Studies on the generalisation of Gaussian processes and Bayesian neural networks

FRANCESCO VIVARELLI

Doctor of Philosophy

ASTON UNIVERSITY

September 1998

ASTON UNIVERSITY

Studies on the generalisation of Gaussian processes and Bayesian neural networks

FRANCESCO VIVARELLI

Doctor of Philosophy, 1998

Thesis Summary

The assessment of the reliability of systems which learn from data is a key issue to investigate thoroughly before the actual application of information processing techniques to real-world problems. Over the recent years Gaussian processes and Bayesian neural networks have come to the fore and in this thesis their generalisation capabilities are analysed from theoretical and empirical perspectives.

Upper and lower bounds on the learning curve of Gaussian processes are investigated in order to estimate the amount of data required to guarantee a certain level of generalisation performance. In this thesis we analyse the effects on the bounds and the learning curve induced by the smoothness of stochastic processes described by four different covariance functions. We also explain the early, linearly-decreasing behaviour of the curves and we investigate the asymptotic behaviour of the upper bounds. The effect of the noise and the characteristic lengthscale of the stochastic process on the tightness of the bounds are also discussed. The analysis is supported by several numerical simulations.

The generalisation error of a Gaussian process is affected by the dimension of the input vector and may be decreased by input-variable reduction techniques. In conventional approaches to Gaussian process regression, the positive definite matrix estimating the distance between input points is often taken diagonal. In this thesis we show that a general distance matrix is able to estimate the effective dimensionality of the regression problem as well as to discover the linear transformation from the manifest variables to the hidden-feature space, with a significant reduction of the input dimension. Numerical simulations confirm the significant superiority of the general distance matrix with respect to the diagonal one.

In the thesis we also present an empirical investigation of the generalisation errors of neural networks trained by two Bayesian algorithms, the Markov Chain Monte Carlo method and the evidence framework; the neural networks have been trained on the task of labelling segmented outdoor images. The two Bayesian algorithms are compared and contrasted with respect to the input feature selection and the investigation of empirical learning curves. We show the use of the Automatic Relevance Determination method in order to estimate the relevance of the input features in training the model. We have also analysed empirical learning curves of the neural networks trained by the two Bayesian methods, assessing the sensitivity of the generalisation errors due to the choice of the training set and the seeds initialising the random number generators of the two Bayesian algorithms.

Keywords: Gaussian processes, learning curves, bounds, input feature reduction, Bayesian neural networks, Automatic Relevance Determination.

To Federico, Paola, Paolo and Russanna

Take, O Lord, and receive
all my liberty,
my memory, my understanding
and my entire will.
Whatever I am, have and hold,
You have given to me;
I restore it all to You
and surrender it wholly
to be governed by your will.
Give me only your love and your grace,
and I am rich enough
and ask for nothing more.

St. Ignatius Loyola (1491-1556)

Acknowledgements

This thesis could not have been carried out without the support and assistance of many colleagues and friends.

First and foremost I wish to thank Dr. Christopher K. I. Williams for his guidance, criticism and supervision; this research project could have been much more difficult without his stimulating ideas and opportune advices.

I am also grateful to Dr. Stephen J. Roberts for the examination of the first draft of the thesis and the useful comments he suggested.

My research has benefited of helpful discussions with many of the members of the Neural Computing Research Group of Aston University and in particular with Dr. Manfred Opper and Prof. Christopher M. Bishop; many thanks also go to Dr. Andy W. Wright of British Aerospace for his assistance with this research. I thank British Aerospace Sowerby Research Centre for making available the Sowerby Image Database and Drs. Neill Campbell and William P. J. Mackeown who provided information about the database.

For the help received, I thank my fellow Ph. D. students Ragnar H. Lesch, Cazhaow S. Qazaz, Ansgar H. West and Krzysztof A. Zapart; I am particularly grateful to Mehdi Azzouzi and Giancarlo Ferrari Trecate who have made my life significantly easier.

I wish to thank Sophia Zoumpou for the continued support, encouragement, confidence and fun we shared during the final part of our adventure at the University of Aston.

This research forms part of the *Validation and Verification of Neural Networks Systems* project funded jointly by EPSRC (GR/K 51792) and British Aerospace. During the three years I have been financially supported by a postgraduate studentship of British Aerospace.

Contents

List of Figures

List of Tables

Declaration

This thesis describes the work carried out between October 1995 and September 1998 in the Neural Computing Research Group of Aston University under the supervision of Dr. Christopher K. I. Williams.

The research presented in the thesis has been executed by myself, unless otherwise stated and explicitly referenced.

This thesis has been composed by myself and it has not, nor any similar dissertation, submitted in any previous application for a degree.

Publications

Some of the work presented in this thesis has been published or submitted for publication as listed below.

- F. Vivarelli and C. K. I. Williams, *Discovering hidden features with Gaussian processes regression.* Accepted for publication in *Advances in Neural Information Processing Systems* 11*: Proceedings of the* 1998 *Conference*, (MIT Press, Cambridge, MA, 1999).
- M. Opper and F. Vivarelli, *General Bounds on Bayes errors for regression with Gaussian process.* Accepted for publication in *Advances in Neural Information Processing Systems* 11*: Proceedings of the* 1998 *Conference*, (MIT Press, Cambridge, MA, 1999).
- C. K. I. Williams and F. Vivarelli, *Upper and lower bounds on the learning curve for Gaussian processes.* Accepted for publication in *Machine Learning* (1999).
- F. Vivarelli and C. K. I. Williams, *Bayesian neural networks for classifying segmented outdoor images.* Submitted for publication to *Neural Networks*.
- F. Vivarelli and C. K. I. Williams, *Using Bayesian neural networks to classify segmented images.* In *Proceedings of the* 5^{th} *IEE International Conference on Artificial Neural Networks* Conference Publication 440, p. 268–273 (1997).

Chapter 1

Introduction

1.1 General background

Understanding the generalisation capability of systems which learn from data is an important task. Since information processing systems have to take part in decision-making processes, investigation on their reliability with respect to the tasks covered should be carried out; this is even more crucial when information processing techniques are embedded in safety critical systems (e.g. the diagnosis of diseases).

In recent years neural networks have been applied to a variety of problems due to their flexibility and the possibility to learn from data. Their actual use should be allowed only after a thorough investigation on their reliability, estimating the generalisation performance which can be obtained in real-world applications. In this context it is relevant to assess and quantify the effects that the inclusion of these methods may have on systems embedding neural networks, knowing under which conditions a given network performs its task reliably, evaluating the probability of success and failure and highlighting the ability to handle novel data. This can be achieved by studying the generalisation capabilities of neural networks.

In the past years neural networks have been interpreted as *black boxes* which process information; after completing the training procedure using a data set, neural networks could be applied in order to perform a certain task. However, the way in which this task could be completed was somehow obscure; since the results obtained by a neural network were not completely explainable, their use in real-world tasks was not completely safe. Recent research has shown that following a principled approach, statistical theory is the most suitable framework in which a study of the generalisation capabilities of neural networks can be developed; Bishop (1995) presents a review of this topic.

There are two main frameworks in which a statistical approach of the generalisation of neural networks can be carried out, the frequentist and the Bayesian schools. They mainly differ in the treatment of the network's parameters; while the frequentist framework finds a single estimate of the network's weights, the Bayesian analysis assigns an uncertainty to the parameters through a probability distribution. Of course these approaches affect also the process of prediction. Having only one estimate of the network's parameters, within the frequentist framework only one prediction for new data can be obtained. Conversely, in the Bayesian framework a posterior distribution over the parameters determines a distribution over the network's output and predictions are obtained averaging over that distribution.

The investigation of the network's generalisation performance addressed by the frequentist analysis has focussed on what is known as the *bias-variance* dilemma (Geman et al., 1992); it can be informally defined as follows. Let us suppose that a dataset is available for training a neural network on a regression problem. If the network is trying to model a regressor with a small number of parameters, it is not very flexible and may have a poor fitting of the underlying function; because of this behaviour the model is said to show a strong bias, underfitting the data. An opposite effect can be observed when a regressor is modelled by a neural network with too many parameters; since the dataset can be interpolated with a low training error, the neural network is characterised by a large variance, overfitting the data. In both cases, generalisation performances of the model are very poor.

The effects of the bias-variance dilemma can be reduced by a number of methods (e.g. Breiman, 1998). The addition of a term in the objective function penalising models with too many parameters helps in decreasing the effect due to the variance; this method is known as regularisation. Another method is the cross-validation which consists in training several models with a given dataset and choosing the best one on the basis of the error obtained on a validation set; as overfitting on the validation set may arise, the generalisation error is evaluated on a test set disjointed from the validation set. More recently the method of Bagging (Bootstrap Aggregating) has been proposed (Breiman, 1996) which aggregates predictions made by several predictors decreasing the variance and reducing the generalisation error of a model.

In a Bayesian perspective (Bishop, 1995), every free parameter of a neural network is affected by an uncertainty denoted as a probability density distribution. The training procedure consists in evaluating the posterior distribution of the parameters by using Bayes' theorem.

This framework can be illustrated in a hierarchical fashion made up by several levels. The first level of the hierarchy involves the parameters of the neural network. The prior distribution of the parameters is chosen on the basis of the prior knowledge about the function we want to infer; the analytic form of the prior distribution depends upon a set of parameters called hyperparameters. Training data can modify the prior belief about a density distribution of the parameters of the neural network through the evaluation of the likelihood of the data. The posterior distribution of the parameters is estimated from the prior probability and the likelihood of the data by using

Bayes' theorem.

The second level of the hierarchy involves the estimation of the posterior distribution of the hyperparameters by applying an argument similar to the previous one. The prior belief about the distribution of the hyperparameters can be modified by evaluating the evidence of the data and generating the posterior distribution through Bayes' theorem.

In principle this approach could be iterated further, introducing hyper-hyperparameters controlling the distribution of the hyperparameters. The last level of the hierarchy interests the evaluation of the belief of a model (e.g. the structure of a neural network) in explaining the data; similarly to the previous steps, the posterior distribution of the models can be carried out by combining a prior distribution and the evidence of the models in Bayes' theorem. To a first approximation, this level is concerned with the issue of model selection, since the model with the highest posterior probability can be picked out, discarding the others.

Within the Bayesian framework, prediction is made by considering the output of a neural network given a certain input vector and the posterior distribution of the parameters. Bayesian prediction for a new input point is given by the integration of the output of the neural network with respect to the posterior distribution of the parameters of the model; the posterior distribution allows also the evaluation of error bars on test data by calculating the variance on the prediction.

Although Bayesian framework provides a principled approach to training neural networks and making predictions, a fully-theoretical investigation of the generalisation capabilities of neural networks is technically difficult because the analytic expression of the probability distribution of the parameters is complicated and equations become analytically intractable. However Gaussian processes allow an appropriate theoretical analysis of this topic. Following the Bayesian approach to prediction with neural networks, inference with Gaussian processes has become popular over the last few years due to their analytical tractability, the interpretability of their parameters and the clear assumptions about prior knowledge embodied in the model; a review of Gaussian processes is presented by MacKay (1997).

Gaussian process predictions can be considered as obtained by a neural network model with a Gaussian prior distribution over the weights in the limit of an infinite number of hidden units (Neal, 1996). Rasmussen (1996) compared Gaussian process predictions to the results obtained by Bayesian neural networks on a number of problems, verifying that the former were at least as good as the latter. The link between covariance functions of Gaussian processes and a Bayesian treatment of neural networks with certain weight priors and transfer functions has been proposed by Williams (1997).

Loosely speaking, regression with Gaussian process is achieved by describing the covariance between datapoints with a function, whose parameters are estimated after an optimisation procedure on training data. A key point in Gaussian process regression concerns the choice of the covariance function as it affects inference on datapoints as well as the evaluation of the variance on prediction. Prior knowledge about the function we wish to model (e.g. its degree of differentiability) can help

in selecting the most appropriate covariance function suitable to model the regressor.

1.2 Structure of the thesis

This thesis considers theoretical and empirical approaches to investigate and to assess the generalisation capability of neural network models and Gaussian processes. This issue is one of the major topics of pattern recognition and several years of research are required to investigate it thoroughly; we restrict our analysis to the following two topics.

i **Learning curves.**

One major problem in using a model which learns from examples is to quantify the amount of data required to guarantee a certain level of performance; this can be addressed by the study of learning curves, which relate the value of the generalisation error to the amount of training data. One of the principal difficulties in carrying out a study on the learning curve is due to the evaluation of the generalisation error averaged over the distribution of the training data, although this problem can be overcome by evaluating numerically the learning curves. In the thesis we investigate bounds on the learning curve of Gaussian process and we compare the learning curves of a neural network trained by two Bayesian training algorithms.

ii **Input variables reduction.**

In any problem a number of features may be superfluous or correlated and this affects the overall generalisation capability of the model. In the thesis we deal with two aspects of input variables reduction. If the input vectors are assumed to lie on a low-dimensional manifold in a higher dimensional space then it may be possible to reduce the number of features, making the input coding to the network more efficient; this topic is investigated in the context of Gaussian process regression. A second aspect of the problem is concerned with the evaluation of the relevance of features in training a model; Bayesian training of neural networks can naturally implement this aspect, reducing the number of irrelevant features.

Investigation on the generalisation capabilities of Gaussian processes is illustrated in Chapter 2 where we present the results of a study of upper and lower bounds on the learning curve for Gaussian processes. For a Gaussian process it is possible to carry out analytically the Bayesian generalisation error (i.e. the generalisation error averaged over the distribution of the function values) given a training set. The integration of that result over the distribution of the datasets in order to obtain the true learning curve is not analytically tractable; this motivates the search for bounds on the learning curve. By considering the properties of the Bayesian generalisation error, we developed two upper bounds on the learning curve. In order to investigate the quality of the bounds with respect to the smoothness of the underlying stochastic processes, we studied the upper bounds for several covariance functions, explaining the early and asymptotic behaviour of

the curves. In the Chapter we also show the results of the experiments concerning a lower bound on the learning curve proposed by Opper (Opper and Vivarelli, 1999).

Many factors can affect the generalisation performance of a Gaussian process; one of them is related to the dimension of the input vector because large input-dimensionality requires large amount of data to train the model properly. This remark motivates the search for a mapping which is able to reduce the components of the input vector. In particular, if the regressor is actually a function of a set of hidden features generated by a linear combination of the manifested variables, better generalisation results should be obtained by performing predictions in the hidden space. In Chapter 3 we illustrate how to discover hidden features with Gaussian process regression. By using a certain parametrisation of the covariance prior, it is possible to determine relationships between the components of the input vector, discovering the space of the hidden features. This is affected by the level of the noise on the data and the number of components of the input vector; these effects are discussed in the Chapter. We also tested our approach on an example taken from Breiman (1993).

An empirical approach to the issue of the generalisation of neural networks is presented in Chapter 4, where we compare the generalisation capabilities obtained by neural networks trained by two Bayesian methods, the evidence framework (MacKay, 1992a; MacKay, 1992b) and a Markov Chain Monte Carlo method (Neal, 1996). The two algorithms have been compared on the task of labelling segmented images. To support this, British Aerospace provided a database consisting of 119 colour images which have been segmented and the subsequent regions hand-labelled. Using this database, a set of 35 features describing the internal and contextual characteristics of each region of the dataset has been obtained; neural networks have been trained to classify each region as one out of eleven classes (e.g. sky, building, vegetation, road, car). Bayesian neural networks trained with the evidence framework and Markov Chain Monte Carlo algorithms have been compared and contrasted with respect to their ability in selecting input features and their generalisation performances.

Conventional approaches to the issue of feature selection involve subset regression, where subgroups of the available features are used to make predictions. Due to the complexity of the problem, sequential forward or backward selection of features are practically used. An alternative approach (which arises naturally within the Bayesian training of neural networks) is the Automatic Relevance Determination technique of MacKay and Neal (Neal, 1996). This enables the detection of the relative importance of the feature set in training the classifier by looking at the distribution of the network's parameters after the training procedure. To our knowledge, this is the first study which demonstrates the application of ARD on a practical problem.

Empirical learning curves have been investigated by setting up a number of experiments to estimate the generalisation errors of neural networks trained by the evidence framework and the Markov Chain Monte Carlo methods. Since the generalisation performances are affected by stochastic effects (such as the composition of the test and training sets and the initialisation of the random

number generators) and by the training algorithm chosen, we evaluated the effects that these contributions have on the empirical learning curves of a neural network. This study enables one to verify which one of the two Bayesian methods leads to a better generalisation performance of the neural network on the problem at hand.

In the last Chapter we summarise the work presented in the thesis and we sketch some future directions of research for addressing some of the issues that have arisen during the present work.

Chapter 2

Upper and lower bounds on the learning curve for Gaussian processes

In this Chapter we introduce and illustrate non-trivial upper and lower bounds on the learning curves for one-dimensional Gaussian Processes. The analysis is carried out emphasising the effects induced on the bounds by the smoothness of the random process described by the Modified Bessel and the Squared Exponential covariance functions. We present an explanation of the early, linearly-decreasing behaviour of the learning curves and the bounds as well as a study of the asymptotic behaviour of the curves. The effects of the noise level and the lengthscale on the tightness of the bounds are also discussed.

2.1 Introduction

A fundamental problem for systems learning from examples is to estimate the amount of training samples needed to guarantee satisfactory generalisation capabilities on new data. This is of theoretical interest but also of vital practical importance; for example, algorithms which learn from data should not be used in safety-critical systems until a reasonable understanding of their generalisation capabilities has been obtained. In recent years several authors have carried out analysis on this issue and the results presented depend on the theoretical formalisation of the learning problem.

Approaches to the analysis of generalisation include those based on asymptotic expansions around optimal parameter values (e.g. AIC (Akaike, 1974), NIC (Murata et al., 1994)); the Prob-

ably Approximately Correct (PAC) framework (Valiant, 1984); uniform convergence approaches (e.g. Vapnik (1995)); and Bayesian methods.

The PAC and uniform convergence methods are concerned with frequentist-style confidence intervals derived from randomness introduced with respect to the distribution of inputs and noise on the target function. A central concern in these results is to identify the flexibility of the hypothesis class $\mathcal{F}$ to which approximating functions belong, for example, through the Vapnik-Chervonenkis dimension of $\mathcal{F}$. Note that these bounds are independent of the input and noise densities, assuming only that the training and test samples are drawn from the same distribution.

The problem of understanding the generalisation capability of systems can also be addressed in a Bayesian framework, where the fundamental assumption concerns the kinds of function our system is required to model. In other words, from a Bayesian perspective we need to put *priors* over target functions. In this context learning curves and their bounds can be analysed by an average over the probability distribution of the functions. In this Chapter we use Gaussian priors over functions which have the advantage of being more general than simple linear regression priors, but they are more analytically tractable than priors over functions obtained from neural networks.

Neal (1996) has shown that for fixed hyperparameters, a large class of neural network models will converge to Gaussian process priors over functions in the limit of an infinite number of hidden units. The hyperparameters of the Bayesian neural network define the parameters of the corresponding Gaussian process (GP). Williams (1997) calculated the covariance functions of GPs corresponding to neural networks with certain weight priors and transfer functions.

The investigation of GP predictors is motivated by the results of Rasmussen (1996), who compared the performances obtained by GPs to those obtained by Bayesian neural networks on a range of tasks. He concluded that GPs were at least as good as neural networks. Although the present study deals with regression problems, GPs have also been applied to classification problems (e.g. Barber and Williams (1997)).

In this Chapter we are mainly concerned with the analysis of upper and lower bounds on the learning curve of GPs. A graph of the expected generalisation error against the number of training samples n is known as a learning curve. There are many results available concerning learning curves under different theoretical scenarios. However, many of these are related to the asymptotic behaviour of these curves, which is not usually of great practical importance as it is unlikely that we will have enough data to reach the asymptotic regime. Our main goal is to explain some of the early behaviour of learning curves for Gaussian processes.

The structure of the Chapter is as follows. GPs for regression problems are introduced in Section 2.2. As will be shown, the whole theory of GPs is based on the choice of the prior covariance function $C_p(\mathbf{x}, \mathbf{x}')$; Section 2.3 presents the covariance functions we have been using in this study. In Section 2.4 the learning curve of a GP is introduced. We present some properties of the learning curve of GPs as well as some problems may arise in evaluating it. Upper and lower bounds on the learning curve of a GP in a non-asymptotic regime are presented in Section 2.5.

These bounds have been derived from two different approaches: one makes use of main properties of the generalisation error, whereas the other is derived from an eigenfunction decomposition of the covariance function. The asymptotic behaviour of the upper bounds is also discussed.

A set of experiments have been run in order to assess the upper and lower bounds of the learning curve. In Section 2.6 we present the results obtained and investigate the link between tightness of the bounds and the smoothness of the stochastic process modelled by a GP. A summary of the results and some open questions are presented in the last Section.

2.2 Gaussian processes

A collection of random variables $\{Y(\mathbf{x}) | \mathbf{x} \in \mathcal{X}\}$ indexed by a set $\mathcal{X}$ defines a stochastic process. In general the domain $\mathcal{X}$ might be $\mathbb{R}^d$ for some dimension d although it could be even more general. A joint distribution characterising the statistics of the random variables gives a complete description of the stochastic process.

A GP is a stochastic process whose joint distribution is Gaussian; it is fully defined by giving a Gaussian prior distribution for every finite subset of variables.

In the following we concentrate to the regression problem assuming that the value of the target function $t(\mathbf{x})$ is generated from an underlying function $y(\mathbf{x})$ corrupted by Gaussian noise with mean 0 and variance σ_ν^2. Given a collection of n training data $\mathcal{D}_n = \{(\mathbf{x}^i, t^i), i = 1 \ldots n\}$ (where each t^i is the observed output value at the input point $\mathbf{x}^i$), we would like to determine the posterior probability distribution $p(y|\mathbf{x}, \mathcal{D}_n)$.

In order to set up a statistical model of the stochastic process, the set of n random variables $\mathbf{y} = (y^1, y^2, \ldots y^n)^{\mathrm{T}}$ modelling the function values at $\mathbf{x}^1, \mathbf{x}^2, \ldots \mathbf{x}^n$ respectively, is introduced. Similarly $\mathbf{t}$ is the collection of target values $\mathbf{t} = (t^1, \ldots t^n)^{\mathrm{T}}$. We also denote with $\tilde{\mathbf{y}}$ the vector whose components are $\mathbf{y}$ and the test value y at the point $\mathbf{x}$. The distribution $p(\tilde{\mathbf{y}}|\mathbf{x}, \mathcal{D}_n)$ can be inferred using Bayes' theorem. In order to do so, we need to specify a prior over functions as well as to evaluate the likelihood of the model and the evidence for the data.

A choice for a prior distribution of the stochastic vector $\tilde{\mathbf{y}}$ is a Gaussian prior distribution:

$$p(\tilde{\mathbf{y}}|\mathbf{x}, \mathbf{x}^1, \ldots \mathbf{x}^n) \propto \exp\left[-\frac{1}{2}\tilde{\mathbf{y}}^{\mathrm{T}}\Sigma^{-1}\tilde{\mathbf{y}}\right].$$

This is a prior as it describes the distribution of the true underlying values without any reference to the target values $\mathbf{t}$. The covariance matrix Σ can be partitioned as

$$\Sigma = \begin{pmatrix} K_p & \mathbf{k}(\mathbf{x}) \\ \mathbf{k}^{\mathrm{T}}(\mathbf{x}) & C_p(\mathbf{x}, \mathbf{x}) \end{pmatrix}.$$

The element $(K_p)_{ij}$ is the covariance between the i-th and the j-th training points, i.e. $(K_p)_{ij} = \mathcal{E}\left[\left(y(\mathbf{x}^i) - \mu(\mathbf{x}^i)\right)\left(y(\mathbf{x}^j) - \mu(\mathbf{x}^j)\right)\right]$. The components of the vector $\mathbf{k}(\mathbf{x})$ are the covariances of the test point with all the training data $\left(\mathbf{k}_i(\mathbf{x}) = C_p(\mathbf{x}, \mathbf{x}^i)\right)$; $C_p(\mathbf{x}, \mathbf{x})$ is the prior covariance of the test point with itself.

A GP is fully specified by its mean $\mathcal{E}[y(\mathbf{x})] = \mu(\mathbf{x})$ and covariance function $C_p(\mathbf{x}, \mathbf{x}') = \mathcal{E}[(y(\mathbf{x}) - \mu(\mathbf{x}))(y(\mathbf{x}') - \mu(\mathbf{x}'))]$. Below we set $\mu(\mathbf{x}) = 0$; this is a valid assumption provided that any known offset or trend in the data has been removed. We can also deal with $\mu(\mathbf{x}) \neq 0$, but this introduces some extra notational complexity. A discussion about the possible choices of the covariance function $C_p(\mathbf{x}, \mathbf{x}')$ is given in Section 2.3. For the moment we note that the covariance function is assumed to depend upon the input variables $(\mathbf{x}, \mathbf{x}')$. Thus the correlation between function values depends upon the spatial position of the input vectors; usually this will be chosen so that the closer the input vectors, the higher the correlation of the function values.

The likelihood relates the underlying values of the function to the target data. Assuming a Gaussian noise corrupting the data, we can write the likelihood as

$$p(\mathbf{t}|\mathbf{y}) \propto \exp\left[-\frac{1}{2}(\mathbf{y} - \mathbf{t})^{\mathrm{T}}\Omega^{-1}(\mathbf{y} - \mathbf{t})\right]$$

where $\Omega = \sigma_\nu^2 \mathbb{I}$. The likelihood refers to the stochastic variables representing the n data; so $\mathbf{t}, \mathbf{y} \in \mathbb{R}^n$ and Ω is an $n \times n$ matrix.

Given the prior distribution over the values of the function $p(\tilde{\mathbf{y}}|\mathbf{x}, \mathbf{x}^1, \ldots \mathbf{x}^n)$, Bayes' rule specifies the distribution $p(\tilde{\mathbf{y}}|\mathbf{x}, \mathcal{D}_n)$ in terms of the likelihood of the model $p(\mathbf{t}|\mathbf{y})$ and the evidence of the data $p(\mathcal{D}_n)$ as

$$p(\tilde{\mathbf{y}}|\mathbf{x}, \mathcal{D}_n) = \frac{p(\mathbf{t}|\mathbf{y})\, p(\tilde{\mathbf{y}}|\mathbf{x}, \mathbf{x}^1, \ldots \mathbf{x}^n)}{p(\mathcal{D}_n)}.$$

Given such assumptions, it is a standard result (e.g. Whittle, 1963) to derive the analytic form of the predictive distribution marginalising over $\mathbf{y}$. The predictive distribution turns out to be a Gaussian distribution whose mean $\hat{y}(\mathbf{x})$ and variance $\sigma^2_{\hat{y},n}(\mathbf{x})$ are

$$\hat{y}(\mathbf{x}) = \mathbf{k}^{\mathrm{T}}(\mathbf{x}) K^{-1} \mathbf{t} \tag{2.1}$$

$$\sigma^2_{\hat{y},n}(\mathbf{x}) = C_p(\mathbf{x}, \mathbf{x}) - \mathbf{k}^{\mathrm{T}}(\mathbf{x}) K^{-1} \mathbf{k}(\mathbf{x}). \tag{2.2}$$

The most probable value $\hat{y}(\mathbf{x})$ is regarded as the prediction of the GP on the test point $\mathbf{x}$; K is the covariance matrix of the targets $\mathbf{t}$: $K = K_p + \sigma_\nu^2 \mathbb{I}$. The estimate of the variance $\sigma^2_{\hat{y},n}(\mathbf{x})$ of the posterior distribution is considered as the error bar of $\hat{y}(\mathbf{x})$. In the following, we always omit the subscript $\hat{y}$ in $\sigma^2_{\hat{y},n}$, taking it as understood. Since the estimate 2.1 is a linear combination of the training targets, GPs are regarded as linear smoothers (Hastie and Tibshirani, 1990).

2.3 Covariance functions

The choice of the covariance function is a crucial one. The properties of two GPs, which differ only in the choice of the covariance function, can be remarkably diverse. This is due to the rôle of the covariance function which has to incorporate in the statistical model the prior belief about the underlying function. In other words the covariance function is the analytical expression of the prior knowledge about the function being modelled. A misspecified covariance function affects the model inference as it has influence on the evaluation of Equations 2.1 and 2.2.

Formally every function which produces a symmetric, positive semi-definite covariance matrix K for any set of the input space $\mathcal{X}$ can be chosen as covariance function. From an applicative point of view we are interested only in functions which contain information about the structure of the underlying process being modelled.

For instance let us suppose to know that a function $y(\mathbf{x})$ has a linear trend, i.e. its values lie on the plane $y(\mathbf{x}) = \mathbf{a}^{\mathrm{T}}\mathbf{x} + b$. The effect of a prior distribution over the function is formalised as a distribution over the parameters a_i and b. Assuming that $a_i \sim \mathcal{N}\left(0, \sigma_{a_i}^2\right), i = 1 \ldots d$ and $b \sim \mathcal{N}\left(0, \sigma_b^2\right)$, the mean and the covariance of $y(\mathbf{x})$ with respect to the distribution of the parameters a_i and b are $\mathcal{E}[y] = \mathcal{E}\left[\mathbf{a}^{\mathrm{T}}\mathbf{x} + b\right] = 0$ and $\mathcal{E}[yy'] = \mathcal{E}\left[\mathbf{x}^{\mathrm{T}}\mathbf{a}\mathbf{a}^{\mathrm{T}}\mathbf{x}' + b^2\right] = \sum_{i=1}^{d} \sigma_{a_i}^2 x_i x_i' + \sigma_b^2$ respectively. Thus the covariance function corresponding to such a kind of prior knowledge is $C_p(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{d} \sigma_{a_i}^2 x_i x_i' + \sigma_b^2$. As the $\sigma_{a_i}^2$ and σ_b^2 control the prior distribution of $y(\mathbf{x})$, they are the parameters of the GP.

The choice of the covariance function is also linked to the *a priori* knowledge about the smoothness of the function $y(\mathbf{x})$ for there is a connection between the differentiability of the covariance function and the degree of smoothness of the stochastic process.

The smoothness is related to the maximum degree of differentiability of the function. In order to apply this property to the random functions, we introduce the definition of stochastic differentiation. A random function $y(\mathbf{x})$ is mean square differentiable at $\mathbf{x}$ along the i-th coordinate direction, with derivative $y_i(\mathbf{x})$, if

$$\lim_{t \to 0} \mathcal{E}_y \left[\left| \frac{y(\mathbf{x} + t\mathbf{e}_i) - y(\mathbf{x})}{t} - y_i(\mathbf{x}) \right|^2 \right] = 0,$$

where $t \in \mathbb{R}$, $\mathbf{e}_i$ is the normalised vector indicating the direction i of the input space and the vectors $\mathbf{x}, (\mathbf{x} + t\mathbf{e}_i) \in \mathcal{X}$ (see e.g. Adler (1981)).

Thus the smoothness of a stochastic function depends upon the differentiability (on average) of the process in $\mathbf{x}$. The relation between smoothness of a process and its covariance function is guaranteed by the following theorem (see e.g. Adler, 1981): if $\partial^2 C_p(\mathbf{x}, \mathbf{x}') / \partial x_i \partial x_i'$ exists and is finite at $(\mathbf{x}, \mathbf{x})$, then the stochastic process $y(\mathbf{x})$ is mean square differentiable in the i-th Cartesian direction at $\mathbf{x}$. This theorem is relevant as it links the differentiability properties of the covariance function with the smoothness of the random process and justifies the choice of a covariance function depending upon the prior belief about the degree of smoothness of $y(\mathbf{x})$.

In this work we are mainly concerned with stationary covariance functions. A stationary covariance function is translation invariant (i.e. $C_p(\mathbf{x}, \mathbf{x}') = C_p(\mathbf{x} - \mathbf{x}')$) and depends only upon the distance between two data points. In the following, the covariance functions we have been using are presented. In order to simplify the notation, we consider the case $\mathcal{X} \subseteq \mathbb{R}$.

The stationary covariance function *squared exponential* (SE) is defined as

$$C_p(x - x') = \exp\left[-\frac{(x - x')^2}{2\lambda^2} \right] \tag{2.3}$$

where λ is the lengthscale of the process. The parameter λ defines the characteristic length of

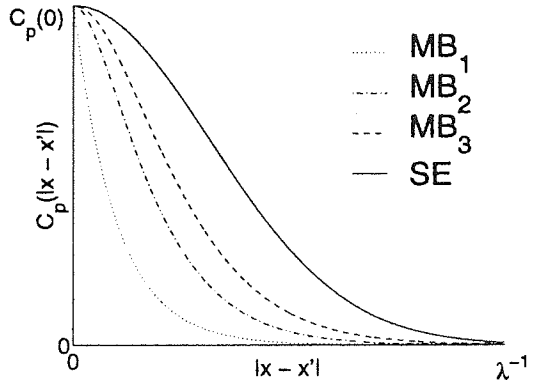


Figure 2.1: The Figure shows the covariance functions used in this work. The solid line is the SE covariance function; the dotted, dash-dotted and dashed lines draw the graph of the MB_r covariance functions with $r = 1, 2$ and 3 respectively. The values of $|x - x'|$ are reported on the x-axis. The larger the lengthscale λ, the slower the decay to 0 of the functions.

the process, estimating the distance in the input space in which the function $y(x)$ is expected to vary significantly. A large value of λ indicates that the function is almost constant over the input space, whereas a small value of the lengthscale designates a function which varies rapidly. The graph of this covariance function is shown by the continuous line in Figure 2.1. As the SE function has infinitely many derivatives it describes smooth random processes ($y(x)$ is mean-square differentiable up to order ∞).

It is possible to tune the differentiability of a process, introducing the modified Bessel covariance function of order r (MB_r). It is defined as

$$C_p(x - x') = \kappa_\nu \left(\frac{|x - x'|}{\lambda}\right)^\nu \mathcal{K}_\nu \left(\frac{|x - x'|}{\lambda}\right) = \kappa_\nu \sum_{k=0}^{r-1} a_k \left(\frac{|x - x'|}{\lambda}\right)^k \exp\left[-\frac{|x - x'|}{\lambda}\right], \tag{2.4}$$

where $\mathcal{K}_\nu(\cdot)$ is the modified Bessel function of order ν (see e.g. Equation 8.468 in Gradshteyn and Ryzhik (1993)), with $\nu = r - 1/2$ for integral r. In what follows, we set the constant κ_ν such that $C_p(0) = 1$. The factors a_k are constants depending on the order ν of the Bessel function. Matérn (1980) shows that the functions MB_r define a proper covariance. Stein (1989) also noted that the process with covariance function MB_r is $r - 1$ times mean-square differentiable.

In this study we deal with modified Bessel covariance function of orders $r = 1, 2, 3$; their explicit analytic form for $x \in \mathcal{X} \subseteq \mathbb{R}$ is

$$r = 1,\ C_p(x - x') = \exp\left[-\frac{|x - x'|}{\lambda}\right] \tag{2.5}$$

$$r = 2,\ C_p(x - x') = \exp\left[-\frac{|x - x'|}{\lambda}\right]\left(1 + \frac{|x - x'|}{\lambda}\right) \tag{2.6}$$

$$r = 3,\ C_p(x - x') = \exp\left[-\frac{|x - x'|}{\lambda}\right]\left(1 + \frac{|x - x'|}{\lambda} + \frac{1}{3}\left(\frac{|x - x'|}{\lambda}\right)^2\right). \tag{2.7}$$

We note that MB_1 corresponds to the Ornstein-Uhlenbeck covariance function which describes a process which is not mean square differentiable.

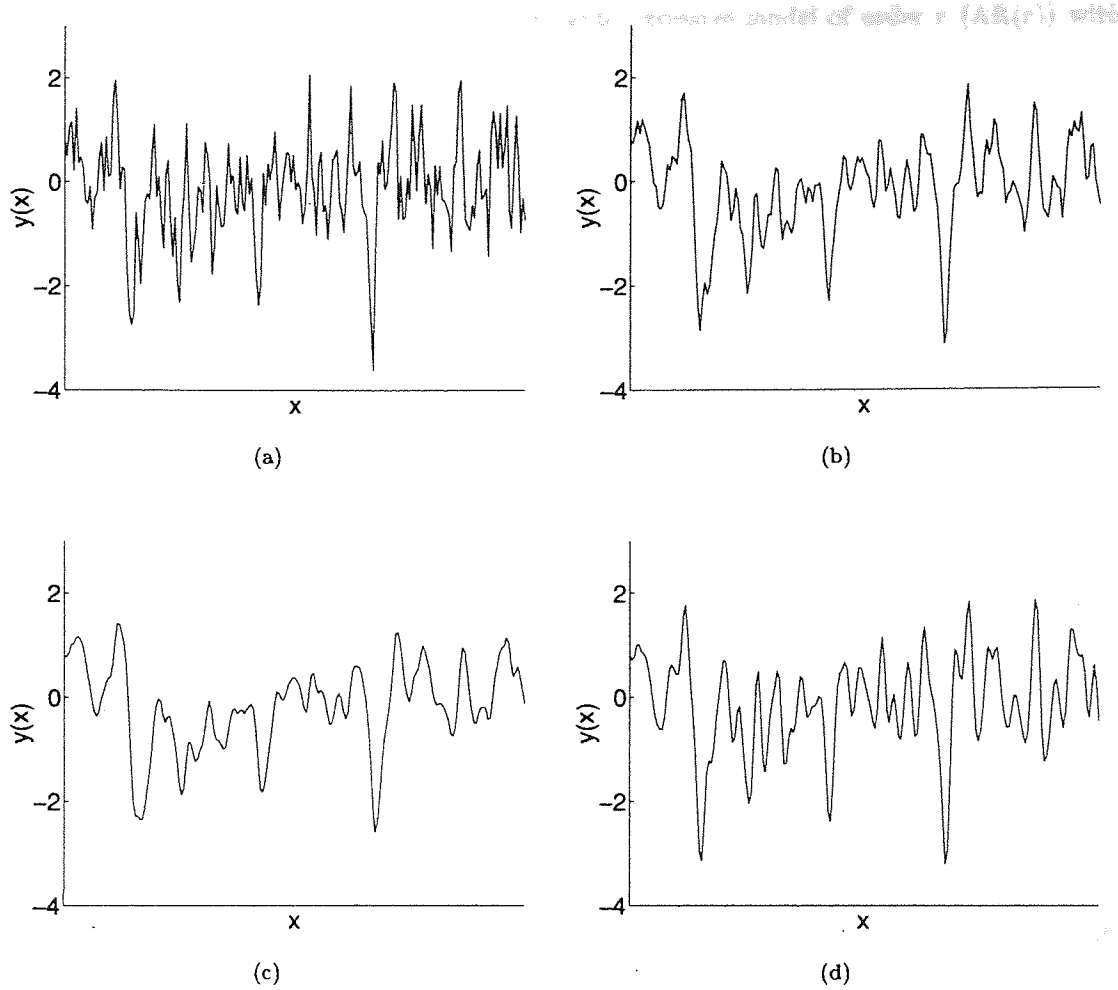


Figure 2.2: Discretised sample of random functions generated from the MB covariance functions of first (2.2(a)), second (2.2(b)), third order (2.2(c)), and the SE function (2.2(d)) with $\lambda = 0.01$. The order $r-1$ of a process refers to the number of mean square derivatives of the random process.

If $r \to \infty$, the MB_r behaves like the SE covariance function; this can be easily shown by considering the power spectra of MB_r and SE which are

$$S_r(\omega) \propto \frac{\lambda}{(1+\omega^2\lambda^2)^r} \text{ and } S_{se}(\omega) \propto \lambda \exp\left[-\frac{\omega^2\lambda^2}{2}\right].$$

Since

$$\lim_{r\to\infty}\left(1+\frac{\omega^2\lambda^2}{2r}\right)^{-r} = \exp\left[-\frac{\omega^2\lambda^2}{2}\right],$$

the MB_r behaves like SE for large r, provided that λ is rescaled accordingly.

Modified Bessel covariance functions are also interesting because they describe Markov processes of order r. Ihara (1991) defines $Y(x)$ to be a *strict sense* Markov process of order r if it is $r-1$ times mean-square differentiable at every $x \in \mathbb{R}$ and if $P(Y(x+s) \leq y|Y(u), u \leq x) = P(Y(x+s) \leq y|Y(x), Y'(x), \ldots Y^{r-1}(x))$[1]. Ihara also states that a Gaussian process is a Markov process of

[1] Note that the definition of a Markov process in discrete and continuous time is rather different. In discrete time, a Markov process of order r depends only on the previous r times, but in continuous time the dependence is on the

order r in the strict sense if and only if it is an autoregressive model of order r (AR(r)) with a power spectrum (in the Fourier domain) of the form

$$S(\omega) \propto \prod_{k=1}^{r} \frac{1}{|i\omega + \alpha_k|^2}.$$

As the power spectrum of MB_r has the same form of the power spectrum of an AR(r) model, the stochastic process (whose covariance function is MB_r) is a strict sense r-ple Markov process. This characteristic of the MB_r covariance functions is important as it ultimately affects the evaluation of the generalisation error (as we shall see in Section 2.6).

Figure 2.2 shows the graphs of four (discretised) random functions generated using the MB_r covariance functions (with $r = 1, 2, 3$) and the SE function. We note how the smoothness of the random function specified is dependent of the choice of the covariance function. In particular, the roughest function is generated by the Ornstein-Uhlenbeck covariance function (Figure 2.2(a)) whereas the smoothest one is produced by the SE (Figure 2.2(d)). An intermediate level of regularity characterises the functions of figures 2.2(b) and 2.2(c), whose covariance function are the MB_2 and MB_3 respectively.

2.4 Learning curve for Gaussian processes

A learning curve of a model is a function which relates the generalisation error to the amount of training data; it is independent of the test points as well as the locations of the training data and depends only upon the amount of data in the training set. The learning curve for a GP is evaluated from the estimation of the generalisation error averaged over the distribution of the training and test data.

For regression problems, a measure of the generalisation capabilities of a GP is the squared difference $E^g_{\mathcal{D}_n}(\mathbf{x}, t)$ between the target value on a test point $\mathbf{x}$ and the prediction made by using Equation 2.1:

$$E^g_{\mathcal{D}_n}(\mathbf{x}, t) = \left(t - \mathbf{k}^{\mathrm{T}}(\mathbf{x}) K^{-1} \mathbf{t}\right)^2. \tag{2.8}$$

The Bayesian generalisation error at a point $\mathbf{x}$ is defined as the expectation of $E^g_{\mathcal{D}_n}(\mathbf{x}, t)$ over the actual distribution of the stochastic process t:

$$E^g_{\mathcal{D}_n}(\mathbf{x}) = \mathcal{E}_t\left[E^g_{\mathcal{D}_n}(\mathbf{x}, t)\right]. \tag{2.9}$$

Under the assumption that the data set is actually generated from a GP, it is possible to read Equation 2.2 as the Bayesian generalisation error at $\mathbf{x}$ given training data $\mathcal{D}_n$. To see this, let us consider the $(n+1)$-dimensional distribution of the target values at $\mathbf{x}^1, \mathbf{x}^2, \ldots \mathbf{x}^n$ and $\mathbf{x}$. This is a

derivatives at the last time. However, function values at previous times clearly allow approximate computation of derivatives (e.g. via finite differences) and thus one would expect that in the continuous-time situation the previous r process values will contain most of the information needed for prediction at the next time. Note that for the Ornstein-Uhlenbeck process $Y(x + s)$ depends only on the previous observation (x).

zero-mean multivariate Gaussian. The prediction at the test point $\mathbf{x}$ is $\hat{y}(\mathbf{x}) = \mathbf{k}^{\mathrm{T}}(\mathbf{x}) K^{-1}\mathbf{t}$, where $K = K_p + \sigma_\nu^2 \mathbb{I}$. Hence the expected generalisation error at $\mathbf{x}$ is given by

$$\begin{aligned} E^g_{\mathcal{D}_n}(\mathbf{x}) &= \mathcal{E}\left[\left(t - \mathbf{k}^{\mathrm{T}}(\mathbf{x}) K^{-1}\mathbf{t}\right)^2\right] \\ &= \mathcal{E}\left[t^2\right] - 2\mathbf{k}^{\mathrm{T}}(\mathbf{x}) K^{-1}\mathcal{E}\left[t\mathbf{t}\right] + \mathcal{E}\left[\mathbf{k}^{\mathrm{T}}(\mathbf{x}) K^{-1}\mathbf{t}\mathbf{t}^{\mathrm{T}}K^{-1}\mathbf{k}(\mathbf{x})\right] \\ &= C_p(\mathbf{0}) + \sigma_\nu^2 - 2\mathbf{k}^{\mathrm{T}}(\mathbf{x}) K^{-1}\mathbf{k}(\mathbf{x}) + \mathrm{Tr}\left[K^{-1}\mathbf{k}(\mathbf{x})\mathbf{k}^{\mathrm{T}}(\mathbf{x}) K^{-1}\mathcal{E}\left[\mathbf{t}\mathbf{t}^{\mathrm{T}}\right]\right] \\ &= C_p(\mathbf{0}) + \sigma_\nu^2 - \mathbf{k}^{\mathrm{T}}(\mathbf{x}) K^{-1}\mathbf{k}(\mathbf{x}), \end{aligned} \tag{2.10}$$

where we have used $\mathcal{E}[t\mathbf{t}] = \mathbf{k}(\mathbf{x})$ and $\mathcal{E}\left[\mathbf{t}\mathbf{t}^{\mathrm{T}}\right] = K$. Equation 2.10 is identical to $\sigma_n^2(\mathbf{x})$ as given in Equation 2.2 with the addition of the noise variance σ_ν^2 (since we are dealing with noisy data). The variance of $(t - \mathbf{k}^{\mathrm{T}}(\mathbf{x}) K^{-1}\mathbf{t})^2$ can also be calculated and it is derived in Appendix A.3.

The covariance matrix pertinent for these calculations is the true prior; if a GP predictor with a different covariance function is used, this increases the expected error (see Appendix A.2).

Another property of the generalisation error can be derived from the following observation; adding more data points never increases the size of the error bars on prediction ($\sigma^2_{n+1}(\mathbf{x}) \leq \sigma_n^2(\mathbf{x})$). This is an intuitive property which can be supported by an analytical proof under the assumption that the process is Gaussian (and unimodal).

Let us suppose that a GP has been trained using a training set with $n+1$ data points $(\mathbf{x}^1, \mathbf{x}^2, \ldots \mathbf{x}^n, \mathbf{x}^{n+1})$. The variance on a test point $\mathbf{x}$ is $\sigma^2_{n+1}(\mathbf{x}) = C(\mathbf{x},\mathbf{x}) - \mathbf{k}^{\mathrm{T}}(\mathbf{x}) K^{-1}\mathbf{k}(\mathbf{x})$ (see Equation 2.2), where $K = K_p + \sigma_\nu^2\mathbb{I}$ and $C(\mathbf{x},\mathbf{x}) = C_p(\mathbf{x},\mathbf{x}) + \sigma_\nu^2$ is the covariance of one point with itself. All of the vector and matrices in Equation 2.2 can be partitioned isolating the elements related to the $(n+1)$-th data point. Thus $\mathbf{k}^{\mathrm{T}}(\mathbf{x}) = \left(C_p(\mathbf{x},\mathbf{x}^1), C_p(\mathbf{x},\mathbf{x}^2), \ldots C_p(\mathbf{x},\mathbf{x}^{n+1})\right)$ can be split up in the n-dimensional vector $\mathbf{v}^{\mathrm{T}}(\mathbf{x}) = \left(C_p(\mathbf{x},\mathbf{x}^1), C_p(\mathbf{x},\mathbf{x}^2), \ldots C_p(\mathbf{x},\mathbf{x}^n)\right)$ and the scalar $c(\mathbf{x}) = C\left(\mathbf{x},\mathbf{x}^{n+1}\right)$. Similarly the $(n+1)\times(n+1)$ matrices K and K^{-1} can be partitioned in 4 sub-matrices

$$K = \begin{pmatrix} P & \mathbf{v}\left(\mathbf{x}^{n+1}\right) \\ \mathbf{v}^{\mathrm{T}}\left(\mathbf{x}^{n+1}\right) & c\left(\mathbf{x}^{n+1}\right) \end{pmatrix} \text{ and } K^{-1} = \begin{pmatrix} \tilde{P} & \tilde{\mathbf{v}}\left(\mathbf{x}^{n+1}\right) \\ \tilde{\mathbf{v}}^{\mathrm{T}}\left(\mathbf{x}^{n+1}\right) & \tilde{c}\left(\mathbf{x}^{n+1}\right) \end{pmatrix},$$

where (see e.g. Press et al. (1992))

$$\begin{aligned} \tilde{P} &= P^{-1} + P^{-1}\mathbf{v}\left(\mathbf{x}^{n+1}\right) M^{-1}\mathbf{v}^{\mathrm{T}}\left(\mathbf{x}^{n+1}\right) P^{-1} \\ M &= c\left(\mathbf{x}^{n+1}\right) - \mathbf{v}\left(\mathbf{x}^{n+1}\right)^{\mathrm{T}} P^{-1}\mathbf{v}\left(\mathbf{x}^{n+1}\right) \\ \tilde{\mathbf{v}}\left(\mathbf{x}^{n+1}\right) &= -P^{-1}\mathbf{v}\left(\mathbf{x}^{n+1}\right) M^{-1} \\ \tilde{c}\left(\mathbf{x}^{n+1}\right) &= M^{-1}. \end{aligned}$$

P and $\tilde{P}$ are $n \times n$ matrices. As P is the covariance matrix of a GP trained with the n training data $\left\{(\mathbf{x}^1, t^1), \ldots (\mathbf{x}^n, t^n)\right\}$ and $\mathbf{v}\left(\mathbf{x}^{n+1}\right)$ is the vector collecting the covariances of $\mathbf{x}^{n+1}$ with the n training point, the scalar M turns out to be the variance on the prediction over the point $\mathbf{x}^{n+1}$:

$$M = \sigma_n^2\left(\mathbf{x}^{n+1}\right).$$

Hence the error bar associated to the test point $\mathbf{x}$ can be rewritten as:

$$\begin{aligned}
\sigma_{n+1}^2(\mathbf{x}) &= C(\mathbf{x},\mathbf{x}) - \mathbf{k}^{\mathrm{T}}(\mathbf{x}) K^{-1}\mathbf{k}(\mathbf{x}) \\
&= C(\mathbf{x},\mathbf{x}) - (\mathbf{v}(\mathbf{x}), c(\mathbf{x})) \begin{pmatrix} \tilde{P} & \tilde{\mathbf{v}}(\mathbf{x}^{n+1}) \\ \tilde{\mathbf{v}}^{\mathrm{T}}(\mathbf{x}^{n+1}) & \tilde{c}(\mathbf{x}^{n+1}) \end{pmatrix} \begin{pmatrix} \mathbf{v}(\mathbf{x}) \\ c(\mathbf{x}) \end{pmatrix} \\
&= C(\mathbf{x},\mathbf{x}) - \mathbf{v}^{\mathrm{T}}(\mathbf{x}) P^{-1}\mathbf{v}(\mathbf{x}) - \frac{\left(v - c(\mathbf{x}^{n+1})\right)^2}{\sigma_n^2(\mathbf{x}^{n+1})} \\
&= \sigma_n^2(\mathbf{x}) - \frac{\left(v - c(\mathbf{x}^{n+1})\right)^2}{\sigma_n^2(\mathbf{x}^{n+1})}
\end{aligned}$$

where $v = \mathbf{v}^{\mathrm{T}}(\mathbf{x}) P^{-1}\mathbf{v}(\mathbf{x}^{n+1})$. Since $\left(v - c(\mathbf{x}^{n+1})\right)^2 / \sigma_n^2(\mathbf{x}^{n+1}) \geq 0$, it follows that

$$\sigma_{n+1}^2(\mathbf{x}) \leq \sigma_n^2(\mathbf{x}). \tag{2.11}$$

As $\sigma_n^2(\mathbf{x}) = E_{\mathcal{D}_n}^g(\mathbf{x})$, the relation of Equation 2.11 applies also to the generalisation errors and hence

$$E_{\mathcal{D}_{n+1}}^g(\mathbf{x}) \leq E_{\mathcal{D}_n}^g(\mathbf{x}). \tag{2.12}$$

This remark will be applied in Section 2.5 for evaluating upper bounds on the learning curve.

Equation 2.10 calculates the generalisation error at a point $\mathbf{x}$. Averaging $E_{\mathcal{D}_n}^g(\mathbf{x})$ over the density distribution of the test points $p(\mathbf{x})$, the expected generalisation error $E_{\mathcal{D}_n}^g$ is

$$E_{\mathcal{D}_n}^g = \int \left(C_p(\mathbf{0}) + \sigma_\nu^2 - \mathbf{k}^{\mathrm{T}}(\mathbf{x}) K^{-1}\mathbf{k}(\mathbf{x})\right) p(\mathbf{x})\, d\mathbf{x}. \tag{2.13}$$

For particular choices of $p(\mathbf{x})$ and $C_p(\mathbf{x},\mathbf{x}')$ the computation of this expression can be reduced to a $n \times n$ matrix computation as $\mathcal{E}_{\mathbf{x}}\left[\mathbf{k}^{\mathrm{T}}(\mathbf{x}) K^{-1}\mathbf{k}(\mathbf{x})\right] = \mathrm{Tr}\left[K^{-1}\mathcal{E}_{\mathbf{x}}\left[\mathbf{k}(\mathbf{x})\mathbf{k}^{\mathrm{T}}(\mathbf{x})\right]\right]$. We also note that Equation 2.13 is independent of the test point $\mathbf{x}$ but still depends upon the choice of the training data $\mathcal{D}_n$. In order to obtain a proper learning curve for GP, $E_{\mathcal{D}_n}^g$ needs to be averaged[2] over the possible choices of the training data $\mathcal{D}_n$. However, it is very difficult to obtain the analytical form of E^g for a GP as a function of n. Because of the presence of the $\mathbf{k}^{\mathrm{T}}(\mathbf{x}) K^{-1}\mathbf{k}(\mathbf{x})$ term in Equation 2.10, the matrix K and vector $\mathbf{k}(\mathbf{x})$ depend on the location of the training points; the calculations of the averages with respect to the data points seems very hard. This motivates looking for upper and lower *bounds* on the learning curve for GP.

2.5 Bounds on the learning curve

For the noiseless case, a lower bound on the generalisation error after n observations is due to Michelli and Wahba (1981). Let $\eta_1, \eta_2, \ldots$ be the ordered eigenvalues of the covariance function on some domain of the input space $\mathcal{X}$. They showed that $E^g(n) \geq \sum_{k=n+1}^{\infty} \eta_k$. Plaskota (1996) gives a bound on the learning curve for the noisy case; since the bound again considers the projection

[2]Hansen (1993) showed that for linear regression models it is possible to average over the distribution of the training sets.

of the random function onto the first n eigenfunctions, it is not expected that it will be tight for observations which consist of function evaluations.

Other results that we are aware of pertain to asymptotic properties of $E^g(n)$. Ritter (1996) has shown that for an optimal sampling of the input space, the asymptotics of the generalisation error is $O\left(n^{-(2s+1)/(2s+2)}\right)$ for a random process which obeys to the Sacks-Ylvisaker[3] conditions of order s (see Ritter et al. (1995) for more details on Sacks-Ylvisaker conditions). In general, the Sacks-Ylvisaker order of the MB_r covariance function is $s = r - 1$. For example an MB_1 process has $s = 0$ and hence the generalisation error shows a $n^{-1/2}$ asymptotic decay. If $\mathcal{X} \subset \mathbb{R}$, the asymptotically optimal design of the input space is the uniform grid.

Silverman (1985) proved a similar result for random designs. Haussler and Opper (1997) have developed general (asymptotic) bounds for the expected log-likelihood of a test point after seeing n training points.

In the following we introduce upper and lower bounds on the learning curve of a GP in a non-asymptotic regime. An upper bound is particularly useful in practice as it provides an (over)estimate of the number of examples needed to give a certain level of performance. A lower bound is similarly important because it contributes to fix the limit which can not be outperformed by the model.

The bounds presented are derived from two different approaches. The first approach makes use of the particular form assumed by the generalisation error at $\mathbf{x}$ ($E^g_{\mathcal{D}_n}(\mathbf{x}) = \sigma^2_n(\mathbf{x})$). As the error bar generated by one data point is greater than that generated by n data points, the former can be considered as an upper bound of the latter. Since this observation holds for the variance due to each one the data points, the envelope of the surfaces generated by the variances due to each data point is also an upper bound of $\sigma^2_n(\mathbf{x})$. In particular as $\sigma^2_n(\mathbf{x}) = E^g_{\mathcal{D}_n}(\mathbf{x})$ (cf. Equation 2.10), the envelope is an upper bound of the generalisation error of the GP. Following this argument, we can assert that an upper bound on $E^g_{\mathcal{D}_n}(\mathbf{x})$ is the one generated by every GP trained with a subset of $\mathcal{D}_n$. The larger the subset of $\mathcal{D}_n$ the tighter the bound.

The two upper bounds we present differ in the number of training points considered in the evaluation of the covariance; the derivation of the one-point upper bound $E^u_1(n)$ and the two-point upper bound $E^u_2(n)$ are presented in Section 2.5.1 and Section 2.5.2 respectively. Section 2.5.3 reports the asymptotic expansion of $E^u_1(n)$ and $E^u_2(n)$ in terms of λ and σ^2_ν.

The second approach is based on the expansion of the stochastic process in terms of the eigenfunctions of the covariance function. Within this framework, Opper found bounds on the training and generalisation error (Opper and Vivarelli, 1999) in terms of the eigenvalues of $C_p(\mathbf{x}, \mathbf{x}')$; the lower bound $E^l(n)$ obtained is presented in Section 2.5.4. One can also derive a second lower bound from the estimation of the Kullback-Leibler distance between the modelled and the true joint distribution of the stochastic function (Opper, 1997). The derivation of this lower bound is

[3]Loosely speaking, a stochastic process possessing s mean-square derivatives but not $s+1$ is said to satisfy the Sacks-Ylvisaker conditions of order s.

presented in Appendix A.6.

In order to have tractable analytical expressions, all the bounds have been derived by introducing three assumptions;

i The input space $\mathcal{X}$ is restricted to the interval $[0,1]$;

ii The probability density distribution of the input points is uniform: $p(x) = 1, x \in [0,1]$;

iii The prior covariance function $C_p(x,x')$ is stationary.

2.5.1 The one-point upper bound $E_1^u(n)$

For the derivation of the one-point upper bound, let us consider the error bar generated by one data point x^i. Since $C(0) = C_p(x^i,x^i) + \sigma_\nu^2 = K$, Equation 2.2 becomes

$$\sigma_1^2(x) = C(0) - \frac{C_p^2(x-x^i)}{C(0)}.$$

For x far away from the training point x^i, $\sigma_1^2(x) \sim C(0)$; the confidence on the prediction for a test point lying far apart from the data point x^i is quite low as the error bar is large. The closer x to x^i, the smaller the error bar on $\hat{y}(x)$. When $x = x^i$, $\sigma_1^2(x) = \sigma_\nu^2(1+r)$ where $r = C_p(0)/C(0)$. Irrespective of the value of $C_p(0)$, r varies from 0 to 1. As normally $C_p(0) \gg \sigma_\nu^2$, $r \sim 1$ and thus $\sigma_1^2(x) \sim 2\sigma_\nu^2$. So far we have not used any hypothesis concerning the dimension of the variable x, thus this observation holds regardless the dimension of the input space.

The effect of just one data point helps in introducing the first upper bound. The interval $[0,1]$ is split up in n subintervals $[a^i,b^i], i = 1 \ldots n$ (where $a^i = (x^i + x^{i-1})/2$ and $b^i = (x^{i+1} + x^i)/2$) centred around the i-th data point x^i, with $a^1 = 0$ and $b^n = 1$.

Let us consider the i-th training point and the error bar $\sigma_1^2(x)$ generated by x^i. When $x \in [a^i,b^i]$, $E_{\mathcal{D}_n}^g(x) \leq \sigma_1^2(x)$; this relation is illustrated in Figure 2.3, where the envelope of the surfaces of the errors due to each datapoint (denoted by $E_{\mathcal{D}_1}^g(\mathbf{x})$) is an upper bound of the overall generalisation error. Since we are dealing with positive functions, an upper bound of the expected generalisation error on the interval $[a^i,b^i]$ can be written as

$$\int_{a^i}^{b^i} E_{\mathcal{D}_n}^g(x)\,p(x)\,dx \leq \int_{a^i}^{b^i} \sigma_1^2(x)\,p(x)\,dx, \tag{2.14}$$

where $p(x)$ is the distribution of the test points. Summing up the contributions coming from each training datapoint in both sides of Equation 2.14 and setting $p(x) = 1$, we obtain

$$E_{\mathcal{D}_n}^g = \sum_{i=1}^{n} \int_{a^i}^{b^i} E_{\mathcal{D}_n}^g(x)\,dx \leq \sum_{i=1}^{n} \int_{a^i}^{b^i} \sigma_1^2(x)\,dx. \tag{2.15}$$

The interval $[a^i,b^i]$ (where the variance $\sigma_1^2(x)$ due to x^i contributes to Equation 2.14) is also shown in Figure 2.3.

Under the assumption of the stationarity of the covariance function, integrals such as those in the right hand side of Equation 2.15 depend only upon differences of adjacent training points

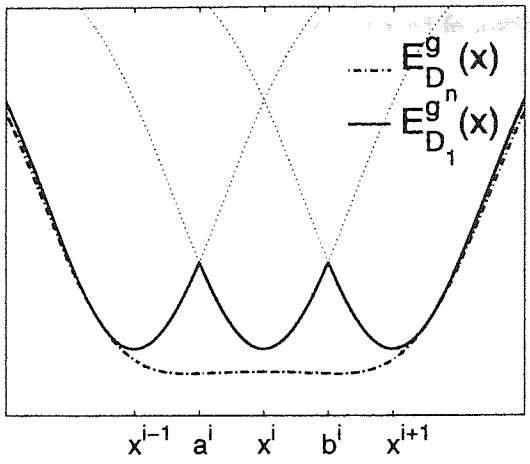


Figure 2.3: The figure suggests a pictorial argument for the upper bound $E_1^u(n)$. The solid and the dash-dotted lines indicate the bound and the actual generalisation error, respectively. The dotted lines are the generalisation errors evaluated considering training sets composed by each training point singularly, i.e. $\mathcal{D}_1 = \{x^{i-1}\}$, $\mathcal{D}_1 = \{x^i\}$ and $\mathcal{D}_1 = \{x^{i+1}\}$. As explained in the text, $E^g_{\mathcal{D}_n}(x) \leq E^g_{\mathcal{D}_1}(x)$ for all the input points of the input space and thus $E^g_{\mathcal{D}_1}(x)$ is regarded as an upper bound of $E^g_{\mathcal{D}_n}(x)$. $[a^i, b^i]$ specifies the interval of integration of Equation 2.14, as explained in the text.

(i.e. $x^i - x^{i-1}$ and $x^{i+1} - x^i$). Thus the right hand side of Equation 2.15 can be rewritten as

$$E^g_{\mathcal{D}_n} \leq \sum_{i=1}^{n} \int_{a^i}^{b^i} \sigma_1^2(x)\, dx \tag{2.16}$$

$$= C(0) \sum_{i=1}^{n} (b^i - a^i) - \frac{1}{C(0)} \sum_{i=1}^{n} \left[\int_{a^i}^{x^i} C_p^2 (x^i - x)\, dx + \int_{x^i}^{b^i} C_p^2 (x - x^i)\, dx \right]$$

$$= C(0) - \frac{1}{C(0)} \left[I(x^1) + 2 \sum_{i=2}^{n} I\left(\frac{x^i - x^{i-1}}{2} \right) + I(1 - x^n) \right] \tag{2.17}$$

where

$$I(\tau) = \int_0^{\tau} C_p^2(\xi) d\xi. \tag{2.18}$$

Equation 2.17 can be derived changing the variables in the two integrals of Equation 2.16 as $\xi = x^i - x$ and $\xi = x - x^i$, respectively. Equation 2.17 is an upper bound on $E^g_{\mathcal{D}_n}$ and still depends upon the choice of the training data $\mathcal{D}_n$ through the intervals of integration. We note that the arguments of the integrals $I(\cdot)$ in Equation 2.17 are the differences between adjacent training points. Denoting those differences with $\omega^i = x^{i+1} - x^i$ (where $\omega^0 = x^1$ and $\omega^n = 1 - x^n$), we can model their probability density distribution by using the theory of order statistics (David, 1970). Given a uniform distribution of n training data over the interval $[0, 1]$, the density distribution of the differences between adjacent points is $p(\omega) = n(1 - \omega)^{n-1}$. Since this is true for all the differences ω^i we can omit the superscript i and thus the expectation of the integrals in Equation 2.17 over $p(\omega)$ is

$$\mathcal{E}_\omega \left[I(\omega^0) + 2 \sum_{i=2}^{n} I\left(\frac{\omega^i}{2} \right) + I(\omega^n) \right] = 2(n-1)\mathcal{E}_\omega [I(\omega/2)] + 2\mathcal{E}_\omega [I(\omega)]. \tag{2.19}$$

Both the integrals $\mathcal{E}_\omega [I(\omega/2)]$ and $\mathcal{E}_\omega [I(\omega)]$ can be calculated following a similar procedure. Let us consider $\mathcal{E}_\omega [I(\omega)]$:

$$\begin{aligned}\mathcal{E}_\omega [I(\omega)] &= \int_0^1 I(\omega)\, n(1-\omega)^{n-1}\, d\omega \\ &= -[I(\omega)(1-\omega)^n]_0^1 + \int_0^1 C_p^2(\omega)(1-\omega)^n\, d\omega \\ &= \int_0^1 C_p^2(\omega)(1-\omega)^n\, d\omega,\end{aligned}$$

where the second line has been obtained integrating by parts. The last line follows from the fact that $[I(\omega)(1-\omega)^n]_0^1 = 0$.

We are now able to write an upper bound on the learning curve as

$$E^g(n) \leq E_1^u(n) \doteq C(0) - \frac{1}{C(0)}\left[(n-1)\int_0^1 C_p^2\left(\frac{\omega}{2}\right)(1-\omega)^n\, d\omega + 2\int_0^1 C_p^2(\omega)(1-\omega)^n\, d\omega\right]. \tag{2.20}$$

The calculations of the integrals in the above expression are straightforward though they involve the evaluation of hyper-geometric functions (because of the term $(1-\omega)^n$). As the evaluation of such functions is computationally intensive, and we found it preferable to evaluate Equation 2.20 numerically.

2.5.2 The two-points upper bound $E_2^u(n)$

The second bound we introduce is the natural extension of the previous idea: it applies the property of the variance of Equation 2.11 by using two data points. By construction, we expect that it will be tighter than the one introduced in Section 2.5.1.

Let us consider two adjacent data points x^i and x^{i+1} of the interval $[0,1]$, with $x^i < x^{i+1}$. By the same argument presented in the previous section, the following inequality holds:

$$\int_{x^i}^{x^{i+1}} E_{\mathcal{D}_n}^g(x)\, p(x)\, dx \leq \int_{x^i}^{x^{i+1}} \sigma_2^2(x)\, p(x)\, dx, \tag{2.21}$$

where $\sigma_2^2(x)$ is the variance on the prediction $\hat{y}(x)$ generated by the data points x^i and x^{i+1}. Similarly to Equation 2.15, summing up the contributions of both sides of Equation 2.21 we obtain an upper bound on the generalisation error:

$$E_{\mathcal{D}_n}^g = \sum_{i=0}^{n}\int_{x^i}^{x^{i+1}} E_{\mathcal{D}_n}^g(x)\, dx \leq \sum_{i=0}^{n}\int_{x^i}^{x^{i+1}} \sigma_2^2(x)\, dx, \tag{2.22}$$

where we have defined $x^0 = 0$ and $x^{n+1} = 1$. As the covariance matrix generated by two data points is a 2×2 matrix, it is straightforward to evaluate Equation 2.22. Considering the two training data x^i and x^{i+1}, the covariance matrix of the GP is

$$K = \begin{pmatrix} C(0) & C_p(x^{i+1} - x^i) \\ C_p(x^{i+1} - x^i) & C(0) \end{pmatrix}.$$

From the evaluation of the determinant of K as $\Delta\left(x^{i+1}-x^{i}\right)=\left(C\left(0\right)\right)^{2}-\left(C_{p}\left(x^{i+1}-x^{i}\right)\right)^{2}$, it follows that

$$K^{-1}=\frac{1}{\Delta\left(x^{i+1}-x^{i}\right)}\begin{pmatrix} C\left(0\right) & -C_{p}\left(x^{i+1}-x^{i}\right) \\ -C_{p}\left(x^{i+1}-x^{i}\right) & C\left(0\right)\end{pmatrix}.$$

As the covariance vector for the test point x is $\mathbf{k}\left(x\right)=\left(C_{p}\left(x-x^{i}\right),C_{p}\left(x^{i+1}-x\right)\right)^{\mathrm{T}}$, the variance assumes the form

$$\sigma_{2}^{2}\left(x\right)=C\left(0\right)-\frac{C\left(0\right)\left(C_{p}^{2}\left(x^{i+1}-x\right)+C_{p}^{2}\left(x-x^{i}\right)\right)-2C_{p}\left(x^{i+1}-x^{i}\right)C_{p}\left(x-x^{i}\right)C_{p}\left(x^{i+1}-x\right)}{\Delta\left(x^{i+1}-x^{i}\right)}.$$

Changing variables in the covariances $C_{p}\left(x^{i+1}-x^{i}\right)$ and $C_{p}\left(x-x^{i}\right)$ (as $\xi=x^{i+1}-x$ and $\xi=x-x^{i}$, respectively), it turns out that the upper bound generated by $\sigma_{2}^{2}\left(x\right)$ in the interval $\left[x^{i},x^{i+1}\right]$ (when $i\neq 0,n$), is

$$\int_{x^{i}}^{x^{i+1}}\sigma_{2}^{2}\left(x\right)dx=C\left(0\right)\left(x^{i+1}-x^{i}\right)-\frac{2\left(I_{1}\left(x^{i+1}-x^{i}\right)-I_{2}\left(x^{i+1}-x^{i}\right)\right)}{\Delta\left(x^{i+1}-x^{i}\right)},$$

where

$$\begin{aligned} I_{1}\left(\tau\right) &= C\left(0\right)\int_{0}^{\tau}C_{p}^{2}\left(\xi\right)d\xi \text{ and} \\ I_{2}\left(\tau\right) &= C_{p}\left(\tau\right)\int_{0}^{\tau}C_{p}\left(\xi\right)C_{p}\left(\tau-\xi\right)d\xi. \end{aligned}$$

It is noticeable that, similarly to Equation 2.17, also the integrals $I_{1}\left(\cdot\right)$, $I_{2}\left(\cdot\right)$ and the determinant $\Delta\left(x^{i+1}-x^{i}\right)$ depend upon the length of the interval of integration $\omega^{i}=x^{i+1}-x^{i}$. We evaluate the contributions to the upper bound over the intervals $\left[0,x^{1}\right]$ and $\left[x^{n},1\right]$ by integrating the variance $\sigma_{1}^{2}\left(x\right)$ generated by x^{1} and x^{n} over $\left[0,x^{1}\right]$ and $\left[x^{n},1\right]$ respectively. Hence the right hand side of Equation 2.22 can be rewritten as

$$E_{\mathcal{D}_{n}}^{g}\leq C\left(0\right)-2\sum_{i=2}^{n-1}\frac{I_{1}\left(\omega^{i}\right)-I_{2}\left(\omega^{i}\right)}{\Delta\left(\omega^{i}\right)}-\frac{1}{C\left(0\right)}\left(I\left(\omega^{1}\right)+I\left(\omega^{n}\right)\right) \tag{2.23}$$

where $I\left(\cdot\right)$ is defined in Equation 2.18. Details of the calculations of Equation 2.23 for $\mathrm{MB_r}$ and SE are reported in Appendix A.4.

Equation 2.23 is still dependent on the distribution of the training data because it is a function of the distances between adjacent training points ω^{i}. Similar to Equation 2.17, we obtain an upper bound independent of the training data by integrating Equation 2.19 over the distribution of the differences $p\left(\omega\right)=n\left(1-\omega\right)^{n-1}$:

$$E^{g}\left(n\right)\leq E_{2}^{u}\left(n\right)\doteq C\left(0\right)-2\left(n-1\right)\mathcal{E}_{\omega}\left[\frac{\left(I_{1}\left(\omega\right)-I_{2}\left(\omega\right)\right)}{\Delta\left(\omega\right)}\right]-\frac{2}{C\left(0\right)}\mathcal{E}_{\omega}\left[I\left(\omega\right)\right]. \tag{2.24}$$

The calculation of the integrals with respect to ω in $E_{2}^{u}\left(n\right)$ are complicated by the determinant $\Delta\left(\omega\right)$ in the denominator and by the distribution $n\left(1-\omega\right)^{n-1}$, so we preferred to evaluate them numerically as we did for $E_{1}^{u}\left(n\right)$.

2.5.3 Asymptotics of the upper bounds

From Equation 2.20, an expansion of $E_1^u(n)$ in terms of λ and σ_ν^2 in the limit of a large amount of training data can be obtained. The expansion depends upon the covariance function we are dealing with. Expanding the covariance function around 0 (see Appendix A.5), the asymptotic form of $E_1^u(n)$ for MB_1 is

$$E_1^u(n) \sim C(0)\left[1 - r^2 + \frac{r^2}{n\lambda}\right] + O\left(n^{-2}\right), \tag{2.25}$$

whereas for the functions MB_2, MB_3 and SE it is

$$E_1^u(n) \sim C(0)\left[1 - r^2 + \frac{r^2}{n^2\lambda^2}\right] + O\left(n^{-3}\right), \tag{2.26}$$

where $r = C_p(0)/C(0)$.

The asymptotic value of $E_1^u(n)$ depends neither on the lengthscale of the process nor on the covariance function but is a function of the ratio r:

$$\lim_{n\to\infty} E_1^u(n) = C(0)\left(1 - r^2\right) = \sigma_\nu^2(1 + r). \tag{2.27}$$

As we pointed out in Section 2.5.1, this is the minimum generalisation error achievable by a GP when it is trained with just one datapoint. The $n \to \infty$ scenario corresponds to the situation in which every test point is close to a datapoint. As mentioned at the beginning of this Section, the asymptotics of the learning curve for the MB_r and SE covariance functions are $O\left(n^{(2r-1)/2r}\right)$ and $O\left(n^{-1}\log n\right)$ respectively. Although the expansions of $E_1^u(n)$ decay asymptotically faster than the learning curves, they reach an asymptotic plateau $\sigma_\nu^2(1 + r) \geq \sigma_\nu^2$. We also note that the asymptotic values $E_1^u(n)$ becomes closer to the true noise level when $r \ll 1$, i.e. for the unrealistic case $\sigma_\nu^2 \gg C_p(0)$.

The smoothness of the process enters into the asymptotics through a factor $O\left(r^2/(\lambda n)\right)$ for MB_1 and $O\left(r^2/\left(\lambda^2 n^2\right)\right)$ for MB_2, MB_3 and SE. This factor affects the rate of approach to the asymptotic value $\sigma_\nu^2(1 + r)$ of $E_1^u(n)$. We notice that larger lengthscales and noise levels increase the rate of decay of $E_1^u(n)$ to the asymptotic plateau.

The asymptotic form of $E_2^u(n)$ for the MB_1, MB_2, MB_3 and SE covariance functions is

$$E_2^u(n) \sim C(0)\left(1 - \frac{2r^2}{1+r}\right) + \frac{a}{n+1} + O\left(n^{-2}\right), \tag{2.28}$$

where the value of a depends upon the choice of the covariance function and $r = C_p(0)/C(0)$; details of the derivation are reported in Appendix A.5. Similarly to the expansion of $E_1^u(n)$, the decay rate of $E_2^u(n)$ is faster than the asymptotic decay of the actual learning curves but it reaches an asymptotic plateau of

$$\lim_{n\to\infty} E_2^u(n) = C(0)\left(1 - \frac{2r^2}{1+r}\right) = \sigma_\nu^2\left(1 + \frac{r}{1+r}\right). \tag{2.29}$$

It is straightforward to verify that the asymptotic plateau of $E_2^u(n)$ is lower than the one of $E_1^u(n)$ and that it corresponds to the error bar estimated by a GP with two observations located at the test point.

2.5.4 The lower bound $E^l(n)$

Opper proposed a bound on the learning curve and on the training error based on the decomposition of the stochastic process $y(\mathbf{x})$ in terms of the eigenfunctions of the covariance $C_p(\mathbf{x}, \mathbf{x}')$ (Opper and Vivarelli, 1999).

Denoting with $\varphi_k(\mathbf{x}), k = 1 \ldots \infty$ a complete set of functions satisfying the integral equation

$$\int C_p(\mathbf{x}, \mathbf{x}') \varphi_k(\mathbf{x}) p(\mathbf{x}) d\mathbf{x} = \eta_k \varphi_k(\mathbf{x}),$$

the Bayesian generalisation error $E^g(\mathbf{x}, \mathcal{D}_n) = \mathcal{E}_y\left[(y(\mathbf{x}) - \hat{y}(\mathbf{x}))^2\right]$ (where $y(\mathbf{x})$ is the true underlying stochastic function and $\hat{y}(\mathbf{x})$ is the GP prediction) can be written in terms of the eigenvalues of $C_p(\mathbf{x}, \mathbf{x}')$. In particular, after an average over the distribution of the input data, $E^g(\mathcal{D}_n)$ can be written as $E^g(\mathcal{D}_n) = \sigma_\nu^2 \mathrm{Tr}\left[\Lambda\left(\sigma_\nu^2 \mathbb{I} + \Lambda V\right)^{-1}\right]$, where Λ is the infinite dimension diagonal matrix of the eigenvalues and V is a matrix depending on the training data, i.e. $V_{kl} = \sum_{i=1}^n \varphi_k(\mathbf{x}^i) \varphi_l(\mathbf{x}^i)$.

By using Jensen's inequality, it is possible to show that a lower bound of the learning curve and an upper bound of the training error is (Opper and Vivarelli, 1999)

$$E_y^l(n) \doteq \sigma_\nu^2 \sum_{k=1}^{\infty} \frac{\eta_k}{(\sigma_\nu^2 + n\eta_k)}. \tag{2.30}$$

In this Chapter we mean to compare this lower bound to the actual learning curve of a GP. As our bounds are on t rather than y, we must add σ_ν^2 to the expression obtained in Equation 2.30 obtaining an actual lower bound of

$$E^l(n) \doteq \sigma_\nu^2 \left(1 + \sum_{k=1}^{\infty} \frac{\eta_k}{(\sigma_\nu^2 + n\eta_k)}\right). \tag{2.31}$$

2.6 Numerical simulations

As we pointed out in Section 2.4, the analytic calculation of the learning curve of a GP is not analytically tractable. Since the generalisation error

$$E_{\mathcal{D}_n}^g = \int \left(C_p(\mathbf{0}) + \sigma_\nu^2 - \mathbf{k}^{\mathrm{T}}(\mathbf{x}) K^{-1} \mathbf{k}(\mathbf{x})\right) p(\mathbf{x}) d\mathbf{x} \tag{2.32}$$

is a complicated function of the training data (which are inside the elements of $\mathbf{k}(\mathbf{x})$ and K^{-1}), it is problematic to perform an integration over the distribution of the training points. For comparing the learning curve of the GP with the bounds we found, we need to evaluate the expectation of the integral in Equation 2.32 over the distribution of the data: $E^g(n) = \mathcal{E}_{\mathcal{D}_n}\left[E_{\mathcal{D}_n}^g\right]$. An estimation of $E^g(n)$ can be obtained using a Monte Carlo approximation of the expectation. We used 50 generations of training data, sampling uniformly the input space $[0, 1]$. For each generation, the expected generalisation error for a GP has been evaluated using up to 1000 datapoints. Using the 50 generations of training data, we can obtain an estimate of the learning curve $E^g(n)$ and its 95% confidence interval.

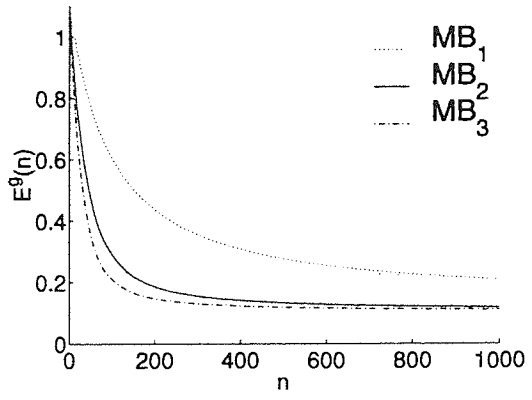


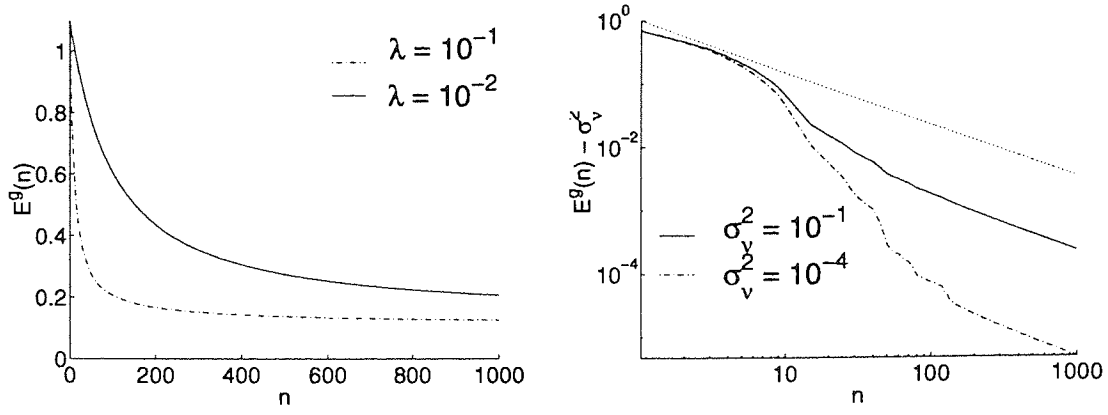
Figure 2.4: The Figure shows the graph of the learning curve computed for the covariance functions MB_1, MB_2 and MB_3 indicated by the dotted, solid and dash-dotted lines, respectively.

Since this study is focused on the behaviour of bounds on learning curve on GP, we assume the true values of the parameters of the GP are known. So we chose the value of the constant κ_ν for the covariance functions MB_1, MB_2 and MB_3 (see Equation 2.4) such that $C_p(0) = 1$ and we allowed the lengthscale λ and the noise level σ_ν^2 to assume several values ($\lambda = 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1$ and $\sigma_\nu^2 = 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1$).

To begin with, we study how the smoothness of a process affects the behaviour of the learning curve. The empirical learning curves of Figure 2.4 have been obtained for processes whose covariance functions are MB_1, MB_2 and MB_3, with $\lambda = 0.01$ and $\sigma_\nu^2 = 0.1$. We can notice that all the learning curves exhibit an initial linear decrease. This can be explained considering that without any training data, the generalisation error is the maximum allowable by the model ($C(0) = C_p(0) + \sigma_\nu^2$). The introduction of a training point x^1 creates an hole on the error surface; the volume of the hole is proportional to the value of the lengthscale and depends on the covariance function. The addition of a new data point x^2 will have the effect of generating a new hole in the surface. With such a few data points it is likely that the two data lie down far apart one from the other, giving rise to two distinct holes. Thus the effect that a small dataset exerts to *pull down* the error surface is proportional to the amount of training points and explains the initial linear trend.

Concerning the asymptotic behaviour of the learning curves, we have verified that they agree with the theoretical analysis carried out by Ritter (1996). In particular, a log-log graph of the learning curves with the MB_r covariance functions shows an asymptotic behaviour as $O\left(n^{-(2r-1)/2r}\right)$. A log-log graph of the learning curve obtained with the SE covariance function shows an asymptotic decay rate of $O\left(n^{-1}\log n\right)$ (Opper, 1997). We have also noted that the smoother the process described by the covariance function the smaller the amount of training data needed to reach the asymptotic regime.

The behaviour of the learning curves is affected also by the value of the lengthscale of the process and by the noise level and this is illustrated in Figure 2.5. The learning curves shown in Figure 2.5(a) have been obtained for the MB_1 covariance function setting the noise level $\sigma_\nu^2 = 0.1$



(a) $\sigma_\nu^2 = 0.1$ (b) $\lambda = 0.1$

Figure 2.5: Figure 2.5(a) shows the graphs of the learning curves for the MB_1 covariance function obtained for a fixed noise level $\sigma_\nu^2 = 0.1$ and lengthscales $\lambda = 10^{-2}, 10^{-1}$; the lengthscale contributes to stretch the input domain and a similar effect is observed on the learning curves. A log-log graph of the learning curve of a MB_3 stochastic process is shown in Figure 2.5(b), with $\lambda = 10^{-1}$ and the noise variance is set to 10^{-4} (solid line) and 10^{-1} (dash-dotted line); the dotted line draws the asymptotic behaviour of the learning curve. The curve with a larger noise level attains the asymptotic regime with fewer datapoints than with a lower noise variance.

and varying the values of the parameters $\lambda = 10^{-2}, 10^{-1}$. Intuitively, Figure 2.5(a) suggests that decreasing the lengthscale stretches the early behaviour of the learning curve and the approach to the asymptotic plateau lasts longer; this is due to the effect induced by different values of the lengthscale which stretch or compress the input space. We have verified that rescaling the amount of data n by the ratio of the two lengthscales, the two curves of Figure 2.5(a) lay on top of each other.

The variation of the noise level shifts the learning curves from the prior value $C_p(0)$ by an offset equal to the noise level itself (cf. Equation 2.10); in order to see any significant effect of the noise on the learning curve, Figure 2.5(b) shows a log-log graph of $E^g(n) - \sigma_\nu^2$ obtained for a stochastic process with MB_3 covariance function, setting $\lambda = 0.1$ and noise variance $\sigma_\nu^2 = 10^{-4}, 10^{-1}$. We can notice two main effects. The noise variance affects the actual values of the generalisation error since the learning curve obtained with high noise level is always above the one obtained with a low noise level. A second effect concerns the amount of data necessary to reach the asymptotic regime; the learning curve characterised by a small noise level needs fewer datapoints to attain to the asymptotic regime.

Stochastic processes with different covariance functions and different values of lengthscales and noise variance behave in a similar way.

In the following we discuss the results in two main subsections. Results about the bounds $E_1^u(n)$ and $E_2^u(n)$ are presented in Section 2.6.1, whereas results concerning the lower bound of Section 2.5.4 are shown in Section 2.6.2. As the results we obtained for these experiments show

common characteristics, we present the bounds of the learning curve obtained by setting $\lambda = 0.01$ and $\sigma_\nu^2 = 0.1$.

2.6.1 The upper bounds $E_1^u(n)$ and $E_2^u(n)$

Each graph in Figure 2.6 shows the empirical learning curve with its confidence interval and the two upper bounds $E_1^u(n)$ and $E_2^u(n)$. The curves are shown for the MB_1, MB_2, MB_3 and the SE covariance functions.

For a limited amount of training data it is possible to notice that the upper error bar associated to $\mathcal{E}_{\mathcal{D}_n}[E^g(n)]$ lies above the actual upper bounds. This effect is due to the variability of the generalisation error for small data sets and suggests that the bounds are quite tight for small n. The effect disappears for large n, when the estimate of the generalisation error is less sensitive to the composition of the training set.

As expected, the two-point upper bound $E_2^u(n)$ is tighter than the one-point upper bound $E_1^u(n)$.

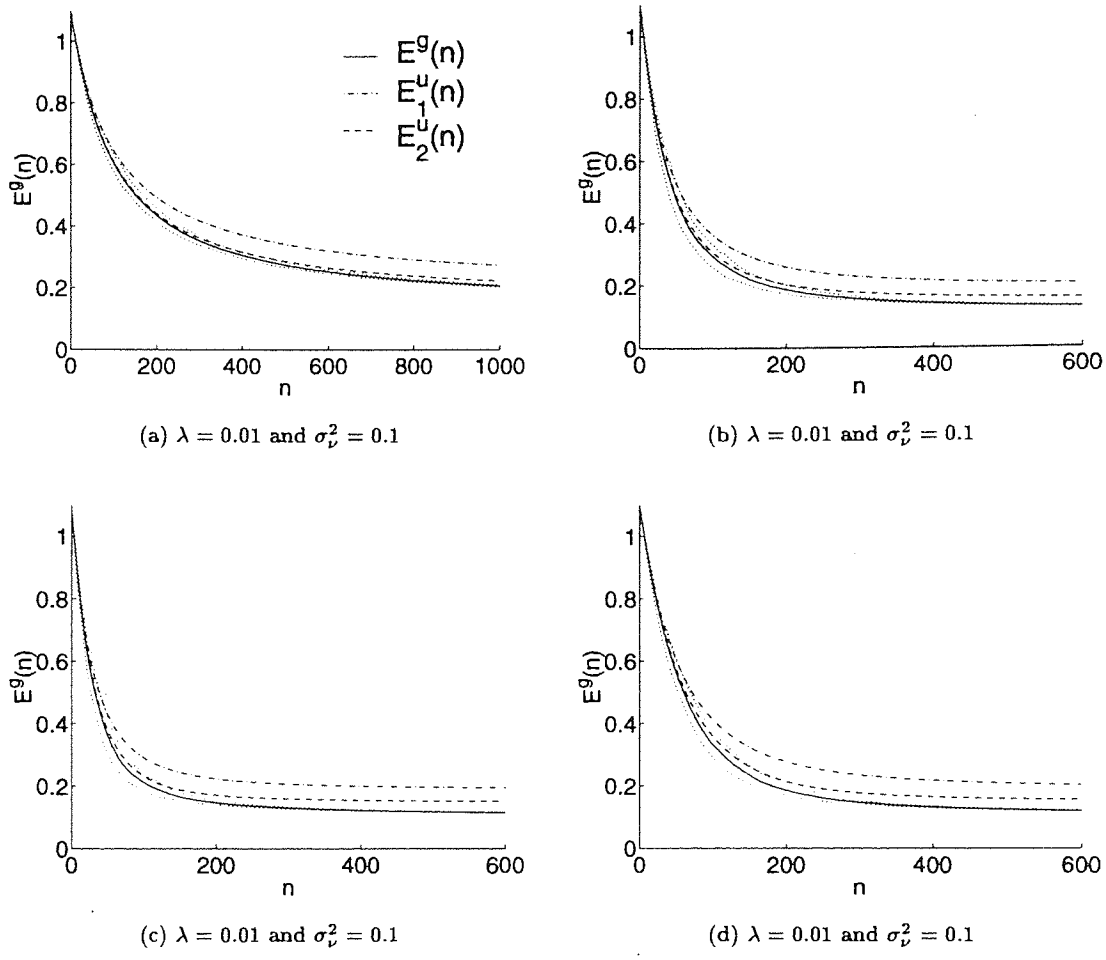
We note that the tightness of the upper bound depends upon the covariance function, being tighter for rougher processes (such as MB_1) and getting worse for smoother processes. This can be explained by recalling that covariance functions such as the MB_r correspond to Markov processes of order r (cf. Section 2.3). Although the Markov process is actually hidden by the presence of the noise, $E^g(n)$ is still more dependent on training data lying close to the test point x than on more distant points. Since the bounds $E_1^u(n)$ and $E_2^u(n)$ have been calculated by using only local information (namely the closest datapoint to the test point, or the closest datapoints to the left and right, respectively), it is natural that the more the variance at x depends on local data points, the tighter the bounds become.

For instance, let us consider MB_1, the covariance function of a first order Markov process. For the noise-free process, knowledge of data-points lying beyond the left and right neighbours of x does not reduce the generalisation error at x[4]. Although in the noisy case more distant data-points reduce the generalisation error (because of the term σ_ν^2 in the covariance matrix K), it is likely that local information is still the most important.

The bounds on the learning curves computed for MB_2 and MB_3 confirm this remark, as they are looser than for MB_1. For the SE covariance function, this effect still holds and is actually enlarged.

In Section 2.5.3 we have shown that the asymptotic behaviour of the bound $E_1^u(n)$ depends on the covariance function, being $O(n^{-1})$ for MB_1 and $O(n^{-2})$ for MB_2 and MB_3. Log-log graphs of the upper bounds confirm the analysis carried out in Section 2.5.3, where we showed that $E_1^u(n)$ and $E_2^u(n)$ approach asymptotic plateaux. In particular, $E_1^u(n)$ tends to $\sigma_\nu^2(1+r)$ as $O(n^{-1})$ for MB_1 and $O(n^{-2})$ for MB_2 and MB_3, whereas $E_2^u(n)$ tends to $\sigma_\nu^2(1+r/(1+r))$ as $O(n^{-1})$.

[4] This is because the process values at the training points and test point form a Markov chain, and knowledge of the process values to the left and right of the test point "blocks" the influence of more remote observations.



(a) $\lambda = 0.01$ and $\sigma_\nu^2 = 0.1$

(b) $\lambda = 0.01$ and $\sigma_\nu^2 = 0.1$

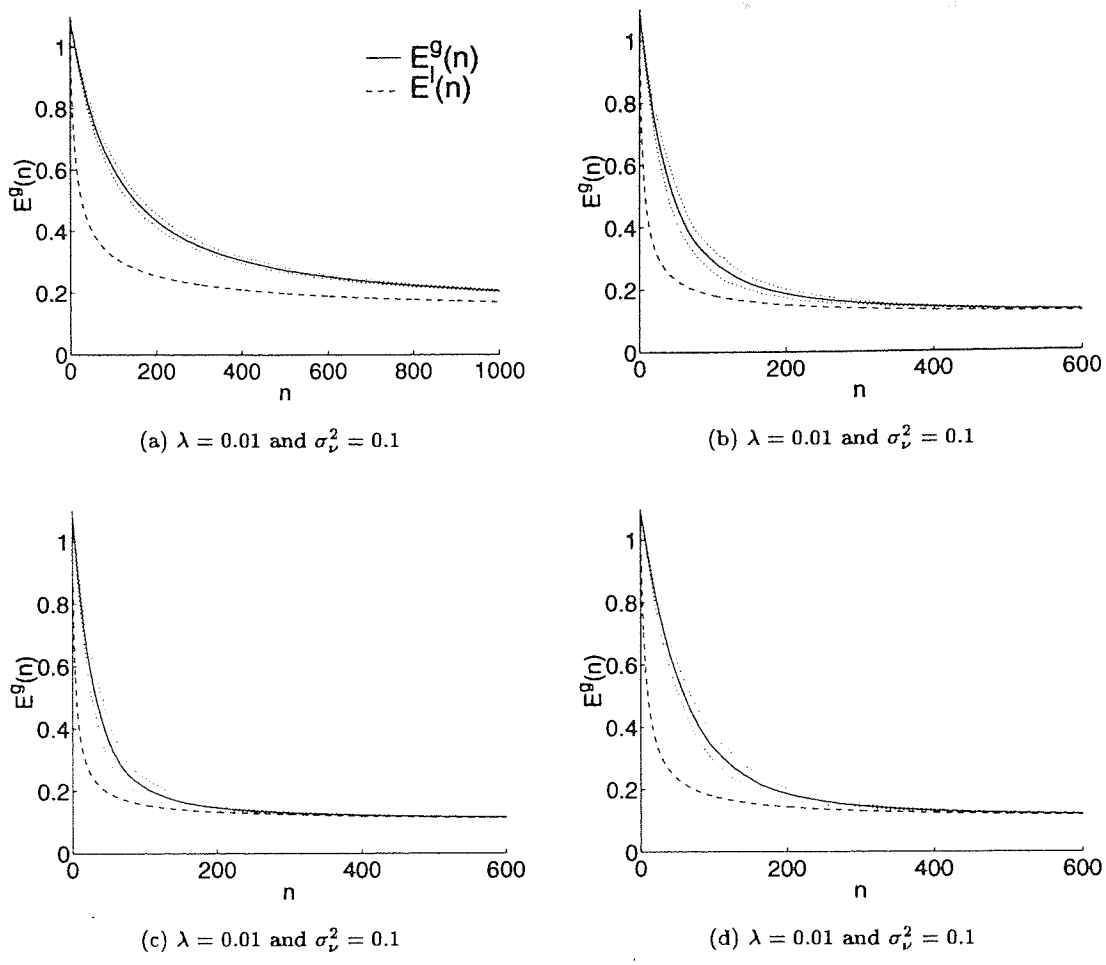
(c) $\lambda = 0.01$ and $\sigma_\nu^2 = 0.1$

(d) $\lambda = 0.01$ and $\sigma_\nu^2 = 0.1$

Figure 2.6: Figures 2.6(a), 2.6(b), 2.6(c) and 2.6(d) show the graphs of the learning curves and their upper bounds computed for the covariance functions MB_1, MB_2, MB_3 and the SE respectively. In all the graphs, the learning curve is drawn by the solid line and its 95% confidence interval is indicated by the dotted curves. The upper bounds $E_1^u(n)$ and $E_2^u(n)$ are indicated by the dash dotted and the dashed lines, respectively.

The quality of the bounds for processes characterised by different lengthscales and different noise levels are comparable to the ones described so far; the tightness of $E_1^u(n)$ and $E_2^u(n)$ still depend on the smoothness of the process. As explained at the beginning of this section, a variation of the lengthscale has the same effect of a rescaling in the number of training data.

For a fixed covariance function, we note that the bounds are tighter for lower noise variance; this is due to the fact that the lower the noise level the better the hidden Markov process manifests itself. For smaller noise levels the learning curve becomes closer to the bounds because the generalisation error relies on the local behaviour of the processes around the test data; on the contrary, a larger noise level hides the underlying Markov process thus loosening the bounds.



(a) $\lambda = 0.01$ and $\sigma_\nu^2 = 0.1$

(b) $\lambda = 0.01$ and $\sigma_\nu^2 = 0.1$

(c) $\lambda = 0.01$ and $\sigma_\nu^2 = 0.1$

(d) $\lambda = 0.01$ and $\sigma_\nu^2 = 0.1$

Figure 2.7: Figures 2.7(a), 2.7(b), 2.7(c) and 2.7(d) show the graphs of the learning curves and their lower bounds computed for the covariance functions MB_1, MB_2, MB_3 and the SE respectively. In all the graphs, the learning curve is drawn by the solid line and its 95% confidence interval is signed by the dotted curves. The lower bound $E^l(n)$ is indicated by the dashed line.

2.6.2 The bound $E^l(n)$

We have also run experiments computing the lower bound we obtained from Equation 2.31 for processes generated by the covariance priors MB_1, MB_2, MB_3 and SE .

Equation 2.31 shows that the evaluation of $E^l(n)$ involves the computation of an infinite sum of terms; we evaluated only those terms which add a significant contribution to the series. In particular, for each value of the lengthscale λ adopted during our experiments, we evaluated numerically a number of eigenvalues whose contributions to the lower bound $E^l(n)$ were not negligible, i.e. $\eta_k/\sigma_\nu^2 \geq \varepsilon$, where ε is the machine precision.

Figure 2.7 shows the results of the experiment in which we set $\lambda = 0.01$ and $\sigma_\nu^2 = 0.1$. The graphs of the lower bound lies below the empirical learning curve, being tighter for large amount of data; in particular for the smoothest processes with large amount of data, the 95% confidence intervals lay below the actual lower bound.

For $n \to \infty$, the lower bound tends to the noise level σ_ν^2. As with the empirical learning curve, log-log graphs of $E_y^l(n)$ show an asymptotic decay to zero as $O(n^{-(2r-1)/2r})$ and $O\left(n^{-1}\log n\right)$ for the MB_r and the SE covariance functions, respectively.

The graphs of Figure 2.7 show also that the tightness of the bound depends on the smoothness of the stochastic process; in particular smooth processes are characterised by a tight lower bound on the learning curve $E^g(n)$. This can be explained by observing that $E^l(n)$ is a lower bound on the learning curve and an upper bound of the training error. The values of smooth functions do not have large variation between training points and thus the model can infer better on test data; this reduces the generalisation error pulling it closer to the training error. Since the two errors sandwich the bound of Equation 2.31, $E^l(n)$ becomes tight for smooth processes.

We can also notice that the tightness of the lower bound depends on the noise level, being tight for high noise level and loose for small noise level. This is consistent with the fact that $E^l(n)$ is a function monotonically decreasing of the noise variance (Opper and Vivarelli, 1999).

2.7 Discussion

In this Chapter we have presented non-asymptotic upper and lower bounds for the learning curve of GPs. The theoretical analysis has been carried out for one-dimensional GPs characterised by several covariance functions and has been supported by numerical simulations.

Starting from the observation that increasing the amount of training data never worsens the Bayesian generalisation error, an upper bound on the learning curve can be estimated as the generalisation error of a GP trained with a reduced dataset. This means that for a given training set the envelope of the generalisation errors generated by one and two datapoints is an upper bound of the actual learning curve of the GP. Since the expectation of the generalisation error over the distribution of the training data is not analytically tractable, we introduced the two upper bounds $E_1^u(n)$ and $E_2^u(n)$ which are amenable to average over the distribution of the test and training points. In this study we have evaluated the expected value of the bounds; future directions of research should also deal with the evaluation of the variances.

In order to highlight the behaviour of the bounds with respect to the smoothness of the stochastic process, we investigated the bounds for the modified Bessel covariance function of order r (describing stochastic processes $r-1$ times mean-square differentiable) and the squared exponential function (describing processes mean square-differentiable up to the order ∞).

The experimental results have shown that the learning curves and their bounds are characterised by an early, linearly decreasing behaviour because of the effect exerted by each datapoint in pulling down the surface of the prior generalisation error. We also noticed that the tightness of the bounds depends on the smoothness of the stochastic processes. This is due to the facts that the bounds rely on subsets of the training data (i.e. one or two datapoints) and the modified Bessel covariance functions describe Markov processes of order r; although in our simulations the Markovian processes

were hidden by noise, the learning curves depend mainly on local information and our bounds become tighter for rougher processes.

We also investigated the behaviour of the curves with respect to the variation of the correlation lengthscale of the process and the variance of the noise corrupting the stochastic process. We noticed that the lengthscale stretches the behaviour of the curves effectively rescaling the number of training data. As the noise level has the effect of hiding the underlying Markov process, the upper bounds become tighter for smaller noise variance.

The expansion of the bounds in the limit of large amount of data highlights an asymptotic behaviour depending upon the covariance function; $E_1^u(n)$ approaches the asymptotic plateau as $O\left(n^{-1}\right)$ (for the MB_1 covariance function) and as $O\left(n^{-2}\right)$ for smoother processes; the rate of decay to the plateau of $E_2^u(n)$ is $O\left(n^{-1}\right)$. Numerical simulations supported our analysis.

One limitation of our analysis is the dimension of the input space; the bounds have been made analytically tractable by using order statistics results after splitting up the one dimensional input space of the GP. In higher dimensional spaces the partition of the input space can be replaced by a Voronoi tessellation that depends on the data $\mathcal{D}_n$ but averaging over this distribution appears to be difficult. One can suggest an approximate evaluation of the upper bounds by an integration over a ball whose radius depends upon the number of examples and the volume of the input space in which the bound holds. In any case we expect that the effect due to larger input dimension is to loosen the upper bounds.

We also ran some experiments by using the lower bound proposed by Opper, based on the knowledge of the eigenvalues of the covariance function of the process. Since the bound $E^l(n)$ is also an upper bound on the training error, we observed that the bound is tighter for smooth processes, when the learning curve becomes closer to the training error. Also the noise can vary the tightness of $E^l(n)$; a low noise level loosens the lower bound. Unlike the upper bounds, the lower bound can be applied also in multivariate problems, as it is easily extended to high dimension input space; however it has been verified (Opper and Vivarelli, 1999) that the bound becomes less tight in input space of higher dimension.

Chapter 3

Discovering hidden features with Gaussian process regression

In Gaussian process regression the covariance between the outputs at input locations $\mathbf{x}$ *and* $\mathbf{x}'$ *is usually assumed to depend on the distance* $(\mathbf{x}-\mathbf{x}')^{\mathrm{T}} W (\mathbf{x}-\mathbf{x}')$*, where* W *is a positive semidefinite matrix.* W *is often taken to be diagonal, but if we allow* W *to be a general positive semidefinite matrix which can be tuned on the basis of training data, then an eigen-analysis of* W *shows that we are effectively creating hidden features, where the dimensionality of the hidden-feature space is determined by the data. We demonstrate the superiority of predictions using the general matrix over those based on a diagonal matrix on two test problems.*

3.1 Introduction

An important aspect of data analysis concerns the projection of multivariate data onto low dimensional manifolds. Generalisation capabilities of a system which learns from examples are significantly affected by the dimensionality of the input vector; in particular, input vectors with some irrelevant components may lead to a degradation of the generalisation capabilities since the learning system becomes more prone to overfit the training data.

Finding mappings which allow the discovery of relevant *directions* in the input space is therefore crucial; in this Chapter we show how to discover hidden features in data by using Gaussian Processes (GPs) for regression. Due to the equivalence between certain GPs and infinite neural networks (Neal, 1996), over the last few years predictions with GP have come to the fore; Rasmussen (1996) has demonstrated good performance of GP predictors on a number of tasks compared to Bayesian neural networks.

In GP prediction as applied by Rasmussen (1996), Williams and Rasmussen (1996) and others, the covariance between the outputs at locations $\mathbf{x}$ and $\mathbf{x}'$ is usually assumed to depend on the distance $(\mathbf{x}-\mathbf{x}')^T W(\mathbf{x}-\mathbf{x}')$, where W is a positive definite, *diagonal* matrix. This allows different dimensions in the input space can have different relevances to the prediction problem, cf. MacKay and Neal's idea of Automatic Relevance Determination (Neal, 1996).

The discovery of hidden features has been achieved in the past by a number of techniques (see e.g. the projection pursuit algorithm of Friedman and Tukey (1974) or the Sliced Inverse Regression of Li (1991)); clearly the ARD model is a special case, where these directions are parallel to the axes in the input feature space. In this Chapter we allow W to be a general positive semidefinite matrix (defining a Mahalanobis distance in the input space), thereby allowing general directions in the input space to be selected. We then compare the performance of GP predictors using the diagonal and full distance matrices on some regression problems.

The structure of the Chapter is as follows. GPs for regression have already been introduced in Section 2.2 for a general covariance function. In Section 3.2 we introduce the covariance function used in this work, explaining the rôle played by the distance matrix W in GP regression for multivariate problem; the parameterisation of the general matrix W and the criterion of optimisation of its elements are also presented. Section 3.2.1 shows how we compared the generalisation performances of the diagonal and the general distance matrices. The two methods have been compared on two regression tasks and the results of our experiments are shown in Section 3.3. A summary of the work done and some open questions are presented in Section 3.4.

3.2 Gaussian processes and prediction

In Section 2.2 we showed that GPs can be regarded as a subset of stochastic processes which can be fully specified by a mean $\mu(\mathbf{x}) = \mathcal{E}[Y(\mathbf{x})]$ and a covariance function $C_p(\mathbf{x}, \mathbf{x}') = \mathcal{E}[Y(\mathbf{x})\, Y(\mathbf{x}')]$. For the work below we shall set $\mu(\mathbf{x}) \equiv 0$. Although the GP formulation provides a prior over functions, for our purposes it suffices to note that the y-values $Y(\mathbf{x}^1), Y(\mathbf{x}^2), \ldots, Y(\mathbf{x}^n)$ corresponding to $\mathbf{x}$-values $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^n$ have a multivariate Gaussian distribution $\mathcal{N}(\mathbf{0}, K_p)$, where $(K_p)_{ij} = C_p(\mathbf{x}^i, \mathbf{x}^j)$. The specific form of the covariance function that we shall use is

$$C_p(\mathbf{x}, \mathbf{x}') = \sigma_p^2 \exp\left[-\frac{1}{2}(\mathbf{x}-\mathbf{x}')^{\mathrm{T}} W (\mathbf{x}-\mathbf{x}')\right]. \tag{3.1}$$

When W is a diagonal matrix the entry w_{ii} is the inverse of the squared *correlation length-scale* of the process along the direction i. In particular, we note that this model is closely related to the Automatic Relevance Determination method of MacKay and Neal (Neal, 1996); assuming that the inputs are normalised, a small lengthscale along a certain direction of the space highlights the relevance of the corresponding input feature.

Denoting with t^i the output-value corresponding to the input $\mathbf{x}^i$, let us suppose to have n data points $\mathcal{D}_n = \{(\mathbf{x}^1, t^1), (\mathbf{x}^2, t^2), \ldots, (\mathbf{x}^n, t^n)\}$. The t's are assumed to be generated from the true

y-values by adding Gaussian noise of variance σ_ν^2. Given the assumption of a Gaussian process prior over functions, it is a standard result (e.g. Whittle, 1963) that the predictive distribution $p(t|\mathbf{x}, \mathcal{D}_n)$ corresponding to a new input is $\mathcal{N}(\hat{y}(\mathbf{x}), \sigma^2(\mathbf{x}))$, with mean and variance

$$\hat{y}(\mathbf{x}) = \mathbf{k}^{\mathrm{T}}(\mathbf{x}) K^{-1} \mathbf{t} \tag{3.2}$$

$$\sigma^2(\mathbf{x}) = C_p(\mathbf{x}, \mathbf{x}) - \mathbf{k}^{\mathrm{T}}(\mathbf{x}) K^{-1} \mathbf{k}(\mathbf{x}), \tag{3.3}$$

where $K = K_p + \sigma_n^2 \mathbb{I}$, $\mathbf{k}^{\mathrm{T}}(\mathbf{x}) = \left(C_p(\mathbf{x}, \mathbf{x}^1), C_p(\mathbf{x}, \mathbf{x}^2), \ldots, C_p(\mathbf{x}, \mathbf{x}^n)\right)$ and $\mathbf{t}^{\mathrm{T}} = \left(t^1, t^2, \ldots t^n\right)$. The parameters of the GP are the elements of the distance matrix W, the prior covariance σ_p^2 and the variable σ_n^2 modelling the variance of the noise.

This method of prediction assumes that the process $y(\mathbf{x})$ we are modelling is really a function of the observable $\mathbf{x}$. However it is often the case that for real world problems the y is actually a function of a set of hidden features $\mathbf{z} \in \mathcal{Z} \subseteq \mathbb{R}^q$ which arise from a combination of the manifest variables $\mathbf{x}$. In particular we wish to study the problem in which the hidden features are a linear combination of the observable coordinates through a $q \times d$ matrix M, where $q < d$ (i.e. $\mathbf{z} = M\mathbf{x}$). In this case, the covariance of the function y is specified by Equation 3.1 but turns out to depend upon the estimation of the distance between hidden features $(\mathbf{z} - \mathbf{z}')^{\mathrm{T}} \Psi (\mathbf{z} - \mathbf{z}')$. Since $\mathbf{z} = M\mathbf{x}$, $(\mathbf{z} - \mathbf{z}') = M(\mathbf{x} - \mathbf{x}')$ and $W = M^{\mathrm{T}} \Psi M$.

A GP model depends on the parameters which describe the covariance function (i.e. σ_p^2, σ_n^2 and the elements of W). The training of a GP can be carried out by either estimating the parameters of the covariance function (for example, using the maximum likelihood method) or using a Bayesian approach and sampling from the posterior distribution over the parameters (Williams and Rasmussen, 1996). We follow the first approach, maximising the logarithm of the likelihood

$$\mathcal{L} = \log p(\mathcal{D}_n|\theta) = -\frac{1}{2} \log \det K - \frac{1}{2} \mathbf{t}^{\mathrm{T}} K^{-1} \mathbf{t} - \frac{n}{2} \log 2\pi \tag{3.4}$$

by optimising the free parameters of the GP which are in K.

The number of free parameters depends on the number of non-zero elements of the matrix W. Usually, W is chosen to be diagonal and thus the number of free parameters is $d+2$ (the d diagonal elements, σ_p^2 and σ_n^2). We notice that this parametrisation of W allows the discovery of relevant directions in the observed space; however, it does not lead to an estimation of a general mapping of $\mathcal{X}$ onto the feature space $\mathcal{Z}$ as the relevant directions are parallel to the axes in the input manifest space.

If q is not known in advance, it is preferable to use a general symmetric positive semidefinite matrix W. A parametrisation of such a matrix follows from the Choleski decomposition as $W = U^{\mathrm{T}} U$, where U is an upper triangular matrix with positive entries on the diagonal (Williams, 1996). Hence the factorisation of U turns out to be

$$U = \begin{pmatrix} \exp[u_{1,1}] & u_{1,2} & \ldots & u_{1,d} \\ 0 & \exp[u_{2,2}] & \ldots & u_{2,d} \\ 0 & 0 & \ldots & u_{3,d} \\ \ldots & \ldots & \ldots & \exp[u_{d,d}] \end{pmatrix}. \tag{3.5}$$

The elements on the diagonal are positive because of the exponential. Being symmetric, W has at most $d(d+1)/2$ independent entries and thus the total number of free parameters of the GP model is $2 + d(d+1)/2$.

We note that such a full distance matrix W allows an estimation of the matrix M from an eigen-decomposition of $W = V\Lambda V^{\mathrm{T}}$, where Λ is a diagonal matrix of the eigenvalues of W and V is the matrix whose columns are composed by the eigenvectors of W. The dimension of the hidden feature space $\mathcal{Z}$ can be inferred by the number of relevant eigenvalues of the matrix Λ (which are the inverse of the squared correlation lengths of the process along the directions of the hidden space). The directions of the hidden feature space are defined by the eigenvectors corresponding to the relevant eigenvalues; in particular the matrix composed by these eigenvectors gives an estimate of the mapping from $\mathcal{X}$ to $\mathcal{Z}$. In the following the diagonal and the general full correlation matrices are designated by W_d and W_f.

We notice that a dimensional reduction through a general distance matrix could also be achieved by parametrising $\tilde{U}$ as a general $m \times d$ matrix, being $\tilde{W}_f = \tilde{U}^{\mathrm{T}}\tilde{U}$ still a symmetric positive semidefinite matrix. With this parametrisation, the number of rows m is an estimate of the dimension q of the hidden space. However there are some drawbacks in using this parametrisation; i) since we need to optimise m searching for the better estimate of the the hidden dimension, the training procedure has to be repeated several times (unlike the parametrisation of W_f according to Equation 3.5, whose eigen-analysis is able to discover the hidden feature space); ii) if we do not impose any prior constraint on the elements of $\tilde{U}$ (i.e. the orthogonality of the column vectors), the number of free parameters turns out to be smaller than in in Equation 3.5 only for certain dimensions (i.e. $q \leq (d+1)/2$).

It is important to observe that the predictor obtained using W_f is not equivalent to an additive model (Hastie and Tibshirani, 1990), as the predictor is a multivariate function of $\mathbf{z}$ rather than being an additive function of the components of $\mathbf{z}$. However, it would be possible to produce an additive function in the GP context, using a covariance function which is the sum of one-dimensional covariance functions based on projections of $\mathbf{x}$.

3.2.1 Relative generalisation error

Consider predicting the value of a function $y(\mathbf{x})$ with a predictor $\hat{y}(\mathbf{x})$. A commonly-used measure of the generalisation error given a dataset $\mathcal{D}_n$ is the average squared error

$$E^g(\mathcal{D}_n) = \int \left(t(\mathbf{x}) - \hat{y}_{\mathcal{D}_n}(\mathbf{x})\right)^2 p(\mathbf{x})\, d\mathbf{x}, \tag{3.6}$$

where $\hat{y}_{\mathcal{D}_n}(\mathbf{x})$ is the GP prediction (cf. Equation 3.2) and $t(\mathbf{x})$ is the target generated by the underlying function corrupted by Gaussian noise. The average generalisation error $E^g(n)$ for a dataset of size n is obtained by averaging over the choice of training dataset, i.e. $E^g(n) = \mathcal{E}_{\mathcal{D}}\left[E^g(\mathcal{D}_n)\right]$. $E^g(\mathcal{D}_n)$ can sometimes be evaluated analytically or by numerical integration, but it is usually necessary to use samples to perform the average over training datasets $\mathcal{D}_n$.

In order to investigate the generalisation capabilities of GPs using a diagonal and full distance matrices W_d and W_f, we trained the GP predictors on some regression tasks. The generalisation errors are compared by looking at the relative error

$$\rho(\mathcal{D}_n) = \frac{E_d^g(\mathcal{D}_n) - E_f^g(\mathcal{D}_n)}{E_d^g(\mathcal{D}_n)}, \tag{3.7}$$

where $E_d^g(\mathcal{D}_n)$ and $E_f^g(\mathcal{D}_n)$ are the generalisation errors reported using a diagonal and a full distance matrix respectively. This ratio allows us to perform a fair comparison between the pairwise differences of the generalisation error $E_f^g(\mathcal{D}_n)$ and $E_d^g(\mathcal{D}_n)$ for each dataset. The expected value $\rho(n)$ is the average over the sampling of the training data $\mathcal{D}_n$: $\rho(n) = \mathcal{E}_{\mathcal{D}}[\rho(\mathcal{D}_n)]$.

3.3 Experimental results

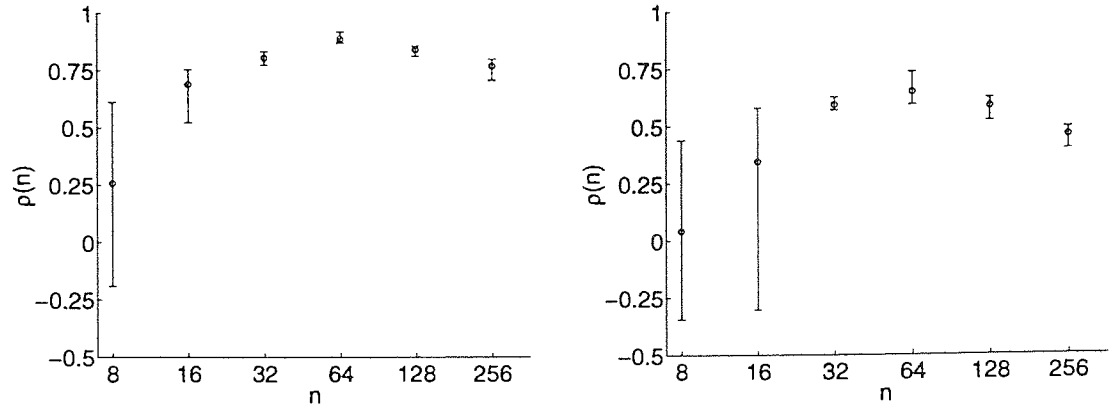
We have conducted experiments to compare the generalisation capabilities of a GP predictor with full and diagonal distance matrices. In this section we illustrate the results we obtained by training a GP on two regression tasks. The aim of the first experiment (Section 3.3.1) is to verify how the noise affect the relative generalisation error between GPs using the general and the diagonal distance matrices; in the second experiment (Section 3.3.2), we study the effects of the input-dimensionality of the manifested space on the relative generalisation error. Both the above experiments have been carried out performing a regression of a trigonometric function. Section 3.3.3 presents the results we obtained on the regression of a high-interaction surface, as proposed by Breiman (1993).

3.3.1 Regression of a trigonometric function: The effect of the noise level

In the first experiments, a GP has been trained on observations drawn from the function $y(z) = \sin(2\pi z)$ corrupted by Gaussian noise of mean zero and variance $\sigma_\nu^2 = 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1$. The hidden feature $z \in \mathbb{R}$ has been generated from the observable variables $\mathbf{x} \in \mathbb{R}^2$ through the transformation $z = \mathbf{m}^{\mathrm{T}}\mathbf{x}$, where $\mathbf{m}^{\mathrm{T}} = (1/\sqrt{2}, 1/\sqrt{2})$ and $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbb{I})$. We wish to infer the function $y(z)$ (where z is a one-dimensional feature) by using a GP on the manifest space $\mathbb{R}^2$. The expected generalisation errors of GPs using W_d and W_f distance matrices have been carried out by analytical evaluation of Equation 3.6 as shown in Appendix B; the expected relative error $\rho(n)$ has been estimated by averaging over 10 different samples of the training set.

In our experiments, the parameters of the covariance function (i.e. the prior variance σ_p^2, the variance modelling the noise σ_n^2 and the parameters of the distance matrices W_d and W_f) have been optimised by maximising the likelihood of the data (see Equation 3.4) with the conjugate gradient algorithm (Press et al., 1992) on each of the 10 training datasets.

Figure 3.1 reports the value of $\rho(n)$ on the vertical axis as a function of the amount of training data (x axis). The variance of the noise has been set to 0.01 in Figure 3.1(a) and 0.1 in Figure 3.1(b).



(a) $\sigma_\nu^2 = 0.01$ (b) $\sigma_\nu^2 = 0.1$

Figure 3.1: The Figures report on the y axis the graphs of $\rho(n)$ (see Equation 3.7) as a function of the amount of training data (x axis); the noise level is set to 0.01 (Figure 3.1(a)) and to 0.1 (Figure 3.1(b)). The error bars are generated by the minimum and the maximum value of $\rho(\mathcal{D}_n)$ which occurred over the 10 training datasets.

The graphs show that the use of W_f significantly improves the generalisation performance with respect to a diagonal matrix as the relative error $\rho(n)$ lies well above zero within its confidence interval. This is particularly highlighted in Figure 3.1(a) where for datasets larger than 32 data, $\rho(n)$ is larger than 75%. These results have also been confirmed by a t-test at the $p \geq 0.95$ level on the pairwise differences between the generalisation errors reported by W_d and W_f.

We notice that for small datasets, $\rho(n)$ is close to zero, as the distribution of its values are spread out around zero with wide confidence intervals. This is due to the fact that with small amounts of data it is not possible to train the GP properly; in particular, as the number of free parameters of W_f is larger than that of W_d, the former needs larger datasets for the training than the latter in order to avoid overfitting. A fully Bayesian treatment of the training of a GP (see Section 3.2) would not be so seriously affected by this problem since the prediction of the GP would be marginalised over the posterior distribution of the parameters. For large datasets, the relative error declines after having reached its maximum value; this agrees with the intuition that with large amounts of data, both methods will be become good predictors. Similar remarks apply also to Figure 3.1(b) (where $\sigma_\nu^2 = 0.1$) although we notice that the relative error $\rho(n)$ assumes lower values due to the higher noise variance.

The better performance of W_f with respect to W_d can be explained by an eigen-analysis of the two distance matrices; Figure 3.2 displays a pictorial representation of the eigenvalue analysis of the matrices W_f and W_d. In the Cartesian space (drawn by the orthonormal set $\{\mathbf{x}_1, \mathbf{x}_2\}$) three families of vectors are represented, the $\mathcal{X}$ to $\mathcal{Z}$ transformation $\mathbf{m}$ and the eigenvectors of W_d and W_f ($\{\mathbf{e}_1^d, \mathbf{e}_2^d\}$ and $\mathbf{e}_1^f$ respectively); the lengths of the eigenvectors are proportional to the correspondent eigenvalues.

As one eigenvalue of W_f is much larger than the other ($O(10)$ vs. $O(10^{-4})$), the full rank

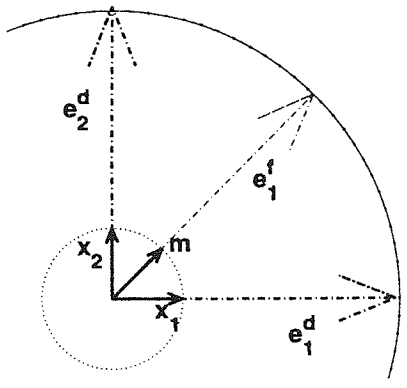


Figure 3.2: Eigenvalue-decomposition of W_f and W_d. The orthonormal basis $\{\mathbf{x}_1, \mathbf{x}_2\}$ represents the manifest space whereas $\mathbf{m}$ shows the direction of the hidden feature. $\{\mathbf{e}_1^d, \mathbf{e}_2^d\}$ are the two eigenvectors of W_d lying in the space of the observable variables; $\mathbf{e}_1^f$ is the eigenvector of W_f lying in the hidden-feature space. The lengths of the eigenvectors are proportional to the corresponding eigenvalues; since the second eigenvalue of W_f is negligible, $\mathbf{e}_2^f$ could not be represented in the picture.

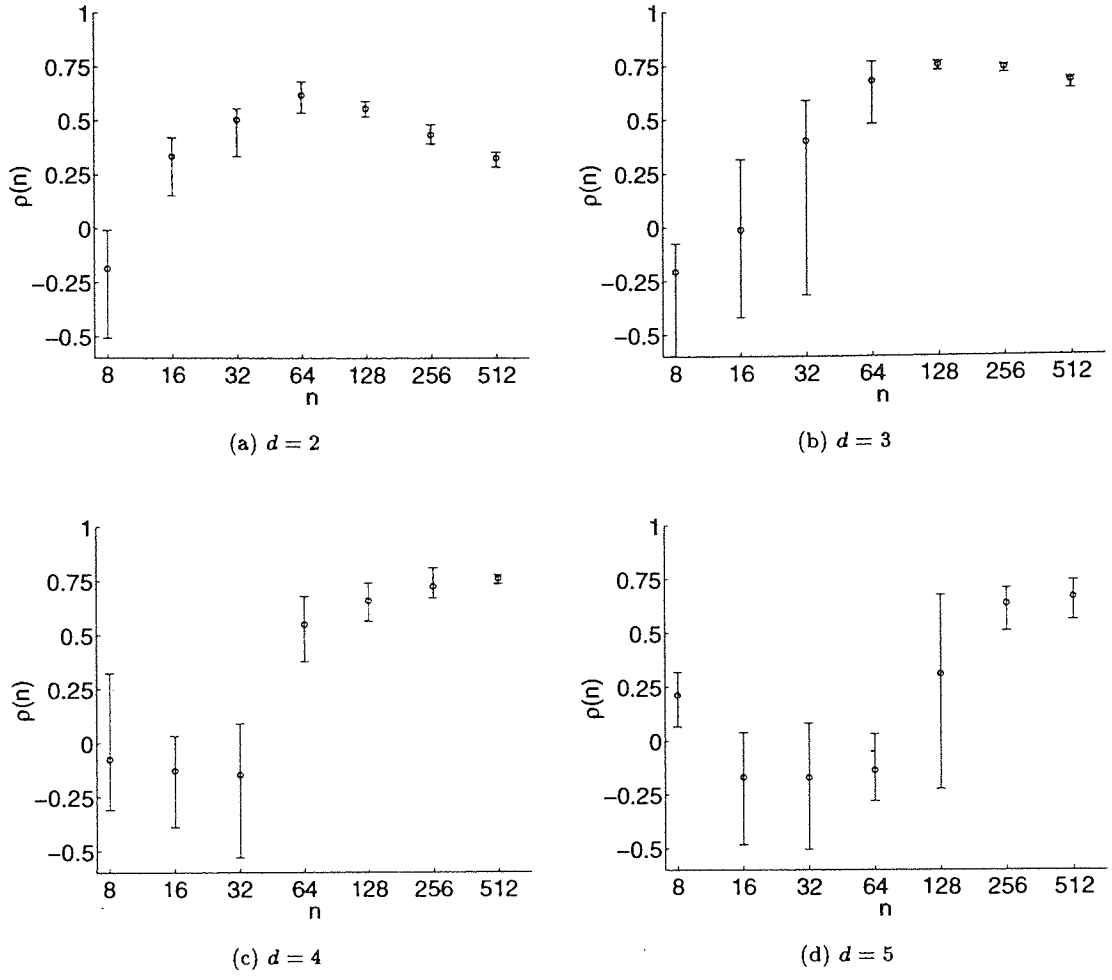
distance matrix is able to discover the relevant true dimension of the process. The figure shows clearly that W_f is able to to discover both the relevant true dimension of the process and the transformation from the observables to the hidden features. The eigenvector $\mathbf{e}_1^f$ lays over $\mathbf{m} = \left(1/\sqrt{2}, 1/\sqrt{2}\right)^{\mathrm{T}}$ and it represents the operator which maps the space of the observables onto the hidden feature space.

W_d fails to find out the effective dimension of the problem as it is characterised by two eigenvalues of similar magnitude ($O\left(10\right)$).

It is worth noting that increasing the noise variance degrades the generalisation performances obtained by W_f with regard to W_d but it does not affect the discovery of the hidden space; during our experiments the general matrix W_f successfully determines the relevant dimension of the process, irrespective of the value of σ_ν^2, provided that enough data are used in optimising the parameters of the GP.

3.3.2 Regression of a trigonometric function: The effect of the input-dimensionality

In order to investigate how the dimension of the manifest space $\mathcal{X}$ affects the generalisation performances of W_f, we also conducted experiments varying the dimensionality of the input space. We considered $\mathbf{x} \in \mathbb{R}^d$ and the latent variable $z = \mathbf{m}^{\mathrm{T}}\mathbf{x}$ (with $\mathbf{m}^{\mathrm{T}} = \mathbf{1}^{\mathrm{T}}/\sqrt{d}, \mathbf{1} \in \mathbb{R}^d$). Target values were generated from the function $y(z) = \sin\left(2\pi z\right) = \sin\left(2\pi\mathbf{m}^{\mathrm{T}}\mathbf{x}\right)$ corrupted by Gaussian noise normally distributed around zero with variance $\sigma_\nu^2 = 0.1$. In order to highlight the effect due to the dimensionality of the input space, we kept the variance of the noise fixed. Choosing $\mathbf{x} \sim \mathcal{N}\left(\mathbf{0}, \sigma_x^2\mathbb{I}\right)$, it is possible to carry out analytically the evaluation of Equation 3.6 and this is shown in Appendix B; the expected relative error $\rho\left(n\right)$ has been estimated by averaging over 10 different training sets. In our simulations we set $\sigma_x^2 = 1$.



(a) $d = 2$ (b) $d = 3$

(c) $d = 4$ (d) $d = 5$

Figure 3.3: The Figures show the graphs of $\rho(n)$ (see Equation 3.7) as a function of the amount of training data obtained for several dimensions of the input space, i.e. $d = 2$ (Figure 3.3(a)), $d = 3$ (Figure 3.3(b)), $d = 4$ (Figure 3.3(c)) and $d = 5$ (Figure 3.3(d)). The error bars are generated by the minimum and the maximum value of $\rho(\mathcal{D}_n)$ which occurred over the 10 training datasets. The minimum value reached in Figure 3.3(b) for $n = 8$ is $\rho(\mathcal{D}_8) = -0.7$.

The maximisation of the likelihood (Equation 3.4) has been carried out by optimising the prior variance σ_p^2, the variance of the noise σ_n^2 and the parameters of the distance matrices W_d and W_f with the conjugate gradient algorithm (Press et al., 1992); the training has been carried out on each of the 10 datasets.

Figure 3.3 shows some results we obtained by using four different dimensions of the input vector, from $d = 2$ (Figure 3.3(a)) to $d = 5$ (Figure 3.3(d)). These suggest that the dimension of the space can have a significant effect on the generalisation performance obtained by using W_f and W_d for given sizes of dataset. However the experiments confirmed that, when there are enough data to train the GP properly, the W_f distance matrix is able to improve generalisation performances of the model, discovering the hidden space; this remark has also been confirmed by a t-test at the $p \geq 0.95$ level on the pairwise differences between the generalisation errors reported by W_d and W_f.

The eigen-analysis of W_f showed that only one out of d dimensions is relevant to reconstruct the underlying function. Furthermore the eigenvector corresponding to the largest eigenvalue draws the direction of the underlying hidden space, i.e. $\mathbf{m}^{\mathrm{T}} = \mathbf{1}^{\mathrm{T}}/\sqrt{d}, \mathbf{1} \in \mathbb{R}^d$.

We note in Figures 3.3(a) and 3.3(b) that the values of $\rho(n)$, after having reached a maximum, tends to decrease; this reveals that the generalisation errors obtained by GPs with a diagonal and a full distance matrices become closer. This is due to the fact that the GP using the general distance matrix plateaus to a value very close to the actual noise variance σ_ν^2 when the datasets are large enough (i.e. $n = 64$ for $d = 2$ and $n = 128$ for $d = 3$). This does not hold for GP using W_d whose generalisation error gradually decreases, approaching the noise level and becoming closer to the generalisation error obtained with W_f for large data sizes.

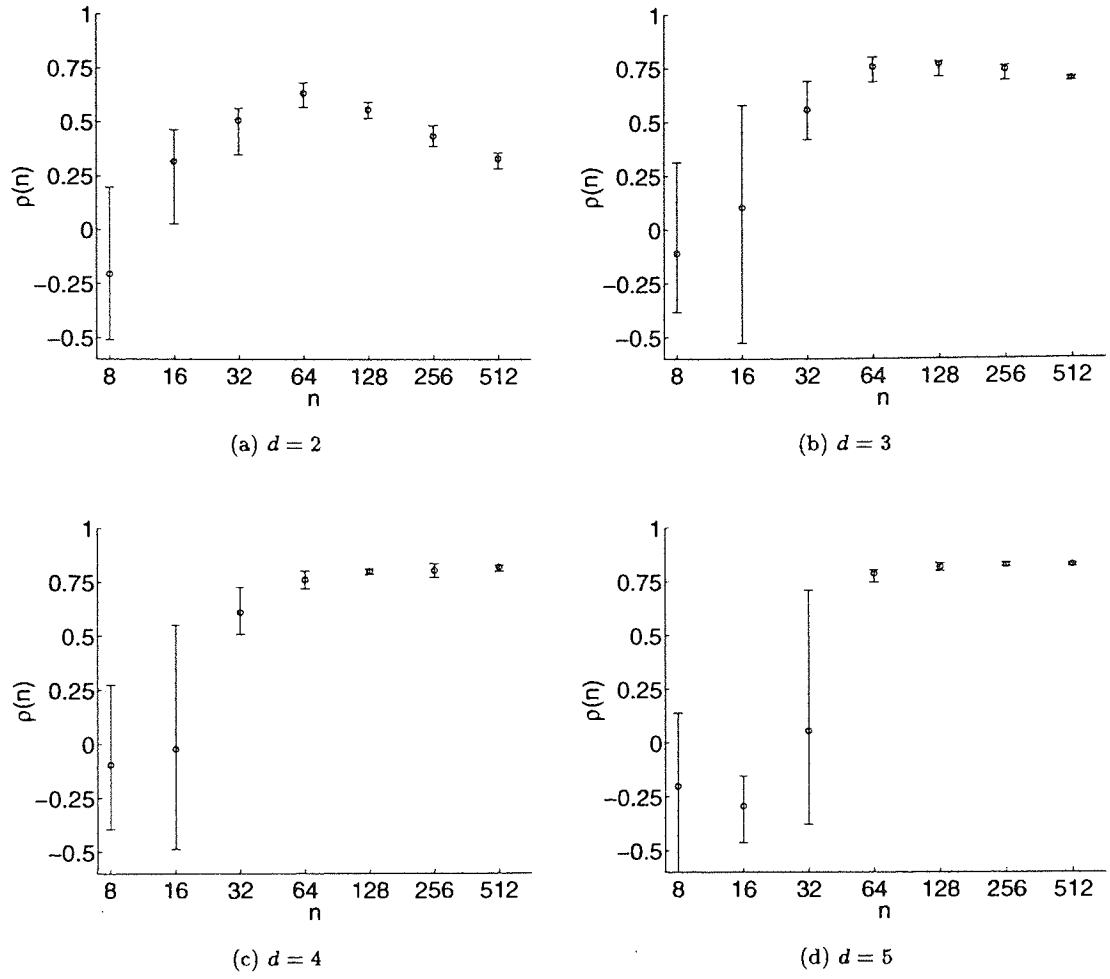
The generalisation performances we obtain vary with respect to the dimensionality of the problem and the amount of data available. The larger the dimension of the input space the larger the amount of datapoints needed to optimise suitably the parameters of the GP; this is shown by the lower average value of $\rho(n)$ obtained for the smallest datasets (i.e. $n = 8, 16, 32$) when the input-dimensionality increases (e.g. $d = 4, 5$); this effect is weakened for $d = 3$, although the variance around the average value of $\rho(n)$ is large. This is due to the fact that the larger the dimension of the input space the larger the amount of data required to train the parameters of W_f properly in order to avoid overfitting and increasing the generalisation error; this effect can be reduced by a full Bayesian approach to the learning. However we note that the best value of $\rho(n)$ does not depend on the input-dimensionality, being always around 0.75.

The results obtained by varying the dimension of the input space have been verified by introducing a different parametrisation of the distance matrix W. As mentioned above (see Section 3.2), the distance matrix W can also be parametrised by using a vector of parameters $\tilde{\mathbf{u}} \in \mathbb{R}^d$, such as $\tilde{W}_f = \tilde{\mathbf{u}}\tilde{\mathbf{u}}^{\mathrm{T}}$. In this case $\tilde{W}_f$ is a rank 1 distance matrix whose number of free parameters (d) is lower than that of W_f (being $d(d+1)/2$). Since $\tilde{W}_f$ contains already knowledge of the hidden dimensionality of the regression problem, the generalisation error obtained by a GP using $\tilde{W}_f$ should behave as a sort of lower bound on the generalisation error of a GP using W_f.

Running experiments with settings similar to those presented above (with $\sigma_\nu^2 = 0.1$), we estimated the relative error $\rho(n)$ of GPs using the W_d and $\tilde{W}_f$ distance matrices and varying the input-dimensionality of the regression problem. In our experiments GPs using a $\tilde{W}_f$ distance matrix have revealed to be prone to optimisation problems, especially when large datasets are used for $d = 3, 4$ and 5; in order to allow the convergence of the algorithm to acceptable solutions, we randomly initialised the elements of the vector $\tilde{\mathbf{u}}$ according to a Gaussian distribution centred around *reasonable*[1] values of the parameters.

From the comparison of the results obtained (shown in Figure 3.4) with those displayed in

[1] During our experiments we noticed that the problem of initialisation is crucial for GPs using $\tilde{W}_f$. This problem can be overcome by an appropriate choice of the centre of the Gaussian distribution from which initial values of $\tilde{\mathbf{u}}$ are drawn. A *reasonable* mean μ of the Gaussian has been determined by considering the value $\tilde{u}_{opt}$ of the parameter $\tilde{\mathbf{u}}$ maximising the likelihood of a one-dimensional GP and translating it in higher dimensions (i.e. $\mu = \mathbf{1}\tilde{u}_{opt}/\sqrt{d}, \mathbf{1} \in \mathbb{R}^d$). It is worth noticing that this kind of problem does not arise in GP using W_f.



(a) $d = 2$ (b) $d = 3$

(c) $d = 4$ (d) $d = 5$

Figure 3.4: The Figures show the graphs of $\rho(n)$ (see Equation 3.7) as a function of the amount of training data obtained for several dimensions of the input space, i.e. $d = 2$ (Figure 3.4(a)), $d = 3$ (Figure 3.4(b)), $d = 4$ (Figure 3.4(c)) and $d = 5$ (Figure 3.4(d)); the distance matrix used by the GP is $\tilde{W}_f = \tilde{\mathbf{u}}\tilde{\mathbf{u}}^{\mathrm{T}}$, $\tilde{\mathbf{u}} \in \mathbb{R}^d$ (see the text). The error bars are generated by the minimum and the maximum value of $\rho(\mathcal{D}_n)$ which occurred over the 10 training datasets. The minimum value reached in Figure 3.4(d) for $n = 8$ is $\rho(\mathcal{D}_8) = -0.8$.

Figure 3.3 we can notice that there are some similarities in the values attained by the relative errors; in particular the graph obtained when $d = 2$ (see Figure 3.4(a)) is not significantly different from Figure 3.3(a).

For dimensions $d > 2$, the large confidence interval of $\rho(n)$ typical of small datasets shows that GP using $\tilde{W}_f$ is prone to overfitting. From Figure 3.4 we also note that the confidence interval of variability reduces significantly for medium sizes of the datasets (i.e. $n = 32$ for $d = 4$ and $d = 3$, $n = 64$ for $d = 5$) with respect to that displayed in Figure 3.3.

Performances obtained by GPs using $\tilde{W}_f$ and W_f become closer for large datasets (when the relative errors lie around 0.75), although the interval of variability of $\rho(n)$ is smaller for GP using the rank 1 distance matrix; this suggests that the use of a full distance matrix obtains generalisation errors similar to those obtained by the rank 1 distance matrix.

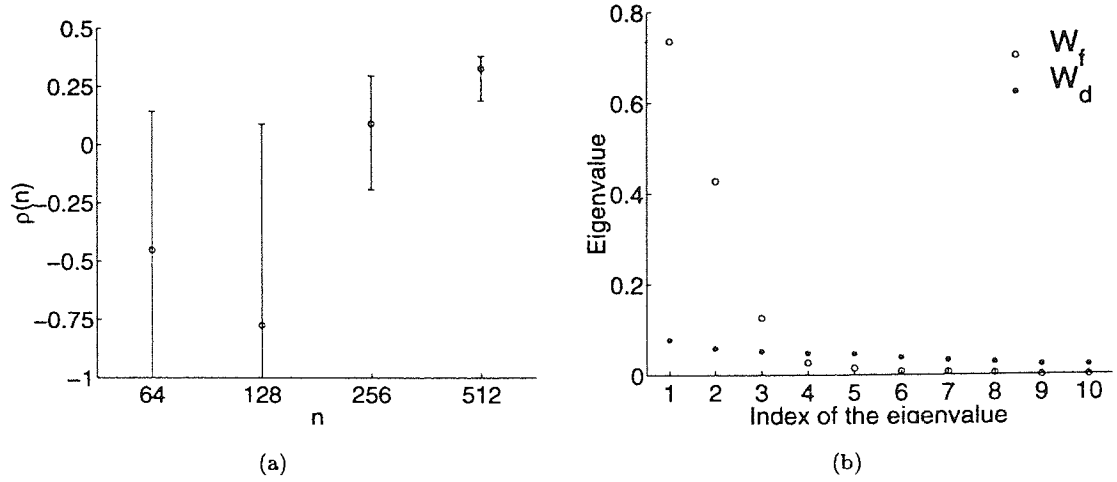


Figure 3.5: Figure 3.5(a) reports on the y axis the graph of $\rho(n)$ (see Equation 3.7) as a function of the amount of training data (x axis); the error bars are generated by the minimum and the maximum value of $\rho(\mathcal{D}_n)$ that occurred over the 10 training datasets. Figure 3.5(b) shows the graph of the ten eigenvalues of the W_d ($*$) and the W_f ($\circ$) distance matrices obtained using one training set of 512 data. The lower values reached by training sets with 64 and 128 data are $\rho(\mathcal{D}_{64}) = -1.06$ and $\rho(\mathcal{D}_{128}) = -1.97$.

3.3.3 A high-interaction surface

We also tested our method on an example taken from Breiman (1993) which is concerned with a regression problem of a surface in a high dimensional space. The target function is $y(\mathbf{x}) = \sigma(z_1) + \sigma(z_2) + \sigma(z_3)$, where $\sigma(z)$ is the sigmoid function

$$\sigma(z) = \frac{\exp[z]}{(1 + \exp[z])}.$$

The hidden features z_1, z_2 and z_3 are derived from the transformation $z_i = 2(l_i - 2)$, $i = 1 \ldots 3$, where the l_i are the normalised inner products $\mathbf{m}_i^{\mathrm{T}}\mathbf{x}$. The observed variables $\mathbf{x} \in \mathbb{R}^{10}$ are uniformly distributed over $[0,1]^{10}$; the three vectors $\mathbf{m}_i$ are $\mathbf{m}_1 = (10,9,3,7,-6,-5,-9,-3,-2,-1)^{\mathrm{T}}$, $\mathbf{m}_2 = (-1,-2,-3,-4,-5,-6,7,8,9,10)^{\mathrm{T}}$ and $\mathbf{m}_3 = (-1,-2,-3,4,5,4,-3,-2,-1,0)^{\mathrm{T}}$. The values of the true function are also corrupted by Gaussian noise of mean zero and variance σ_ν^2. Following Breiman (1993) the noise level was such that the ratio between the standard deviations of the signal $y(\mathbf{x})$ and the the noise was 4.0, i.e.

$$\sqrt{\mathcal{E}\left[(y(\mathbf{x}))^2\right] - (\mathcal{E}[y(\mathbf{x})])^2} = 4.0 \cdot \sigma_\nu.$$

It turns out that $\sigma_\nu^2 = 2 \cdot 10^{-3}$.

We have run experiments, training GPs with diagonal and full distance matrices on 10 data sets of size 64, 128, 256 and 512; in his work, Breiman used training sets with 400 datapoints. The parameters have been optimised by a conjugate gradient algorithm. The generalisation errors of W_d and W_f have been estimated using 1024 test data points; the relative generalisation error $\rho(n)$ (cf. Equation 3.7) is shown in Figure 3.5(a). We observe that for datasets of size 512 the use of W_f

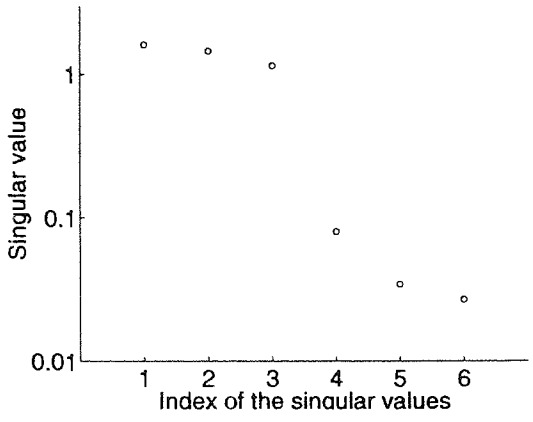


Figure 3.6: The Figure shows the six singular values of the matrix M as defined in Equation 3.8. As three out of six singular values are significantly smaller than the others, the space spanned by the first three columns of M well approximates the hidden space generated by the others.

significantly reduces the relative error with respect to the diagonal matrix. Models trained with smaller training sets do not have such good generalisation performance because the large number of parameters in W_f (57) tends to overfit the data.

An eigenvalue decomposition of the distance matrices shows that W_f is able to discover the underlying structure of the process. Figure 3.5(b) displays the eigenvalues of W_f and W_d optimised from one of the training sets with 512 data. W_f is characterised by three large eigenvalues, whose eigenvectors indicate the three main directions in the feature space; thus the full matrix is able to find out three out of ten directions which are responsible of the variation of the function. Conversely, W_d fails to discover the hidden features in the data; as all the eigenvalues have almost the same magnitude, all the input dimensions of the observed variable are equally relevant in training the GP.

The eigenvectors $\mathbf{e}_i^f, i = 1,2,3$ of W_f define a basis in the space generating a subspace of features. In order to verify that the subspace spanned by the $\mathbf{e}_i^f$ actually overlaps the hidden feature space, we tried to express the former set of vectors as a linear combination the latter. Thus we computed the Singular Value Decomposition (Press et al., 1992) of the matrix

$$M = \left[\mathbf{e}_1^f, \mathbf{e}_2^f, \mathbf{e}_3^f, \mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3\right], \tag{3.8}$$

where the $\mathbf{n}_i = \mathbf{m}_i / \sqrt{\mathbf{m}_i^T \mathbf{m}_i}, i = 1 \ldots 3$ are the normalised vectors generating the hidden space. The six singular values are displayed in Figure 3.6.

We note that three out of six singular values are smaller than the others, being $O\left(10^{-1}\right) - O\left(10^{-2}\right)$ compared to $O(1)$. This shows that the three vectors $\mathbf{n}_i$ generating the hidden space can be well approximated as a linear combination of the new basis of eigenvectors and thus the eigenspace of W_f is a good approximation of the hidden feature space.

3.4 Discussion

In this Chapter we have shown how to discover hidden features with GP regression. We also note that this technique could be applied to problems where Gaussian process predictors are used in classification problems. An attractive feature of the method is that it allows the appropriate dimensionality of the $\mathbf{z}$ space to be discovered; this may help in obtaining a better understanding of the generalisation capability of the model (Opper and Vivarelli, 1999). If we wish to restrict the maximum dimensionality of $\mathcal{Z}$ to be q then one could use a distance matrix of rank-q, i.e. $W = (\Psi^{\frac{1}{2}} M)^{\mathrm{T}} (\Psi^{\frac{1}{2}} M)$.

Conventionally GP regression has been realised using a diagonal distance matrix W_d for the estimation of the covariance between two points. The discovery of hidden features can be achieved by using a general positive semidefinite matrix W_f as distance matrix; the eigen-analysis of W_f determines the matrix of the general transformation from the input to the hidden space whose columns are composed by the eigenvectors of W_f. The idea of allowing a general transformation of the input space has been mentioned before in the literature, for example in Girosi et al. (1995). However, Girosi et al. suggest setting the parameters in W_f by cross-validation; we believe that this is not very practical in high-dimensional spaces.

Our investigation has highlighted the superiority of the general distance matrix W_f with respect to W_d; the effect was particularly evident when larger amounts of data were used because of the maximisation of the likelihood leads to overfitting if enough data are not used in training the model. This problem can be reduced by a full Bayesian approach which allows to perform GP regression marginalising GP predictions over the posterior distribution.

We compared the generalisation performances of GPs using W_d and W_f on two regression problems; the results obtained show that the use of a full distance matrix can reduce significantly the relative error with respect to the use of a diagonal distance matrix. The experiments have also shown that W_f can discover the linear transformation from the input to the hidden space since the eigenspace of W_f effectively overlaps the actual hidden space.

During the first experiment, two-dimensional GPs have been trained on the regression of a trigonometric function and varying the variance of the noise. Although higher values of the noise worsen the generalisation performances of W_f with respect to W_d, the general distance matrix is able to discover the one-dimensional hidden space. However the generalisation performances depend upon the noise level, becoming better for lower noise level.

The results obtained varying the input-dimensionality of the manifest space (for fixed dimensionality of the feature space $\mathcal{Z}$) have revealed that W_f is able to discover the hidden space, although it is more prone to overfit the data due to a greater number of parameters than W_d. However generalisation performances of GP with a general distance matrix are still better than those obtained with a diagonal one since the expected relative generalisation error is around 0.75. We also compared the relative generalisation errors of GPs using W_f and a rank 1 distance matrix

$\tilde{W}$, showing that the former attains relative generalisation errors $\rho(n)$ similar to those obtained by the latter, when large datasets are used.

Better generalisation performances of W_f with respect to W_d have been also observed for GP regression of a high-interaction surface, confirming the superiority of the general distance matrix in discovering the hidden feature space. This experiment has also confirmed that the eigenspace span by the general distance matrix W_f is a good approximation of the original hidden feature space, showing that W_f is able to discover a three-dimensional data structure hidden within a ten-dimensional manifest space.

Our experiments have revealed that the use of the general distance matrix W_f improves the generalisation capabilities of GPs and enables to discover hidden features estimating the transformation from the manifested to the hidden spaces; one limitation is due to the larger number of parameters to be optimised with the danger of overfitting the data. The effect may be reduced by a full Bayesian optimisation of the distance matrix, allowing the application of the general distance matrix W_f to real-world problem.

Chapter 4

Using Bayesian neural networks for classifying segmented images

In this Chapter we investigate the Bayesian training of neural networks for region labelling of segmented outdoor scenes; the database is drawn from the Sowerby Image Database of British Aerospace. We present results that compare the performance of neural networks trained with two Bayesian methods, (i) the evidence framework of MacKay (1992) and (ii) a Markov Chain Monte Carlo method due to Neal (1996). The performance of the two methods are compared evaluating the empirical learning curves of neural networks trained with the two Bayesian algorithms. The sensitivity of the two methods to the choice of the starting point in the weight space and to the amount of training data has been investigated. We also present the use of the Automatic Relevance Determination method for input feature selection.

4.1 Introduction

Computer Vision is a challenging area aimed at implementing in computer-based systems some of the processes used for visual perception; the problem is complicated because it is concerned with processes which human vision carries out automatically, without explicit knowledge of how they actually work.

This Chapter deals with the specific problem of investigating the use of neural networks for the classification of regions of outdoor scenes.

Scene interpretation is usually split in a two stage procedure: the segmentation and the interpretation of images. The segmentation process divides an image in a set of regions, where ideally each region corresponds to one object; the interpretation consists in the labelling of the regions.

To carry out the task successfully it is important to classify each region not only using its own attributes (e.g. colour, shape, texture) but also taking into account the *context*. The context can be handled in two ways, either by searching for a consistent interpretation of the whole scene, or by classifying each single region according to its local context.

An example of the interpretation of the whole scene is based on techniques of Artificial Intelligence (Ohta, 1985; Draper et al., 1989; McKeown et al., 1985). In this approach, images are described with some attributes extracted from regions which are used by a rule-based system and classified on the basis of contextual relationships.

An example of classification of each individual region on the basis of local context makes use of statistical methods (Wright, 1989; Wright et al., 1995; Mackeown, 1994; Clark, 1995). Segmented regions are classified individually by using statistical methods rather than a set of rules. Neural networks have been applied to this task and contextual information is used in the classification of each region.

The methods discussed above deal with generic scenes. However if specific information is available (e.g. airplane types), scene interpretation can use this knowledge by matching geometric models to image features. Similarly to the rule-based system, the matching of models can be performed by using logic rules. An example of vision system based on this approach was ACRONYM (Brooks, 1983), which was designed for the recognition of airplanes from aerial images.

An approach to image interpretation based on matching of geometric model does not seem applicable to the Sowerby Image Database because of the great variety of the objects represented. Even though a model based approach may be suitable for recognition of man-made objects, it would be difficult to apply it to the classification of regions with complex shape (such as trees, vegetation). Rule based systems may have problems when dealing with real valued attributes as their decisions are based upon binary logic.

The present study follows the same approach as Wright (1989), but compares and contrasts two implementations of Bayesian learning of neural networks: the evidence framework approximation (EF) and the Markov Chain Monte Carlo method (MCMC). It is important to carry out such a comparison if a neural network has to be applied to real-world tasks, in order to understand which algorithm can guarantee better generalisation performance.

The structure of the Chapter is as follows. The next Section introduces the image database used for the training of the neural networks. The images have been pre-processed in order to derive a set of features describing numerically the regions of the dataset. Two kind of features have been computed; the internal features (representing the internal properties of each region) and the contextual features (describing the relationship between a region and its surround). A linear normalising transformation has been applied to the features.

The two implementations of Bayesian learning of neural networks are introduced in Section 4.3 and applied to the training of a Multiple Logistic Regression network (MLR) and a Multi Layer Perceptron (MLP).

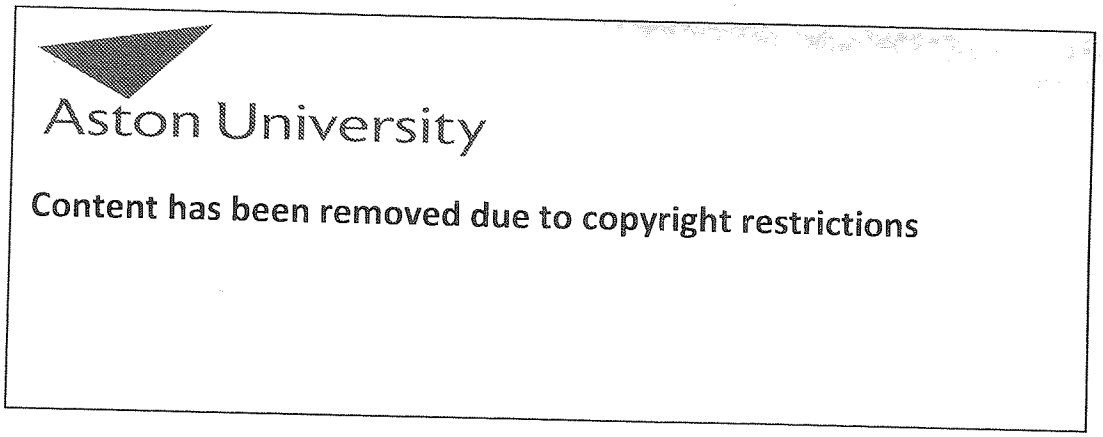


Figure 4.1: The Figure shows the colour image of a typical scene drawn from the database.

In Section 4.4 we compare and contrast the two Bayesian methods with respect to the use of the Automatic Relevance Determination technique (ARD) of MacKay and Neal (Neal, 1996) for feature selection and to the learning curves of neural networks trained on datasets of various sizes.

The Chapter ends with a discussion of the study undertaken.

4.2 The Database

The database consists of 96 coloured images extracted from the Sowerby Image Database of British Aerospace. The main subjects of the images are rural and urban scenes; the database was constructed so that objects with diverse characteristics appear in it (e.g. cars with different colours, different shapes of buildings, etc.). All the scenes have been photographed using small-grain 35mm transparency film. In order to ensure clear pictures, all the images have been shot with good atmospheric visibility, in dry and fully overcast conditions. Each image of the database has been digitised with a calibrated scanner, generating a high quality 24-bit colour representation of size 768 by 512 pixels.

The database contains three different representations of each image, the digitised image, the segmented image and the labelled image.

The digitised image is the pixelated representation of a scene, made up by three matrices whose elements are the levels of red, green and blue composing the colour of the pixel of an image. There are 256 intensity levels in the interval $[0, 255]$. In the following the intensity levels of the pixel (i, j) are indicated as R_{ij}, G_{ij} and B_{ij}. An example of the digitised representation of an image is shown in Figure 4.1.

Figure 4.2: The Figure shows the segmented representation of the picture in Figure 4.1.

Each digitised image has been segmented. The segmentation process consists in splitting up an image into a number of regions, each one corresponding ideally to an object (see Figure 4.2). None of the techniques available execute this task reliably, generating scenes which are often either over-segmented (when one object is split up in several regions) or under-segmented (when one region is composed by more than one object). The segmentation algorithm used in the Sowerby's database is based on the region growing method of Gay (1989) and it is described in Mackeown (1994).

Each region of the database has been hand labelled in one of eleven classes as `clouds`, `vegetation`, `road marking`, `road surface`, `road border`, `building`, `bounding object`, `road sign`, `telegraph pole`, `illumination shadow` and `mobile object`. The categories have been selected using prior knowledge about the subject of the images, and thus they provide a complete description of all the objects occurring in the database. An example of the hand labelled format of Figure 4.1 is shown in Figure 4.3.

The 96 images of the database have been divided randomly in two independent sets: a training set and a testing set. Each region of the two sets has been hand labelled in one of the eleven categories. Table 4.1 shows a list of the eleven categories and the composition of the two sets.

A region $\mathcal{R}$ is described by a set of features grouped in a vector $\mathbf{x} = (x_1, x_2, \ldots x_d)$ whose elements are the numerical description of characteristics of the given region. All of the features have been computed from the digitised and the segmented representation of the images; they have been rescaled using a linear normalising transformation, obtaining rescaled features which have mean 0 and variance 1.

The task on which neural networks have been trained is the classification of feature vectors

Figure 4.3: The Figure shows the hand labelled representation of Figure 4.2. The eleven classes (and the colours labelling them) are `clouds` (light blue), `vegetation` (green), `road marking` (bright yellow), `road surface` (grey), `road border` (dark yellow), `building` (brick red), `bounding object` (brown), `road sign` (bright red), `telegraph pole` (white), `illumination shadow` (grey blue) and `mobile object` (pink). Black regions label unclassified objects.

which encode the characteristics of each region of the database; obviously, the set of features chosen plays an important rôle in the classification task. Following previous work of Mackeown (1994), Wright et al. (1995) and Clark (1995), the vector $\mathbf{x}$ is composed by 35 features divided in two groups, the internal and the contextual features. The features are described in the following Sections and listed in Table 4.2.

4.2.1 The internal features

The internal features describe colour, topology, size, position, shape and texture of a given region.

There are three colour features based on the mean intensity and the hue of a region. Defining as $I_{ij} = (R_{ij} + G_{ij} + B_{ij})/3$ the intensity of a pixel whose coordinates in the image are (i, j), the mean intensity of the region $\mathcal{R}$ is computed averaging the intensity of all the pixels belonging to the region as

$$x_1 = \frac{1}{\mathcal{A}_\mathcal{R}} \sum_{(i,j)\in\mathcal{R}} I_{ij},$$

where $\mathcal{A}_\mathcal{R}$ is the area of $\mathcal{R}$, namely the number of pixels in the region $\mathcal{R}$.

An estimate of the hue is given by (Ballard and Brown, 1982)

$$\varphi_\mathcal{R} = \arccos \frac{(R-G)+(R-B)}{2\sqrt{(R-G)^2+(R-B)(G-B)}},$$

where R, G and B are the mean values of red, blue and green obtained by averaging the colour level of all the pixels in the region $\mathcal{R}$. The components x_2 and x_3 of the feature vector are defined

		Training set		Test set	
	Number of regions	5832		1505	
	Class	Region (%)	Area (%)	Region (%)	Area (%)
1	Clouds	7.90	8.18	6.38	11.03
2	Vegetation	27.35	35.37	26.25	33.19
3	Road Marking	1.68	0.20	1.79	0.17
4	Road Surface	15.29	39.30	13.02	40.40
5	Road Border	9.88	9.32	8.04	5.57
6	Building	20.82	4.20	24.98	5.19
7	Bounding object	7.13	2.08	6.45	2.17
8	Road sign	0.77	0.05	0.93	0.15
9	Telegraph Pole	2.45	0.26	2.79	0.23
10	Illumination Shadow	2.52	0.40	2.13	0.34
11	Mobile Object	4.20	0.65	7.24	1.56

Table 4.1: The Table shows the labels used and the composition of the training and testing sets. For both data sets the first column gives the percentage of each class computed from the number of regions of the class in the data set against the total number of regions; the second column reports the percentage of each class computed from the total area of that class against the total area of all the regions in the data set.

as

$$x_2 = \cos\varphi_{\mathcal{R}} \text{ and } x_3 = \sin\varphi_{\mathcal{R}}.$$

The topological characteristics are the hollowness, the Euler number and the compactness of a region.

The hollowness is defined as the ratio between the total area occupied by holes inside a region ($\mathcal{A}_h$), and the area $\mathcal{A}_{\mathcal{R}}$ of the region itself:

$$x_4 = \frac{\mathcal{A}_h}{\mathcal{A}_{\mathcal{R}}}.$$

The computation of the sizes of the holes of a region is a difficult task because of the presence of holes within holes which can not be counted as holes of $\mathcal{R}$, but they must be taken into account computing $\mathcal{A}_h$; the routine we implemented takes into account this effect by using recursive functions.

The Euler number of the region $\mathcal{R}$ is defined as the difference between the number of connected components of the region and the number of holes lying inside the region. Since all the regions contained in the database are single component, the Euler number of the region $\mathcal{R}$ is $\epsilon_{\mathcal{R}} = 1 - h_{\mathcal{R}}$, where $h_{\mathcal{R}}$ is the number of holes of the region. In order to discriminate small values of $\epsilon_{\mathcal{R}}$ (typical of compact regions), this topological feature has been chosen as

$$x_5 = \log(2 - \epsilon_{\mathcal{R}}) = \log(1 + h_{\mathcal{R}}).$$

The compactness of a region is defined as

$$x_6 = \frac{4\pi\mathcal{A}_{\mathcal{R}}}{p_{\mathcal{R}}^2},$$

where $p_{\mathcal{R}}$ is the perimeter of $\mathcal{R}$.

Three components of the feature vector depend upon the size of the region and the Cartesian coordinates of its centroids:

$$x_7 = \mathcal{A}_{\mathcal{R}},\ x_8 = i_{c_{\mathcal{R}}} \text{ and } x_9 = j_{c_{\mathcal{R}}},$$

where the coordinates of the centroids are

$$i_{c_{\mathcal{R}}} = \frac{1}{\mathcal{A}_{\mathcal{R}}} \sum_{i \in \mathcal{R}} i,\ j_{c_{\mathcal{R}}} = \frac{1}{\mathcal{A}_{\mathcal{R}}} \sum_{j \in \mathcal{R}} j.$$

We implemented seven shape descriptors which are invariant with respect to linear transformations such as translation, rotation and scale change (Hu, 1962). Defining the normalised central moments of order $(p+q)$ (Gonzales and Woods, 1992) as

$$\eta_{pq} = \frac{1}{\mathcal{A}_{\mathcal{R}}^{\gamma}} \sum_{(i,j) \in \mathcal{R}} (i - i_{c_{\mathcal{R}}})^p (j - j_{c_{\mathcal{R}}})^q$$

(where $\gamma = (p+q)/2$), the seven invariant moments are defined from the normalised central moments of second and third order as

$$\begin{aligned}
x_{10} &= \eta_{20} + \eta_{02} \\
x_{11} &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\
x_{12} &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
x_{13} &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
x_{14} &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12}) \left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] + \\
&\quad (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03}) \left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right] \\
x_{15} &= (\eta_{20} - \eta_{02}) \left[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right] + \\
&\quad 4\eta_{11}(\eta_{30} - \eta_{12})(\eta_{21} + \eta_{03}) \\
x_{16} &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12}) \left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] + \\
&\quad (3\eta_{12} - \eta_{30})(\eta_{21} + \eta_{03}) \left[(3\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right].
\end{aligned}$$

The last internal features describe the texture of a region. The texture is an intrinsic property of all the objects and it is defined from the statistical distribution of the intensity levels of a region; following Wright et al. (1995) we described the texture of the regions by using the eigenvalues of the mean co-occurrence matrix (Haralick et al., 1973).

Given a region $\mathcal{R}$, the element c_{ij} of the co-occurrence matrix $C(\theta)$ is given by the total number of pairs of nearest neighbour pixels whose grey levels are i and j respectively. Since a pixel has eight nearest neighbours, each region has eight co-occurrence matrices $C(\theta)$, labelled with the angle θ. They were reduced to four mean co-occurrence matrices by the linear combination

$$\overline{C}(\theta) = \frac{C(\theta) + C(\theta + \pi)}{2},$$

with $\theta = 0, \pi/4, \pi/2, 3\pi/4$. For each $\overline{C}(\theta)$ the two largest eigenvalues $\lambda_1(\theta), \lambda_2(\theta)$ have been computed, where $\lambda_1(\theta) > \lambda_2(\theta)$. The first one measures the total amount of smooth variation of

Component	Internal features
1	Mean intensity
2-3	Hue
4-6	Topological features
7	Size
8-9	Position of the centroid
10-16	Shape descriptors
17-19	Texture descriptors
	Contextual features
20-23	Intensity ratios
24-27	Size ratios
28-35	Relative position

Table 4.2: List of the features used for the numerical description of the regions. The table is split in two parts: the first one shows the internal features, the other the contextual features.

intensity, whereas the second is related to the *amount* of texture of the image (Ballard and Brown, 1982).

The three features computed (as used in Wright et al. (1995)) are

$$
\begin{aligned}
x_{17} &= \frac{1}{4}\sum_{\theta}\frac{\lambda_2(\theta)}{\lambda_1(\theta)} \\
x_{18} &= \sum_{\theta}\left(\lambda_1(\theta)+\lambda_2(\theta)\right), \\
x_{19} &= \begin{cases} 0 & \text{if } \arg_\theta \max\left(\lambda_1(\theta)+\lambda_2(\theta)\right) = 0, \pi/2 \\ 0.5 & \text{if } \arg_\theta \max\left(\lambda_1(\theta)+\lambda_2(\theta)\right) = \pi/4 \\ 1 & \text{if } \arg_\theta \max\left(\lambda_1(\theta)+\lambda_2(\theta)\right) = 3\pi/4. \end{cases}
\end{aligned}
$$

4.2.2 The contextual features

The contextual features describe the relationships between a region $\mathcal{R}$ and its context. Following Wright et al. (1995), the contextual features are defined as the relative size ($x_{20}, \ldots x_{23}$), intensity ($x_{24}, \ldots x_{27}$) and position ($x_{28}, \ldots x_{35}$) of a region with respect to the four largest regions surrounding it. The relative positions are described by the cosine and the sine of the vector connecting the centroid of $\mathcal{R}$ with the centroids of the neighbouring regions. Because of the fixed length of the contextual features, if the number of surrounding regions is less than four there are some features which are not defined; in this case the undefined contextual features are set to the average of the defined components.

4.3 Bayesian training of neural networks

A feed-forward neural network (Hertz et al., 1991) consists of processing units (also called *nodes*) allocated in layers; each node of one layer is connected with all those of the previous layer and is

Figure 4.4: The Figure shows the test image of Figure 4.1 labelled by one of the neural networks used in our experiments. The eleven classes (and the colours labelling them) are `clouds` (light blue), `vegetation` (green), `road marking` (bright yellow), `road surface` (grey), `road border` (dark yellow), `building` (brick red), `bounding object` (brown), `road sign` (bright red), `telegraph pole` (white), `illumination shadow` (grey blue) and `mobile object` (pink). Black regions label unclassified objects.

characterised by a numerical value called *activation*.

For the classification of segmented images, neural networks with one and two layers of adaptable weights have been used; the first neural network model is called Multiple Logistic Regression (MLR) whereas the second is the Multi Layer Perceptron (MLP). Both of the models have an input layer made up by 35 units whose activation values are the components of the feature vector.

In order to discriminate patterns belonging to one out of c classes, the output units should be interpreted as the posterior probabilities that the pattern $\mathbf{x}$ belongs to the class C_k ($P(C_k|\mathbf{x})$). This is achieved by imposing two conditions on the output functions: $y_k(\mathbf{x}) \in [0,1]$ and $\sum_{k=1}^{c} y_k(\mathbf{x}) = 1$. The activation of the output units is given by the softmax function

$$y_k(\mathbf{x}) = \frac{\exp[a_k(\mathbf{x})]}{\sum_{l=1}^{c} \exp[a_l(\mathbf{x})]}, \tag{4.1}$$

where $a_k(\mathbf{x})$ is given by the sum of the output values of all the units lying in the previous layer weighted by a set of adjustable parameters $\mathbf{w}$. Of course, in our experiments $c = 11$.

With this choice of output functions the neural network can be used as discriminant function. The probability of misclassification is minimised by assigning the pattern $\mathbf{x}$ to the class k for which $y_k(\mathbf{x}) > y_l(\mathbf{x}) \, \forall l = 1, \ldots, c, l \neq k$. An example of classification of the test image of Figure 4.1 is shown in Figure 4.4.

The activation function of the j-th hidden unit of the MLP is the sigmoid function $z_j(\mathbf{x}) = \tanh a_j(\mathbf{x})$.

Preliminary experiments using a maximum likelihood approach to the training of neural networks are reported by Vivarelli (1996); in the following we consider the Bayesian training of neural

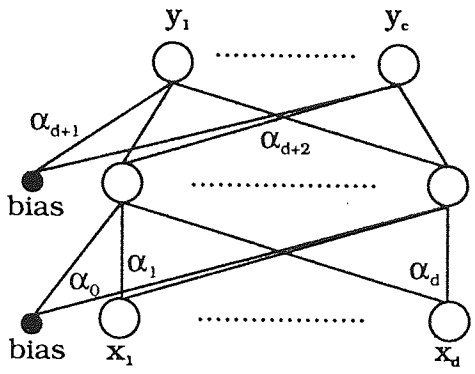


Figure 4.5: Graphical representation of a 2-layer neural network (MLP) with Automatic Relevance Determination. α_0 controls the biases of the hidden units; the hyperparameters $(\alpha_1, \ldots \alpha_d)$ control the groups of synaptic weights connecting each input to the hidden layer; α_{d+1} controls the biases of the output layer and α_{d+2} controls the weights connecting the hidden to the output units.

networks.

It is natural to describe the Bayesian training of neural networks in a three-level hierarchy. The first level involves the probability distribution $p(\mathbf{w}|\boldsymbol{\alpha})$ over the synaptic weights of a neural network given the vector of *hyperparameters* $\boldsymbol{\alpha}$. The second level of the hierarchical description is concerned with the probability distribution of the hyperparameters $p(\boldsymbol{\alpha})$. A third hierarchical level allows the comparison between different statistical models. Given a set of models H_i, $p(H_i|\mathcal{D}_n)$ can be calculated by using Bayes' theorem and thus it is possible to choose the model which is more suitable to describe the dataset or to average predictions over the models. Due to limited time, issues related to model selection have not been considered in this analysis.

In order to detect the relevant components of the input vector, one hyperparameter can be associated with each group of weights which connects one input unit to all of the units in the next layer. This is the Automatic Relevance Determination (ARD) method of MacKay and Neal (Neal, 1996). Two further hyperparameters control the distribution of the hidden-to-output weights and the biases of the output units of the MLP. This is shown in Figure 4.5. The hyperparameter α_g controls the size of the group of weights g through a Gaussian prior distribution with 0 mean and variance $\sigma_g^2 = 1/\alpha_g$. The larger the hyperparameter α_g the smaller the variance σ_g^2 of the weights belonging to the group g; the smaller α_g the larger σ_g^2. Different values of σ_g^2 affects the norm of the synaptic weight; as we see in Section 4.4.1, the relevance of the input units to the training of neural networks is determined on the basis of this idea.

After the training of the model, an estimate of the distribution of the network parameters is given by Bayes' theorem

$$p(\mathbf{w}|\boldsymbol{\alpha}, \mathcal{D}_n) = \frac{p(\mathcal{D}_n|\mathbf{w})\, p(\mathbf{w}|\boldsymbol{\alpha})}{p(\mathcal{D}_n|\boldsymbol{\alpha})}, \tag{4.2}$$

where $\mathcal{D}_n$ is a training set with n data, $p(\mathcal{D}_n|\mathbf{w})$ is the likelihood and

$$p(\mathcal{D}_n|\boldsymbol{\alpha}) = \int p(\mathcal{D}_n|\mathbf{w})\, p(\mathbf{w}|\boldsymbol{\alpha})\, d\mathbf{w}$$

is a normalising factor.

The likelihood is defined as the probability of the data set given the value of the parameters of the statistical model. From the Equation 4.1, the output of a network $y_l(\mathbf{x}^i, \mathbf{w})$ can be interpreted as the posterior probability of assigning the input vector $\mathbf{x}^i$ to the class C_l. Considering the training point $(\mathbf{x}^i, \mathbf{t}^i)$, where the l-th component of the target vector $t_l^i = \delta_{kl}$ if $\mathbf{x}^i$ belongs to the class C_k, the probability of observing $\mathbf{t}^i$ is given by $p(\mathbf{t}^i|\mathbf{w}, \mathbf{x}^i) = \prod_{l=1}^{c} \exp\left[t_l^i \ln y_l(\mathbf{x}^i, \mathbf{w})\right]$; the likelihood of a training set $\mathcal{D}_n$ composed by data independently drawn from the same distribution can be written as (see e.g. Bishop (1995))

$$p(\mathcal{D}_n|\mathbf{w}) = \prod_{i=1}^{n} p(\mathbf{t}^i|\mathbf{w}, \mathbf{x}^i) = \prod_{i=1}^{n}\prod_{l=1}^{c} \left[y_l(\mathbf{x}^i, \mathbf{w})\right]^{t_l^i} = \exp\left[\left(\sum_{i=1}^{n}\sum_{l=1}^{c} t_l^i \ln y_l(\mathbf{x}^i, \mathbf{w})\right)\right]. \tag{4.3}$$

Bayes' theorem also expresses the probability distribution of $\boldsymbol{\alpha}$ given the data $\mathcal{D}_n$ as

$$p(\boldsymbol{\alpha}|\mathcal{D}_n) = \frac{p(\mathcal{D}_n|\boldsymbol{\alpha})\, p(\boldsymbol{\alpha})}{p(\mathcal{D}_n)}, \tag{4.4}$$

where $p(\boldsymbol{\alpha})$ is the prior distribution. The factor $p(\mathcal{D}_n|\boldsymbol{\alpha})$ is called the evidence for $\boldsymbol{\alpha}$. Equations 4.2 and Equation 4.4 show the hierarchy of the Bayesian approach: the term depending upon the data in the denominator of Equation 4.2 appears also in the numerator of Equation 4.4.

Bayesian prediction for a new input $\mathbf{x}$ is given by the *marginalisation* of the classification performed by the model $p(C_k|\mathbf{x}, \mathbf{w}, \boldsymbol{\alpha})$ with respect to the posterior distribution of the parameters:

$$p(C_k|\mathbf{x}, \mathcal{D}_n) = \int p(C_k|\mathbf{x}, \mathbf{w}, \boldsymbol{\alpha})\, p(\mathbf{w}, \boldsymbol{\alpha}|\mathcal{D}_n)\, d\mathbf{w} d\boldsymbol{\alpha}. \tag{4.5}$$

The second term of the integrand represents the degree of belief in the values of the parameters $\mathbf{w}$ and $\boldsymbol{\alpha}$ given the data $\mathcal{D}_n$ and can be expressed as

$$p(\mathbf{w}, \boldsymbol{\alpha}|\mathcal{D}_n) = p(\mathbf{w}|\boldsymbol{\alpha}, \mathcal{D}_n)\, p(\boldsymbol{\alpha}|\mathcal{D}_n). \tag{4.6}$$

In this Chapter we compare two Bayesian methods for the evaluation of Equation 4.5, the evidence framework (EF) and the Markov Chain Monte Carlo method (MCMC).

Implementations of Bayesian training for neural networks differ in the way they evaluate the posterior distribution $p(\mathbf{w}, \boldsymbol{\alpha}|\mathcal{D}_n)$. EF estimates the posterior distribution of the network's parameters approximating Equation 4.6 with a Gaussian function centred around the most probable value $\mathbf{w}^{mp}$ and optimising the hyperparameters $\boldsymbol{\alpha}$.

The MCMC algorithm generates a Markov chain, obtaining values of the full posterior density distribution in order to evaluate Equation 4.5 with a Monte Carlo approximation.

Both of the methods are briefly outlined in the following sections.

Barber and Bishop (1998) suggested a different approach to the Bayesian training of neural networks based on the framework of the Ensemble learning (Hinton and van Camp, 1993). They approximated the posterior distribution of the network's parameters by optimising the Kullback-Leibler divergence between an ensemble $Q(\mathbf{w}, \boldsymbol{\alpha})$ and the true posterior. Although it seems a promising approach, due to limited time we did not consider it in our experiments.

4.3.1 The evidence framework

The EF (based on work by Gull (1988)) has been discussed by MacKay (MacKay, 1992a; MacKay, 1992b) and is similar to the *type II maximum likelihood* method.

The EF computes an approximation to equation (4.6) by assuming that the posterior probability of the hyperparameters $p(\boldsymbol{\alpha}|\mathcal{D}_n)$ is sharply peaked around its maximum $\boldsymbol{\alpha}^{mp}$. To reflect the lack of knowledge about the best value of $\boldsymbol{\alpha}$, the hyper-prior $p(\boldsymbol{\alpha})$ is chosen as a constant function on a logarithmic scale. Thus the value of $\boldsymbol{\alpha}$ maximising the posterior $p(\boldsymbol{\alpha}|\mathcal{D}_n)$ can be found maximising the evidence $p(\mathcal{D}_n|\boldsymbol{\alpha})$. Integrating over the parameter $\mathbf{w}$

$$p(\mathcal{D}_n|\boldsymbol{\alpha}) = \int p(\mathcal{D}_n|\mathbf{w})\, p(\mathbf{w}|\boldsymbol{\alpha})\, d\mathbf{w} \tag{4.7}$$

and approximating the integrand as a Gaussian centred around $\mathbf{w}^{mp}$, it is possible to maximise $p(\mathcal{D}_n|\boldsymbol{\alpha})$ with respect to $\boldsymbol{\alpha}$. The mean of the Gaussian $\mathbf{w}^{mp}$ is the point of the weight space maximising the posterior $p(\mathbf{w}|\boldsymbol{\alpha}, \mathcal{D}_n)$. It turns out that the components of the optimal values of $\boldsymbol{\alpha}^{mp}$ are given by

$$\alpha_g^{mp} = \frac{\gamma_g}{\sum_{i \in g} (w_i^{mp})^2}, \tag{4.8}$$

where

$$\gamma_g = W_g - \alpha_g \sum_{i \in g} \left((\nabla\nabla G + \boldsymbol{\alpha}\mathbb{I})^{-1} \right)_{ii}.$$

W_g is the total number of weights controlled by the hyperparameter α_g and G is the negative logarithm of the penalised likelihood ($G = -\ln p(\mathcal{D}_n|\mathbf{w}) - \ln p(\mathbf{w}|\boldsymbol{\alpha})$); the sum over $i \in g$ takes into account the elements of the matrix $(\nabla\nabla G + \boldsymbol{\alpha}\mathbb{I})^{-1}$ corresponding to the weights belonging to the group g. The factor γ_g measures the effective number of parameters controlled by the data rather than by the prior.

From the estimation of the optimal value of the hyperparameter α_g (Equation 4.8), we can construct a Gaussian approximation of $p(\mathcal{D}_n|\boldsymbol{\alpha})$. Since the hyperparameters are scale factors for the weights, their uncertainty is usually represented on a logarithmic scale (Bishop, 1995). From the Gaussian approximation, the inverse of the variance for $\log \alpha_g$ turns out to be

$$\frac{1}{\sigma^2_{\log \alpha_g}} = -\alpha_g \frac{d}{d\alpha_g} \left(\alpha_g \frac{d}{d\alpha_g} p(\mathcal{D}_n|\alpha_g) \right) = \frac{\gamma_g}{2}. \tag{4.9}$$

(see e.g. Bishop (1995)).

The EF proceeds following a two-step iterative procedure by computing the value $\mathbf{w}^{mp}$ maximising the penalised likelihood while periodically re-estimating the hyperparameter $\boldsymbol{\alpha}^{mp}$. In our experiments the first step has been achieved by optimising the penalised likelihood with a scaled conjugate algorithm (Press et al., 1992) for 70 iterations (for the MLP) or 50 iterations (for the MLR). The re-estimation of the hyperparameters $\boldsymbol{\alpha}^{mp}$ was carried out 10 times, by when the optimisation of $\boldsymbol{\alpha}$ has converged.

Some problems may arise during the implementation of EF. Since it is based on an approximation of the posterior around a local minimum, EF assumes that the Hessian matrix is positive

definite at $\mathbf{w}^{mp}$. Sometimes this is not the case because the optimisation of the weights has not fully reached a local minimum and thus some of the eigenvalues can be negative, introducing instability in the implementation. In order to avoid this, we set the negative eigenvalues in $\nabla\nabla G$ to 0. Another problem is due to the fact that EF is based upon the computation of a Hessian matrix; for large networks the amount of storage required for this matrix is considerably large, because its size grows as the square of the number of weights.

4.3.2 The Markov Chain Monte Carlo method

The second Bayesian algorithm computes Equation 4.5 by Markov Chain Monte Carlo (Neal, 1996). An MCMC algorithm constructs a Markov chain whose equilibrium distribution is the desired probability density $p(\mathbf{w}, \alpha|\mathcal{D}_n)$; the Monte Carlo algorithm (Press et al., 1992) approximates an integral as $I = \int f(\mathbf{z})\, p(\mathbf{z})\, d\mathbf{z}$ with the arithmetic mean $I \sim \sum_i f(\mathbf{z}^i)/N$, where $\{\mathbf{z}^i, i = 1 \ldots N\}$ are samples from the distribution $p(\mathbf{z})$. In our case we have $\mathbf{z} = (\mathbf{w}, \alpha)$, $f(\mathbf{z}) = p(C_k|\mathbf{x}, \mathbf{w}, \alpha)$ and $p(\mathbf{z}) = p(\mathbf{w}, \alpha|\mathcal{D}_n)$. Although samples from the chain are not independent, they can be used for computing the necessary integrals.

The algorithm[1] used in this thesis was written by Neal (Neal, 1996); a brief outline is given below.

Samples from the posterior distribution $p(\mathbf{w}, \alpha|\mathcal{D}_n) = p(\alpha|\mathcal{D}_n)\, p(\mathbf{w}|\alpha, \mathcal{D}_n)$ follow from a two-step procedure; in the first step the hyperparameters are constant and the posterior distribution of the network weights is sampled using the *Hybrid Monte Carlo* method (Duane et al., 1987). This algorithm merges the Metropolis algorithm with a dynamical simulation. As it avoids random walks, it performs better than a simple Metropolis algorithm.

In the Hybrid Monte Carlo method each network variable has an associated fictitious momentum, thereby creating a dynamical system which is described in the phase space by a set of coordinates $(\mathbf{q}, \mathbf{p})$; for neural networks, the vector position $\mathbf{q}$ is interpreted as the network weights while $\mathbf{p}$ is the associated momentum. This system is described by an Hamiltonian function H given by the sum of the kinetic and the potential energies (denoted by $K(\mathbf{p})$ and $V(\mathbf{q})$ respectively). The kinetic energy is a function of the momentum vector $K(\mathbf{p}) = \mathbf{p}^T\mathbf{p}/2$, whereas the potential energy is a function of the position $\mathbf{q}$. The Hybrid Monte Carlo method samples from the canonical distribution for $\mathbf{q}$ and $\mathbf{p}$ defined as

$$p(\mathbf{q}, \mathbf{p}) \propto \exp(-H(\mathbf{q}, \mathbf{p})) = \exp(-(K(\mathbf{p}) + V(\mathbf{q}))).$$

Provided that $V(\mathbf{q}) = V(\mathbf{w}) = -\ln p(\mathbf{w}|\mathcal{D}_n, \alpha)$, a set of values of $\mathbf{q}$ (whose posterior probability distribution is $p(\mathbf{q}) = p(\mathbf{w}|\mathcal{D}_n, \alpha)$ is obtained by generating samples from $p(\mathbf{q}, \mathbf{p})$ and ignoring $\mathbf{p}$.

Sampling from the joint distribution $p(\mathbf{q}, \mathbf{p})$ is achieved by generating new points in the phase space with constant value of H and then by doing a Gibbs sampling of the momentum $\mathbf{p}$ to change

[1] The source code of the implementation of Bayesian learning of neural networks is available at the ftp address: `ftp://ftp.cs.utoronto.ca/pub/radford/`.

the value of H. The *leap-frog* method is used to approximate the Hamilton's first order differential equations. At the end of a chain of L leapfrog steps, the state of the system can be accepted or rejected depending upon the average value of the energy H over a window of states. For all the MCMC simulations, the number of leap-frog steps L is 100, the window size is composed by 10 states and the step size correction factor (for the approximation of the Hamilton's equations) is 0.3.

The samples of the hyperparameters are generated during the second step, when samples of the hyperparameters are obtained from the posterior distribution $p(\alpha|\mathcal{D}_n)$ via Gibbs sampling.

The prior distributions for each group of weights is a 0 mean Gaussian whose precision (α_g) is specified by a vague Gamma prior. We also used scaling on the hidden-to-output weight as recommended by Neal (1996) so that the prior variances of the activations of the softmax units do not grow with the number of hidden units. The prototypes of scripts we used for running the MCMC simulations are reported in Appendix C. The sampling phase of the MCMC simulations has been run for 200 iterations; the first third of these have been discarded letting the simulation to reach the equilibrium distribution (Gelman et al., 1995). Note that in general it is very difficult to know when a MCMC simulation has reached equilibrium, see (Cowles and Carlin, 1996). During the simulations, the rejection rates were around 1%.

4.4 Experimental results

We used the two Bayesian techniques to train a MLR and a MLP networks with 30 hidden units; this is a relatively large number of hidden units, chosen with a view to observing a difference between the EF and MCMC runs.

We compare and contrast the training of the EF and the MCMC with respect to two issues.

- Feature selection. The problem of input selection is a crucial one. A smaller input vector can reduce training time and overfitting to noise by reducing the number of free parameters. Practically, there are two main schemes for the selection of the inputs (Ripley, 1996); the *forward* selection (where each feature is added one at time) and the *backward* selection (where the original feature set is reduced by deleting irrelevant components of the input vector). As we noticed in Section 4.3, Bayesian techniques enable to study the input features of the model. Section 4.4.1 reports the region based and area based performances obtained for the MLR and the MLP networks trained by the two Bayesian techniques, with a particular stress on the use of ARD.

- Empirical learning curves. In order to assess how the training algorithms rely on the amount of training data, we trained the MLP on several independent training sets by varying the quantity of data available and assessing the sensitivity to the choice of the seed initialising the random number generator of the two algorithms. The study has been carried out comparing

	E^r (%)		E^a (%)	
MLR	2HYP	ARD	2HYP	ARD
EF	34.1 ± 2.1	34.8 ± 2.4	11.4 ± 12.5	11.8 ± 14.8
MCMC	33.1 ± 2.4	32.4 ± 2.4	11.6 ± 14.7	8.6 ± 12.9
MLP	4HYP	ARD	4HYP	ARD
EF	32.8 ± 2.4	32.8 ± 2.4	11.5 ± 14.7	9.8 ± 13.6
MCMC	31.1 ± 2.3	31.0 ± 2.3	10.9 ± 14.3	9.7 ± 13.6

Table 4.3: The Table shows the region and area based misclassification rate achieved by the MLP on the test set. The overall region-based misclassification rate is defined as the fraction of regions which have not been correctly classified, i.e. $E^r = N/N_t$, where N is the number of regions incorrectly classified and N_t is the total number of regions of the data set. Similarly the area-based misclassification rate is defined as $E^a = A/A_t$, where A is the area of the regions incorrectly labelled and A_t is the overall size of the regions of the test set. The error bars associated to the region-based and area-based misclassification rate can be estimated modelling the probabilities of incorrect classification with binomial probabilities distribution with parameters (N_t, E^r) and (A_t, E^a) respectively (Ripley, 1996). Thus the variance of E^r is $E^r(1-E^r)/N_t$ and that of E^a is $E^a(1-E^a)/A_t$.

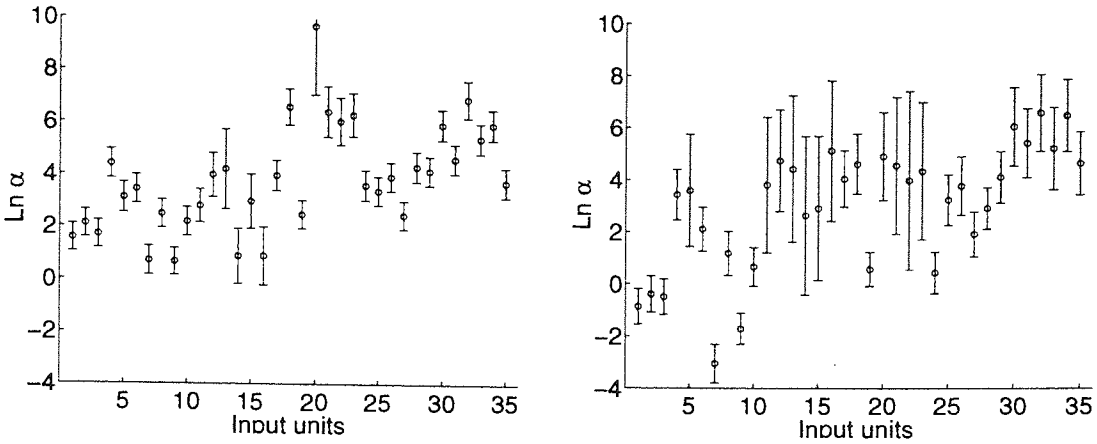
empirical learning curves of the MLP trained with the EF and the MCMC. These experiments are time consuming and have been running on our 200 MHz Silicon Graphics Challenge machine during several months before their completion; the results are presented in Section 4.4.2.

4.4.1 Automatic Relevance Determination

Bayesian training of the MLR and the MLP can be approached following two main directions.

In the first one the distribution of the input weights is controlled by one hyperparameter, regardless the input unit those weights are connected with. The hyperparameter $\boldsymbol{\alpha}$ for the MLR has 2 components, one controlling the distribution of the biases and one controlling the distribution of the synaptic weights. The hyperparameter $\boldsymbol{\alpha}$ for the MLP has 4 components which control the distribution of the biases of the hidden units, the input-to-hidden weights, the biases of the output units and the hidden-to-output weights, respectively. This leads to the specification 2HYP and 4HYP in Table 4.3 where we present the region and area based misclassification rates obtained during the experiments by the MLR and MLP networks.

The second approach actually implements ARD and is achieved by associating one hyperparameter to the group of weights which connects one input unit to all of the units in the next layer. Thus 35 elements of $\boldsymbol{\alpha}$ control the distribution of the synaptic weights connected to the input layer $(\alpha_g,\ g = 1, \ldots, 35)$ and one (α_0) controls the distribution of the biases. Of course, for the MLP $\boldsymbol{\alpha}$ has two additional components α_{36} and α_{37} controlling the distribution of the biases of the output unit and the hidden-to-output weights, respectively. From the values of the hyperparameters α_g it is possible to determine the inputs which are more relevant than others. A set of weights associated with a very small α will likely have a large norm, since the variance of their distribution will be large; the weights controlled by such a Gaussian are spread out around 0 and therefore the input



(a) ARD with evidence framework

(b) ARD with Markov Chain Monte Carlo

Figure 4.6: Graphs of the natural logarithm of the hyperparameters determined by EF (Figure 4.6(a)) and MCMC (Figure 4.6(b)) for the MLP with ARD. The labels of the input units are displayed on the x axis. The features are the mean intensity (input 1), hue angle (2, 3), topological descriptors (4 – 6), size of the region (7), coordinates of the centroid (8,9), shape (10 – 16) and texture (17 – 19) descriptors, and contextual features describing the intensity ratios (20 – 23), size ratios (24 – 27), and the relative positions (28 – 35) of the four largest surrounding regions. The natural logarithm of α_i, $i = 1, \ldots 35$ are reported on the y-axis. The error bars shown in Figure 4.6(a) have been estimated from Equation 4.9; the error bars shown in Figure 4.6(b) have been computed from the sample variance.

unit connected to those weights is relevant for the classification of the pattern. Conversely weights associated with a large hyperparameter value will likely have a small norm, since their distribution will be peaked around 0. Inputs that have small weights associated with them are thus determined to be irrelevant.

The results in Table 4.3 show that although the best results were obtained with a MLP trained with the MCMC method with ARD, the differences between the different methods are not statistically significant when the full training set is used.

The values of the hyperparameters determined by ARD for the MLP network trained with both MCMC and EF approaches are shown in Figure 4.6. The ARD parameter values for the MLR models trained with the EF and MCMC methods are not shown because they are broadly similar to the MLP results.

The two training methods give similar results: almost all of the hyperparameters determined by MCMC and EF lie well within the range of 95% confidence intervals; they also give similar relevance to a common subset of features, although some differences can be noticed.

The graphs show that the features describing the colour, size and the y position of the regions (inputs 1, 2, 3, 7 and 9) are the most relevant in training the classifier; the hyperparameters corresponding to those features have small values for both the MCMC and the EF methods. The lesser relevance of the topological features and the x position (x_8) of the regions is also recognised by both of the methods.

There is some disagreement between the EF and MCMC methods over the relevance of the shape descriptors (inputs $10-16$), although we note that the MCMC in particular yields large error bars. Although EF gives relevance to the the whole set of the shape descriptors, the MCMC method reduces the subset of the relevant shape descriptors to the first two features x_{10} and x_{11}.

Concerning the textural features, EF and MCMC point out the importance of the last feature x_{19}, whereas x_{17} and x_{18} seem less relevant for the labelling of the regions.

Among the contextual features (inputs $20-35$), the relative intensity of one region $(x_{24},\ldots x_{27})$ with respect to the four surrounding it appear the most useful.

All of the experiments have shown that the coordinates (x,y) of the centroids have a different weight in training the networks. Because images are taken with the y axis closely aligned to the vertical, the classification of the regions is unlikely to depend upon the x coordinate of the centroid; for example, regions representing cars appear in the data base in many positions along the x axis, whereas their y coordinates are ranged in a well defined interval. A similar consideration applies for the contextual features $28-35$, where a different relevance is accorded to the x and the y offsets of the relative position.

The determination of the irrelevance of some of the features does not mean that those attributes are *absolutely* irrelevant but that they have not properly encoded the information about a region.

4.4.2 Empirical learning curve

A thorough investigation of the differences between the EF and the MCMC methods training of neural networks can be carried out by a statistical analysis of empirical learning curves. We should note in advance that the task of comparing two learning techniques, determining which one actually out-performs the other and detecting all the relevant sources of variation, is a difficult one and every statistical test has to be considered as an approximate and heuristic test (Dietterich, 1998).

Generalisation capabilities of neural networks depend upon many stochastic factors (e.g. Rasmussen, 1996); for our problem, the variations in generalisation error are mainly due to the randomness in the learning algorithms and in the samples from the distribution of the test data and the training sets. The learning curve can be defined as the expectation of the generalisation error averaged over the distribution of all the stochastic components

$$E^g(n) = \int E^g(\mathcal{D}_n, r, \mathbf{x}, t)\, p(\mathcal{D}_n, r, \mathbf{x}, t)\, d\mathcal{D}_n dr d\mathbf{x} dt, \tag{4.10}$$

where $(\mathbf{x},t)$ is the test point, $\mathcal{D}_n$ is the training set and r represents stochastic effects due to the training method; in our case, the stochastic effect r depends on the seed initialising the random number generator for the two algorithms.

Generalisation errors of the MLP trained with the two Bayesian algorithms have been estimated on the basis of the classification performances obtained on the original test set. At least two different evaluation of generalisation errors can be employed. The most common measure is the $0/1$ loss function; in this case the neural network is considered as a discriminant function assigning

the test vector $\mathbf{x}$ to the class C_k for which $y_k(\mathbf{x}) > y_l(\mathbf{x}), \forall l = 1, \ldots c$. A second performance measure is based on the likelihood of the true class (see Equation 4.3) averaged over the test points.

Although the latter is more suitable to describe the class conditional distribution estimated by the neural network, the former provides a more intuitive representation of the overall performance of the classifier and in all our experiments we evaluate the generalisation error of the MLP with the 0/1 loss function.

In our simulations, we conducted three kinds of experiments evaluating the effect on the generalisation error due to the choice of (i) the training set for a given random seed, (ii) the random seed for a given dataset and (iii) randomising over the choice of the dataset and the random seed. The sets of experiments (i) and (ii) are aimed to estimate the variances in generalisation error due to the training dataset and the random seed effects, respectively. The aim of the set of experiments (iii) is to find evidence of which learning algorithms can improve the generalisation performance of the MLP for the task at hand.

For all of the experiments, Bayesian algorithms with ARD have trained the MLP with 30 hidden units, using several initialisations of the random number generators; as explained above, the ARD approach has been implemented by using 38 hyperparameters (35 controlling the distribution of the input-to-hidden weights, 1 controlling the biases of the hidden units, 1 controlling the distribution of the hidden-to-output weights and 1 for the output's biases). In order to obtain several datasets with different sizes, the original training set made up by 5832 data has been successively subdivided obtaining two data sets of half size (2916 examples), four of size 1458 and eight of size 729. Below this, ten data sets each for the sizes 365, 182, 91 and 46 have been generated. All the datasets were disjoint.

(i)The effect of the training data

In the first group of experiments we trained the MLP varying the amount of training data; for both the MCMC and EF methods the value of the random seed has been fixed.

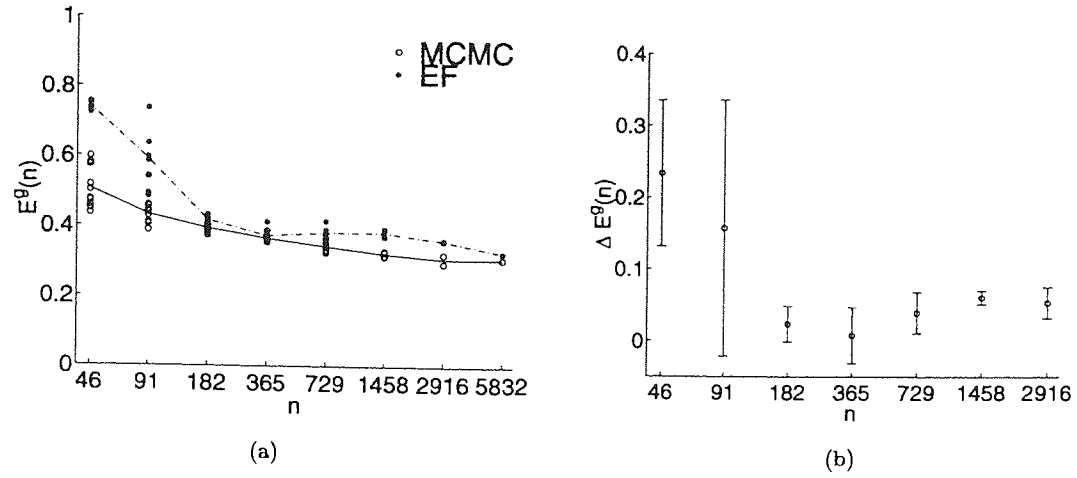
We note that a formal evaluation of Equation 4.10 is not analytically tractable since in many real problems a formal expression of all its elements is not available. An evaluation of the expected generalisation error (with respect to the distribution the test data) can be estimated via Monte Carlo integration by considering the number of misclassified regions obtained by the MLP trained with each learning algorithm as in

$$\hat{E}^g_{\text{EF}}(\mathcal{D}_n, r_{\text{EF}}) = \frac{N_{\text{EF}}}{N_t} \tag{4.11}$$

$$\hat{E}^g_{\text{MCMC}}(\mathcal{D}_n, r_{\text{MCMC}}) = \frac{N_{\text{MCMC}}}{N_t}, \tag{4.12}$$

where N_t is the total number of test regions and N_{EF} and N_{MCMC} are the number of regions of the test set incorrectly classified by the MLP trained with the EF and the MCMC, respectively.

The test set performance of both methods is shown for each training size in Figure 4.7(a), where the region-based misclassification rate is shown against the amount of data in the training set. Table



(a) (b)

Figure 4.7: Figure 4.7(a) shows the learning curves for the EF and MCMC methods obtained by reducing the number of data points in the training sets. The symbols ∘ and ∗ label the performances of the MCMC and EF respectively. The average values of the region based misclassification rates are drawn by the continuous (MCMC) and the dash-dotted (EF) lines. Figure 4.7(b) shows the average of the pairwise difference of the generalisation errors reported during the experiments with its 95% interval of variability due to the choice of the dataset. Since we have only one training set with 5832 data, Figure 4.7(b) does not report the difference of the generalisation errors as the average and the variance can not be estimated.

4.4 reports the expected value of the generalisation errors with their intervals of variability due to the choice of the training set. The third column shows the pairwise difference in generalisation error and its 95% confidence interval; note that this is a paired comparison, i.e. the differences between the EF and MCMC methods are computed on the *same* training set. Since we fixed the random initialisation of the algorithms, the expected pairwise difference can be averaged over the choice of the training sets.

As expected an improvement in performance is observed as the size of the training set is increased. We also note that the differences between the two algorithms in the generalisation errors increase as the size of the dataset is reduced; in particular the MCMC method performs significantly better than the EF method for all the sizes of the data set but for $n = 365$. The difference between the two methods is actually enlarged for small datasets ($n = 46, 91$). As the number of data is much smaller than the number of parameters in the network, it is unlikely that the Gaussian distribution approximates reliably the posterior distribution of the parameters and the EF breaks down.

(ii)The effect of the random seed

The second set of experiments has been carried out to investigate the effect of the random seed of the algorithms on the generalisation performances; thus we chose to fix one training dataset for each size and we trained the MLP with the EF and the MCMC initialising the random generators with 10 different seeds.

n	$\hat{E}^g_{\text{EF}}$	$\hat{E}^g_{\text{MCMC}}$	$\Delta\hat{E}^g$
46	0.74 ± 0.02	0.51 ± 0.11	0.23 ± 0.10
91	0.59 ± 0.18	0.43 ± 0.05	0.16 ± 0.18
182	0.42 ± 0.03	0.39 ± 0.02	0.02 ± 0.02
365	0.37 ± 0.03	0.37 ± 0.02	0.01 ± 0.04
729	0.38 ± 0.03	0.34 ± 0.03	0.04 ± 0.03
1458	0.38 ± 0.02	0.32 ± 0.01	0.06 ± 0.01
2916	0.36 ± 0.01	0.31 ± 0.04	0.05 ± 0.02
5832	0.33	0.31	0.02

Table 4.4: The Table shows the average of the generalisation error reported by the MLP trained with EF and MCMC by varying the composition of the datasets; the third column shows the expected pairwise difference $\Delta\hat{E}^g = \hat{E}^g_{\text{EF}} - \hat{E}^g_{\text{MCMC}}$. The uncertainty associated with each value is the 95% confidence interval of variability due to the choice of the dataset. We do not estimate the variances in the last line because we have only one training set with 5832 data.

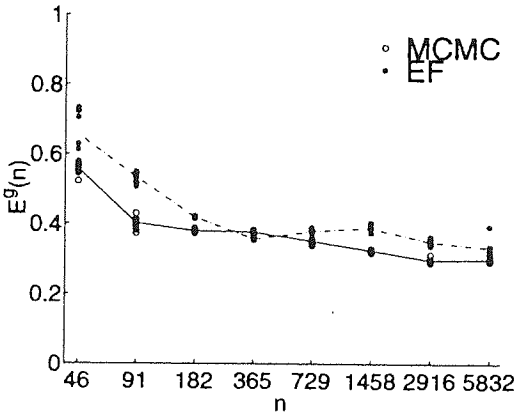


Figure 4.8: The Figures show the learning curves for the EF and MCMC methods obtained by varying the random seeds. The symbols ∘ and ∗ label the performances of the MCMC and EF respectively. The average values of the region based misclassification rates are drawn by the continuous (MCMC) and the dash-dotted (EF) lines.

n	$\hat{E}^g_{\text{EF}}$	$\hat{E}^g_{\text{MCMC}}$
46	0.66 ± 0.15	0.56 ± 0.03
91	0.53 ± 0.03	0.41 ± 0.03
182	0.42 ± 0.01	0.38 ± 0.01
365	0.36 ± 0.01	0.38 ± 0.01
729	0.38 ± 0.02	0.35 ± 0.01
1458	0.39 ± 0.02	0.33 ± 0.01
2916	0.35 ± 0.02	0.30 ± 0.01
5832	0.33 ± 0.04	0.30 ± 0.01

Table 4.5: The Table shows the average of the generalisation error reported by the MLP trained with EF and MCMC by varying the random seed. The uncertainty associated with each value is the 95% confidence interval of variability due to the initialisation of the random number generator.

Figure 4.8 shows the region-based misclassification rates we obtained during these experiments and Table 4.5 reports the average misclassification rates of the two methods with their variance due to the choice of the random seed; the generalisation errors have been evaluated according to Equations 4.11 and 4.12. We note that a *paired* comparison between the two Bayesian approaches can not be carried out because the two methods do not use the *same* initialisation[2].

From Table 4.5 we note that the two learning algorithms attain a similar level of generalisation as the average misclassification rates reported by the MCMC and the EF lie well within the 95% interval of variability. However we notice that the average generalisation error $\hat{E}^g_{\text{MCMC}}(n)$ lies below $\hat{E}^g_{\text{EF}}(n)$ for most of the training sizes and that the variance on $\hat{E}^g_{\text{MCMC}}(n)$ is always smaller than (or at most equal to) the variance affecting $\hat{E}^g_{\text{EF}}(n)$.

(iii)The effect of the learning algorithms

The investigation of the effect induced by the two Bayesian algorithms onto the empirical learning curves has been investigated with a third set of experiments, where the MLP has been trained randomising the choice of the dataset and the random seed and evaluating the generalisation performance on the original test set.

The experiments have been analysed following a two-way ANOVA design (DeGroot, 1984; Rasmussen et al., 1996). In the two-way ANOVA analysis, the generalisation error obtained after training the MLP with a given learning algorithms is modelled by the Equation

$$E^g\left(\mathcal{D}_n^i, \mathbf{x}^j, t^j\right) = \mu_n + \alpha_n^i + \beta_n^j + \varepsilon_n^{ij}, \tag{4.13}$$

where μ_n is the mean generalisation error obtained with a training set with n data, α_n^i and β_n^j model the effect on μ_n due to choice of the training set $\mathcal{D}_n^i$ and to the test point $\left(\mathbf{x}^j, t^j\right)$, respectively. The residual variations in generalisation error due to stochastic effects and to the interaction between the training set and the test point are described by the noise term ε_n^{ij}. The stochastic effects include the variability of $E^g\left(\mathcal{D}_n^i, \mathbf{x}^j, t^j\right)$ due to the choice of the random seed in the learning algorithm. We assume that the values of α, β and ε are normally distributed around zero with variance σ_α^2, σ_β^2 and σ_ε^2, respectively; these assumptions are not strictly applicable when the 0/1 loss function for classification is used, but a more appropriate model is not straightforward to analyse. However, according to Rasmussen (1996), these assumptions are commonly used and they lead to reasonable conclusions[3].

[2] In Equations 4.11 and 4.12 we have indicated the random seed in the EF and MCMC algorithms with two distinct variables r_{EF} and r_{MCMC} because the random initialisation does not have the same effect on the two methods. The EF uses the initial random seed at the beginning of the optimisation, generating randomly the weight vector $\mathbf{w}_0$ from a Gaussian distribution whose variance is specified by the hyperparameters $\boldsymbol{\alpha}_0$; the weight vector and the hyperparameters are then optimised by maximising the posterior distribution using information concerning the gradient of the penalised likelihood. The MCMC algorithm uses the initial random seed in order to initialise the random generator from which samples of the weight vectors and the hyperparameters are generated according to the posterior distributions during the whole training and test procedures; unlike in the EF, the stochastic element plays a key rôle when predictions with MLP are obtained by the MCMC algorithm.

[3] An alternative approach to empirical study on difference in classification performance has been presented by Dietterich (1998).

The superscripts i and j in Equation 4.13 label the training set $\mathcal{D}_n^i$ and the test points $j = 1 \ldots 1505$, respectively. As above, the number of training sets $\mathcal{D}_n^i$ varies with the amount of data n, being $i = 1, 2$ for $n = 2916$ data, $i = 1 \ldots 4$ for $n = 1458$, $i = 1 \ldots 8$ for $n = 729$ and $i = 1 \ldots 10$ for $n = 365$, 182, 91 and 46. We do not carry out such analysis when the full amount of data is used (i.e. $n = 5832$) because we can not estimate the variance on the expected generalisation error generated by only one training set[4].

The estimate of the mean value μ_n and its variance have been computed according to the two-way ANOVA design with one observation for each coupling $(\mathcal{D}_n^i, j)$ as

$$\hat{\mu}_n = \frac{1}{N_t D} \sum_{i=1}^{D} \sum_{j=1}^{N_t} \hat{E}^g \left(\mathcal{D}_n^i, \mathbf{x}^j, t^j \right), \tag{4.14}$$

$$\hat{\sigma}_{\hat{\mu}_n} = \sqrt{\frac{\hat{\sigma}_a^2}{D} + \frac{\hat{\sigma}_b^2}{N_t} + \frac{\hat{\sigma}_\epsilon^2}{N_t D}} \tag{4.15}$$

(Rasmussen et al., 1996), where N_t and D indicate the overall number of test examples and training sets, respectively; $\hat{E}^g \left(\mathcal{D}_n^i, \mathbf{x}^j, t^j \right)$ indicates the generalisation error of the MLP evaluated for the training set $\mathcal{D}_n^i$ and the test point $(\mathbf{x}^j, t^j)$. The $\hat{}$ in Equations 4.14 - 4.15 denotes the empirical estimates of the variables; in particular, the values of σ_α^2, σ_β^2 and σ_ϵ^2 follow from the formulæ

$$\hat{\sigma}_\alpha^2 = \frac{(mse_\alpha - mse_\epsilon)}{N_t}, \hat{\sigma}_\beta^2 = \frac{(mse_\beta - mse_\epsilon)}{D}, \hat{\sigma}_\epsilon^2 = mse_\epsilon,$$

where the *mean squared errors* are evaluated as

$$mse_\alpha = \frac{N_t}{D-1} \sum_{i=1}^{D} (\hat{\mu}_{i\cdot} - \hat{\mu}_n)^2,$$

$$mse_\beta = \frac{D}{N_t - 1} \sum_{j=1}^{N_t} (\hat{\mu}_{\cdot j} - \hat{\mu}_n)^2,$$

$$mse_\epsilon = \frac{1}{(D-1)(N_t-1)} \sum_{i=1}^{D} \sum_{j=1}^{N_t} \left[\left(\hat{E}^g \left(\mathcal{D}_n^i, \mathbf{x}^j, t^j \right) - \hat{\mu}_n \right) - (\hat{\mu}_{i\cdot} - \hat{\mu}_n) - (\hat{\mu}_{\cdot j} - \hat{\mu}_n) \right]^2,$$

and

$$\hat{\mu}_{i\cdot} = \frac{1}{N_t} \sum_{j=1}^{N_t} \hat{E}^g \left(\mathcal{D}_n^i, \mathbf{x}^j, t^j \right), \hat{\mu}_{\cdot j} = \frac{1}{D} \sum_{i=1}^{D} \hat{E}^g \left(\mathcal{D}_n^i, \mathbf{x}^j, t^j \right)$$

(cf. Equations 12 – 17 in Rasmussen et al. (1996)).

In order to quantify the difference in generalisation errors obtained by the MLP trained by EF and MCMC, we carried out also the comparison of the two methods by modelling the difference of the expected generalisation errors according to the two-way ANOVA design, i.e.

$$\begin{aligned} \Delta E^g \left(\mathcal{D}_n^i, \mathbf{x}^j, t^j \right) &= E_{\text{EF}}^g \left(\mathcal{D}_n^i, \mathbf{x}^j, t^j \right) - E_{\text{MCMC}}^g \left(\mathcal{D}_n^i, \mathbf{x}^j, t^j \right) \\ &= \Delta\mu_n + \Delta\alpha_n^i + \Delta\beta_n^j + \Delta\epsilon_n^{ij}. \end{aligned} \tag{4.16}$$

In Equation 4.16 the variable $\Delta\mu_n$ models the variation in the mean generalisation error obtained with a training set with n data, $\Delta\alpha_n^i$ and $\Delta\beta_n^j$ model the variations of the effects due to choice of the

[4] In this case, the variability in the generalisation error is due to the initialisation of the random number generator which has been presented in (ii).

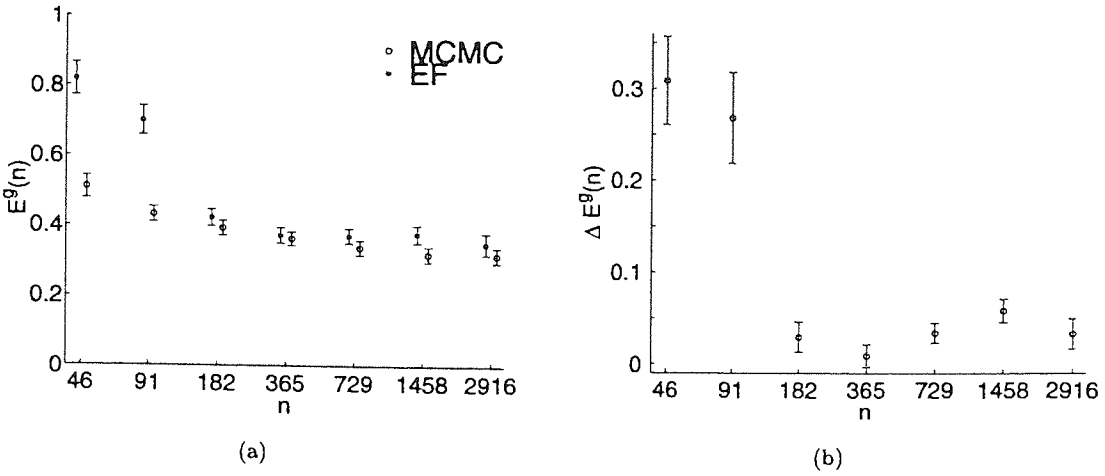


Figure 4.9: Figure 4.9(a) shows the empirical learning curves of the MLP trained with the EF and MCMC algorithms; the symbols $*$ and $\circ$ label the performances of the EF and MCMC respectively. Figure 4.9(b) shows the graph of the empirical $\Delta E^g(n)$ as a function the amount of training data. The estimate of the means and their intervals of variability are obtained according to the two-way ANOVA analysis, as explained in the text. The error bars are generated by the 95% confidence interval due to the stochastic effects of the training data, the test points and the initialisation of the random seeds.

training set $\mathcal{D}_n^i$ and to the test point $(\mathbf{x}^j, t^j)$, respectively; $\Delta\varepsilon_n^{ij}$ describes residual variations in the difference of the generalisation errors due to stochastic effects and to the interaction between the training set and the test point. After the substitution of the generalisation error $E^g(\mathcal{D}_n^i, \mathbf{x}^j, t^j)$ with the difference $\Delta E^g(\mathcal{D}_n^i, \mathbf{x}^j, t^j)$, the formulæ for the numerical estimation of $\Delta\hat{\mu}$ and its variance are similar to Equations 4.14 and 4.15, respectively.

Figure 4.9(a) shows the empirical learning curves of the MLP trained with the EF and the MCMC algorithms, as calculated from Equations 4.14 and 4.15; the numerical estimation of $\Delta\hat{E}^g(n)$ with its 95% interval of variability as a function of the amount of training data is shown in Figure 4.9(b). The numerical values of $\hat{E}^g_{\text{EF}}(n)$, $\hat{E}^g_{\text{MCMC}}(n)$ and $\Delta\hat{E}^g(n)$ are reported in Table 4.6.

The evidence that one Bayesian algorithm has significantly better generalisation error than the other can be obtained by investigating whether the estimated overall difference $\Delta\hat{E}^g(n)$ is significantly different from zero; of course the conclusions drawn from this analysis depend upon the assumptions concerning the model adopted in Equation 4.16.

A quasi-F test (Lindman, 1974) can verify any significant deviation of $\Delta\hat{E}^g(n)$ from 0 by using the F statistics. The result of the test is the probability that the null hypothesis $\Delta\hat{E}^g(n) = 0$ is true; a low value of this probability indicates a significant deviation of from $\Delta\hat{E}^g(n)$ zero, highlighting a significant difference of the two training algorithms.

Using the quasi-F test at the 0.05 level of significance, we find that for most of the sizes of the training sets the use of the MCMC algorithm has a relevant effect in improving significantly the

n	$\hat{E}^g_{\text{EF}}$	$\hat{E}^g_{\text{MCMC}}$	$\Delta\hat{E}^g$
46	0.82 ± 0.05	0.51 ± 0.03	0.30 ± 0.05
91	0.70 ± 0.04	0.43 ± 0.02	0.26 ± 0.05
182	0.42 ± 0.02	0.39 ± 0.02	0.03 ± 0.02
365	0.37 ± 0.02	0.36 ± 0.02	0.01 ± 0.01
729	0.37 ± 0.02	0.34 ± 0.02	0.03 ± 0.01
1458	0.38 ± 0.03	0.32 ± 0.02	0.06 ± 0.01
2916	0.35 ± 0.03	0.32 ± 0.02	0.03 ± 0.02

Table 4.6: The Table shows the average of the generalisation error reported by the MLP trained with EF and MCMC by varying the composition of the datasets; the third column shows the expected pairwise difference. The estimate of the means and their intervals of variability are obtained according to the two-way ANOVA analysis. The uncertainty associated with each value is the 95% confidence interval due to the stochastic effects of the training data, the test points and the initialisation of the random seeds.

generalisation error of the MLP with respect to the one obtained by the MLP trained with the EF, although the difference varies with the number of training examples. In particular $\Delta\hat{E}^g(n)$ is significantly different from zero for dataset with $n = 2916$, 1458, 729, 182, 91 and 46 data. For large datasets the difference in generalisation error and its interval of variability are always below 7%; the difference is enlarged when small datasets are used (being in the range $[20\%, 30\%]$ for $n = 91$ and $[25\%, 35\%]$ for $n = 46$), although we note that the results have a larger variability. MLP trained with the MCMC obtains better generalisation error than that obtained by the MLP trained with EF when small datasets are used; we expect that this is due to the inaccurate Gaussian approximation of the posterior distribution of the network's parameters estimated by the EF. This is not unexpected as it is improbable that a small amount of data can approximate reliably the posterior distribution of the parameters of the MLP with a Gaussian distribution.

For dataset with $n = 365$, $\Delta\hat{E}^g(n)$ is not significantly different from zero at the 0.05 level of significance, and thus the generalisation errors of the MLP trained by the EF and the MCMC must be considered similar.

Due to questions concerning the applicability of the Gaussian assumptions when using the two-way ANOVA design of Equation 4.13 with 0/1 class, we also analysed the results when j indexes not a single test case but a pool of test cases. Pool sizes of 2, 4, 8, 16 and 32 were created. The results obtained were very similar to those shown above, with $\Delta\hat{E}^g(n)$ being significantly different from zero for training sets of size 1458, 729, 182, 91 and 46. The only difference was that $\Delta\hat{E}^g(n)$ for $n = 2916$ was now found to be not significantly different from zero.

During our experiments we noticed another disadvantage of the EF for small data sets, namely that a large part of the computational effort is taken up with the inversion of the Hessian; on the contrary, the MCMC method we have used provides its results using less CPU time for small data sets.

4.5 Discussion

In this Chapter we have compared and contrasted the evidence framework and a Markov Chain Monte Carlo method for training neural networks; this is the first study we are aware of that has carried out such a comparison. During our experiments, a Multiple Logistic Regression and a Multi Layer Perceptron with 30 hidden nodes have been trained on the task of classifying segmented outdoor images.

The two Bayesian methods have been contrasted in many respects, namely investigating the relevance of each feature in training the networks and analysing their sensitivity due to the composition of the datasets, the seeds initialising the random generators of the two algorithms and the amount of training data.

Investigation of the relevance of the input feature has been carried out training the MLR and the MLP on the full dataset. The two Bayesian methods achieve also similar results when ARD evaluates the relevance of different input variables; they highlight the relevance of some features such as the mean intensity, the size and the y coordinate of the regions, some shape descriptors and a subset of the contextual features. Less relevance is assigned to other features, although with some difference in the two methods.

The generalisation performances obtained by the MLP trained with EF and MCMC has been thoroughly investigated by analysing empirical learning curves; this analysis is aimed to assess the dependence of the two training methods on the size of the datasets. The original training set has been split up, obtaining several datasets composed by 2916, 1458, 729, 365, 182, 91 and 46 examples.

In order to assess the variances on the generalisation error due to the training dataset and the random seed effects, we evaluated the generalisation error of the MLP obtained by varying the datasets (fixing the random seeds of the two algorithms) and varying the initialisation of the random generator (with a fixed dataset).

We evaluated the sensitivity of the generalisation capabilities to the choice of the learning algorithms training the MLP with the Bayesian algorithms and randomising over the choice of the dataset and the initialisation of the random number generator. Assuming the additivity and the Gaussian distribution around zero of the stochastic effects due to the choice of the datasets, the test points and the initialisation of the random number generator, we investigated the difference of the generalisation errors of the MLP due to the EF and the MCMC training algorithms using a two-way ANOVA design. Any statistically significant deviation from zero has been highlighted by using a quasi-F test. Our results suggest that on the task of labelling images, the MCMC method obtains better generalisation performances than the EF, revealing statistically significant differences in the generalisation errors for most of the sizes of the datasets. For large amount of training data, the difference in generalisation errors (with its 95% interval of variability) lies within the interval $\Delta E^g(n) \in [0\%, 7\%]$; the expected difference between the two methods becomes

larger for datasets with 91 and 46 data (i.e. $\Delta E^g(n) \in [20\%, 30\%]$ and $\Delta E^g(n) \in [25\%, 35\%]$, respectively), although it is affected by large variance due to the choice of the datasets and the random seed. This difference in generalisation errors can be explained since the EF is based upon a Gaussian approximation of the posterior distribution around the most probable weight vector; when the amount of data in the training set is much smaller than the number of parameters this approximation becomes inaccurate, worsening the training of the network.

The analysis presented suggests that, for the problem at hand and with the model of analysis considered, the MCMC method yields generalisation errors that are significantly better than those of the EF algorithm.

Chapter 5

Conclusion

5.1 Summary and future work

In this thesis we presented some studies on generalisation capabilities of Gaussian processes and Bayesian neural networks. This topic is one of the major issue of pattern recognition since an actual embedding of such learning-systems in safety critical applications will be allowed only when a thoroughly investigation of their reliability has been carried out. We focussed the research work on two issues, the study of learning curves and the reduction of input variables; both the issues have been investigated from theoretical and empirical perspectives. The theoretical analysis has been carried out by considering Gaussian processes, whose predictions can be regarded as generated by a neural network with a Gaussian prior over the weights in the limit of an infinite number of hidden units.

In the following, we review the topics discussed in the thesis, presenting the main results obtained as well as some open questions which motivate future work.

5.1.1 Upper and lower bounds on the learning curve for Gaussian processes

In Chapter 2 we discussed non-asymptotic upper and lower bounds on the learning curves for one-dimensional Gaussian processes. From the intuitive remark that increasing the number of training points never worsens the generalisation error, we proposed two upper bounds ($E_1^u(n)$ and $E_2^u(n)$) on the learning curve by considering the generalisation error of a Gaussian process trained on a subset of the training data and averaged over the distribution of the training points. We have also tested the lower bound ($E^l(n)$) proposed by Opper (Opper and Vivarelli, 1999).

In order to investigate the tightness of the bounds in relation to the smoothness of the stochastic process, experiments have been run considering four covariance functions (the modified Bessel of

first, second and third order and the squared exponential), describing processes characterised by different levels of regularity.

We noticed and explained that the learning curves and their upper bounds exhibit an initial phase when they are linearly dependent on the amount of training data.

The analysis of the experiments has shown that the behaviour of the learning curves as well as the tightness of the bounds depends upon the choice of the covariance function, the value of the characteristic length of the process and the variance of the noise corrupting the stochastic process. The smoothness of the stochastic process affects the bounds because the rougher the process the more localised the information. Since the bounds have been determined considering subsets of the training data composed by one or two datapoints around a test point, the bounds become closer to the learning curves for rougher processes.

The lengthscale affects the behaviour of the learning curve and its bounds rescaling the amount of training data and stretching the curves. The value of the noise variance affects the tightness of the bounds; since the noise level hides the underlying stochastic process, the upper bounds become looser increasing the noise variance.

We have also developed asymptotic expansions of the upper bounds, showing that the bounds tend to asymptotic plateaux. The theoretical analysis have been supported by the results of the numerical simulations.

The experiments concerning the lower bound $E^l(n)$ proposed by Opper have revealed that the tightness depend on the smoothness of the process; in particular since $E^l(n)$ is also an upper bound on the expected training error, the lower bound is tighter when the learning curve becomes closer to the training error, i.e. when the stochastic process becomes smoother. The variance of the noise affects the lower bound as $E^l(n)$ becomes closer to the learning curve for lower noise levels.

The analysis for the upper bounds carried out in this thesis was limited to a one dimensional input space; with this assumption, the bounds have been analytically tractable by using order statistics (David, 1970). In order to be able to apply such upper bounds on more realistic tasks we should be able to extend those results to multivariate problems. This problem becomes difficult since in higher dimension we are not able to use the results of order statistics, although the results on the initial phase of learning still remain valid. This limitation does not affect the lower bound $E^l(n)$ since it still holds for multivariate problems (Opper and Vivarelli, 1999). However, we expect that the tightness of the bounds will depends on the dimension of the input space, being tighter for lower input-dimensionality.

In the Chapter we have calculated bounds on the learning curve of Gaussian processes averaging over the distribution of the training data. A further step in this research should carry out the estimation of the variance of the bounds due to the distribution of the training sets; similarly to the bounds presented in the Chapter, it is reasonable to expect that the size of the variance will depend on the smoothness of the stochastic process as well as the lengthscale and the noise level.

5.1.2 Discovering hidden features with Gaussian process regression

Chapter 3 shows how to discover hidden feature with Gaussian process for regression. In conventional Gaussian process regression, a diagonal matrix is used in the covariance function in order to estimate the distance between datapoints. Although the entries in the diagonal matrix can detect the relevance of each input in training the model (the Automatic Relevance Determination), this distance matrix is not able to discover the transformation from the the manifest and the hidden space.

In the Chapter we show how to discover the linear mapping between the manifest input space and the hidden manifold by using a general distance matrix. This allows the discovery of the hidden feature space through an eigen-analysis of the distance matrix; in particular the number of relevant eigenvalues is an estimate of the dimension of the hidden space and the matrix of the corresponding eigenvectors approximates the actual transformation from the manifest to the hidden-feature space.

Gaussian processes using a diagonal and a general distance matrices have been compared and contrasted on two regression tasks.

The regression of a trigonometric function is aimed to verify how the noise and the input-dimensionality of the manifested space affect the relative generalisation error between Gaussian processes using the two distance matrices. The use of a general matrix improves significantly the generalisation error of a GP as it is able to discover the underlying transformation from the manifest to the hidden feature spaces. Our experiments also showed that a Gaussian process using a general distance matrix attains relative generalisation errors similar to those obtained by a Gaussian process using prior knowledge about the hidden dimensionality of the regression problem.

The regression of a high-interaction surface (as reported by Breiman (1993)) showed that, although generalisation performances obtained for some sizes of the training set are affected by overfitting, the general distance matrix improves significantly the generalisation error of the Gaussian process. The eigenvectors of the general distance matrix span a subspace which well approximates the original hidden feature space; in particular the use of a general distance matrix allows to reduce the ten-dimensional manifest space to a three-dimensional feature space.

The problem of overfitting exhibited during our experiments by Gaussian processes using the general distance matrix may be reduced by a full Bayesian approach to the parameters. One future direction of research may be to develop the Bayesian training of the Gaussian process's parameters, allowing the use of the general distance matrix on real-world tasks.

The study we presented is able to discover only linear transformation between the hidden manifold and the manifest space. A future extension of this work could follow an idea suggested by Sampson and Guttorp (1992), by studying the discovery of non-linear mapping; this may involve the use of a further level of GP in order to model the coordinate functions describing the non-linear transformation (Williams, 1998).

5.1.3 Using Bayesian neural networks for classifying segmented images

A comparison of the performances of neural networks trained with the evidence framework (EF) of MacKay (1992) and a Markov Chain Monte Carlo method (MCMC) due to Neal (1996) on a task of classifying segmented outdoor images has been presented in Chapter 4. The comparison has been carried out training a Multiple Logistic Regression and a Multi Layer Perceptron with 30 hidden nodes.

The two Bayesian algorithms highlight some differences when an evaluation of the relevance of the input features in training the neural networks is carried out by Automatic Relevance Determination. Although both the Markov Chain Monte Carlo and the evidence framework reveal that the most relevant features are the colour, the size and the y position of the objects of an image, the algorithms assign different relevance to the other components of the input vectors such as the shape descriptors, the topological features and the contextual features.

To further investigate any differences between the two Bayesian methods, we have also analysed the empirical learning curves of the MLP considering the effects on the generalisation error due to the composition of the datasets and the initialisation of the random number generators of the two Bayesian algorithms. A number of datasets with different sizes have been generated from the original database.

We assessed the variances on the generalisation error of the MLP trained with the Bayesian training algorithms due to the choice of the datasets (for a fixed random seed) and the initialisation of the random number generators (for a given training set).

The overall improvements in generalisation performance of the MLP due to the learning algorithm has been estimated randomising over the choice of the dataset and the initialisation of the random number generator. Following a two-way ANOVA analysis, we have shown that on this task the Markov Chain Monte Carlo approach provides lower misclassification rates than the evidence framework, although the differences in misclassification depend upon the size of datasets. In particular when large amount of data are used the expected difference is below 7%. When training sets with 365 data are used the generalisation performance obtained with the two algorithms does not show significant differences, highlighting the similarity of the the Markov Chain Monte Carlo and the evidence framework in training the MLP on the task at hand.

When a small amount of data are used in the training set, the Gaussian posterior distribution of the synaptic weights computed by the evidence framework does not approximate reliably the posterior distribution of the network's weights; this turns out to increase the expected difference in generalisation error of the MLP in the interval $[20\%, 30\%]$, highlighting the superiority of the Markov Chain Monte Carlo algorithm for small datasets. Our experiments have also highlighted another disadvantage of the EF for small data sets, namely that a large part of the CPU time is taken up computing the inversion of the Hessian matrix; on the contrary, this disadvantage does not arise using the MCMC algorithm for training the MLP.

In conclusion the analysis presented shows that, for the problem at hand and with the model

of analysis considered, the MCMC algorithm improves the generalisation error of the MLP and yields performances which are significantly better than those of the EF algorithm.

5.2 Conclusions

This thesis has investigated two topics concerning the generalisation capabilities of data-driven statistical models; we focused our research on Gaussian processes and Bayesian neural networks, studying the accuracy of prediction in relation to the number of training data and examining the relevance of the input variables with respect to the generalisation capabilities.

The study of the bounds on the learning curves of Gaussian process has lead to the estimation of upper and lower extrema of the interval in which the expected generalisation error lies as a function of the number of training data. We also noticed that the tightness of the bounds depends on the regularity of the stochastic process described by the covariance function of the Gaussian process. These results can give a deeper insight about the minimum amount of data required in order to obtain a certain level of generalisation using Gaussian processes for regression. The analysis has also explained the initial linear behaviour of the curves as well as the approach of the upper bounds to asymptotic plateaux.

In this thesis we have also shown how to discover hidden structure in the data using Gaussian process regression. We have seen that a new parametrisation of the matrix estimating the distance between the input variables can lead to a significant improvement of the generalisation error of the model with respect to the prediction made by a conventional approach. This new parametrisation allows the discovery of the actual set of features hidden among the input variables, estimating the relevance of each hidden feature in training the model as well as the mapping between the manifest and the hidden feature spaces.

We have also presented an empirical assessment of two Bayesian algorithms training neural networks in labelling segmented outdoor scene images. According to the analysis carried out, the comparison of the learning curves of the neural network has revealed the superiority of the Markov Chain Monte Carlo implementation with respect to the evidence framework on the task at hand; however the differences in generalisation performance depend upon the amount of training data, being below 7% for large datasets and increasing up to 35% for small datasets. The comparison has also been investigated by using the Automatic Relevance Determination implemented by the two algorithms; the analysis has revealed that the Markov Chain Monte Carlo method and the evidence framework have assigned different relevance to the input features in training the neural networks on segmented image classification, highlighting the practical use of Automatic Relevance Determination on a real-world problem.

Appendix A

Mathematical derivations

A.1 The eigenvalues of the covariance functions

In this appendix we present the methods we have used for evaluating the eigenvalues of the covariance functions MB_1, MB_2, MB_3 and SE .

The eigenvalue η_k of a stationary covariance function $C_p(t-x)$ is a scalar such that

$$\int_{-\infty}^{\infty} C_p(t-x)\,\varphi_k(x)\,p(x)\,dx = \eta_k \varphi_k(t), \tag{A.1}$$

where $\varphi_k(x)$ is the eigenfunction of $C_p(\cdot)$ and $p(x)$ is the probability density of the input variable x; in our study we considered $p(x)$ as the uniform distribution over the interval $[0,1]$.

To begin with, let us consider the modified Bessel covariance function of order 1

$$C_p(x-t) = \exp\left[-\frac{|x-t|}{\lambda}\right] \tag{A.2}$$

(cf. Equation 2.5). Hawkins (1989) found the solution of Equation A.1 for the covariance function in Equation A.2 showing that the set of eigenfunctions $\varphi_k(x)$ satisfies the second order differential equation

$$\lambda^2 \frac{d^2\varphi_k(x)}{dx^2} - \left(1 - \frac{2\lambda}{\eta_k}\right)\varphi(x) = 0 \tag{A.3}$$

whose general solution is $\varphi_k(x) = a_{1,k}\cos\omega_k x + a_{2,k}\sin\omega_k x$. The values of the constant $a_{1,k}$, $a_{2,k}$ and the frequency ω_k can be obtained by imposing boundary conditions on $\varphi_k(x)$; in particular it turns out that the frequencies ω_k belong to the intervals $[2(k-1)\pi, 2k\pi]$, $k \in \mathbb{N}$ and satisfy the transcendental equation

$$\tan[\omega_k] = \frac{2\omega_k}{(\omega_k^2 - 1)}. \tag{A.4}$$

The numerical solutions of Equation A.4 enable us to estimate the set of frequencies ω_k of the eigenfunction $\varphi_k(x)$. Plugging the particular solution $\varphi_k(x)$ in Equation A.3, it turns out that we can evaluate the spectrum of eigenvalues of the MB_1 covariance function according to the formula

$$\eta_k = \frac{2\lambda}{(1+\omega_k^2\lambda^2)}. \tag{A.5}$$

We have determined the eigenfunctions and eigenvalues of the MB_2 covariance function as follows. Considering the actual expression of MB_2 (cf. Equation 2.6) as

$$C(x-t) = \left(1+\frac{|x-t|}{\lambda}\right)\exp\left[-\frac{|x-t|}{\lambda}\right],$$

we can rewrite Equation A.1 as

$$\eta_k\varphi_k(t) = \int_{-\infty}^{\infty}\left(1+\frac{|x-t|}{\lambda}\right)\exp\left[-\frac{|x-t|}{\lambda}\right]\varphi_k(x)\,p(x)\,dx \tag{A.6}$$

and calculate the first four derivatives of $\varphi_k(t)$, obtaining the following set of equations:

$$\frac{d\varphi_k(t)}{dt} = \frac{1}{\eta_k\lambda}\int_{-\infty}^{\infty}\left(\frac{x-t}{\lambda}\right)\exp\left[-\frac{|x-t|}{\lambda}\right]\varphi_k(x)\,p(x)\,dx \tag{A.7}$$

$$\frac{d^2\varphi_k(t)}{dt^2} = \frac{1}{\eta_k\lambda^2}\int_{-\infty}^{\infty}\left(\frac{|x-t|}{\lambda}-1\right)\exp\left[-\frac{|x-t|}{\lambda}\right]\varphi_k(x)\,p(x)\,dx \tag{A.8}$$

$$\frac{d^3\varphi_k(t)}{dt^3} = \frac{1}{\eta_k\lambda^3}\int_{-\infty}^{\infty}\left(\frac{x-t}{\lambda}-2\mathrm{sgn}(x-t)\right)\exp\left[-\frac{|x-t|}{\lambda}\right]\varphi_k(x)\,p(x)\,dx \tag{A.9}$$

$$\frac{d^4\varphi_k(t)}{dt^4} = \frac{1}{\eta_k\lambda^4}\int_{-\infty}^{\infty}\left(\frac{|x-t|}{\lambda}-3+4\lambda\delta(x-t)\right)\exp\left[-\frac{|x-t|}{\lambda}\right]\varphi_k(x)\,p(x)\,dx. \tag{A.10}$$

Equation A.10 has been obtained from Equation A.9 using the relation

$$\frac{d\mathrm{sgn}(x-t)}{dt} = -2\delta(x-t),$$

where $\mathrm{sgn}(\cdot)$ is the sign function and $\delta(\cdot)$ is the Dirac delta function. A linear combination of Equations A.6, A.8 and A.10 shows that the set of eigenfunctions must fulfil the following 4^{th} order differential equation

$$\lambda^4\frac{d^4\varphi_k(t)}{dt^4} - 2\lambda^2\frac{d^2\varphi_k(t)}{dt^2} + \left(1-\frac{4\lambda}{\eta_k}\right)\varphi_k(t) = 0. \tag{A.11}$$

Searching for solution of the form $\exp[\xi t]$, we obtain the characteristic equation of A.11

$$\lambda^4\xi^4 - 2\lambda^2\xi^2 + \left(1-\frac{4\lambda}{\eta_k}\right) = 0, \tag{A.12}$$

whose solutions are

$$\xi_{1,2} = \pm i\omega = \pm\frac{i}{\lambda}\sqrt{2\sqrt{\frac{\lambda}{\eta_k}}-1} \text{ and } \xi_{3,4} = \pm\kappa = \pm\frac{1}{\lambda}\sqrt{2\sqrt{\frac{\lambda}{\eta_k}}+1}. \tag{A.13}$$

From a linear combination of the solutions $\exp[\xi_j x], j = 1,\ldots 4$, it turns out that the general solution of Equation A.11 is the function

$$\varphi_k(t) = a_{1,k}\sin(\omega_k t) + a_{2,k}\cos(\omega_k t) + a_{3,k}\sinh(\kappa_k t) + a_{4,k}\cosh(\kappa_k t), \tag{A.14}$$

where $\omega_k, \kappa_k \in \mathbb{R}$ are the frequencies of the eigenfunctions. It is straightforward to verify that Equation A.14 satisfies Equation A.11 when ω_k and κ_k are given by Equation A.13.

A particular solution of Equation A.11 can be obtained from Equation A.14 determining the value of the six constants $a_{1,k}$, $a_{2,k}$, $a_{3,k}$, $a_{4,k}$, ω_k and $\kappa_k, \forall k \in \mathbb{N}$. This can be done by imposing the following six conditions on the eigenfunctions $\varphi_k(t)$.

i First condition is the normalisation of the eigenfunctions:

$$\int_0^1 (\varphi_k(x))^2 dx = 1, \tag{A.15}$$

which allows to expand a stochastic process in a linear combination of the eigenfunctions $\varphi_k(x)$, i.e. $y(x) = \sum_k c_k \sqrt{\eta_k} \varphi_k(x)$, where $c_k = \int y(x) \varphi_k(x) p(x) dx$.

ii From Equation A.13 it is also straightforward to verify that the set of frequencies must fulfil the condition

$$\kappa_k^2 - \omega_k^2 = \frac{2}{\lambda^2}. \tag{A.16}$$

iii-vi Four more constraints follow by imposing boundary conditions on the derivatives of $\varphi_k(t)$ (cf. Equations A.7 - A.10). Recalling that in our study we restricted the input space to the interval $[0, 1]$, from the evaluation of Equations A.7 - A.10 and Equation A.6 on the boundary of the interval, the following set of conditions must be satisfied:

$$\left[\varphi_k(t) - 2\lambda \frac{d\varphi_k(t)}{dt} + \lambda^2 \frac{d^2\varphi_k(t)}{dt^2}\right]_{t=0} = 0 \tag{A.17}$$

$$\left[\varphi_k(t) + 2\lambda \frac{d\varphi_k(t)}{dt} + \lambda^2 \frac{d^2\varphi_k(t)}{dt^2}\right]_{t=1} = 0 \tag{A.18}$$

$$\left[\varphi_k(t) - \lambda \frac{d\varphi_k(t)}{dt} - \lambda^2 \frac{d^2\varphi_k(t)}{dt^2} + \lambda^3 \frac{d^3\varphi_k(t)}{dt^3}\right]_{t=0} = 0 \tag{A.19}$$

$$\left[\varphi_k(t) + \lambda \frac{d\varphi_k(t)}{dt} - \lambda^2 \frac{d^2\varphi_k(t)}{dt^2} - \lambda^3 \frac{d^3\varphi_k(t)}{dt^3}\right]_{t=1} = 0. \tag{A.20}$$

The set of Equations A.17 - A.20, together with the conditions on the normalisation of the eigenfunctions and the frequencies (cf. Equations A.15 and A.16, respectively) are functions of the six constants $a_{1,k}$, $a_{2,k}$, $a_{3,k}$, $a_{4,k}$, ω_k and κ_k characterising the general solution of the differential equation A.14. The numerical solution of the transcendental system composed by those six relations allows to determine the value of the six constants of Equation A.14 obtaining the particular solution of the differential equation A.11.

Finally plugging the numerical values of the six constants in Equation A.14, we obtain the spectrum of eigenvalues from Equation A.11 as

$$\eta_k = \frac{4\lambda}{\left(1 + \lambda^2 \omega_k^2\right)^2}, \tag{A.21}$$

where the frequencies belong to the interval $\omega_k \in [2(k-1)\pi, 2k\pi], k \in \mathbb{N}$.

The result of Equation A.5 and Equation A.21 can also be derived by carrying out the analysis in the Fourier space.

In this thesis we have also used covariance functions of smoother processes (described by the MB_3 and SE covariance functions) whose eigenfunctions are the solutions of a differential equation order higher than 4. This increases the complexity of the solution of the differential equation; for instance, the particular solution of a 6^{th} order differential equation (characteristics of MB_3 covariance function), is made up by nine constants whose values can be calculated by imposing twelve conditions on the eigenfunctions.

We preferred to evaluate the set of eigenvalues numerically, being also aware of the fact that the power spectrum of covariance functions describing smooth processes decays to zero quicker than that characteristic of a rough process (given a process satisfying the Sacks-Ylvisaker conditions of order s (cf. Section 2.5), the asymptotic behaviour of the eigenvalues of its covariance function decreases asymptotically as $\eta_k \propto (\pi k)^{-2(s+1)}$ (Ritter, 1996), where for the modified Bessel covariance function $s = r - 1$). The use of this property of the spectrum allows a significant reduction of numerical inaccuracies in the estimation of the eigenvalues of smooth processes for both the MB_3 and the SE covariance functions.

A.2 The use of an incorrect covariance matrix

In this appendix we show that the use of an incorrect covariance matrix increases the generalisation error. To begin with, we recall the definitions of the generalisation error of Equation 2.8 on a test point $(\mathbf{x}, t)$

$$E_c^g(\mathbf{x}, t) = \left(t - \mathbf{k}_c^{\mathrm{T}}(\mathbf{x}) K_c^{-1}\mathbf{t}\right)^2 \text{ and } E_i^g(\mathbf{x}, t) = \left(t - \mathbf{k}_i^{\mathrm{T}}(\mathbf{x}) K_i^{-1}\mathbf{t}\right)^2,$$

where the subscripts c and i label the matrix K and the vector $\mathbf{k}(\mathbf{x})$ computed with the correct and incorrect parameters of the covariance function, respectively.

Denoting with $\Delta E^g(\mathbf{x}, t)$ the error induced by the use of the incorrect covariance matrix with respect to the proper one, i.e.

$$\Delta E^g(\mathbf{x}, t) = E_i^g(\mathbf{x}, t) - E_c^g(\mathbf{x}, t),$$

we need to prove that the Bayesian expectation of $\Delta E^g(\mathbf{x}, t)$ (i.e. the expectation over the distribution of the targets t) is greater than 0. From the definition of the generalisation error follows that

$$\begin{aligned} \mathcal{E}\left[E_c^g(\mathbf{x}, t)\right] &= C(0) - \mathbf{k}_c^{\mathrm{T}} K_c^{-1}\mathbf{k}_c(\mathbf{x}) \text{ and} \\ \mathcal{E}\left[E_i^g(\mathbf{x}, t)\right] &= C(0) - 2\mathbf{k}_c^{\mathrm{T}}(\mathbf{x}) K_i^{-1}\mathbf{k}_i(\mathbf{x}) + \mathbf{k}_i^{\mathrm{T}}(\mathbf{x}) K_i^{-1} K_c K_i^{-1}\mathbf{k}_i(\mathbf{x}), \end{aligned}$$

where we used the relations $\mathcal{E}\left[\mathbf{t}\mathbf{t}^{\mathrm{T}}\right] = K_c$ and $\mathcal{E}[t\mathbf{t}] = \mathbf{k}_c(\mathbf{x})$. Hence the expected difference

between the two generalisation errors can be written as

$$\begin{aligned}\mathcal{E}\left[\Delta E^g(\mathbf{x},t)\right] &= \mathcal{E}\left[E_i^g(\mathbf{x},t)\right] - \mathcal{E}\left[E_c^g(\mathbf{x},t)\right] \\ &= \mathbf{k}_i^{\mathrm{T}}(\mathbf{x}) K_i^{-1} K_c K_i^{-1} \mathbf{k}_i(\mathbf{x}) - 2\mathbf{k}_c^{\mathrm{T}}(\mathbf{x}) K_i^{-1}\mathbf{k}_i(\mathbf{x}) + \mathbf{k}_c^{\mathrm{T}} K_c^{-1}\mathbf{k}_c(\mathbf{x}) \quad \text{(A.22)} \\ &= \left(K_i^{-1}\mathbf{k}_i(\mathbf{x}) - K_c^{-1}\mathbf{k}_c(\mathbf{x})\right)^{\mathrm{T}} K_c \left(K_i^{-1}\mathbf{k}_i(\mathbf{x}) - K_c^{-1}\mathbf{k}_c(\mathbf{x})\right). \quad \text{(A.23)}\end{aligned}$$

Equation A.23 follows from the property of symmetry of the covariance matrices and rewriting the second term in Equation A.22 as $\mathbf{k}_c^{\mathrm{T}}(\mathbf{x}) K_i^{-1}\mathbf{k}_i(\mathbf{x}) = \mathbf{k}_c^{\mathrm{T}}(\mathbf{x}) K_c^{-1} K_c K_i^{-1}\mathbf{k}_i(\mathbf{x})$. Since the covariance matrix K_c is positive semi-definite ($\mathbf{v}^{\mathrm{T}} K_c \mathbf{v} \geq 0, \forall \mathbf{v} \in \mathbb{R}^n$), Equation A.23 shows that $\mathcal{E}\left[\Delta E^g(\mathbf{x},t)\right] \geq 0$ proving the thesis.

A.3 The variance of the Bayesian generalisation error

In Section 2.4 the Bayesian generalisation error (i.e. the expectation of the generalisation error $E^g_{\mathcal{D}_n}(\mathbf{x},t)$ over the actual distribution of the stochastic process $t(\mathbf{x})$) has been calculated as

$$\mathcal{E}\left[E^g_{\mathcal{D}_n}(\mathbf{x},t)\right] = C(\mathbf{x},\mathbf{x}) - \mathbf{k}^{\mathrm{T}}(\mathbf{x}) K^{-1}\mathbf{k}(\mathbf{x}) \quad \text{(A.24)}$$

(cf. Equation 2.10), where $E^g_{\mathcal{D}_n}(\mathbf{x},t) = \left(t - \mathbf{k}^{\mathrm{T}}(\mathbf{x}) K^{-1}\mathbf{t}\right)^2$; in this Section we derive the variance of the generalisation error, which is defined as

$$\begin{aligned}\mathcal{V}\left[E^g_{\mathcal{D}_n}(\mathbf{x},t)\right] &= \mathcal{E}\left[\left(E^g_{\mathcal{D}_n}(\mathbf{x},t) - \mathcal{E}\left[E^g_{\mathcal{D}_n}(\mathbf{x},t)\right]\right)^2\right] \\ &= \mathcal{E}\left[\left(E^g_{\mathcal{D}_n}(\mathbf{x},t)\right)^2\right] - \left(\mathcal{E}\left[E^g_{\mathcal{D}_n}(\mathbf{x},t)\right]\right)^2. \quad \text{(A.25)}\end{aligned}$$

Expanding the term $\mathcal{E}\left[\left(E^g_{\mathcal{D}_n}(\mathbf{x},t)\right)^2\right]$, it turns out that

$$\begin{aligned}\mathcal{E}\left[\left(E^g_{\mathcal{D}_n}(\mathbf{x},t)\right)^2\right] &= \mathcal{E}\left[\left(t - \mathbf{k}^{\mathrm{T}}(\mathbf{x}) K^{-1}\mathbf{t}\right)^4\right] \\ &= \mathcal{E}\left[t^4\right] + 6\left(\mathbf{k}^{\mathrm{T}}(\mathbf{x}) K^{-1}\mathbf{t}\right)^2 t^2 - 4\mathcal{E}\left[\left(\mathbf{k}^{\mathrm{T}}(\mathbf{x}) K^{-1}\mathbf{t}\right) t^3\right] - \\ &\quad 4\mathcal{E}\left[\left(\mathbf{k}^{\mathrm{T}}(\mathbf{x}) K^{-1}\mathbf{t}\right)^3 t\right] + \mathcal{E}\left[\left(\mathbf{k}^{\mathrm{T}}(\mathbf{x}) K^{-1}\mathbf{t}\right)^4\right]. \quad \text{(A.26)}\end{aligned}$$

Assuming that the true process has a Gaussian joint probability distribution with zero mean, each expectation in the right hand side of Equation A.26 is the moment of order 4 of the distribution. The general expression of the moment of 4^{th} order is (Iranpour and Chacon, 1986)

$$\begin{aligned}\mathcal{E}\left[t^i t^j t^l t^m\right] &= \mathcal{E}\left[t^i t^j\right]\mathcal{E}\left[t^l t^m\right] + \mathcal{E}\left[t^i t^l\right]\mathcal{E}\left[t^j t^m\right] + \mathcal{E}\left[t^i t^m\right]\mathcal{E}\left[t^j t^l\right] \\ &= C\left(\mathbf{x}^i,\mathbf{x}^j\right) C\left(\mathbf{x}^l,\mathbf{x}^m\right) + C\left(\mathbf{x}^i,\mathbf{x}^l\right) C\left(\mathbf{x}^j,\mathbf{x}^m\right) + C\left(\mathbf{x}^i,\mathbf{x}^m\right) C\left(\mathbf{x}^j,\mathbf{x}^l\right) \quad \text{(A.27)}\end{aligned}$$

where the t's indicate the elements of the $(n+1)$-dimensional vector $\left(\mathbf{t}^{\mathrm{T}}, t\right)$; by applying Equation A.27, all of the contributions in Equation A.26 can be evaluated. The first term contributing to Equation A.26 can be calculated as

$$\mathcal{E}\left[t^4\right] = 3C^2(\mathbf{x},\mathbf{x}). \quad \text{(A.28)}$$

The second term in Equation A.26 can be evaluated as

$$\begin{aligned}
\mathcal{E}\left[\left(\mathbf{k}^{\mathrm{T}}(\mathbf{x})K^{-1}\mathbf{t}\right)^2 t^2\right] &= \mathcal{E}\left[\mathrm{Tr}\left[\mathbf{k}^{\mathrm{T}}(\mathbf{x})K^{-1}\mathbf{t}\mathbf{t}^{\mathrm{T}}K^{-1}\mathbf{k}(\mathbf{x})t^2\right]\right] \\
&= \sum_{i,j=1}^{n}\left(\mathbf{k}^{\mathrm{T}}(\mathbf{x})K^{-1}\right)_i\left(K^{-1}\mathbf{k}(\mathbf{x})\right)_j\mathcal{E}\left[t^i t^j t^2\right] \\
&= \sum_{i,j=1}^{n}\left(\mathbf{k}^{\mathrm{T}}(\mathbf{x})K^{-1}\right)_i\left(K^{-1}\mathbf{k}(\mathbf{x})\right)_j \times \\
&\qquad \left(C(\mathbf{x},\mathbf{x})C\left(\mathbf{x}^i,\mathbf{x}^j\right)+2C\left(\mathbf{x},\mathbf{x}^i\right)C\left(\mathbf{x},\mathbf{x}^j\right)\right) \\
&= C(\mathbf{x},\mathbf{x})\mathbf{k}^{\mathrm{T}}(\mathbf{x})K^{-1}KK^{-1}\mathbf{k}(\mathbf{x})+2\left(\mathbf{k}^{\mathrm{T}}(\mathbf{x})K^{-1}\mathbf{k}(\mathbf{x})\right)^2 \\
&= C(\mathbf{x},\mathbf{x})\mathbf{k}^{\mathrm{T}}(\mathbf{x})K^{-1}\mathbf{k}(\mathbf{x})+2\left(\mathbf{k}^{\mathrm{T}}(\mathbf{x})K^{-1}\mathbf{k}(\mathbf{x})\right)^2, \qquad \text{(A.29)}
\end{aligned}$$

where we used the definitions $(K)_{ij}=C\left(\mathbf{x}^i,\mathbf{x}^j\right)$ and $(\mathbf{k}(\mathbf{x}))_i=C\left(\mathbf{x},\mathbf{x}^i\right)$.

The other terms in Equation A.26 can be calculated by applying algebraic techniques similar to those used to evaluate $\mathcal{E}\left[\left(\mathbf{k}^{\mathrm{T}}(\mathbf{x})K^{-1}\mathbf{t}\right)^2 t^2\right]$. It turns out that the following Equations hold:

$$\mathcal{E}\left[\left(\mathbf{k}^{\mathrm{T}}(\mathbf{x})K^{-1}\mathbf{t}\right)^4\right]=3\left(\mathbf{k}^{\mathrm{T}}(\mathbf{x})K^{-1}\mathbf{k}(\mathbf{x})\right)^2 \qquad \text{(A.30)}$$

$$\mathcal{E}\left[\mathbf{k}^{\mathrm{T}}(\mathbf{x})K^{-1}\mathbf{t}t^3\right]=3C(\mathbf{x},\mathbf{x})\mathbf{k}^{\mathrm{T}}(\mathbf{x})K^{-1}\mathbf{k}(\mathbf{x}) \qquad \text{(A.31)}$$

$$\mathcal{E}\left[\left(\mathbf{k}^{\mathrm{T}}(\mathbf{x})K^{-1}\mathbf{t}\right)^3 t\right]=3\left(\mathbf{k}^{\mathrm{T}}(\mathbf{x})K^{-1}\mathbf{k}(\mathbf{x})\right)^2. \qquad \text{(A.32)}$$

Plugging Equations A.28 - A.32 in Equation A.26, follows that

$$\begin{aligned}
\mathcal{E}\left[\left(E^g_{\mathcal{D}_n}(\mathbf{x},t)\right)^2\right] &= 3C^2(\mathbf{x},\mathbf{x})-12C(\mathbf{x},\mathbf{x})\mathbf{k}^{\mathrm{T}}(\mathbf{x})K^{-1}\mathbf{k}(\mathbf{x})+6C(\mathbf{x},\mathbf{x})\mathbf{k}^{\mathrm{T}}(\mathbf{x})K^{-1}\mathbf{k}(\mathbf{x})+ \\
&\quad 12\left(\mathbf{k}^{\mathrm{T}}(\mathbf{x})K^{-1}\mathbf{k}(\mathbf{x})\right)^2-12\left(\mathbf{k}^{\mathrm{T}}(\mathbf{x})K^{-1}\mathbf{k}(\mathbf{x})\right)^2+3\left(\mathbf{k}^{\mathrm{T}}(\mathbf{x})K^{-1}\mathbf{k}(\mathbf{x})\right)^2 \\
&= 3\left(C(\mathbf{x},\mathbf{x})-\mathbf{k}^{\mathrm{T}}(\mathbf{x})K^{-1}\mathbf{t}\right)^2=3\left(\mathcal{E}\left[E^g_{\mathcal{D}_n}(\mathbf{x},t)\right]\right)^2. \qquad \text{(A.33)}
\end{aligned}$$

Inserting Equation A.33 in Equation A.25, we can evaluate the variance of the generalisation error with the formula

$$\mathcal{V}\left[E^g_{\mathcal{D}_n}(\mathbf{x},t)\right]=3\left(\mathcal{E}\left[E^g_{\mathcal{D}_n}(\mathbf{x},t)\right]\right)^2-\left(\mathcal{E}\left[E^g_{\mathcal{D}_n}(\mathbf{x},t)\right]\right)^2=2\left(\mathcal{E}\left[E^g_{\mathcal{D}_n}(\mathbf{x},t)\right]\right)^2.$$

A.4 Calculation of the upper bound $E_2^u(n)$

In Section 2.5.2, the bound generated by the variance $\sigma_2^2(x)$ in the interval $\left[x^i,x^{i+1}\right]$, has been written as

$$E^u_{2,\mathcal{D}_n}(\omega)=C(0)\omega-\frac{2\left(I_1(\omega)-I_2(\omega)\right)}{\Delta(\omega)}, \qquad \text{(A.34)}$$

where $\omega=x^{i+1}-x^i$. The integrals $I_1(\omega)$ and $I_2(\omega)$ are defined as

$$I_1(\omega) = C(0)\int_0^{\omega}C_p^2(\xi)\,d\xi \qquad \text{(A.35)}$$

$$I_2(\omega) = C_p(\omega)\int_0^{\omega}C_p(\xi)C_p(\omega-\xi)\,d\xi, \qquad \text{(A.36)}$$

where ξ is the distance between the test point x from x^i and $\Delta(\omega) = (C(0))^2 - (C_p(x^{i+1} - x^i))^2$ (cf. Section 2.5.2). The analytical evaluation of the integrals in Equations A.35 and A.36 for the MB and the SE covariance functions are presented in the following Sections A.4.1 and A.4.2, respectively.

A.4.1 The Modified Bessel covariance functions

The Modified Bessel covariance functions MB_r (see Equation 2.4) can be rewritten in term of distances between training points as

$$\begin{aligned} C_p(\xi) &= \kappa_\nu \left(\frac{\xi}{\lambda}\right)^\nu \mathcal{K}_\nu\left(\frac{\xi}{\lambda}\right) \\ &= \kappa_\nu \sum_{k=0}^{r-1} a_k \left(\frac{\xi}{\lambda}\right)^k \exp\left[-\frac{\xi}{\lambda}\right], \end{aligned} \tag{A.37}$$

where $\nu = r - 1/2$ is the order of the Bessel function $\mathcal{K}_\nu(\cdot)$ and r is integer; the factors a_k are constants depending upon the value of r.

Plugging Equation A.37 in Equation A.35, the integral $I_1(\omega)$ can be calculated as

$$I_1(\omega) = C(0)\,\kappa_\nu^2 \sum_{k,l=1}^{r-1} \frac{a_k a_l}{\lambda^{k+l}} \int_0^\omega \xi^{k+l} \exp\left[-\frac{2\xi}{\lambda}\right] d\xi. \tag{A.38}$$

Since

$$\int_0^\omega \xi^{k+l} \exp\left[-\frac{2\xi}{\lambda}\right] d\xi = (k+l)! \left(\frac{\lambda}{2}\right)^{k+l+1} \left[1 - \exp\left[-\frac{2\omega}{\lambda}\right] \sum_{p=0}^{k+l} \frac{1}{p!} \left(\frac{2\omega}{\lambda}\right)^p\right]$$

(see e.g. Equation 2.3212 in Gradshteyn and Ryzhik, 1993), Equation A.38 can be written as

$$I_1(\omega) = C(0)\,\kappa_\nu^2 \sum_{k,l=0}^{r-1} \frac{a_k a_l \lambda\,(k+l)!}{2^{k+l+1}} \left[1 - \exp\left[-\frac{2\omega}{\lambda}\right] \sum_{p=0}^{k+l} \frac{1}{p!} \left(\frac{2\omega}{\lambda}\right)^p\right]. \tag{A.39}$$

For $I_2(\omega)$, the term $C_p(\xi)\,C_p(\omega - \xi)$ in Equation A.36 can be expressed as

$$C_p(\xi)\,C_p(\omega - \xi) = \kappa_\nu^2 \sum_{k,l=1}^{r-1} \frac{a_k a_l}{\lambda^{k+l}} \xi^k (\omega - \xi)^l \exp\left[-\frac{\xi}{\lambda}\right] \exp\left[-\frac{\omega - \xi}{\lambda}\right].$$

Since $\exp[-\xi/\lambda] \exp[-(\omega - \xi)/\lambda] = \exp[-\omega/\lambda]$, expanding $(\omega - \xi)^l$ in the binomial series, $I_2(\omega)$ can be rewritten as

$$\begin{aligned} I_2(\omega) &= C_p(\omega)\,\kappa_\nu^2 \sum_{k,l=1}^{r-1} \frac{a_k a_l}{\lambda^{k+l}} \exp\left[-\frac{\omega}{\lambda}\right] \sum_{p=0}^{l} \frac{l!\,\omega^{l-p}}{p!\,(l-p)!} (-1)^p \int_0^\omega \xi^{k+p} d\xi \\ &= C_p(\omega)\,\kappa_\nu^2 \sum_{k,l=1}^{r-1} a_k a_l \lambda \exp\left[-\frac{\omega}{\lambda}\right] \left(\frac{\omega}{\lambda}\right)^{k+l+1} \sum_{p=0}^{l} \frac{l!}{p!\,(l-p)!} \frac{(-1)^p}{k+p+1}. \end{aligned} \tag{A.40}$$

Inserting Equations A.39 and A.40 in Equation A.34 we have an estimate of an upper bound which is dependent on the training set. The expectation with respect to the training set is calculated integrating over the distribution of the distances ω (where $p(\omega) = n(1-\omega)^{n-1}$ (David, 1970)). Although the integrations are feasible, their final expressions contain hyper-geometric functions whose computation is time consuming; for this reason we preferred to evaluate numerically the integrations over ω.

A.4.2 The Squared Exponential

As the SE covariance function (expressed in terms of the distance ξ) is (cf. Equation 2.3)

$$C_p(\xi) = \kappa_\nu \exp\left[-\frac{\xi^2}{2\lambda^2}\right], \tag{A.41}$$

$I_1(\omega)$ can be calculated as

$$I_1(\omega) = C(0)\,\kappa_\nu^2 \int_0^\omega \exp\left[-\frac{\xi^2}{\lambda^2}\right] d\xi = C(0)\,\kappa_\nu^2 \lambda \frac{\sqrt{\pi}}{2} \operatorname{erf}\left[\frac{\omega}{\lambda}\right],$$

where the error function is defined as (e.g. Equation 8.250 in Gradshteyn and Ryzhik, 1993)

$$\operatorname{erf}\left[\frac{\omega}{\lambda}\right] = \frac{2}{\sqrt{\pi}} \int_0^{\frac{\omega}{\lambda}} \exp\left[-\xi^2\right] d\xi. \tag{A.42}$$

For carrying out the calculations of $I_2(\omega)$, we consider

$$\begin{aligned} C_p(\xi)\,C_p(\omega-\xi) &= \kappa_\nu^2 \exp\left[-\frac{\xi^2 + (\omega-\xi)^2}{2\lambda^2}\right] \\ &= \kappa_\nu^2 \exp\left[-\frac{\omega^2}{4\lambda^2}\right] \exp\left[-\frac{(\xi-\omega/2)^2}{\lambda^2}\right] \end{aligned}$$

After changing the variable of integration ξ to $\tau = (\xi - \omega/2)/\lambda$ (with $d\xi = \lambda d\tau$) $I_2(\omega)$ can be evaluated as

$$\begin{aligned} I_2(\omega) &= C_p(\omega)\,\kappa_\nu^2 \lambda \exp\left[-\frac{\omega^2}{4\lambda^2}\right] \int_{-\omega/2\lambda}^{\omega/2\lambda} \exp\left[-\tau^2\right] d\tau \\ &= C_p(\omega)\,\kappa_\nu^2 \lambda \sqrt{\pi} \exp\left[-\frac{\omega^2}{4\lambda^2}\right] \operatorname{erf}\left[\frac{\omega}{2\lambda}\right]. \end{aligned} \tag{A.43}$$

Plugging Equations A.42 and A.43 in Equation A.34 we obtain an expression of the bound depending on the location of the datapoints whose numerical integration gives an estimate of $E_2^u(n)$ for the SE covariance function.

A.5 Asymptotics of the upper bounds

In this section we present the asymptotic expansion of the upper bounds $E_1^u(n)$ and $E_2^u(n)$ in terms of λ and σ_ν^2 in the limit of large amount of training data.

According to Equation 2.20, $E_1^u(n)$ is

$$E_1^u(n) = C(0) - \frac{1}{C(0)} \left[(n-1) \int_0^1 C_p^2\left(\frac{\omega}{2}\right)(1-\omega)^n\, d\omega + 2\int_0^1 C_p^2(\omega)\,(1-\omega)^n\, d\omega\right]. \tag{A.44}$$

For large n, the major contribution to the integral in Equation A.44 is given by *small* ω because the terms $(1-\omega)^n$ decays quickly to zero as n increases (Figure A.1). Therefore it is possible to expand the functions $C_p^2(\omega)$ and $C_p^2(\omega/2)$ in polynomial forms for $\omega \ll 1$ and to compute the

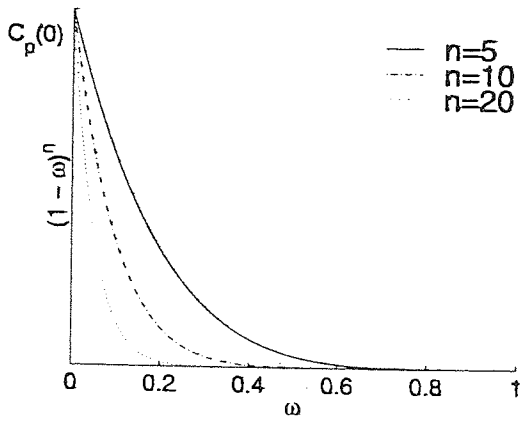


Figure A.1: The Figure shows the graphs of the function $(1-\omega)^n$ for $n = 5$ (solid line), $n = 10$ (dash-dotted line) and $n = 20$ (dotted line).

integrals in Equation A.44 with such approximations. In particular, the Taylor expansion of the MB_1 covariance function for $\omega \ll 1$ is

$$C_p^2\left(\frac{\omega}{a}\right) \sim C_p^2(0)\left(1 - \frac{2\omega}{a\lambda} + O\left(\frac{\omega^2}{a^2\lambda^2}\right)\right), \tag{A.45}$$

where $a = 1$ or $a = 2$ according to the arguments ω or $\omega/2$ of $C_p^2(\cdot)$ (cf. Equation A.44). Over the interval $\omega \in [0, a\lambda]$ the terms $C_p^2(\cdot)$ in the integrands of Equation A.44 can be approximated by Equation A.45 for large n; when $\omega > a\lambda$ the integrands are negligible and their contribution to the integrals are effectively zero. Hence one can approximate the integrals of Equation A.44 as

$$\begin{aligned}\frac{1}{C(0)}\int_0^1 C_p^2\left(\frac{\omega}{a}\right)(1-\omega)^n\,d\omega &\sim \frac{C_p^2(0)}{C(0)}\int_0^{a\lambda}\left(1-\frac{2\omega}{a\lambda}\right)(1-\omega)^n\,d\omega \\ &\sim \frac{C(0)\,r^2\,(a\lambda(n+2)-2)}{a\lambda(n+1)(n+2)} + O\left(\exp\left[-n\lambda\right]\right), \end{aligned} \tag{A.46}$$

where $r = C_p(0)/C(0)$. Plugging Equation A.46 in place of the integrals in Equation A.44 with the proper values of a, we obtain that an approximation of $E_1^u(n)$ for large n is

$$E_1^u(n) \sim C(0) - C(0)\,r^2\left[\frac{(n-1)(2\lambda(n+2)-2)}{2\lambda(n+1)(n+2)} + \frac{2(\lambda(n+2)-2)}{\lambda(n+1)(n+2)} + O\left(\exp\left[-n\lambda\right]\right)\right].$$

After some simplifications, we have that the asymptotic expansion of the one-point upper bound is

$$E_1^u(n) \sim C(0)\left(1-r^2\right) + \frac{C(0)\,r^2}{\lambda(n+1)} + O\left(n^{-2}\right). \tag{A.47}$$

A similar arguments applies for the covariance functions MB_2, MB_3 and SE . In particular the integrands in Equation A.44 can be expanded as

$$C_p^2\left(\frac{\omega}{a}\right) \sim C_p^2(0)\left(1 - \frac{\omega^2}{ba^2\lambda^2} + O\left(\frac{\omega^4}{a^4\lambda^4}\right)\right)$$

where the expansion holds for $\omega \in \left[0, a\lambda\sqrt{b}\right]$; when $\omega > a\lambda\sqrt{b}$ the two integrands are negligible. The factor b comes from the Taylor expansion of the covariance functions MB_2, MB_3 and SE and

turns out to be either 1 (for MB_2 and SE) or 3 for MB_3. Evaluating the integrals of Equation A.44 with this expansion, we have

$$\frac{1}{C(0)}\int_0^1 C_p^2\left(\frac{\omega}{a}\right)(1-\omega)^n\,d\omega \sim \frac{C_p^2(0)}{C(0)}\int_0^{a\lambda\sqrt{b}}\left(1-\frac{\omega^2}{ba^2\lambda^2}\right)(1-\omega)^n\,d\omega$$
$$\sim \frac{C(0)\,r^2\left(ba^2\lambda^2(n+2)(n+3)-2\right)}{ba^2\lambda^2(n+1)(n+2)(n+3)} + O\left(\exp\left[-n\lambda\right]\right), \quad \text{(A.48)}$$

where $r = C_p(0)/C(0)$. Plugging Equation A.48 in Equation A.44, after some simplifications, we obtain that the one-point upper bound for the covariance functions MB_2, MB_3 and SE can be approximated for large n as

$$E_1^u(n) \sim C(0)\left(1-r^2\right) + \frac{C(0)\,r^2}{2b\lambda^2(n+1)(n+2)} + O\left(n^{-3}\right). \quad \text{(A.49)}$$

We note that the asymptotic value of $E_1^u(n)$ does not depend either on the lengthscale of the process or on the covariance function but is a function of the relative ratio r between the prior variance $C_p(0)$ and $C(0) = C_p(0) + \sigma_\nu^2$. This can be shown calculating the limit of Equations A.45 and A.49 for $n \to \infty$

$$\lim_{n\to\infty} E_1^u(n) = C(0)\left(1-r^2\right) = \sigma_\nu^2(1+r).$$

The second upper bound $E_2^u(n)$ can be written as (cf. Equation 2.24)

$$E_2^u(n) = C(0) - 2(n-1)\int_0^1 f(\omega)\,p(\omega)\,d\omega - \frac{2}{C(0)}\int_0^1 I(\omega)\,p(\omega)\,d\omega, \quad \text{(A.50)}$$

where $f(\omega) = (I_1(\omega) - I_2(\omega))/\Delta(\omega)$, $\Delta(\omega) = (C(0))^2 - \left(C_p\left(x^{i+1}-x^i\right)\right)^2$ (as it has been defined in Section A.4) and $I_1(\omega)$ and $I_2(\omega)$ are defined as in Equations A.35 and A.36, respectively. $I(\omega)$ is defined as (cf. Section 2.5.1)

$$I(\omega) = \int_0^\omega C_p^2(\xi)d\xi. \quad \text{(A.51)}$$

According to the theory of order statistics (David, 1970), the distribution of the distances between two training points is $p(\omega) = n(1-\omega)^{n-1}$.

Similarly to the previous upper bound, when $n \to \infty$ the integrals in Equation A.50 are dominated by small ω. This allows us to search for a Taylor expansion of the function $f(\omega)$ around 0 in terms of λ and σ_ν^2, as

$$f(\omega) \underset{\omega\ll 1}{\sim} f(0) + \omega\left.\frac{df(\omega)}{d\omega}\right|_{\omega=0} + \frac{\omega^2}{2}\left.\frac{d^2f(\omega)}{d\omega^2}\right|_{\omega=0} + O\left(\omega^3\right)$$
$$= \alpha\omega + \frac{\beta\omega^2}{2} + O\left(\omega^3\right), \quad \text{(A.52)}$$

where we used the equivalence $f(0) = 0$ and defined

$$\alpha \doteq \left.\frac{df(\omega)}{d\omega}\right|_{\omega=0} = \frac{C(0)\,r^2}{1+r} \text{ and } \beta \doteq \left.\frac{d^2f(\omega)}{d\omega^2}\right|_{\omega=0} = \frac{2r^2\gamma}{(1+r)^2}.$$

The term γ takes into account the first derivative of $C_p(\omega)$ evaluated in $\omega = 0$ divided by the prior covariance $C_p(0)$. It turns out that for the MB_2, MB_3 and SE covariance functions, $\gamma = 0$; γ is

not zero only for the MB_1 covariance function, for which

$$\gamma = \frac{\mathrm{I}}{C_p(0)} \left. \frac{dC_p(\omega)}{d\omega} \right|_{\omega=0} = -\lambda^{-1}.$$

The Taylor expansion of $f(\omega)$ allows to approximate the first integral of Equation A.50 as

$$\begin{aligned} \int_0^1 f(\omega) p(\omega) d\omega &\sim \int_0^1 \left(\alpha\omega + \frac{\beta\omega^2}{2} + O(\omega^3) \right) n(1-\omega)^{n-1} d\omega \\ &= -\left[(1-\omega)^n \left(\alpha\omega + \frac{\beta\omega^2}{2} + O(\omega^3) \right) \right]_0^1 + \int_0^1 (\alpha + \beta\omega + O(\omega^2)) (1-\omega)^n d\omega \\ &= \frac{1}{n+1} \left(\alpha + \frac{\beta}{n+2} \right) + O(n^{-3}). \end{aligned} \tag{A.53}$$

The expansion of $I(\omega)$ is similar to that presented for $E_1^u(n)$; in particular, the expansion of the term $I(\omega)$ in the limit of large n is (cf. Equation A.46 for MB_1 and Equation A.48 for MB_2, MB_3 and SE functions with $a = 1$) is

$$\frac{2}{C(0)} \int_0^1 I(\omega) p(\omega) d\omega \sim \frac{2C(0) r^2}{n+1} + O(n^{-2}). \tag{A.54}$$

Plugging Equations A.53 and A.54 in Equation A.50 we obtain the asymptotic expansion of the upper bound $E_2^u(n)$ as

$$E_2^u(n) \sim C(0) \left(1 - \frac{2r^2}{1+r} \right) + \frac{2C(0)}{n+1} \left(\frac{r(1-r)}{1+r} - \frac{2r^2\gamma}{(1+r)^2} \right) + O(n^{-2}). \tag{A.55}$$

Equation A.55 shows that $E_2^u(n)$ approaches the asymptotic plateau

$$\lim_{n\to\infty} E_2^u(n) = C(0) \left(1 - \frac{2r^2}{1+r} \right) = \sigma_\nu^2 \left(1 + \frac{r}{1+r} \right)$$

as $O(n^{-1})$.

A.6 Derivation of an alternative lower bound

In this appendix we present an alternative lower bound; it can be derived from the evaluation of the mutual entropy (Opper, 1997) between the modelled and the true distribution of the values of the stochastic process. Although we derive the result when input space is a subset of $\mathbb{R}$, the lower bound holds also for higher dimensions.

In section 2.2 we have seen how a GP models the posterior distribution of functions and estimates the mean and the variance of the posterior distribution (cf. Equations 2.1 and 2.2). In order to evaluate the correctness of the predictive distribution we compare the solution found by the GP model with the true distribution of the function. Since we need to contrast two quantities (the true and the predictive function), we recall that $\hat{y}(x)$ and $y(x)$ designate the GP prediction and the true underlying function, respectively.

A practical evaluation of the performance of the model is given by a measure of the discrepancy between $y(x)$ and $\hat{y}(x)$. This measure differs accordingly to the problem at hand. For regression

problem a suitable measure is the squared error $f_n(y,\hat{y}) = (y(x) - \hat{y}(x))^2$, where n is the number of training data. This quantity can be averaged over the estimated distribution $p(\hat{y}|y)$ obtaining $f_n(y) = \int f(y,\hat{y})\, p(\hat{y}|y)\, d\hat{y}$. We note that $p(\hat{y}|y)$ is a probability density defined over the $\infty-$ dimensional space of the functions. Although usually we don't know the distribution $p(\hat{y}|y)$, it is possible to find a bound on $f_n(y)$ (Opper, 1997), that is

$$f_n(y) \geq -\frac{\sigma_\nu^2}{2}\frac{\partial}{\partial n}\log\left[\int \exp\left[-\frac{n}{\sigma_\nu^2}\int f_n(y,\hat{y})\, p(x)\, dx\right] p(\hat{y})\, d\hat{y}\right]. \tag{A.56}$$

Averaging equation A.56 over the distribution of the values of the true function y, we can estimate a lower bound on the learning curve of the model. Assuming that the true underlying process is a GP, we are able to estimate the expectation of Equation A.56 over $p(y)$, evaluating a lower bound of the Bayesian generalisation error.

The integrals of Equation A.56 can be evaluated by expanding the random functions $y(x)$ and $\hat{y}(x)$ in terms of the eigenfunctions of the prior covariance $C_p(x,x')$ as

$$y(x) = \sum_{k=1}^{\infty} c_k \sqrt{\eta_k}\varphi_k(x) \text{ and } \hat{y}(x) = \sum_{k=1}^{\infty} \hat{c}_k \sqrt{\eta_k}\varphi_k(x).$$

The eigenvalues $\eta_k, k \in \mathbb{N}$ are the solutions of the equation

$$\int C_p(x,x')\,\varphi_k(x)\, p(x)\, dx = \eta_k \varphi_k(x')$$

where the set of eigenfunctions $\varphi_k(x)$ define a complete system of orthonormal function; the factors c_k and $\hat{c}_k$ are the projections of the random functions $y(x)$ and $\hat{y}(x)$ onto the eigenfunctions, i.e.

$$c_k = \int y(x)\,\varphi_k(x)\, p(x)\, dx \text{ and } \hat{c}_k = \int \hat{y}(x)\,\varphi_k(x)\, p(x)\, dx.$$

As the probabilistic model is a GP, the projections c_k on the eigenfunctions are distributed as a Gaussian with mean 0 ($\mathcal{E}[y(x)] = 0 = \sum_k \mathcal{E}[c_k]\sqrt{\eta_k}\varphi_k(x)$) and variance 1. This can be shown comparing $\mathcal{E}[y(x)\,y(x')] = C(x,x') = \sum_{k,l}\mathcal{E}[c_k c_l]\sqrt{\eta_k\eta_l}\varphi_k(x)\,\varphi_l(x')$ with the Mercier's expansion $C(x,x') = \sum_k \eta_k\varphi_k(x)\,\varphi_k(x')$ (e.g. Wong, 1971) from which follows that $\mathcal{E}[c_k c_l] = \delta_{k,l}$.

Defining

$$I(y,n) = \int \exp\left[-\frac{n}{\sigma_\nu^2}\int (y(x) - \hat{y}(x))^2\, p(x)\, dx\right] p(\hat{y})\, d\hat{y}, \tag{A.57}$$

we can expand the inner integral over the input distribution of $I(y,n)$ as

$$\begin{aligned}\int_{-\infty}^{\infty}(y(x) - \hat{y}(x))^2\, p(x)\, dx &= \sum_{k,l=1}^{\infty}\int_{-\infty}^{\infty}\sqrt{\eta_k\eta_l}\,(\hat{c}_k\hat{c}_l + c_k c_k - 2c_k\hat{c}_l)\,\varphi_k(x)\,\varphi_l(x)\, p(x)\, dx \\ &= \sum_{k=1}^{\infty}\eta_k\left(\hat{c}_k^2 + c_k^2 - 2c_k\hat{c}_k\right), \end{aligned}\tag{A.58}$$

where we used the relation

$$\int_{-\infty}^{\infty}\varphi_k(x)\,\varphi_l(x)\, p(x)\, dx = \delta_{kl}.$$

Hence $I(y,n)$ can be expanded as

$$\begin{aligned} I(y,n) &= \int \exp\left[-\sum_{k=1}^{\infty}\frac{n\eta_k}{\sigma_\nu^2}\left(c_k-\hat{c}_k\right)^2\right]\prod_{k=1}^{\infty}\exp\left[-\frac{\hat{c}_k^2}{2}\right]\frac{d\hat{c}_k}{\sqrt{2\pi}} \\ &= \prod_{k=1}^{\infty}\exp\left[-\frac{n\eta_k c_k^2}{\sigma_\nu^2}\right]\int_{-\infty}^{\infty}\frac{d\hat{c}_k}{\sqrt{2\pi}}\exp\left[-\frac{\hat{c}_k^2}{2}-\frac{n\eta_k}{\sigma_\nu^2}\left(\hat{c}_k^2-2\hat{c}_k c_k\right)\right] \\ &= \prod_{k=1}^{\infty}\sqrt{\frac{\sigma_\nu^2}{\sigma_\nu^2+2n\eta_k}}\exp\left[-\frac{n\eta_k c_k^2}{\sigma_\nu^2+2n\eta_k}\right]. \end{aligned} \tag{A.59}$$

Thus the right hand side of Equation A.56 can be evaluated as

$$\begin{aligned} \frac{\partial}{\partial n}\log I(y,n) &= -\frac{\partial}{\partial n}\sum_{k=1}^{\infty}\left(\frac{1}{2}\log\left[\frac{\sigma_\nu^2+2n\eta_k}{\sigma_\nu^2}\right]+\frac{n\eta_k c_k^2}{\sigma_\nu^2+2n\eta_k}\right) \\ &= -\sum_{k=1}^{\infty}\frac{\eta_k\left(\sigma_\nu^2+2n\eta_k+\sigma_\nu^2 c_k^2\right)}{\left(\sigma_\nu^2+2n\eta_k\right)^2}. \end{aligned} \tag{A.60}$$

Under the assumption that the underlying process is actually normally distributed with zero mean and unit variance, $\mathcal{E}_c\left[c_k^2\right]=1$ and hence we can calculate the expectation of Equation A.60 as

$$\int_{-\infty}^{\infty}\frac{\partial}{\partial n}\log I(y,n)\,p(y)\,dy = -2\sum_{k=1}^{\infty}\frac{\eta_k\left(\sigma_\nu^2+n\eta_k\right)}{\left(\sigma_\nu^2+2n\eta_k\right)^2}.$$

Plugging the last expression in Equation A.56, it turns out that the bound on $\mathcal{E}\left[f_n(y)\right]$ is

$$\mathcal{E}\left[f_n(y)\right] \geq \sigma_\nu^2\sum_{k=1}^{\infty}\frac{\eta_k\left(\sigma_\nu^2+n\eta_k\right)}{\left(\sigma_\nu^2+2n\eta_k\right)^2}. \tag{A.61}$$

As our bounds are on t rather than y, we must add σ_ν^2 to the bounds obtained in Equation A.61 obtaining a lower bound of

$$E_1^l(n) \doteq \sigma_\nu^2\left(1+\sum_{k=1}^{\infty}\frac{\eta_k\left(\sigma_\nu^2+n\eta_k\right)}{\left(\sigma_\nu^2+2n\eta_k\right)^2}\right). \tag{A.62}$$

It is straightforward to verify that this bound is always looser than the one proposed in Section 2.5.4

$$E^l(n) \doteq \sigma_\nu^2\left(1+\sum_{k=1}^{\infty}\frac{\eta_k}{\left(\sigma_\nu^2+n\eta_k\right)}\right) \tag{A.63}$$

(cf. Equation 2.31). Rearranging the terms in Equation A.61, we obtain that

$$\mathcal{E}\left[f_n(y)\right] \geq \sigma_\nu^2\sum_{k=1}^{\infty}\frac{\eta_k\left(\sigma_\nu^2+n\eta_k\right)}{a\left(\sigma_\nu^2+n\eta_k\right)^2} = \sigma_\nu^2\sum_{k=1}^{\infty}\frac{\eta_k}{a\left(\sigma_\nu^2+n\eta_k\right)}, \tag{A.64}$$

where $a=\left(1+n\eta_k/\left(\sigma_\nu^2+n\eta_k\right)\right)^2>1$. Since $E_1^l(n)<E^l(n),\forall n$ (as each contribution to the series of Equation A.64 is smaller than each term of the sum in Equation A.63), the lower bound $E_1^l(n)$ is looser than $E^l(n)$.

Appendix B

Derivation of the expected generalisation error

In this Appendix we show the derivation of the generalisation error averaged over the distribution of the input vectors as used in Section 3.3.2. In Equation 3.6 the expected generalisation error is defined as

$$E^g\left(\mathcal{D}_n\right) = \int \left(t\left(\mathbf{x}\right) - \hat{y}_{\mathcal{D}_n}\left(\mathbf{x}\right)\right)^2 p\left(\mathbf{x}\right) d\mathbf{x}, \tag{B.1}$$

where $\hat{y}_{\mathcal{D}_n}\left(\mathbf{x}\right)$ is the GP prediction

$$\hat{y}_{\mathcal{D}_n}\left(\mathbf{x}\right) = \mathbf{k}^{\mathrm{T}}\left(\mathbf{x}\right) K^{-1}\mathbf{t} \tag{B.2}$$

(see Equation 3.2) made by a GP using either a general W_f or a diagonal W_d distance matrix; as in our experiments we evaluated Equation B.1 for both the matrices, in what follows we omit the subscript f and d. The vector $\mathbf{k}^{\mathrm{T}}\left(\mathbf{x}\right) = \left(C_p\left(\mathbf{x},\mathbf{x}^1\right), C_p\left(\mathbf{x},\mathbf{x}^2\right), \ldots, C_p\left(\mathbf{x},\mathbf{x}^n\right)\right)$ contains the covariances of the test point $\mathbf{x}$ with all the input data, i.e.

$$C_p\left(\mathbf{x},\mathbf{x}'\right) = \sigma_p^2 \exp\left[-\frac{1}{2}\left(\mathbf{x}-\mathbf{x}'\right)^{\mathrm{T}} W\left(\mathbf{x}-\mathbf{x}'\right)\right] \tag{B.3}$$

(cf. Equation 3.1) and $\mathbf{t}^{\mathrm{T}} = \left(t^1, t^2, \ldots t^n\right)$ is the vector collecting the data at the input points $\left(\mathbf{x}^1, \mathbf{x}^2 \ldots \mathbf{x}^n\right)$. K is the covariance matrix defined as $K = K_p + \sigma_n^2 \mathbb{I}$ (cf. Equation 3.2), where $\left(K_p\right)_{ij} = C_p\left(\mathbf{x}^i, \mathbf{x}^j\right)$. The parameter σ_n^2 models the variance of the noise and σ_p^2 models the prior covariance. In the experiments of Section 3.3.2, the target $t\left(\mathbf{x}\right)$ is generated by adding Gaussian noise to the underlying function $y\left(\mathbf{x}\right) = \sin\left(2\pi \mathbf{m}^{\mathrm{T}}\mathbf{x}\right)$, where $\mathbf{x}$ is normally distributed in the d dimensional space $\mathbb{R}^d$ with variance σ_x^2 and $\mathbf{m} \in \mathbb{R}^d$ is the vector representing the transformation from the manifest space $\mathcal{X} \in \mathbb{R}^d$ to the hidden feature space $\mathcal{Z} \in \mathbb{R}$ (i.e. $z = \mathbf{m}^{\mathrm{T}}\mathbf{x}$).

The evaluation of Equation B.1 can be decomposed in two parts, one concerning the squared difference of the underlying function $y(\mathbf{x})$ and the GP prediction $\hat{y}_{\mathcal{D}_n}(\mathbf{x})$ (the model error), and the unpredictable noise components (containing linear and squared terms of the noise corrupting $y(\mathbf{x})$). Since the noise is assumed to be independent of $\mathbf{x}$, the integration of the linear and squared terms of the noise over the Gaussian distribution of the $\mathbf{x}$ equalises 0 and σ_ν^2 respectively; thus Equation B.1 can be rewritten as

$$\begin{aligned} E^g(\mathcal{D}_n) &= \sigma_\nu^2 + \int \left(y(\mathbf{x}) - \hat{y}_{\mathcal{D}_n}(\mathbf{x})\right)^2 p(\mathbf{x})\, d\mathbf{x} \\ &= \sigma_\nu^2 + \int \left(y^2(\mathbf{x}) + \hat{y}^2_{\mathcal{D}_n}(\mathbf{x}) - 2y(\mathbf{x})\,\hat{y}_{\mathcal{D}_n}(\mathbf{x})\right) p(\mathbf{x})\, d\mathbf{x} \qquad \text{(B.4)} \\ &\doteq \sigma_\nu^2 + E^g_{y^2} + E^g_{\hat{y}^2} - 2E^g_{y\hat{y}}. \qquad \text{(B.5)} \end{aligned}$$

The first term is the expected value of the squared underlying function, i.e.

$$E^g_{y^2} = \int y^2(\mathbf{x})\, p(\mathbf{x})\, d\mathbf{x} = \frac{1}{\sqrt{(2\pi\sigma_x^2)^d}} \int \sin^2\left(2\pi \mathbf{m}^{\mathrm{T}}\mathbf{x}\right) \exp\left[-\frac{\mathbf{x}^{\mathrm{T}}\mathbf{x}}{2\sigma_x^2}\right] d\mathbf{x} \qquad \text{(B.6)}$$

where in Section 3.3.2 we set $\mathbf{m} = \mathbf{1}/\sqrt{d}$, $\mathbf{1} \in \mathbb{R}^d$.

We note that Equation B.6 is the integral of the product of two functions, one Gaussian (characterised by the variance σ_x^2 along the d directions of the space) and the trigonometric function $\sin^2\left(2\pi\mathbf{m}^{\mathrm{T}}\mathbf{x}\right)$, whose direction of variation is drawn by $\mathbf{m}$; this allows to integrate out all the other directions of the space, evaluating the Equation B.6 as the one-dimensional integral along $\mathbf{m}$. In one dimension (see e.g. Equation 3.898.3 in Gradshteyn and Ryzhik, 1993),

$$\int_{-\infty}^{\infty} \sin^2(a\xi) \exp\left[-p\xi^2\right] d\xi = \frac{1}{2}\sqrt{\frac{\pi}{p}}\left(1 - \exp\left[-\frac{a^2}{p}\right]\right). \qquad \text{(B.7)}$$

Setting $p = 1/\left(2\sigma_x^2\right)$ and $a^2 = 4\pi^2\mathbf{m}^{\mathrm{T}}\mathbf{m} = 4\pi^2$, the multiplying factor $\sqrt{\pi/p}$ simplifies with the normalisation of the Gaussian in Equation B.6 and we thus obtain

$$E^g_{y^2} = \frac{1}{2}\left(1 - \exp\left[-8\pi^2\sigma_x^2\right]\right). \qquad \text{(B.8)}$$

The second contribution to Equation B.5 is the average of the squared prediction over the distribution of the input vectors $\mathbf{x}$, i.e.

$$E^g_{\hat{y}^2} = \int \hat{y}^2_{\mathcal{D}_n}(\mathbf{x})\, p(\mathbf{x})\, d\mathbf{x}. \qquad \text{(B.9)}$$

Plugging the actual formula of $\hat{y}^2(\mathbf{x})$ (see Equation B.2) in Equation B.9 we obtain the following expression which depends upon the covariance matrix K and the vectors $\mathbf{t}$ and $\mathbf{k}(\mathbf{x})$:

$$\begin{aligned} E^g_{\hat{y}^2} &= \int \mathbf{k}^{\mathrm{T}}(\mathbf{x})\, K^{-1}\mathbf{t}\mathbf{t}^{\mathrm{T}}K^{-1}\mathbf{k}(\mathbf{x})\, p(\mathbf{x})\, d\mathbf{x} \\ &= \int \mathrm{Tr}\left[K^{-1}\mathbf{t}\mathbf{t}^{\mathrm{T}}K^{-1}\mathbf{k}(\mathbf{x})\,\mathbf{k}^{\mathrm{T}}(\mathbf{x})\right] p(\mathbf{x})\, d\mathbf{x} \\ &= \sum_{i,j=1}^{n} \left(K^{-1}\mathbf{t}\mathbf{t}^{\mathrm{T}}K^{-1}\right)_{ij} \int C_p(\mathbf{x}, \mathbf{x}^j)\, C_p(\mathbf{x}, \mathbf{x}^i)\, p(\mathbf{x})\, d\mathbf{x} \qquad \text{(B.10)} \\ &= \frac{\sigma_p^4}{\sqrt{(2\pi\sigma_x^2)^d}} \sum_{i,j=1}^{n} \left(K^{-1}\mathbf{t}\mathbf{t}^{\mathrm{T}}K^{-1}\right)_{ij} \exp\left[-\frac{1}{2}\left(\mathbf{x}^{i\mathrm{T}}W\mathbf{x}^i + \mathbf{x}^{j\mathrm{T}}W\mathbf{x}^j\right)\right] I_{ij}. \qquad \text{(B.11)} \end{aligned}$$

Equation B.10 has been obtained substituting each term of the vector $(\mathbf{k}(\mathbf{x}))_i$ with its components $C_p(\mathbf{x}, \mathbf{x}^i)$, where $C_p(\mathbf{x}, \mathbf{x}^i)$ defined in Equation B.3. We also note that in Equation B.11 we have used the property of symmetry of the matrices K and I, where each element of the matrix I is defined as

$$I_{ij} = \int \exp\left[-\frac{1}{2}\left(2\mathbf{x}^{\mathrm{T}} W \mathbf{x} - 2\mathbf{x}^{\mathrm{T}} W\left(\mathbf{x}^i + \mathbf{x}^j\right) + \frac{\mathbf{x}^{\mathrm{T}}\mathbf{x}}{\sigma_x^2}\right)\right] d\mathbf{x}. \tag{B.12}$$

Due to the symmetry of the distance matrix W, we also used the relation $\mathbf{x}^{\mathrm{T}} W \mathbf{x}' = \mathbf{x}'^{\mathrm{T}} W \mathbf{x}$.

Defining $\tilde{A} = \left(2\sigma_x^2 W + \mathbb{I}\right)/\sigma_x^2$ and $\mathbf{v}^{ij} = W\left(\mathbf{x}^i + \mathbf{x}^j\right)$, we note that Equation B.12 can be rewritten as

$$I_{ij} = \int \exp\left[-\frac{1}{2}\mathbf{x}^{\mathrm{T}} \tilde{A} \mathbf{x} + \mathbf{x}^{\mathrm{T}} \mathbf{v}^{ij}\right] d\mathbf{x}. \tag{B.13}$$

The solution of Equation B.13 can be carried out considering the eigen-space of the matrix $\tilde{A}$ and the linear transformation from the manifest input space $\mathcal{X}$ onto the eigen-space. Details of this calculation can be found in Bishop (1995), Appendix B; we report the result of Equation B.13 which is

$$\int \exp\left[-\frac{1}{2}\mathbf{x}^{\mathrm{T}} \tilde{A} \mathbf{x} + \mathbf{x}^{\mathrm{T}} \mathbf{v}^{ij}\right] d\mathbf{x} = \sqrt{(2\pi)^d \det \tilde{A}^{-1}} \exp\left[\frac{1}{2}\mathbf{v}^{ij\mathrm{T}} \tilde{A}^{-1} \mathbf{v}^{ij}\right] \tag{B.14}$$

(cf. Equation B. 22 in Bishop (1995)).

Setting $A = \sigma_x^2 \tilde{A}$, we have that $\det \tilde{A} = \left(\sigma_x^2\right)^d \det A$ and $\tilde{A}^{-1} = \sigma_x^2 A^{-1}$; plugging the actual expression of $\mathbf{v}^{ij}$ in Equation B.14, we can evaluate I_{ij} as

$$I_{ij} = \sqrt{(2\pi\sigma_x^2)^d \det A^{-1}} \exp\left[\frac{\sigma_x^2}{2}\left(\mathbf{x}^i + \mathbf{x}^j\right)^{\mathrm{T}} W^{\mathrm{T}} A^{-1} W \left(\mathbf{x}^i + \mathbf{x}^j\right)\right]. \tag{B.15}$$

Finally, plugging Equation B.15 in Equation B.11, the factor $\sqrt{(2\pi\sigma_x^2)^d}$ simplifies with the normalisation factor of the Gaussian distribution and we calculate the value of the squared GP prediction integrated over the distribution of the input vectors as

$$E_{\tilde{y}^2}^g = \sigma_p^4 \sqrt{\det A^{-1}} \sum_{i,j=1}^{n} \left(K^{-1}\mathbf{t}\mathbf{t}^{\mathrm{T}} K^{-1}\right)_{ij} B_{ij}, \tag{B.16}$$

where

$$B_{ij} = \exp\left[-\frac{1}{2}\left(\mathbf{x}^{i\mathrm{T}} W \mathbf{x}^i + \mathbf{x}^{j\mathrm{T}} W \mathbf{x}^j\right)\right] \exp\left[\frac{\sigma_x^2}{2}\left(\mathbf{x}^i + \mathbf{x}^j\right)^{\mathrm{T}} W^{\mathrm{T}} A^{-1} W \left(\mathbf{x}^i + \mathbf{x}^j\right)\right].$$

The last contribution to Equation B.5 is given by

$$E_{y\hat{y}}^g = \int y(\mathbf{x})\, \hat{y}_{\mathcal{D}_n}(\mathbf{x})\, p(\mathbf{x})\, d\mathbf{x}. \tag{B.17}$$

Plugging the GP prediction $\hat{y}_{\mathcal{D}_n}(\mathbf{x})$ and the actual underlying function $y(\mathbf{x})$ in Equation B.17 we obtain

$$\begin{aligned} E_{y\hat{y}}^g &= \int \mathbf{k}^{\mathrm{T}}(\mathbf{x}) K^{-1}\mathbf{t} \sin\left(2\pi\mathbf{m}^{\mathrm{T}}\mathbf{x}\right) p(\mathbf{x})\, d\mathbf{x} = \int \mathrm{Tr}\left[K^{-1}\mathbf{t}\mathbf{k}^{\mathrm{T}}(\mathbf{x})\right] \sin\left(2\pi\mathbf{m}^{\mathrm{T}}\mathbf{x}\right) p(\mathbf{x})\, d\mathbf{x} \\ &= \sum_{j=1}^{n} \left(K^{-1}\mathbf{t}\right)_j \int C_p\left(\mathbf{x}, \mathbf{x}^j\right) \sin\left(2\pi\mathbf{m}^{\mathrm{T}}\mathbf{x}\right) p(\mathbf{x})\, d\mathbf{x} \\ &= \frac{\sigma_p^2}{\sqrt{(2\pi\sigma_x^2)^d}} \sum_{j=1}^{n} \left(K^{-1}\mathbf{t}\right)_j \exp\left[-\frac{1}{2}\left(\mathbf{x}^{j\mathrm{T}} W \mathbf{x}^j\right)\right] I_j, \end{aligned} \tag{B.18}$$

where we have used the property of symmetry of W (i.e. $\mathbf{x}^{\mathrm{T}}W\mathbf{x}^j = \mathbf{x}^{j\mathrm{T}}W\mathbf{x}$) and we have defined I_j as

$$I_j = \int \sin\left(2\pi\mathbf{m}^{\mathrm{T}}\mathbf{x}\right) \exp\left[-\frac{1}{2}\left(\mathbf{x}^{\mathrm{T}}W\mathbf{x} - 2\mathbf{x}^{\mathrm{T}}W\mathbf{x}^j + \frac{\mathbf{x}^{\mathrm{T}}\mathbf{x}}{\sigma_x^2}\right)\right] d\mathbf{x}.$$

Considering the complex form of the sine function

$$\sin\left(2\pi\mathbf{m}^{\mathrm{T}}\mathbf{x}\right) = \frac{\exp\left[i2\pi\mathbf{m}^{\mathrm{T}}\mathbf{x}\right] - \exp\left[-i2\pi\mathbf{m}^{\mathrm{T}}\mathbf{x}\right]}{2i}$$

with $i = \sqrt{-1}$, we can rewrite I_j as $I_j = \left(I_j^+ - I_j^-\right)/2i$ where

$$\begin{aligned} I_j^{\pm} &= \int \exp\left[-\frac{1}{2}\left(\mathbf{x}^{\mathrm{T}}W\mathbf{x} - 2\mathbf{x}^{\mathrm{T}}W\mathbf{x}^j + \frac{\mathbf{x}^{\mathrm{T}}\mathbf{x}}{\sigma_x^2}\right)\right] \exp\left[\pm 2\pi i\mathbf{x}^{\mathrm{T}}\mathbf{m}\right] d\mathbf{x} \\ &= \int \exp\left[-\frac{1}{2}\left(\mathbf{x}^{\mathrm{T}}W\mathbf{x} - 2\mathbf{x}^{\mathrm{T}}W\mathbf{x}^j \mp 4\pi i\mathbf{x}^{\mathrm{T}}\mathbf{m} + \frac{\mathbf{x}^{\mathrm{T}}\mathbf{x}}{\sigma_x^2}\right)\right] d\mathbf{x}. \end{aligned} \tag{B.19}$$

Similarly to Equation B.12, we can define $\tilde{A} = \left(\sigma_x^2 W + \mathbb{I}\right)/\sigma_x^2$ and $\mathbf{v}^j = W\mathbf{x}^j \pm 2\pi i\mathbf{m}$. Equation B.19 can thus be rewritten as

$$I_j^{\pm} = \int \exp\left[-\frac{1}{2}\mathbf{x}^{\mathrm{T}}\tilde{A}\mathbf{x} + \mathbf{x}^{\mathrm{T}}\mathbf{v}^j\right] d\mathbf{x} \tag{B.20}$$

and evaluated as Equation B.14, substituting the vector $\mathbf{v}^{ij}$ with $\mathbf{v}^j$ and setting $A = \sigma_x^2\tilde{A}$ (from which follows that $\det\tilde{A} = \left(\sigma_x^2\right)^d \det A$ and $\tilde{A}^{-1} = \sigma_x^2 A^{-1}$); this leads to the evaluation of $I_j^{\pm}$ as

$$I_j^{\pm} = \sqrt{(2\pi\sigma_x^2)^d \det A^{-1}} \exp\left[\frac{\sigma_x^2}{2}\mathbf{v}^{j\mathrm{T}}A^{-1}\mathbf{v}^j\right]. \tag{B.21}$$

We note that the quadratic form in the exponent of Equation B.21 can be expanded as

$$\begin{aligned} \mathbf{v}^{j\mathrm{T}}A^{-1}\mathbf{v}^j &= \left(W\mathbf{x}^j \pm 2\pi i\mathbf{m}\right)^{\mathrm{T}} A^{-1}\left(W\mathbf{x}^j \pm 2\pi i\mathbf{m}\right) \\ &= \mathbf{x}^{j\mathrm{T}}W^{\mathrm{T}}A^{-1}W\mathbf{x}^j - 4\pi^2\mathbf{m}^{\mathrm{T}}A^{-1}\mathbf{m} \pm 4i\pi\sigma_x^2\mathbf{m}^{\mathrm{T}}A^{-1}W\mathbf{x}^j, \end{aligned}$$

where we have used the relation $\mathbf{x}^{j\mathrm{T}}W^{\mathrm{T}}A^{-1}\mathbf{m} = \mathbf{m}^{\mathrm{T}}A^{-1}W\mathbf{x}^j$. The value $I_j^{\pm}$ of Equation B.21 can thus be expressed with the formula

$$\begin{aligned} I_j^{\pm} = \sqrt{(2\pi\sigma_x^2)^d \det A^{-1}} \exp\left[\frac{\sigma_x^2}{2}\left(\mathbf{x}^{j\mathrm{T}}W^{\mathrm{T}}A^{-1}W\mathbf{x}^j - 4\pi^2\mathbf{m}^{\mathrm{T}}A^{-1}\mathbf{m}\right)\right] \times \\ \exp\left[\pm 2i\pi\sigma_x^2\mathbf{m}^{\mathrm{T}}A^{-1}W\mathbf{x}^j\right]. \end{aligned} \tag{B.22}$$

Recalling that the definition of I_j is $I_j = \left(I_j^+ - I_j^-\right)/2i$, by using Equation B.22 we have that

$$\begin{aligned} I_j &= \sqrt{(2\pi\sigma_x^2)^d \det A^{-1}} \exp\left[\frac{\sigma_x^2}{2}\left(\mathbf{x}^{j\mathrm{T}}W^{\mathrm{T}}A^{-1}W\mathbf{x}^j - 4\pi^2\mathbf{m}^{\mathrm{T}}A^{-1}\mathbf{m}\right)\right] \times \\ &\qquad \frac{\exp\left[2i\pi\sigma_x^2\mathbf{m}^{\mathrm{T}}A^{-1}W\mathbf{x}^j\right] - \exp\left[-2i\pi\sigma_x^2\mathbf{m}^{\mathrm{T}}A^{-1}W\mathbf{x}^j\right]}{2i} \\ &= \sqrt{(2\pi\sigma_x^2)^d \det A^{-1}} \exp\left[\frac{\sigma_x^2}{2}\left(\mathbf{x}^{j\mathrm{T}}W^{\mathrm{T}}A^{-1}W\mathbf{x}^j - 4\pi^2\mathbf{m}^{\mathrm{T}}A^{-1}\mathbf{m}\right)\right] \times \\ &\qquad \sin\left(2\pi\sigma_x^2\mathbf{m}^{\mathrm{T}}A^{-1}W\mathbf{x}^j\right). \end{aligned}$$

Finally, the value of I_j can be plugged in Equation B.18 simplifying the factor $\sqrt{(2\pi\sigma_x^2)^d}$ with the normalisation factor of the Gaussian distribution; this allows the estimation of the last term of the

generalisation error $E^g_{y\hat{y}}$ as

$$E^g_{y\hat{y}} = \sigma_p^2 \sqrt{\det A^{-1}} \sum_{j=1}^{n} \left(K^{-1}\mathbf{t}\right)_j b_j \tag{B.23}$$

where

$$b_j = \sin\left(2\pi\sigma_x^2 \mathbf{m}^{\mathrm{T}} A^{-1} W \mathbf{x}^j\right) \exp\left[-\frac{1}{2}\left(\mathbf{x}^{j\mathrm{T}} W \mathbf{x}^j\right)\right] \exp\left[\frac{\sigma_x^2}{2}\left(\mathbf{x}^{j\mathrm{T}} W^{\mathrm{T}} A^{-1} W \mathbf{x}^j - 4\pi^2 \mathbf{m}^{\mathrm{T}} A^{-1} \mathbf{m}\right)\right]. \tag{B.24}$$

In conclusion, the evaluation of the expected generalisation error $E^g\left(\mathcal{D}_n\right)$ can be obtained plugging Equations B.6, B.16 and B.23 in

$$E^g\left(\mathcal{D}_n\right) = \sigma_\nu^2 + E^g_{y^2} + E^g_{\hat{y}^2} - 2E^g_{y\hat{y}}$$

(cf. Equation B.5).

Appendix C

Scripts for the MCMC algorithm

This Appendix[1] shows the script used for the numerical simulations concerning the training of a neural network with the Markov Chain Monte Carlo algorithm. The case of the Multi Layer Perceptron with 30 hidden nodes is presented.

A simulation with the MCMC method can be divided in five stages: the set up of the network, the creation of the training and the test set, the starting phase, the sampling phase and the prediction phase.

The simulation starts with the creation of a `log` file to store all the data about the experiment. The file is generated by the command line

```
net-spec mlp-log 35 30 11 / - 0.05:0.5:0.5 0.05:0.5 - x0.05:  0.5 - 0.05:0.5.
```

This command calls the program `net-spec` which stores the network specifications in the file `mlp-log`. In our experiments, a MLP with 35 input units and 30 hidden nodes have been used; the network has also 11 outputs in order to discriminate 11 classes. After the /, several weight parameters are given. They specify the prior distributions of the hyperparameters controlling the input-to-hidden and the hidden-to-output weights as well as the biases of the hidden and the output nodes. The prior `0.05:0.5:0.5` has two values of hyperparameters, indicating that one hyperparameter controls the overall magnitude of the weights and the other controls the magnitude of the weights out of each input. The hyper-priors used are inverse Gamma distributions: the first value `0.05` refers to how vague the prior is, while the value `0.5` specify the location of the distribution.

The command line

[1] The scripts of the commands for running the Markov Chain Monte Carlo software has been drawn from the Introductory documentation for software implementing Bayesian learning for neural networks using Markov Chain Monte Carlo methods, written by Radford Neal.

```
rand-seed mlp-log s
```

initialises the random generator to the numerical value s.

The type of problem the network is dealing with (classification problem in our experiments) is selected by the line

```
model-spec mlp-log class.
```

The training and test data which are used in the numerical simulations are specified with the command

```
data-spec mlp-log 35 1 11 / train.dat@1:ntrain .  test.dat@1:1505 .
```

In this line, the process `data-spec` writes on `mlp-log` the number of inputs (35), the number of targets (1) and the number of the possible values of the targets (11, from 0 to 10). The following part of the line specifies the names of the files containing the training and the test data (`train_hmc.dat` and `test_hmc.dat` respectively) and the number of lines composing the sets; in our simulations we considered data from line 1 to line `ntrain` of the training set, and data from line 1 to line 1505 of the test set. The amount of training data `ntrain` has been varied in our experiments being `ntrain` = 46, 91, 182, 365, 729, 1458, 2916 and 5832.

The command

```
net-gen mlp-log fix 0.5
```

stores a network in `mlp-log` whose synaptic weights are zero and the hyperparameters are set to `0.5`.

The specification of the Markov chain operation to be done in the starting phase are allowed by the command

```
mc-spec mlp-log repeat 10 sample-noise heatbath hybrid 100:10 0.2.
```

The `mc-spec` process repeats 10 times the Gibbs sampling for the noise level, the heatbath generation of the fictious momentum variables p and finally updates the parameters of the hybrid Monte Carlo.

In the generation of the hybrid Monte Carlo chain, the 100 *leapfrog* steps are performed with a window size of 10 and with a step size set to `0.2`. This first generation has been done just to set up both the hyperparameters and the network parameters to reasonable values.

The starting phase only last one iteration as it is specified by the line

```
net-mc mlp-log 1.
```

The command

```
mc-spec mlp-log repeat 10 sample-sigmas heatbath 0.95 hybrid 100:10 0.3 negate
```

appends to the `mlp-log` a new set of Markov Chain operations, negating the momentum variables.

Now the 200 iterations of the Markov Chain Monte Carlo method can be done with the command

```
net-mc mlp-log 200.
```

The prediction can be done using the program `net-pred`. Discarding the first third of the iterations, the predictions are based upon the values sampled when the simulation has reached equilibrium. The command is

```
net-pred itn mlp-log 67:.
```

The output is a text file in which are displayed the input vector (given by the option `i`), the target (option `t`) and the mean output of the networks (option `n`) used for prediction during the iterations `67:200`.

Bibliography

Adler, R. J. (1981). *The Geometry of Random Fields.* John Wiley and Sons, New York.

Akaike, H. (1974). A new look at statistical model identification. *IEEE Transactions on Automatic Control*, 19:716–723.

Ballard, D. H. and Brown, C. M. (1982). *Computer Vision.* Prentice-Hall, London.

Barber, D. and Bishop, C. M. (1998). Ensemble learning for Multi-Layer Networks. In Mozer, M. C.and Jordan, M. I. and Petsche, T., editors, *Advances in Neural Information Processing Systems 10.* MIT Press.

Barber, D. and Williams, C. K. I. (1997). Gaussian Processes for Bayesian Classification via Hybrid Monte Carlo. In Mozer, M. C., Jordan, M. I., and Petsche, T., editors, *Advances in Neural Information Processing Systems 9.* MIT Press.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition.* Oxford University Press, Oxford.

Breiman, L. (1993). Hinging hyperplanes for regression, classification and function approximation. *IEEE Trans. on Information Theory*, 39(3):999–1013.

Breiman, L. (1996). Bagging Predictors. *Machine Learning*, 24:123–140.

Breiman, L. (1998). Instability, bias-variance and regularization. In Bishop, C. M., editor, *Generalisation in Neural Networks and Machine Learning.* Springer-Verlag.

Brooks, R. A. (1983). Model-based 3D interpretation of 2D images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 5(2):140–150.

Clark, A. (1995). Computer Vision for Outdoor Scene Analysis. Master's thesis, University of Bristol, Bristol, United Kingdom.

Cowles, M. K. and Carlin, B. P. (1996). Markov Chain Monte Carlo Convergence Diagnostics: A Comparitive Review. *J. American Statistical Assn.*, 91(434):883–904.

David, H. A. (1970). *Order Statistics.* John Wiley and Sons, New York.

DeGroot, M. H. (1984). *Probability and Statistics.* Addison-Wesley, Reading, Mass., second edition.

Dietterich, T. G. (1998). Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation*, 10(7).

Draper, B. A., Collins, R. T., Brolio, J., Hanson, A. R., and Riseman, E. M. (1989). The Schema System. *The International Journal of Computer Vision*.

Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. (1987). Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222.

Friedman, J. H. and Tukey, J. W. (1974). A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on Computers*, 23:881–890.

Gay, M. (1989). Segmentation using region merging with edges. In *Proceedings* 5^{th} *Alvey Vision Conference*, pages 115–119, Reading, UK. Sheffield University Press.

Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (1995). *Bayesian Data Analysis*. Chapman and Hall.

Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural Networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58.

Girosi, F., Jones, M., and Poggio, T. (1995). Regularization Theory and Neural Networks Architectures. *Neural Computation*, 7(2):219–269.

Gonzales, R. C. and Woods, R. E. (1992). *Digital Image Processing*. Addison-Wesley, Reading, Mass.

Gradshteyn, E. S. and Ryzhik, I. M. (1993). *Table of Integrals, Series and Products*. Academic Press, New York, fifth edition.

Gull, S. (1988). Bayesian inductive inference and maximum entropy. In Erickson, G. J. and Smith, C. R., editors, *Maximum-Entropy and Bayesian Methods in Science and Engineering, Vol. 1: Foundations*, pages 53–74. Kluwer, Dordrecht.

Hansen, L. K. (1993). Stochastic Linear Learning: Exact Test and Training Error Averages. *Neural Networks*, 6:393–396.

Haralick, R. M., Shanmugam, K., and Dinstein, I. (1973). Textural features for image classification. *IEEE Transactions on Systems, Man and Cybernetics*.

Hastie, T. J. and Tibshirani, R. J. (1990). *Generalized Additive Models*. Chapman and Hall, London.

Haussler, D. and Opper, M. (1997). Mutual Information, Metric Entropy and Cumulative Relative Entropy Risk. *The Annals of Statistics*, 25(6):2451.

Hawkins, D. L. (1989). Some practical problems in implementing a certain sieve estimation of the Gaussian mean function. *Communications in Statistics - Simulation and Computation*, 18(2):481–500.

Hertz, J., Krogh, A., and Palmer, G. (1991). *Introduction to the Theory of Neural Networks*. Addison-Wesley, Redwood City, CA.

Hinton, G. E. and van Camp, D. (1993). Keeping neural networks simple by minimising the description length of the weights. In *Proceedings of the Sixth Annual Workshop on Computational Learning Theory*, pages 5–13, New York, NY. ACM Press.

Hu, M. K. (1962). Visual pattern recognition by moment invariants. *IRE Trans.*, IT-8:179–187.

Ihara, S. (1993). *Information Theory*. World Scientific Publishing, Singapore.

Iranpour, R. and Chacon, P. (1986). *Basic Stochastic Processes: the Mark Kac Lectures*. McMillan.

Li, K. C. (1991). Sliced Inverse Regression for Dimension Reduction. *Journal of the American Statistical Association*, 86:316–342.

Lindman, H. R. (1974). *Analysis of variance in Complex Experimental Design*. W. H. Freeman and Company, San Francisco.

MacKay, D. J. C. (1992a). Bayesian interpolation. *Neural Computation*, 4(3):415–447.

MacKay, D. J. C. (1992b). The evidence framework applied to classification networks. *Neural Computation*, 4(5):720–736.

MacKay, D. J. C. (1997). Gaussian processes: A Replacement for Neural Networks. Tutorial at the *Neural Information Processing Systems*. The tutorial is available at the URL address `http://wol.ra.phy.cam.ac/pub/mackay/`.

Mackeown, W. P. J. (1994). *A Labelled Image Database and its Application to Outdoor Scene Analysis*. PhD thesis, University of Bristol, Bristol, United Kingdom.

Matérn, B. (1986). *Spatial Variation*. Springer-Verlag, Berlin, second edition. Lecture Notes in Statistics 36.

McKeown, D. M., Harvey, W. A., and McDermott, J. (1985). Rule-based interpretation of aerial imagery. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 7(5).

Michelli, C. A. and Wahba, G. (1981). Design problems for optimal surface interpolation. In Ziegler, Z., editor, *Approximation Theory and Applications*, pages 329–348. Academic Press.

Murata, N., Yoshizawa, S., and Amari, S. (1994). Network information criterion–determining the number of hidden units for artificial neural network models. *IEEE Transactions on Neural Networks*, 5:865–872.

Neal, R. M. (1996). *Bayesian Learning for Neural Networks*. Springer, New York. Lecture Notes in Statistics 118.

Ohta, Y. (1985). *Knowledge based interpretation of Outdoor Natural Scenes*. Pittmann, London.

Opper, M. (1997). Regression with Gaussian processes: average case performance. In Kwok-Yee, M. W., Irwin, K., and Dit-Yan, Y., editors, *Theoretical Aspects of Neural Computation: A Multidisciplinary Perspective*, Berlin. Springer-Verlag.

Opper, M. and Vivarelli, F. (1999). General Bounds on Bayes errors for regression with Gaussian processes. In *Advances in Neural Information Processing Systems 11*. In press.

Plaskota, L. (1996). *Noisy information and computational complexity*. Cambridge University Press, Cambridge.

Press, W., Teukolsky, S., Vetterling, W., and Flannery, B. (1992). *Numerical Recipes in C. The Art of Scientific Computing*. Cambridge University Press, Cambridge. Second edition.

Rasmussen, C. E. (1996). *Evaluation of Gaussian Processes and Other Methods for Non-linear Regression*. PhD thesis, Department of Computer Science, University of Toronto, Canada.

Rasmussen, C. E., Neal, R. M., Hinton, G. E., van Camp, D., Revow, M., Ghahramani, Z., Kustra, R., and Tibshirani, R. (1996). *The DELVE Manual*. Department of Computer Science, University of Toronto, Canada, 1.1 edition. Delve is available at the URL address `http://www.cs.utoronto.ca/~delve/`.

Ripley, B. D. (1996). *Pattern Recognition and Neural Networks*. Cambridge University Press.

Ritter, K. (1996). Almost Optimal Differentiation Using Noisy Data. *Journal of Approximation Theory*, 86(3):293–309.

Ritter, K., Wasilkowski, G. W., and Wozniakowski, H. (1995). Multivariate Integration and Approximation for Random Fields satisfying Sacks-Ylvisaker Conditions. *Ann. Appl. Prob.*, 5:518–540.

Sampson, P. D. and Guttorp, P. (1992). Nonparametric estimation of nonstationary covariance structure. *Journal of the American Statistical Association*, 87:108–119.

Silverman, B. W. (1985). Some Aspects of the Spline Smoothing Approach to Non-parametric Regression Curve Filtering. *Journal of the Royal Statistical Society B*, 47(1):1–52.

Stein, M. L. (1989). Comment on the paper by Sacks, J. *et al.* Design and Analysis of Computer Experiments. *Statistical Science*, 4(4):432-433.

Valiant, L. G. (1984). A theory of the learnable. *Communication of the Association for Computing Machinery*, 27:1134–1142.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory.* Springer-Verlag, New York.

Vivarelli, F. (1996). The Use of Neural Networks in Classifying Segmented Images. Master's thesis, Neural Computing Research Group, Aston University, Birmingham, United Kingdom.

Whittle, P. (1963). *Prediction and regulation by linear least square methods.* English Universities Press.

Williams, C. K. I. (1997). Computing with infinite networks. In Mozer, M. C., Jordan, M. I., and Petsche, T., editors, *Advances in Neural Information Processing Systems 9.* MIT Press.

Williams, C. K. I. (1998). Prediction with Gaussian processes: from linear regression to linear prediction and beyond. In Jordan, M. I., editor, *Learning and Inference in Graphical Models.* Kluwer Academic Press.

Williams, C. K. I. and Rasmussen, C. E. (1996). Gaussian processes for regression. In Touretzky, M. C., Mozer, M. C., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems 8*, pages 514–520. MIT Press.

Williams, P. M. (1996). Conditional Multivariate Densities. *Neural Computation*, 8(4):843–854.

Wong, E. (1971). *Stochastic Processes in Information and Dynamical Systems.* McGraw-Hill, New York.

Wright, W. A. (1989). Image labelling with a neural network. In *Proceedings* 5^{th} *Alvey Vision Conference*, pages 227–232, Reading, UK. Sheffield University Press.

Wright, W. A., Mackeown, W. P. J., and Greenway, P. (1995). The use of neural networks for region labelling and scene understanding. In Taylor, J. G., editor, *Neural Networks*, pages 165–192. Alfred Waller.