperparameters in a similar manner to that described in chapter 5. It consists of choosing N samples in an M-dimensional space resulting in an error term that decreases as $N^{-1/2}$. However, as (MacKay, 1999a) observes the MCMC approach can be considered to be the most computationally demanding hyperparameter determination method. As part of an MCMC implementation, it is important to determine how long the simulations should be run for, and to discard a number of initial ’burn-in’ iterations (Gilks et al., 1996). Saving all simulations from an MCMC run can consume a large amount of storage, especially when consecutive iterations are highly correlated necessitating a long simulation. Raftery and Lewis (1996) have proposed an alternative method whereby they only save every k^{th} iteration ($k > 1$), a process they refer to as *thinning* the chain. The advantage of this approach is that it reduces the amount of data often saved from an MCMC run. However, a limitation of this approach is that it requires the value of k to be chosen in advance, as such for chains that do not ’mix’ well (see Section 3.7.7.1) this approach may not alleviate the computational expense.

The computational complexity of variational learning can be considered to be a hybrid of both the evidence and the MCMC framework. The BISKIT algorithm developed in chapter 5, has the additional disadvantage that in addition to running time, the algorithm requires the inversion of a full kernel matrix which is an order N^3 process. An analogy can be drawn between the number of samples required to minimise the KL-divergence, and those needed to converge to an acceptable solution in the MCMC framework. Minka (2001) and Lawrence (2000) argue that the evaluation of Jensen's inequality in variational Bayesian learning is also computationally demanding. However, for the SUPANOVA algorithm, in the quadratic loss case the solution for $\boldsymbol{\alpha}^*$ is given by a matrix inversion, and for $\boldsymbol{c}^*$ by a bound constrained quadratic program. In the ϵ-insensitive case the solution for $\boldsymbol{\alpha}^*$ is given by a box constrained quadratic program, and for $\boldsymbol{c}^*$ by a bound constrained linear program with linear constraints. Consequently, they can all be solved readily using a standard quadratic programming optimiser (Mészáros, 1998).

Chapter 7

Conclusions

Advanced inductive methods are increasingly being used for modelling tasks, however, they often overlook the need to provide an interpretable solution. In the last few years many researchers have tried to address this shortcoming from a number of different angles, focusing primarily on selecting inputs which are relevant in predicting the output. An alternative approach, which is advocated in this thesis, is to decompose the model structure into smaller more interpretable portions. Transparent data modelling methods are in demand because they allow a model to be assessed not only on predictive accuracy, but also to be validated and interpreted using expert knowledge. This thesis has developed an interpretable modelling algorithm based on an additive ANOVA spline model capable of deployment within a kernel method. This concluding chapter summarises the work presented in this thesis and suggests some future directions.

7.1 Summary of Work

The aim of this thesis has been to explore the construction of data driven models that are mathematically well founded and yet have the flexibility to model complex phenomena. An important component of this has been to provide an interpretable model allowing structural information to be derived from the model assisting in model selection and model validation. This work has been concerned with the parameterisation of the models being able to perform a continuous search over a large area of the model hypothesis space, and that the parameters of the model should be interpretable, both to facilitate the incorporation of prior beliefs and to allow a deeper understanding of the data generating mechanism. This thesis compares a number of interpretable modelling methods and benchmarks them on artificial and 'real world' modelling problems. The question we are now concerned with is, has interpretable data modelling lived up to our expectations?

In chapter 1 the interpretable data modelling problem was introduced, and its applicability to a wide range of problems was motivated. We argued that interpretable data modelling algorithms are intuitively appealing because they provide a confidence measure in a model by allowing the model structure to be assessed using expert or prior knowledge. They also provide a framework into which expert knowledge can be easily integrated.

Having motivated the need for interpretable data models, chapter 2 introduced the problem of learning from data. Since the learning problem is inherently ill-posed, the concepts of model and structural regularisation were introduced which try to convert the problem to one that is well posed. The features that aid in introducing model interpretability were also reviewed. The limitations of a number of existing interpretable modelling algorithms were also discussed most notably their convergence to local minima and their ease of interpretation.

As Gibbs (1997) observes the easiest way of introducing interpretability into a learning algorithm is to use learning algorithms where the parameters and the related hyperparameters have a clearly interpretable meaning. In chapter 3 Bayesian inference was introduced. The Bayesian learning approach is based upon the expression of this knowledge in terms of a probability distribution. In general, these probabilities can be interpreted as expression of our degrees of belief in the various possibilities. Interpretable modelling is of particular interest in Bayesian learning since the knowledge derived from a model structure can aid in the assignment of Bayesian prior distributions.

The Bayesian method of Automatic Relevance Determination (ARD) has been proposed by MacKay (1994); Neal (1995) as a method of introducing interpretability into neural network models which are regarded as being *black-box* models (Ljung, 1987). The use of Bayesian methods, their application is not always straightforward. The problem centres around the mathematical complexity that often occurs in Bayesian approaches. Approximations often have to be made to avoid the intractable high dimensional integrals that typically exist. In chapter 3, the three current approximation methods for dealing with such high-dimensional integrals are reviewed and their performance with respect to model interpretability, via ARD, was illustrated on two synthetic problems. The datasets which were chosen illustrate two typical problems that occur when modelling "real world" data, i.e. learning in the presence of highly correlated input variables and learning in the presence of irrelevant inputs. The performance of each of the approximation methods on the synthetic problems was evaluated, and discussed the applicability of their inherent assumptions.

Kernel based methods have received a large amount of attention due to a number of attractive features and promising empirical performance on a range of datasets. These methods were reviewed in chapter 4. The solution to a kernel method is a weighted sum of kernel functions. A consequence of this is that the resulting model structure is opaque due to the large number of terms that typically exist in this expansion. This thesis has been concerned with addressing this issue by introducing interpretability into kernel based methods. Chapter 5 highlighted the development of three novel algorithms.

ANOVA Splines are an attractive choice for modelling (Wahba, 1990a) due to their ability to approximate arbitrary functions. In this work we motivate a sparse subset of terms is selected by introducing a separate hyperparameter (or regularisation coefficient) for every term of a complete ANOVA expansion. A number of approaches exist within the signal processing community to develop a sparse model structure from a complex model structure. In chapter 5 these methods are reviewed, and their applicability to kernel methods is evaluated. A new algorithm is developed which combines the representational advantage of a sparse ANOVA decomposition, with the good generalisation ability of a kernel machine. It achieves this by employing two forms of regularisation: a 1-norm based structural regulariser to enforce transparency, and a 2-norm based regulariser to control smoothness. The resulting model structure can be visualised showing the overall effects of different inputs, their interactions, and the strength of the interactions. Using the ideas of Bayesian hyperparameter determination described in chapter 3, ANOVA kernel re-estimation formulae are derived using variational learning giving rise to a novel algorithm (Bayesian Interpretable Sparse Kernel Inference Technique (BISKIT). A limitation of the variational learning approach is that the factorisation assumption over parameters and hyperparameters is often incorrect. For the BISKIT algorithm we show that since the parameters and hyperparameters are independent this factorisation assumption holds and allows the formulation of separate prior distributions. Exact sampling using Monte Carlo sampling is also used to determine the hyperparameters associated with the ANOVA kernel giving rise to the Monte-Carlo BISKIT (McBISKIT) algorithm. The algorithms were evaluted on the same datasets as the ARD approach in chapter 3.

For model interpretability to be of use in a wider sense, in chapter 6 we demonstrated the performance of all the advanced interpretable modelling techniques on three datasets. In conclusion, this thesis has shown that interpretable modelling algorithms can be used to provide structural information about a constructed model, exploiting prior knowledge and assisting in model selection and model validation.

7.2 Future Work

A large part of this thesis has been concerned with the description and illustration of alternative mathematical framework to allow incorporation of interpretability into data models. Whilst some progress has been made in describing the theoretical and practical aspects of interpretable modelling there is much scope for further investigation. In this section some aspects of the current work which could form the basis for additional investigation are discussed.

7.2.1 Computational Requirements

The major limitations of classical modelling algorithms, such as neural networks, are their convergence to local minima, sensitivity to parameter initialisation and lack of parameter interpretability. An important component of any learning algorithm is its computational complexity.

A limitation of the ANOVA spline approach used in this thesis is that there are a large number of ANOVA components (2^F, where F is the number of inputs) that need to be estimated from the data. The examples in this thesis used datasets that are restricted considerably in size. The largest dataset used for training was that for the Boston house price with twelve inputs. In many real world applications, for example bioinformatics problems, there can be many hundreds or thousands of input variables. This then makes the interpretable modelling approaches developed in this thesis infeasible.

In the SUPANOVA approach a quadratic programming routine is employed to solve optimisation problem. An interesting approach would be to use the Sequential Minimal Optimisation (SMO) approach developed by (Platt, 1998) and compare performance. The motivations for approximation strategies, such as SMO, was for application to very large datasets and hence it will be an interesting avenue of further research to compare the performance of this approach.

In the BISKIT algorithm a complete ANOVA matrix of dimensionality $N \times N$ (where N is the number of data points), needs to be inverted many times. This is prohibitive even for small datasets and hence methods such as the Nystroem method developed by Williams and Seeger (2000) should be employed. In McBISKIT despite MCMC sampling being independent of the dimensionality of the sample space, it does affect convergence. Techniques such as reversible jump MCMC may be attractive and work by Brooks has even established a convergence diagnostic and tells you what proposal distribution to choose. Further developments on MCMC are being made in the statistics community and these ideas can be exploited in machine learning.

7.2.2 Alternative Kernel Functions

Many of the ideas for model interpretability expressed in this thesis are drawn from the work of splines. When dealing with interpretability within kernel methods only two kernel functions (namely the 'ARD kernel' used extensively in Gaussian processes) and the ANOVA kernel used in this thesis have been proposed. An interesting area of research would be to find other interpretable kernels. Much work has been done in the statistics community on learning from data, and it is wholely possible that alternative kernel functions have been proposed. An alternative approach which is being considered by (Cristianini and Shawe-Taylor, 2000) is to develop kernels from the data that is available.

7.2.3 Further Applications

In this thesis we have illustrated the performance of interpretable modelling algorithms with respect to some interesting applications. However, these were by no means exhaustive. An interesting avenue of further research would be to consider application of the interpretable modelling algorithms to a wider range of problems. As the volume of data available on the internet continues to increase, the ability to perform feature selection and to develop interpretable models will become more attractive. As Cristianini and Shawe-Taylor (2000) observe large collections of digital images are becoming freely available over the internet, or in specialised databases. Furthermore, the generation of images has become extremely cheap and is exploited in several applications. The ability to extract information from medical images for use in medical diagnosis is an attractive area of research. Given the vast quantity of medical information that exists, collaborations with medical researchers will be needed to evaluate the performance of the interpretable modelling approaches.

7.2.4 Committees of Interpretable Models

There has been considerable interest recently in *voting methods* for pattern recognition, which predict the label of a particular example using a weighted vote over a set of base classifiers. For example, AdaBoost (Freund and Schapire, 1997) and Bagging (Breiman, 1996) have been found to give significant performance improvements over algorithms for the corresponding base classifiers (Freund and Schapire, 1996). Recent work by (Mason et al., 2000) has extended some of these ideas to an SVM resulting in an algorithm they term MarginBoost. An interesting avenue of further research would be to develop these ideas in an interpretable modelling framework. Given a set of interpretable models an AdaBoost or Bagging algorithm could be constructed from these models which results in a model that has improved interpretability over the individual base interpretable models. Such work would require expert prior knowledge to assess whether the AdaBoost or Bagging generated interpretable model was more accurate than any of the base interpretable models.

A related approach which could also be employed in a similar framework to that described above is to use the Bayesian Committee Machine (BCM) of (Tresp, 2000). The Bayesian committee machine (BCM) is a novel approach to combining estimators which were trained on different data sets. The BCM can be applied to the combination of Gaussian process regression and related systems such as regularization networks and smoothing splines for which the degrees of freedom increase with the number of training data. Combination of interpretable models within this framework may provide additional structural information.

In this thesis, we have developed and evaluated a set of interpretable learning algorithms that can be applied to many diverse problems. There still remain a number of open problems relating to improving the convergence and computational requirements of interpretable modelling algorithms that need to be addressed for interpretable modelling to become more effective in a wide variety of applications.

Appendix A

Functional Analysis

The following definitions are used extensively in the kernel methods community and can be found in any book on functional analysis. They are reviewed here for the sake of completeness.

Definition A.1 (Positive Definiteness). A matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ is positive definite if the quadratic form $\mathbf{x}^T \boldsymbol{A} \mathbf{x} > 0$.

Theorem A.2. *If $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ is positive definite and $\boldsymbol{X} \in \mathbb{R}^{N \times k}$ has rank k, then $\boldsymbol{B} = \boldsymbol{X}^T \boldsymbol{A} \boldsymbol{X} \in \mathbb{R}^{k \times k}$ is also positive definite.*

Corollary A.3. *If $\boldsymbol{A}$ is positive definite then all its principal submatrices are positive definite. In particular all the diagonal entries are positive.*

Corollary A.4. *If $\boldsymbol{A}$ is positive definite then the factorisation $\boldsymbol{A} = LDM^T$ exists and $D = diag(d_1, \ldots, d_N)$ has positive diagonal entries.*

Definition A.5 (l_p-norms). If we assume $\mathcal{X} = \mathbb{R}^N$ the l_p-norm can be defined as,

$$\|\boldsymbol{x}\|_{l_p^N} \overset{\text{def}}{=} \|\boldsymbol{x}\|_p = \begin{cases} \left(\sum_{i=1}^N |x_i|^p\right)^{1/p} & if \quad 0 < p < \infty \\ \max_{i=1,\ldots,N} |x_i| & if \quad p = \infty \end{cases}$$

Definition A.6 (Inner Product Space). Given a vector space $\mathcal{X}$, an inner product space $\mathcal{X}$ is defined by the tuple $(\mathcal{X}, \langle \cdot, \cdot \rangle_{\mathcal{X}})$, where $\langle \cdot, \cdot \rangle_{\mathcal{X}} : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is called an inner product and satisfies the following properties: for all $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathcal{X}$ and $c, d \in \mathbb{R}$ the following quantities are defined,

$$\begin{aligned} \langle \boldsymbol{x}, \boldsymbol{x} \rangle_{\mathcal{X}} &\geq 0 \quad and \quad \langle \boldsymbol{x}, \boldsymbol{x} \rangle_{\mathcal{X}} = 0 \quad \Leftrightarrow \boldsymbol{x} = \boldsymbol{0}, \\ \langle c\boldsymbol{x} + d\boldsymbol{y}, \boldsymbol{z} \rangle_{\mathcal{X}} &= c\langle \boldsymbol{x}, \boldsymbol{z} \rangle_{\mathcal{X}} + d\langle \boldsymbol{y}, \boldsymbol{z} \rangle_{\mathcal{X}}, \\ \langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\mathcal{X}} &= \langle \boldsymbol{y}, \boldsymbol{x} \rangle_{\mathcal{X}}. \end{aligned} \tag{A.1}$$

It therefore follows that each inner product space is a normed space by $\|\boldsymbol{x}\|_{\mathcal{X}} = \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle_{\mathcal{X}}}$. If $\mathcal{X} = \mathbb{R}^N$ the Euclidean inner product can be defined by $\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\mathbb{R}^N} \overset{\text{def}}{=} \boldsymbol{x}^T \boldsymbol{y} = \sum_{i=1}^N x_i y_i$

Definition A.7 (Hilbert Space). A Hilbert Space $\mathcal{X}$ is a complete separable inner product space. A space is called separable if there exists a countable subset $X \subseteq \mathcal{X}$, such that every element of $\mathcal{X}$ is the limit of a sequence of elements of X. A space is called complete if each Cauchy sequence converges.

Definition A.8 (Reproducing Kernel Hilbert Space).

Consider a Mercer kernel k, and a probability space $\mathcal{P}$. From Mercer's theorem k can be expanded into,

$$k(\boldsymbol{x}, \boldsymbol{z}) = \sum_{i=1}^{\infty} \lambda_i \psi_i(\boldsymbol{x}) \psi_i(\boldsymbol{z}) \qquad \forall i : \int_x |\psi_i(\boldsymbol{x})|^2 d\mu(\boldsymbol{x}) = 1 \tag{A.2}$$

Suppose, we consider the space $\mathcal{F}$ of all linear functions over mapped input points from $\mathcal{X}$,

$$f_a \in \mathcal{F} \Leftrightarrow f_a(\boldsymbol{x}) = \sum_{i=1}^{\infty} a_i \psi_i(\boldsymbol{x}) \tag{A.3}$$

where the inner product in $\mathcal{F}$ shall be given by,

$$\langle f_a, f_b \rangle_{\mathcal{F}} = \sum_{i=1}^{\infty} \frac{a_i b_i}{\lambda_i} \tag{A.4}$$

Let $\boldsymbol{w} = (\lambda_1 \psi_1(\boldsymbol{z}), \ldots, \lambda_i \psi_i(\boldsymbol{z}))'$ be the image of an input point $\boldsymbol{z}$. Since,

$$f_{\boldsymbol{w}}(\cdot) = \sum_{i=1}^{\infty} \lambda_i \psi_i(\boldsymbol{z}) \psi_i(\cdot) = k(\boldsymbol{z}, \cdot) \tag{A.5}$$

we see that $k(\boldsymbol{z}, \cdot) \in \mathcal{F}$. As a consequence, functions of the form,

$$f(\boldsymbol{x}) = \sum_{i=1}^{m} \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}) \qquad m \in \mathbb{N}, \boldsymbol{x}_i \in \mathcal{X}, \boldsymbol{\alpha} \in \mathbb{R}^m \tag{A.6}$$

are in the space $\mathcal{F}$. If we take the inner product of a function $f_a \in \mathcal{F}$ with $k(\boldsymbol{z}, \cdot), \boldsymbol{z} \in \mathcal{X}$ the following can be obtained,

$$\langle f_a, k(\boldsymbol{z}, \cdot) \rangle_{\mathcal{F}} = \sum_{i=1}^{\infty} \frac{a_i \lambda_i \psi_i(\boldsymbol{z})}{\lambda_i} = \sum_{i=1}^{\infty} a_i \psi_i(\boldsymbol{z}) = f_a(\boldsymbol{z}) \tag{A.7}$$

known as the *reproducing property* of the kernel k. This also implies that,

$$\mathcal{F} = \left\{ \sum_{i=1}^{m} \alpha_i k(\boldsymbol{x}_i, \cdot) : m \in \mathbb{N}, (\boldsymbol{x}_1, \ldots, x_m) \in \mathcal{X}^m, \alpha_i \in \mathbb{R} \right\} \tag{A.8}$$

since $f \perp \mathcal{F}$ implies that for all $\boldsymbol{z} \in \mathcal{X}, \langle f, k(\boldsymbol{z}, \cdot) \rangle_{\mathcal{F}} = 0 = f(\boldsymbol{z})$ implying $f = 0$. For two functions $f(\cdot) = \sum_{i=1}^{p} \alpha_i k(\boldsymbol{x}_i, \cdot)$ and $g(\cdot) = \sum_{j=1}^{q} \beta_j k(\boldsymbol{z}_j, \cdot)$ in $\mathcal{F}$ the inner product is given by,

$$\begin{aligned}\langle f,g\rangle_{\mathcal{F}} &= \left\langle \sum_{i=1}^{p}\alpha_i k(\boldsymbol{x}_i,\cdot), \sum_{j=1}^{q}\beta_j k(\boldsymbol{z}_j,\cdot)\right\rangle_{\mathcal{F}} \\ &= \sum_{i=1}^{p}\alpha_i \sum_{j=1}^{q}\beta_j \langle k(\boldsymbol{x}_i,\cdot), k(\boldsymbol{z}_j,\cdot)\rangle_{\mathcal{F}} \\ &= \sum_{i=1}^{p}\alpha_i \sum_{j=1}^{q}\beta_j k(\boldsymbol{z}_j,\boldsymbol{x}_i) = \sum_{j=1}^{q}\beta_j \sum_{i=1}^{p}\alpha_i k(\boldsymbol{x}_i,\boldsymbol{z}_j) \\ &= \sum_{i=1}^{p}\alpha_i g(\boldsymbol{x}_i) = \sum_{j=1}^{q}\beta_j f(\boldsymbol{z}_j), \qquad \text{(A.9)}\end{aligned}$$

showing that the definition of the inner product is independent of the particular representation of the functions. By the reproducing property, Equation A.7, we know that the evaluation functional $T_{\boldsymbol{x}}[f] = \langle f, k(\boldsymbol{x},\cdot)\rangle_{\mathcal{F}} = f(\boldsymbol{x})$ is linear and bounded. These are the defining properties of a *reproducing kernel Hilbert space (RKHS)*.

Bibliography

J. H. Ahlbery, E. N. Nilson, and J. I. Walsh. *The Theory of Splines and their Applications.* Academic Press Publishers, 1968.

H. Akaike. A new look at statistical model identification. *IEEE Transactions on Automatic Control*, 19:716–723, 1974.

D. Allen. The relationship between variable selection and data augmentation and a method for prediction. *Technometrics*, 16:125–127, 1974.

D. Amos and M. Slater. Polynomial and spline approximations by quadratic programming. *Communications of the Association for Computing Machinery*, 12:379ff, 1969.

P. M. Anselone and P. J. Laurent. A general method for the construction of interpolating or smoothing spline functions. *Numerische Mathematik*, 12:66–82, 1968.

N. Aronszajn. Theory of reproducing kernels. *Trans. Amer. Math. Soc.*, 686:337–404, 1950.

S. Asmussen, P. W. Glynn, and H. Thorisson. Stationarity detection in the initial transient problem. *ACM Transactions on Modelling and Computer Simulation*, 2: 130–157, 1992.

H. Attias. *Inferring Parameters and Structure of Latent Variables Models by Variational Bayes.* In Proc. 15th Conf. on Uncertainty in Artificial Intelligence, 1999.

D. Barber and C. M. Bishop. Ensemble learning for multi-layer networks. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10. The MIT Press, 1998.

R. E. Bellman. *Adaptive Control Processes.* Princeton University Press, 1961.

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

J. Bernardo and A. F. M. Smith. *Bayesian Theory.* John Wiley and sons, 1996.

J. Besag. Markov chain monte carlo for statistical inference. Technical Report Available from http://www.csss.washington.edu/Papers, University of Washington, 2000.

C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

C. M. Bishop. *Variational PCA.* In Proc. Ninth Int. Conf. on Artificial Neural Networks, 1999.

C. M. Bishop and M. E. Tipping. A hierarchical latent variable model for data visualisation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3): 281–293, 1996.

C. M. Bishop and M. E. Tipping. The variational relevance vector machine. In *In Proceedings of the 16th Conference in Uncertainty in Artificial Intelligence*, 2000.

C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

P. S. Bradley, O. L. Mangasarian, and W. N. Street. Feature selection via mathematical programming. *INFORMS Journal on Computing*, 10(2):209–217, 1998.

L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression trees.* Wadsworth Inc., 1984.

S. P. Brooks. Markov chain monte carlo method and its application. *The Statistician*, 47:69–100, 1998.

M. Brown and C. J. Harris. *Neurofuzzy Adaptive Modelling and Control.* Prentice-Hall Publishers, 1994.

A. Buja, T. Hastie, and R. Tibshirani. Linear smoothers and additive models. *Annals of Statistics*, 17:453–510, 1989.

W. Buntine. Theory refinement on Bayesian networks. In B. D. D'Ambrosio, P. Smets, and P. P. Bonissone, editors, *Proc. Seventh Annual Conference on Uncertainty Artificial Intelligence*, pages 52–60, San Francisco, CA, 1991. Morgan Kaufmann Publishers.

C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2, 1998.

C. J. C. Burges. Geometry and invariance in kernel based methods. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 89–116, Cambridge, MA, 1999. MIT Press.

S. Chen. *Basis Pursuit.* PhD thesis, Department of Statistics, Stanford University, 1995.

S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. Technical Report 479, Stanford University, 1995.

S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 41(2):33–61, 1999.

Z. Chen. Fitting multivariate regression functions by interaction spline models. *Journal of the Royal Statistical Society B*, 55:473–491, 1993.

V. Cherkassky and F. Mulier. *Learning from Data: Concepts, Theory and Methods.* John WIley Publishers, 1998.

C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:1–25, 1995.

D. D. Cox. Asymptotics for m-type smoothing splines. *Annals of Statistics*, 11:530–551, 1983.

P. Craven and G. Wahba. Smoothing noisy data with spline functions. *Numerical Mathematics*, 31:377–403, 1979.

N. A. C. Cressie. *Statistics for Spatial Data.* John-Wiley Publishers, 1993.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines.* Cambridge University Press, 2000.

P. Dagum and M. Luby. Approximating probabilistic inference in bayesian belief networks is np-hard. *Artificial Intelligence*, 60:141–153, 1993.

I. Daubechies. *Ten Lectures on Wavelets.* SIAM, 1992.

A. P. Dawid. Conditional independence in statistical theory (with discussion). *Journal of the Royal Statistical Society B*, 41(1):1–31, 1979a.

A. P. Dawid. Some misleading arguments concerning conditional independence. *Journal of the Royal Statistical Society B*, 41(2):249–252, 1979b.

P. A. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach.* Prentice-Hall, 1982.

L. Devroye. *Non-Uniform Random Variate Generation.* Springer-Verlag, 1986.

S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid monte carlo. *Physics Letters B*, (195):216–222, 1987.

R. O. Duda, D. G. Stork, and P. E. Hart. *Pattern Classification and Scene Analysis Part I: Pattern Classification.* John Wiley and sons, 2000.

R. Dybowski and S. J. Roberts. *Confidence Intervals and Prediction Intervals for Feed-Forward Neural Networks.* In: Clinical Applications of Artificial Neural Networks, Cambridge University Press, 2000.

H. W. Engl. Regularisation methods for the stable solution of inverse problems. *Surveys Math. Indust.*, 3:71–143, 1993.

T. Evgeniou and M. Pontil. On the v_γ dimension for regression in reproducing kernel hilbert spaces. Technical Report AI Memo No. 1656, Massachusetts Institute of Technology, 1999.

T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. In A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 171–204, Cambridge, MA, 2000. MIT Press.

R. P. Feynman. *Statistical Mechanics.* W. A. Benjamin Publishers, 1972.

Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Proc. 13th International Conference on Machine Learning*, pages 148–146. Morgan Kaufmann, 1996.

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1): 119–139, 1997.

J. H. Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, 19 (1):1–141, 1991.

K. Fukunga. *Statistical Pattern Recognition.* Academic Press, 1990.

J. B. Gao, S. R. Gunn, C. J. Harris, and M. Brown. A probabilistic framework for svm regression and error bar estimation. *Machine Learning*, 2000.

S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992.

Z. Ghahramani. *Factorial Learning and the EM Algorithm.* In G. Tesauro, D.S. Touretzky and T.K. Leen (EDS), Advances in Neural Information Processing (NIPS) 7, MIT Press, 1994.

Z. Ghahramani and M. J. Beal. *Variational Inference for Bayesian Mixtures of Factor Analysers.* In Advances in Neural Information Processing (NIPS) 12, MIT Press, 2000.

Z. Ghahramani and M. I. Jordan. Factorial hidden markov models. *Machine Learning*, 29:245–273, 1997.

M. Gibbs. *Bayesian Gaussian Processes for Regression and Classification.* PhD thesis, University of Cambridge, Available from http://wol.ra.phy.cam.ac.uk, 1997.

M. Gibbs and D. J. C. MacKay. Efficient implementation of gaussian processes. Technical Report Available from http://wol.ra.phy.cam.ac.uk, University of Cambridge, 1997.

W. R. Gilks, S. Richardson, and D. J. Spiegelhalter. *Markov Chain Monte Carlo in Practice.* Chapman and Hall, 1996.

F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7:219–269, 1995.

F. Girosi and T. Poggio. Neural Networks and the best Approximation Property. *Biol. Cybern.*, 63:169–176, 1990.

I. J. Good and R. A. Gaskins. Non-parametric roughness penalities for probability densities. *Biometrika*, 58:255–277, 1971.

P. J. Green. Iteratively reweighted least squares for maximum likelihood estimation and some robust and resistant alternatives (with discussion). *Journal of the Royal Statistical Society B*, 46:149–162, 1984.

C. W. Groetsch. *Generalised Inverses of Linear Operators.* Marcel-Dekker Publishers, 1977.

S. F. Gull. *Developments in maximum entropy data analysis. In J. Skilling (Ed.), Maximum Entropy and Bayesian Methods.* Kluwer Academic Publishers, 1989.

S. R. Gunn, M. Brown, and K. M. Bossley. Network performance assessment for neurofuzzy data modelling. In *Intelligent Data Analysis*, number 1208 in Lecture Notes in Computer Science, pages 313–323, 1997.

S. R. Gunn and J. S. Kandola. Structural modelling with sparse kernels. *Machine Learning*, Accepted, 2000.

S.R. Gunn. Support vector machines for classification and regression. Technical Report, Dept. of Electronics and Computer Science, University of Southampton, Southampton, U.K., 1998.

S. Gutjahr and C. Nautze. Extended bayesian learning. pages 321–326. Proceedings of ESANN97, European Symposium on Artificial Neural Networks, 1997.

J. F. Hadamard. *Lectures on the Cauchy Problem in Linear Partial Differential Equations.* Yale University Press, 1923.

D. Harrison and D. L. Rubinfield. Hedonic housing prices and the demand for clean air. *Journal of Enviromental Economics and Management*, (5):81–102, 1978.

A. C. Harvey. *Forecasting, Structural Time Series Models and the Kalman Filter.* Cambridge University Press, U.K., 1989.

B. Hassibi, D. G. Stork, and G. J. Wolff. Optimal brain surgeon and general network pruning. In *IEEE International Conference on Neural Networks*, pages 293–299, USA, 1992.

T. Hastie and R. Tibshirani. *Generalized additive models.* Chapman and Hall, 1990.

S. Haykin. *Neural Networks, A Comprehensive Foundation.* Macmillan, 1999.

D. Heckerman. *Learning in Graphical Models*, chapter A Tutorial on Learning with Bayesian Networkd. MIT Press, 1999.

D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995.

G. E. Hinton. Personal communication. Gatsby Computational Neuroscience Research Group, University College London, December 2000.

G. E. Hinton and D. van Camp. Keeping neural networks simple by minimising the description length of the weights. In *Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory*, pages 5–13, USA, 1993.

P. J. Huber. *Robust Statistics*. John Wiley Publishers, 1981.

D. R. Hush and B. G. Horne. Progress in supervised neural networks: What's new since Lippmann? *IEEE Signal Processing Magazine*, 10:8–39, 1993.

D. Husmeier. *Neural Networks for Conditional Probability Estimation*. Springer-Verlag Publishers, 1999.

T. Jaakkola. *Variational Methods for Inference and Estimation in Graphical Models*. PhD thesis, Massachusetts Institute of Technology, 1997.

T. S. Jaakkola and D. Haussler. Probabilistic kernel regression models. In *Proceedings of the 1999 Conference on AI and Statistics*, 1999.

E. T. Jaynes. *Probability Theory the Logic of Science*. Washington University Press, 1994.

T. Jebera and T. Jaakkola. Feature selection and dualities in maximum entropy discrimination. In *Uncertainty in Artificial Intelligence*, 2000.

M. I. Jordan. *Learning in Graphical Models*. MIT Press, 1999.

J. S. Kandola and S. R. Gunn. On the use of advanced inductive methods for knowledge extraction from complex datasets. *Submitted to Journal of Data Mining and Knowledge Discovery*, 2000.

J. S. Kandola, S. R. Gunn, and J. B. Gao. Bayesian inference using variational learning. Technical report, Southampton University, 2000.

J. S. Kandola, S. R. Gunn, I. Sinclair, and P. A. Reed. Data driven knowledge extraction of materials property prediction. USA, 1999. IEEE Intelligent Processing and Manufacturing of Materials.

J. S. Kandola, W. A. Wright, and P. Greenway. Restoring a noisy sequence of images. In *SPIE*, USA, 1998.

R. E. Kass, B. P. Carlin, A. Gelman, and R. M. Neal. Mcmc in practice: A roundtable discussion. *American Statistician*, 1999.

T. Kavli and E. Weyer. On ASMOD - an algorithm for building multivariable spline models. In G.R. Irwin K.J. Hunt and K. Warwick, editors, *Advances in Neural Networks for Control Systems*, Springer series on Advances in Industrial Control, pages 83–104. Springer Verlag, 1995.

G. Kimeldorf and G. Wahba. A correspondence between bayesian estimation on stochastic processes and smoothing by splines. *Annals of Mathematical Statistics*, 41:495–502, 1971.

S. Kirkpatrick, C. D. Gelatt, and M. P. S. Vecchi. Optimisation by simulated annealing. *Science*, 220:671–680, 1983.

P.K. Kitanidis. Parameter uncertainty in estimation of spatial functions: Bayesian analysis. *Water Resources Research*, 22:499–507, 1986.

J. Lamperti. *Stochastic Processes (Applied Mathematical Sciences 23).* Springer-Verlag, 1977.

S. L. Lauritzen. *Graphical Models.* Oxford University Press, 1995.

N. D. Lawrence. *Variational Inference in Probabilistic Models.* PhD thesis, University of Cambridge, U.K., 2000.

Y. L. LeCun, D. S. Denker, and S. A. Solla. *Optimal brain damage.* Advances in Neural Information Processing (NIPS) 2, Morgan Kauffman, 1990.

R. V. Lenth. Robust splines. *Commutative Statistics*, A6:847–854, 1977.

L. Ljung. *System identification: theory for the user.* Prentice-Hall Publishers, 1987.

D.G. Lowe. Similarity metric learning for a variable-kernel classifier. *Neural Computation - submitted*, 7:72–85, 1995.

D. J. C. MacKay. *Bayesian Modelling and Neural Networks.* PhD thesis, California Institute of Technology, 1991.

D. J. C. MacKay. Bayesian non-linear modelling for the prediction competition. *ASHRAE Transactions: Symposia*, OR-94-17-1, 1994.

D. J. C. MacKay. Ensemble learning and evidence maximisation. Techical report. University of Cambridge, 1995.

D. J. C. MacKay. Ensemble learning for hidden markov models. Technical report, University of Cambridge, 1997.

D. J. C. MacKay. *An Introduction to Markov Chain Monte Carlo Methods*, chapter I. MIT Press, 1999a.

D. J. C. MacKay. Personal communication. Cavendish Laboratory, Department of Physics, University of Cambridge, March 1999b.

S. Mallat and Z. Zhang. Matching pursuit in a time frequency dictionary. *IEEE Transactions on Signal Processing*, 41:3397–3415, 1993.

L. Mason, J. Baxter, P.L. Bartlett, and M. Frean. Functional gradient techniques for combining hypotheses. In A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 221–246, Cambridge, MA, 2000. MIT Press.

G. Matheron. Principles of geostatistics. *Economic Geology*, 58:1246–1266, 1963.

C. Mészáros. The bpmpd interior point solver for convex quadratic problems. Technical Report WP 98-8, Computer and Automation Research Institute, Hungarian Academy of Sciences, Budapest, 1998.

T. P. Minka. *Expectation Propagation for Approximate Bayesian Inference.* PhD thesis, M.I.T., 2001.

J. E. Moody and T. S. Rögnvaldsson. Smoothing regularisers for projective basis function networks. Technical Report OGI CSE TR 96-006, Dept. Computer Science and Engineering, Oregan Graduate Institute of Science and Technology, 1996.

K. P. Murphy, Y. Weiss, and M. I. Jordan. Loopy belief propagation for approximate inference: an empirical study. In *In Proceedings of Uncertainty in Artificial Intelligence (UAI)*, 1999.

I. T. Nabney. Personal communication. Neural Computing Research Group, University of Aston, March 1999.

R. M. Neal. *Bayesian Learning for Neural Networks.* Springer-Verlag Publishers, 1995.

A. O'Hagan. Curve fitting and optimal design for prediction. *Journal of the Royal Statistical Society, Ser. B*, 40:1–42, 1978.

M. J. Oldfield. *Advances in Probabilistic Modelling.* PhD thesis, University of Cambridge, U.K., 1995.

H. Omre. Bayesian kriging - merging observations and qualified guesses in kriging. *Mathematical Geology*, 19:25–39, 1987.

M. Opper and O. Winther. Gaussian processes for classification: Mean field algorithms. *Neural Computation*, 12(11), 2000.

F. O'Sullivan. The analysis of some penalized likelihood schemes. Technical Report 72, University of Wisconsin-Madison, 1983.

E. Osuna, R. Freund, and F. Girosi. Support vector machines: Training and applications. Technical Report AIM-1602, MIT A.I. Lab., 1996.

Y. Pati, R. Rezaiifar, and P. Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Proceedings of the 27 th Annual Asilomar Conference on Signals, Systems, and Computers*, 1993.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann Publishers, 1988.

W. Penny and S. J. Roberts. Bayesian classification using neural networks - how useful is the evidence framework? *Neural Networks*, 12:877–892, 1998.

W. D. Penny and S. J. Roberts. Notes on variational learning. Technical report, Oxford University, 2000.

T. A. Plate. Accuracy versus interpretability in flexible modelling: implementing a tradeoff using gaussian process models. *Behaviourmetrika special issue on Interpreting Neural Network Models*, (26):29–50, 1999.

J. C. Platt. Sequential minimal optimisation: A fast algorithm for training support vector machines. Technical report, Microsoft Research, 1998.

T. Poggio and F. Girosi. A theory of networks for approximation and learning. C.B.I.P. Memo No. 31, Center for Biological Information Processing, Whitaker College, 1989.

T. Poggio and F. Girosi. A sparse representation for function approximation. *Neural Computation*, 10(6):1445–1454, 1998.

M. Pontil, S. Mukherjee, and F. Girosi. On a novel class of loss functions for robust estimation. Ai memo, MIT Artificial Intelligence Laboratory, 1998.

S. Qian and D. Chen. Signal representation via adaptive normalized gaussian functions. *IEEE Trans. on Signal Processing*, 36(1), 1994.

J. R. Quinlan. Induction of decision trees. *Machine Learning*, (1):81–106, 1986.

A. E. Raftery and S. M. Lewis. *Implementing MCMC: In Markov Chain Monte Carlo in Practice*. Chapman and Hall Publishers, 1996.

A. V. Rao, D. J. Miller, K. Rose, and A. Gersho. A deterministic annealing approach for parsimonious design of piecewise regression models. *IEEE Trans. PAMI*, pages 8–39, Feb 1999.

C. E. Rasmussen. *Evaluation of Gaussian Processes and other Methods for Non-Linear Regression.* PhD thesis, University of Toronto, Available from http://bayes.imm.dtu.dk/pub/, 1996.

G. O. Roberts and N. G. Polson. On the geometric convergence of the gibbs sampler. *J.R. Stat. Soc. B*, 56:377–384, 1994.

L. K. Saul, T. Jaakkola, and M. I. Jordan. Mean field theory for sigmoid belief networks. *Journal of Artificial Intelligence Research*, 4:61–76, 1996.

B. Scholkopf, S. Mika, A. Smola, G. Ratsch, and K. Muller. Kernel pca pattern reconstruction via approximate pre-images. In *In L. Niklasson, M. Boden, and T. Ziemke, editors, Proceedings of the 8th International Conference on Artificial Neural Networks, Perspectives in Neural Computing, pages 147–152, Berlin, 1998. Springer Verlag.*, 1998.

B. Schölkopf, P. Y. Simard, A. J. Smola, and V. N. Vapnik. Prior knowledge in support vector kernels. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural information processings systems*, volume 10, pages 640–646, Cambridge, MA, 1998. MIT Press.

G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

B. W. Silverman. On the estimation of a probability density function by the maximum penalized likelihood method. *Annals of Statistics*, 10:795–810, 1982.

B. W. Silverman. Some aspects of the spline smoothing approach to non-parametric regression curve fitting (with discussion). *Journal of the Royal Statistical Society Series B*, (47):1–52, 1985.

A. J. Smola. *Learning with Kernels.* PhD thesis, GMD First, Available from http://www.gmd.first.de, 1998.

A. J. Smola and P. Bartlett. Sparse greedy gaussian process regression. In *Advances in Neural Information Processings Systems 13.* MIT Press, 2000.

A. J. Smola, T. Frieß, and B. Schölkopf. Semiparametric support vector and linear programming machines. In *Advances in Neural Information Processing Systems, 11.* MIT Press, 1998.

A. J. Smola and B. Schölkopf. Sparse greedy matrix approximation for machine learning. In *International Conference on Machine Learning*, 2000.

P. Sollich. Probabilistic interpretation and bayesian methods for support vector machines. In *Proceedings of ICANN'99*, pages 91–96. IEE Publications, 1999a.

P. Sollich. Probabilistic methods for support vector machines. In *NIPS 1999, to appear*, 1999b.

P. Sollich. Bayesian methods for support vector machines: Evidence and predictive class probabilities. *Machine Learning*, 2000.

M. Stitson, A. Gammerman, V. Vapnik, V. Vovk, C. Watkins, and J. Weston. Support vector regression with ANOVA decomposition kernels. In B. Schölkopf, C.J.C. Burges, and A.J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 285–292, Cambridge, MA, 1999. MIT Press.

C. Stone. Additive regression and other non-parametric models. *Annals of Statistics*, 13:689–705, 1985.

M. Stone. Cross validity choice and assessment of statistical predictions (with discussion). *Journal of the Royal Statistical Society B*, 36:111–147, 1974.

P. Sykacek, G. Dorffner, P. Rappelsberger, and J. Zeitlhofer. Evaluating confidence measures in neural network based sleep stager. Technical Report Technical Report Tr-97-21, OFAI, 1997.

P. Sykacek, I. Rezek, and S. J. Roberts. Markov chain monte carlo methods for bayesian sensor fusion. Technical Report PARG-00-10, University of Oxford, 2000.

H. H. Thodberg. A review of bayesian neural networkd with an application to near infrared spectroscopy. *IEEE Transacions on Neural Networks*, 7:56–72, 1995.

D. Thouless, P. Anderson, and R. Palmer. Solution of solvable model of a spin glass. *Phil. Mag.*, 35:593–601, 1977.

M. E. Tipping. Personal communication. Microsoft Research, Cambridge, 2000a.

M. E. Tipping. The relevance vector machine. In *Advances in Neural Information Processing Systems 12*, 2000b.

V. Tresp. A bayesian committee machine. *Neural Computation*, 2000.

V. Vapnik. *The Nature of Statistical Learning Theory.* Springer Verlag Publishers, 1995.

V. Vapnik. *Statistical Learning Theory.* Wiley Inter Science, 1998.

P. Vincent and Y. Bengio. Kernel matching pursuit. *Machine Learning*, Accepted, 2000.

G. Wahba. Bayesian confidence intervals for the cross validated smoothing spline. *Journal of the Royal Statistical Society B*, 45:133–150, 1983.

G. Wahba. A comparison of gcv and gml for choosing the smoothing parameter in the generalised splines smoothing problem. *The Annals of Statistics*, 13:1378–1402, 1985.

G. Wahba. *Spline Models for Observational Data.* Series in Applied Mathematics (SIAM), 59, 1990a.

G. Wahba. *Spline Models for Observational Data.* Series in Applied Mathematics, Vol. 59, SIAM, Philadelphia, 1990b.

G. Wahba, Y Wang, C. Gu, R. Klein, and B. Klein. *Structured machine learning for 'soft' classification with smoothing spline ANOVA and stacked tuning, testing and evaluation.* In J. Cowan, G. Tesaro and J. Alspector (EDS), Advances in Neural Information Processing (NIPS) 6, Morgan Kauffman, 1994.

A. M. Walker. On the asymptotic behaviour of posterior distributions. *Journal of the Royal Statistical Society, B*, 31(1):80–88, 1969.

S. Waterhouse, D. J. C. MacKay, and T. Robinson. *Bayesian Methods for Mixtures of Experts.* In G. Tesauro, D.S. Touretzky and T.K. Leen (EDS), Advances in Neural Information Processing (NIPS) 7, MIT Press, 1995.

S. R. Waterhouse and A. J. Robinson. Non-linear prediction of acoustic vectors using hierarchical mixtures of experts. In G. Tesauro, D.S. Touretzky, and T.K. Leen, editors, *Advances in Neural Information Processing Systems 7*, Cambridge, MA, 1997. MIT Press.

E. J. Wegman and I. W. Wright. Splines in statistics. *Journal of the American Statistical Association*, 78:351–365, 1983.

A. S. Weigend, D. E. Rumelhart, and B. Huberman. *Generalisation by weight-elimination with application to forecasting.* Advances in Neural Information Processing (NIPS) 3, Morgan Kauffman, 1991.

J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection in svms. In *Advances in Neural information processings systems 13.* MIT Press, 2000.

J. Whittaker. *Graphical Gaussian Models in Applied Multivariate Statistics.* Wiley Publilshers, 1990.

P. Whittle. On the smoothing of probability density functions. *Journal of the Royal Statistical Society B*, 20:334–343, 1958.

C. K. I. Williams. *Prediction with Gaussian Processes: From Linear Regression to Linear Prediction and Beyond*, chapter Learning and Inference in Graphical Models. MIT Press, 1998.

C. K. I. Williams and D. Barber. Bayesian classification with gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351, 1998.

C. K. I. Williams and C. E. Rasmussen. *Gaussian Processes for Regression.* In Touretsky, Mozer and Hasselmo (EDS), Advances in Neural Information Processing (NIPS) 8, Morgan Kauffman, 1996.

C. K. I. Williams and M. Seeger. Using the nyström method to speed up kernel machines. In *Advances in Neural Information Processings Systems 13*. MIT Press, 2000.

J. Wyatt. Nervous about artificial neural networks? (commentary). *The Lancet*, (346): 1175–1177, 1995.

J. S. Yedidia. An idiosyncratic journey beyond mean field theory. Technical report, MERL, http://www.merl.com/reports/TR2000-27/index.html, 2000.