Bayesian Methods for Neural Networks

João F. G. de Freitas

Trinity College, University of Cambridge
and
Cambridge University Engineering Department.

Dissertation submitted to the University of Cambridge
for the degree of Doctor of Philosophy

Summary

The application of the Bayesian learning paradigm to neural networks results in a flexible and powerful nonlinear modelling framework that can be used for regression, density estimation, prediction and classification. Within this framework, all sources of uncertainty are expressed and measured by probabilities. This formulation allows for a probabilistic treatment of our *a priori* knowledge, domain specific knowledge, model selection schemes, parameter estimation methods and noise estimation techniques.

Many researchers have contributed towards the development of the Bayesian learning approach for neural networks. This thesis advances this research by proposing several novel extensions in the areas of sequential learning, model selection, optimisation and convergence assessment. The first contribution is a regularisation strategy for sequential learning based on extended Kalman filtering and noise estimation via evidence maximisation. Using the expectation maximisation (EM) algorithm, a similar algorithm is derived for batch learning. Much of the thesis is, however, devoted to Monte Carlo simulation methods. A robust Bayesian method is proposed to estimate, jointly, the parameters, number of parameters, noise statistics and signal to noise ratios of radial basis function (RBF) networks. The necessary computations are performed using a reversible jump Markov chain Monte Carlo (MCMC) simulation method. The derivation of an efficient reversible jump MCMC simulated annealing strategy to perform global optimisation of neural networks is also included. The convergence of these algorithms is proved rigorously.

This study also incorporates particle filters and sequential Monte Carlo (SMC) methods into the analysis of neural networks. In doing so, new SMC algorithms are devised to deal with the high dimensional parameter spaces inherent to neural models. The SMC algorithms are shown to be suitable for nonlinear, non-Gaussian and non-stationary parameter estimation. In addition, they are applied to sequential noise estimation and model selection.

Keywords: Bayesian methods, neural networks, Markov chain Monte Carlo, extended Kalman filtering, EM, sequential Monte Carlo, particle filters, model selection and noise estimation.

Declaration

This thesis is the result of my own original work. Where it draws on the work of others, this is acknowledged at the appropriate points in the text. The length of the thesis, including appendices and footnotes, is approximately 64,000 words. The following publications were derived from this work:

Reviewed Journal Articles

- de Freitas, J. F. G., Niranjan, M., Gee, A. H. and Doucet, A. (1999). Sequential Monte Carlo methods to train neural network models. To appear in *Neural Computation*.

- de Freitas, J. F. G., Niranjan, M. and Gee, A. H. (1999). Hierarchical Bayesian models for regularisation in sequential learning. To appear in *Neural Computation*.

- de Freitas, J. F. G., Niranjan, M. and Gee, A. H. (1999). Dynamic learning with the EM algorithm for neural networks. To appear in *VLSI Signal Processing Systems*.

Book Chapters

- Doucet, A., de Freitas, J. F. G. and Gordon, N. J. (To appear in 2000). Introduction to sequential Monte Carlo methods. In Doucet, A., de Freitas, J. F. G., and Gordon, N. J., editors, *Sequential Monte Carlo Methods in Practice*. Springer-Verlag.

- de Freitas, J. F. G., Andrieu, C., Niranjan, M. and Gee, A. H. (To appear in 2000). Sequential Monte Carlo methods for neural networks. In Doucet, A., de Freitas, J. F. G., and Gordon, N. J., editors, *Sequential Monte Carlo Methods in Practice*. Springer-Verlag.

Reviewed Conference Proceedings

- Smith, G. A., de Freitas, J. F. G. Robinson, T. R. and Niranjan, M. (1999). Speech modelling using subspace and EM techniques. To appear in *Advances in Neural Information Processing Systems*, volume 12.

- Andrieu, C., de Freitas, J. F. G. and Doucet, A. (1999). Robust full Bayesian methods for neural networks. To appear in *Advances in Neural Information Processing Systems*, volume 12.

- de Freitas, J. F. G., Milo, M., Clarkson, P., Niranjan, M. and Gee A. H. (1999). Sequential support vector machines. In *IEEE International Workshop on Neural Networks in Signal Processing*, Winsconsin.

- Andrieu, C., de Freitas, J. F. G. and Doucet, A. (1999). Sequential MCMC for Bayesian model selection. In *IEEE Higher Order Statistics Workshop*, pages 130–134, Ceasarea, Israel.

- de Freitas, J. F. G., Niranjan, M. and Gee, A. H. (1999). Hybrid sequential Monte Carlo/Kalman methods to train neural networks in non-stationary environments. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 1057–1060, Arizona.

- de Freitas, J. F. G., Niranjan, M., Gee, A. H. and Doucet, A. (1999). Global optimisation of neural network models via sequential sampling. In Kearns, M. J., Solla, S. A., and Cohn, D., editors, *Advances in Neural Information Processing Systems*, volume 11, pages 410–416, MIT Press.

- de Freitas, J. F. G., Johnson, S. E., Niranjan, M. and Gee, A. H. (1998). Global optimisation of neural network models via sequential sampling-importance resampling. In *International Conference on Spoken Language Processing*, volume 4, pages 1311–1315, Sidney, Australia.

- de Freitas, J. F. G., Niranjan, M. and Gee, A. H. (1998). Nonlinear state space learning with EM and neural networks. In Constantinides, T., Kung, S. Y., Niranjan, M., and Wilson, E., editors, *IEEE International Workshop on Neural Networks in Signal Processing*, pages 254–263, Cambridge, England.

- de Freitas, J. F. G., Niranjan, M. and Gee, A. H. (1998). Regularisation in sequential learning algorithms. In Jordan, M. I., Kearns, M. J., and Solla, S. A., editors, *Advances in Neural Information Processing Systems*, volume 10, pages 458–464, MIT Press.

Journal Articles Under Review

- Andrieu, C., de Freitas, J. F. G. and Doucet, A. (1999). On-line Bayesian learning and model selection for radial basis functions. Submitted to *IEEE Transactions on Signal Processing*.

- Andrieu, C., de Freitas, J. F. G. and Doucet, A. (1999). Robust full Bayesian learning for neural networks. Submitted to *Neural Computation*.

Acknowledgements

During the course of this thesis, I had the fortune of having four people overseeing my work. Andrew Gee gave me the opportunity to explore my ideas, while ensuring that I was following a reasonable path. Mahesan Niranjan introduced me to the field of sequential learning and many interesting ideas in a very relaxed and enjoyable atmosphere. Christophe Andrieu and Arnaud Doucet contributed enormously towards my understanding of Markov chain Monte Carlo and sequential Monte Carlo methods, as well as rekindling my appreciation for mathematical thought.

I am most grateful to Neil Gordon for helping me when I started investigating sequential Monte Carlo methods and for his constructive criticisms at the Isaac Newton Institute.

Many more people have helped me, directly or indirectly, knowingly or unknowingly, at different stages. I would like to thank Mounir Azmy, Andrew Blake, Niclas Bergman, Carlo Berzuini, Laird Breyer, Jonathan Carr, Vasilis Digalakis, Zoubin Ghahramani, Wally Gilks, Patrick Gosling, Chris Holmes, Michael Isard, Sue Johnson, Visakan Kadirkamanathan, David Lovell, David Lowe, Alan Marrs, David Melvin, Marta Milo, Ian Nabney, Alex Nelson, Ben North, Mary Ostendorf, Will Penny, Stephen Roberts, Tony Robinson, Robert Rohling, David Saad, Gavin Smith, David Stoffer, Jaco Vermaak, Eric Wan, Mike West and the anonymous reviewers of my papers.

Over the last three years, I was supported by two University of the Witwatersrand Merit Scholarships, a South African Foundation for Research Development Scholarship, a British Academy Overseas Research Studentship and a Trinity College (Cambridge) External Research Studentship. I am most grateful for this financial assistance.

I want to thank my wife Julie for all the love, happiness and support she brought into my life. My friend Kelvin “el guapo” Meek made the ride all the more enjoyable.

To my wife and family

Notation

Symbols

$\mathbf{z}_{1:t}$	Stacked vector $\mathbf{z}_{1:t} \triangleq (z_1, \ldots, z_{j-1}, z_j, z_{j+1}, \ldots, z_t)'$.
$\mathbf{z}_{-j}$	Vector with j-th component missing $\mathbf{z}_{-j} \triangleq (z_1, \ldots, z_{j-1}, z_{j+1}, \ldots, z_k)'$.
$\mathbf{A}_{i,j}$	Entry of the matrix $\mathbf{A}$ in the i^{th} row and j^{th} column.
$\mathbf{A}_{1:p,1:q,1:r}$	Three-dimensional matrix of size $p \times q \times r$.
$\mathbf{I}_n$	Identity matrix of dimension $n \times n$.
$\mathbb{R}^n$	Euclidean n-dimensional space.
$\mathbb{N}$	The set of natural numbers (positive integers).
$p(\mathbf{z})$	Distribution of $\mathbf{z}$.
$p(\mathbf{z}\|\mathbf{y})$	Conditional distribution of $\mathbf{z}$ given $\mathbf{y}$.
$p(\mathbf{z}, \mathbf{y})$	Joint distribution of $\mathbf{z}$ and $\mathbf{y}$.
$\mathbf{z} \sim p(\mathbf{z})$	$\mathbf{z}$ is distributed according to $p\left(\mathbf{z}\right)$.
$\mathbf{z}\| \, \mathbf{y} \sim p\left(\mathbf{z}\right)$	The conditional distribution of $\mathbf{z}$ given $\mathbf{y}$ is $p\left(\mathbf{z}\right)$.
$\mathcal{B}(\mathbf{\Theta})$	Sigma field of subsets of the space $\mathbf{\Theta}$.
$\mathcal{O}(N)$	The computation complexity is order N operations.

Operators and functions

$\mathbf{A}'$	Transpose of matrix $\mathbf{A}$.
$\mathbf{A}^{-1}$	Inverse of matrix $\mathbf{A}$.
$\mathrm{tr}(\mathbf{A})$	Trace of matrix $\mathbf{A}$.
$\|\mathbf{A}\|$	Determinant of matrix $\mathbf{A}$.
$\mathbb{I}_E(\mathbf{z})$	Indicator function of the set E (1 if $\mathbf{z} \in E$, 0 otherwise).
$\delta_{\mathbf{z}_i}(d\mathbf{z})$	Dirac delta function (impulse function).
$\lfloor z \rfloor$	Highest integer strictly less than z.
$\mathbb{E}(\mathbf{z})$	Expectation of the random variable $\mathbf{z}$.
$var(\mathbf{z})$	Variance of the random variable $\mathbf{z}$.
$\exp(\cdot)$	Exponential function.
$\Gamma(\cdot)$	Gamma function.
$\log(\cdot)$	Logarithmic function of base e (ln).
min, max	Extrema with respect to an integer value.
inf, sup	Extrema with respect to a real value.
$\arg\min\limits_{\mathbf{z}}$	The argument $\mathbf{z}$ that minimises the operand.
$\arg\max\limits_{\mathbf{z}}$	The argument $\mathbf{z}$ that maximises the operand.
$\\|\mu\\|_{\mathrm{TV}}$	Total variation norm $\\|\mu\\|_{\mathrm{TV}} \triangleq \sup\limits_{A \in \mathcal{B}(\mathbf{\Theta})} \mu(A) - \inf\limits_{A \in \mathcal{B}(\mathbf{\Theta})} \mu(A)$.

Standard probability distributions

Bernoulli	$\mathcal{B}r(\alpha)$	$\alpha^{\mathbf{z}}(1-\alpha)^{(1-\mathbf{z})}$
Gamma	$\mathcal{G}a\left(\alpha,\beta\right)$	$\frac{\beta^{\alpha}}{\Gamma(\alpha)}z^{\alpha-1}\exp\left(-\beta z\right)\mathbb{I}_{[0,+\infty)}\left(z\right)$
Gaussian	$\mathcal{N}\left(\mathbf{m},\Sigma\right)$	$\lvert 2\pi\Sigma\rvert^{-1/2}\exp\left(-\frac{1}{2}\left(\mathbf{z}-\mathbf{m}\right)'\Sigma^{-1}\left(\mathbf{z}-\mathbf{m}\right)\right)$
Inverse Gamma	$\mathcal{IG}\left(\alpha,\beta\right)$	$\frac{\beta^{\alpha}}{\Gamma(\alpha)}z^{-\alpha-1}\exp\left(-\beta/z\right)\mathbb{I}_{[0,+\infty)}\left(z\right)$
Poisson	$\mathcal{P}n\left(\lambda\right)$	$\frac{\lambda^{z}}{z!}\exp(-\lambda)\mathbb{I}_{\mathbb{N}}\left(z\right)$
Student t	$\mathcal{S}t(m,\lambda,\alpha)$	$\frac{\Gamma\left(\frac{1}{2}(\alpha+1)\right)}{\Gamma\left(\frac{1}{2}\alpha\right)}\left(\frac{\lambda}{\alpha\pi}\right)^{1/2}\left[1+\alpha^{-1}\lambda(z-m)^{2}\right]^{-(\alpha+1)/2}$
Uniform	$\mathcal{U}_{A}$	$\left[\int_{A}d\mathbf{z}\right]^{-1}\mathbb{I}_{A}\left(\mathbf{z}\right)$

Abbreviations

AIC	Akaike's Information Criterion.
a.s.	Almost Surely.
BIC	Bayesian Information Criterion.
EKF	Extended Kalman Filter.
EM	Expectation Maximisation.
i.i.d.	Independent Identically Distributed.
MAP	Maximum A Posteriori.
MDL	Minimum Description Length.
MCMC	Markov Chain Monte Carlo.
MH	Metropolis-Hastings.
MLP	Multi-Layer Perceptron.
MS	Mean Square.
RBF	Radial Basis Function.
RMS	Root Mean Square.
SA	Simulated Annealing.
SIR	Sampling Importance Resampling.
SIS	Sequential Importance Sampling.
SMC	Sequential Monte Carlo.

Contents

1

Introduction

1.1 Motivation

Models are abstractions of reality to which experiments can be applied to improve our understanding of phenomena in the world. They are at the heart of science and permeate throughout most disciplines of human endeavour, including economics, engineering, medicine, politics, sociology and data management in general. Models can be used to process data to predict future events or to organise data in ways that allow information to be extracted from it.

There are two common approaches to constructing models. The first is of a deductive nature. It relies on subdividing the system being modelled into subsystems that can be expressed by accepted relationships and physical laws. These subsystems are typically arranged in the form of simulation blocks and sets of differential equations. The model is consequently obtained by combining all the sub-models.

The second approach favours the inductive strategy of estimating models from measured data. This estimation process will be referred to as "learning from data" or simply "learning" for short. In this context, learning implies finding patterns in the data or obtaining a parsimonious representation of data that can then be used for several purposes such as forecasting or classification. Learning is of paramount importance in nonlinear modelling due to the lack of a coherent and comprehensive theory for nonlinear systems.

Learning from data is an ill-posed problem. That is, there is a continuum of solutions for any particular data set. Consequently, certain restrictions have to be imposed on the form of the solution. Often *a priori* knowledge about the phenomenon being modelled is available in the form of physical laws, generally accepted heuristics or mathematical relations. This knowledge should be incorporated into the modelling process so as to reduce the set of possible solutions to one that provides reasonable results. The ill-posed nature and other inherent difficulties associated with the problem of learning can be clearly illustrated by means of a simple noisy interpolation example.

Consider the data plotted in Figure 1.1-A. It has been generated by the following equation:

$$y = x^3 + \eta$$

where x^3 represents the true function between the input x and the output y and η denotes zero mean uniformly distributed noise. The learning task is to use the noisy data points plotted in Figure 1.1-A to estimate the true relation between the input and the output.

We could attempt to model the data by polynomials of different order fitted to the data by conventional least squares techniques. Let us assume that we try to fit second and sixth order polynomials to the data. As shown in Figure 1.1-B, the 6th order polynomial approximates the data better than the second order polynomial. However, if we plot the true function and the two estimators as in Figure 1.1-C, we find that the second order estimator provides a better approximation to the true function. Moreover, the second order estimator provides a far better approximation to the true function for novel (extrapolation) data, as depicted in Figure 1.1-D.

In conclusion, very complex estimators will approximate the training data points better but may be worse estimators of the true function. Consequently, their predictions for samples not encountered in the training data set may be worse than the predictions produced by lower complexity estimators. The ability to predict well with samples not encountered in the training data set is usually referred to as generalisation in the machine learning literature. Note that if we had known the attributes of the noise term *a priori*, we could have inferred that the 6th order polynomial was fitting it. Alternatively, if we had had any data in the interval $(1, 1.5)$, we would have noticed the problems associated with using the 6th order polynomial. The last two remarks indicate, clearly, that *a priori* knowledge and the size and scope of the data set play a significant role in learning.

The previous simple example has unveiled several of the difficulties that arise when we try to infer models from noisy data, namely:

- The learning problem is ill-posed. It contains infinitely many solutions.
- Noise and limited training data pose limitations on the generalisation performance of the estimated models.
- We have to select a set of nonlinear model structures with enough capacity to approximate the true function.
- We need techniques for fitting the selected models to the data.

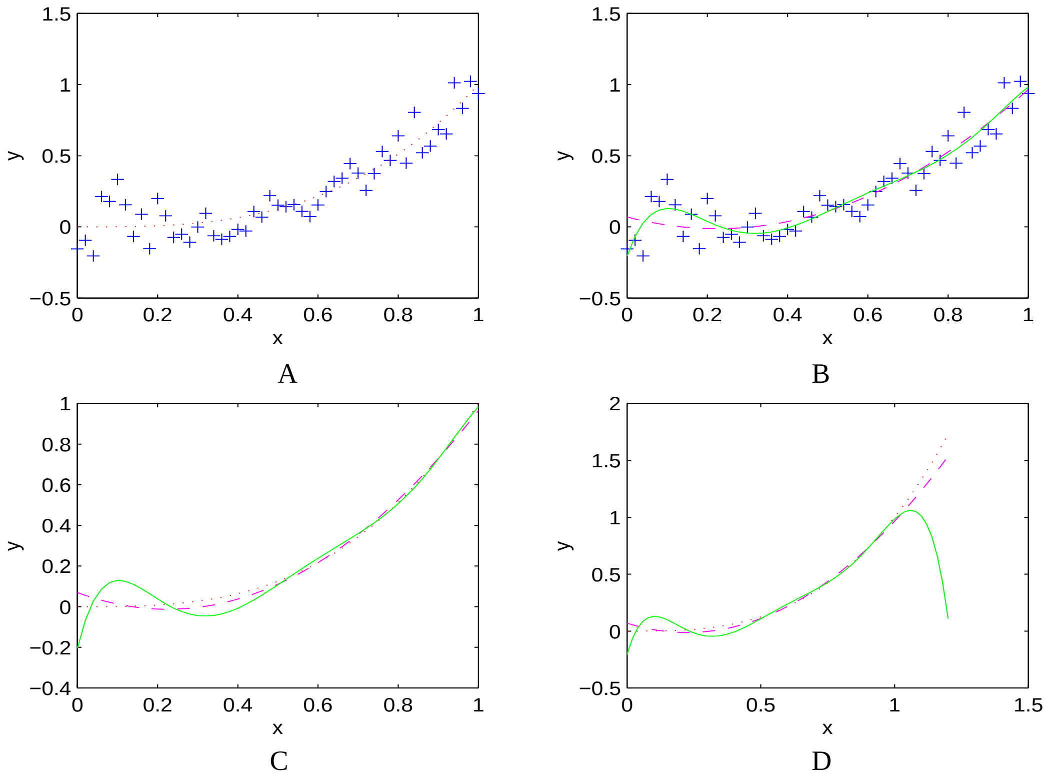

Figure 1.1 *Data generated by the system $y = x^3 + \eta$ and the true function $y = x^3$. (A) shows the noisy data and the true function [$\cdots$], (B) shows the noisy data and the approximations to it by second order [- -] and sixth order [—] polynomials, (C) shows a comparison between the true function and the estimators, and (D) depicts the performance of the estimators on novel data.*

In addition, there are other problems such as:

- The data sequences being approximated might be non-stationary. That is, the statistics of the data might change with time.

- The right set of inputs has to be selected from several possible alternatives.

Before discussing ways of dealing with these difficulties, a more precise statement of the learning problem is needed.

1.1.1 The learning problem and neural networks

Many physical processes may be described by the following nonlinear, multivariate input-output mapping:

$$\mathbf{y}_t = \mathbf{f}_t(\mathbf{x}_t) + \mathbf{n}_t \tag{1.1}$$

where $\mathbf{x}_t \in \mathbb{R}^d$ corresponds to a group of input variables, $\mathbf{y}_t \in \mathbb{R}^c$ to the output or target variables, $\mathbf{n}_t \in \mathbb{R}^c$ to an unknown noise process and $t = \{1, 2, \cdots\}$ is an index variable over the data. Depending on how the data is gathered, we can identify two types of learning: batch learning and sequential learning. In the context of batch learning, the learning problem involves computing an approximation to the function $\mathbf{f}$ and estimating the characteristics of the noise process given a set of N input-output observations:

$$\mathcal{O} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N, \mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_N\}$$

In contrast, in the sequential learning scenario, the observations arrive one at a time.

Typical instances of the learning problem include regression, where $\mathbf{y}_{1:N,1:c}$[1] is continuous; classification, where $\mathbf{y}$ corresponds to a discrete group of classes; and nonlinear dynamical system identification, where the inputs and targets correspond to several delayed versions of the signals under consideration.

The disturbances $\mathbf{n}_t$ may represent both measurement noise and unknown inputs. This study will assume that they can be added directly to the output. The basis for this assumption is that noise in the input together with other system disturbances will propagate through the system and therefore can be lumped into one single measurement noise term (Ljung, 1987). When introducing sequential Monte Carlo methods in Chapter 6, this assumption will be weakened by adopting a more general formulation: $\mathbf{y}_t = \mathbf{f}_t(\mathbf{x}_t, \mathbf{n}_t)$. In some scenarios, one might, however, be interested in modelling the distribution of the input data $p(\mathbf{x})$ (Cornford et al., 1998; Wright, 1998). This topic, however, lies beyond the scope of this thesis.

The goal of learning, as posed here, is to obtain a description of the conditional distribution $p(\mathbf{y}|\mathbf{x})$. As the dimension of this distribution can be very large, it is convenient to adopt a variational approach and project it into a lower dimensional space. This can be accomplished by introducing a set of parameters $\boldsymbol{\theta} \in \mathbb{R}^m$, leading to the distribution $p(\mathbf{y}|\boldsymbol{\theta}, \mathbf{x})$. For example, if we believe that the data has been generated by a Gaussian distribution, we only need two sufficient statistics to describe it, namely its mean and covariance. These statistics can, in turn, be described by a low-dimensional set of parameters. These parameters will allow us to infer the outputs $\mathbf{y}$ whenever we observe new values of the inputs $\mathbf{x}$.

The regression function of $\mathbf{y}$ on $\mathbf{x}$ ($\mathbf{f} : \mathbb{R}^d \mapsto \mathbb{R}^c$) is a multivariate, nonlinear and

[1] $\mathbf{y}_{1:N,1:c}$ is an N by c matrix, where N is the number of data and c the number of outputs. The notation $\mathbf{y}_{1:N,j} \triangleq (\mathbf{y}_{1,j}, \mathbf{y}_{2,j}, \ldots, \mathbf{y}_{N,j})'$ is adopted to denote all the observations corresponding to the j-th output (j-th column of $\mathbf{y}$). To simplify the notation, $\mathbf{y}_t$ is equivalent to $\mathbf{y}_{t,1:c}$. That is, if one index does not appear, it is implied that we are referring to all of its possible values. Similarly, $\mathbf{y}$ is equivalent to $\mathbf{y}_{1:N,1:c}$. The shorter notation will be favoured. The longer notation will only be invoked to avoid ambiguities and emphasise certain dependencies.

possibly time-varying mapping. When the exact nonlinear structure of this mapping cannot be established *a priori*, it may be synthesised as a combination of parametrised basis functions. That is:

$$\widehat{\mathbf{f}}_t(\mathbf{x}_t, \boldsymbol{\theta}_t) = G_{t,k}\left(\boldsymbol{\theta}_{t,k}; \left(\cdots \sum_j G_{t,j}\left(\boldsymbol{\theta}_{t,j}; \sum_i G_{t,i}(\boldsymbol{\theta}_{t,i}; \mathbf{x}_t)\right)\cdots\right)\right) \tag{1.2}$$

where $G_{t,i}(\mathbf{x}_t, \boldsymbol{\theta}_{t,i})$ denotes a multivariate basis function. These multivariate basis functions may be generated from univariate basis functions using radial basis, tensor product or ridge construction methods. This type of modelling is often referred to as "non-parametric" regression because the number of basis functions is typically very large. Equation (1.2) encompasses a large number of nonlinear estimation methods including projection pursuit regression (Friedman and Stuetzle, 1981; Huber, 1985), Volterra series (Billings, 1980; Mathews, 1991), fuzzy inference systems (Jang and Sun, 1993), generalised linear models (Nelder and Wedderburn, 1972), multivariate adaptive regression splines (MARS) (Denison, 1998; Friedman, 1991) and many artificial neural network paradigms such as functional link networks (Pao, 1989), multi-layer perceptrons (MLPs) (Rosenblatt, 1959; Rumelhart et al., 1986), radial basis function networks (RBFs) (Lowe, 1989; Moody and Darken, 1988; Poggio and Girosi, 1990), wavelet networks (Bakshi and Stephanopoulos, 1993; Juditsky et al., 1995) and hinging hyper-planes (Breiman, 1993). For an introduction to neural networks, the reader may consult any of the following books (Bishop, 1995b; Haykin, 1994; Hecht-Nielsen, 1990; Ripley, 1996).

Neural networks can approximate any continuous function arbitrarily well as the number of neurons (basis functions) increases without bound (Cybenko, 1989; Hornik et al., 1989; Poggio and Girosi, 1990). In addition, they have been successfully applied to many complex problems, including speech recognition (Robinson, 1994), hand written digit recognition (Le Cun et al., 1989), financial modelling (Refenes, 1995) and medical diagnosis (Baxt, 1990) among others. This thesis will consider two types of neural network architectures: fixed dimension MLPs and variable dimension RBFs.

MLPs have enjoyed a privileged position in the neural networks community because of their simplicity, approximating power, relation to biological systems and various historical reasons. Figure 1.2 shows a typical two hidden layer MLP with logistic sigmoid basis functions in the hidden layers and a single output linear neuron. Networks of this type can be represented mathematically as follows:

$$\widehat{\mathbf{f}}_t(\mathbf{x}_t, \boldsymbol{\theta}_t) = \sum_l \boldsymbol{\theta}_{t,l}\left(\sum_k \sigma\Big(\boldsymbol{\theta}_{t,k}\Big(\cdots \sum_j \sigma\left(\boldsymbol{\theta}_{t,j}\sum_i \sigma(\boldsymbol{\theta}_{t,i}\mathbf{x}_t + b_{t,i}) + b_{t,j}\right)\cdots\Big) + b_{t,k}\Big)\right) + b_{t,l}$$

where $b_{t,i}$ denotes the bias of the ith neuron in the first layer and $\boldsymbol{\theta}_{t,i}$ is a row vector containing the weights connecting each input with the ith neuron. The logistic sigmoid function σ is given by:

$$\sigma(\varrho) = \frac{1}{1 + \exp(-\varrho)}$$

If our goal is to perform classification, then it is convenient to employ a logistic function in the output layer. This allows us to interpret the outputs of the network as probabilities of class membership. Although the MLPs discussed in this thesis exhibit a feed-forward architecture, they can be easily extended to recurrent schemes by the addition of multiple feedback connections or tapped delay lines (de Freitas et al., 1996; Narendra and Parthasarathy, 1990; Puskorius and Feldkamp, 1994; Qin et al., 1992; Sjöberg, 1995; Yamada and Yabuta, 1993).

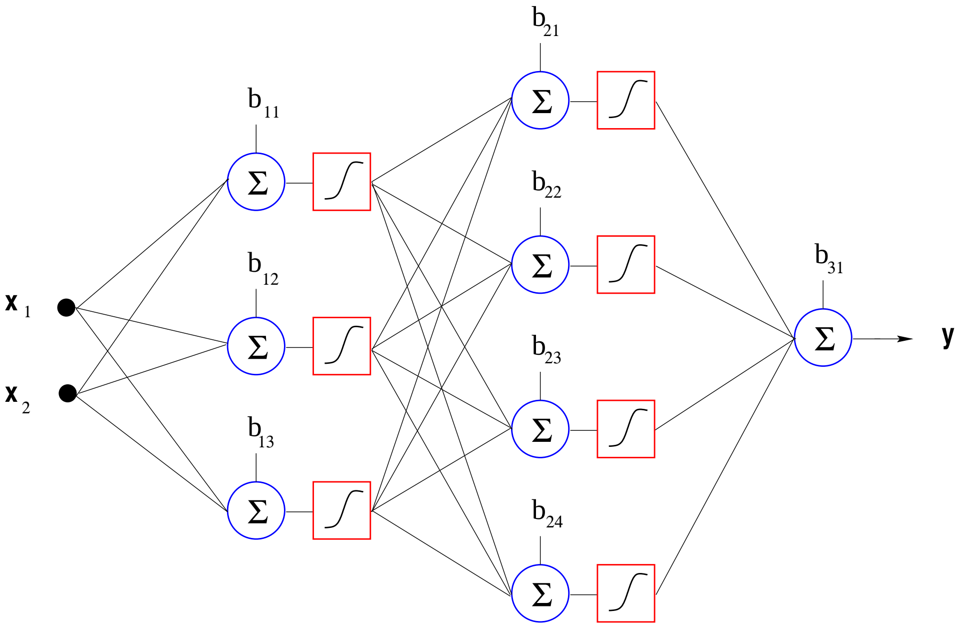

Figure 1.2 *Typical multi-layer perceptron architecture.*

RBF networks tend to be more tractable than MLPs. In these models, the training of the parameters corresponding to different layers is, to a large extent, decoupled. Chapters 5 and 7 will discuss an approximation scheme consisting of a mixture of k RBFs and a linear regression term (Holmes and Mallick, 1998). The number of basis functions will be estimated from the data. Thus, unless the data is nonlinear, the model collapses to a standard linear model. More precisely, the linear-RBF model $\mathcal{M}$ is given by:

$$\begin{array}{llr} \mathcal{M}_{t,0}: & \widehat{\mathbf{f}}_t(\mathbf{x}_t, \boldsymbol{\theta}_t) = \boldsymbol{b}_t + \boldsymbol{\beta}_t'\mathbf{x}_t & k_t = 0 \\ \mathcal{M}_{t,k}: & \widehat{\mathbf{f}}_t(\mathbf{x}_t, \boldsymbol{\theta}_t) = \sum_{j=1}^{k_t} \mathbf{a}_{t,j}\phi_t(\|\mathbf{x}_t - \boldsymbol{\mu}_{t,j}\|) + \boldsymbol{b}_t + \boldsymbol{\beta}_t'\mathbf{x}_t & k_t \geq 1 \end{array} \qquad (1.3)$$

where, in this case, $\boldsymbol{\theta} = \{\boldsymbol{b}, \boldsymbol{\beta}, \mathbf{a}, \mu\}$, $\|\cdot\|$ denotes a distance metric (usually Euclidean or Mahalanobis), $\boldsymbol{\mu}_j \in \mathbb{R}^d$ denotes the j-th RBF centre for a model with k RBFs, $\mathbf{a}_j \in \mathbb{R}^c$ denotes the j-th RBF amplitude and $\boldsymbol{b} \in \mathbb{R}^c$ and $\boldsymbol{\beta} \in \mathbb{R}^d \times \mathbb{R}^c$ denotes the linear regression parameters. Figure 1.3 depicts the approximation model for $k = 3$, $c = 2$ and $d = 2$. Depending on our *a priori* knowledge about the smoothness of the mapping, we can choose different types of basis functions (Girosi et al., 1995). The most common choices are:

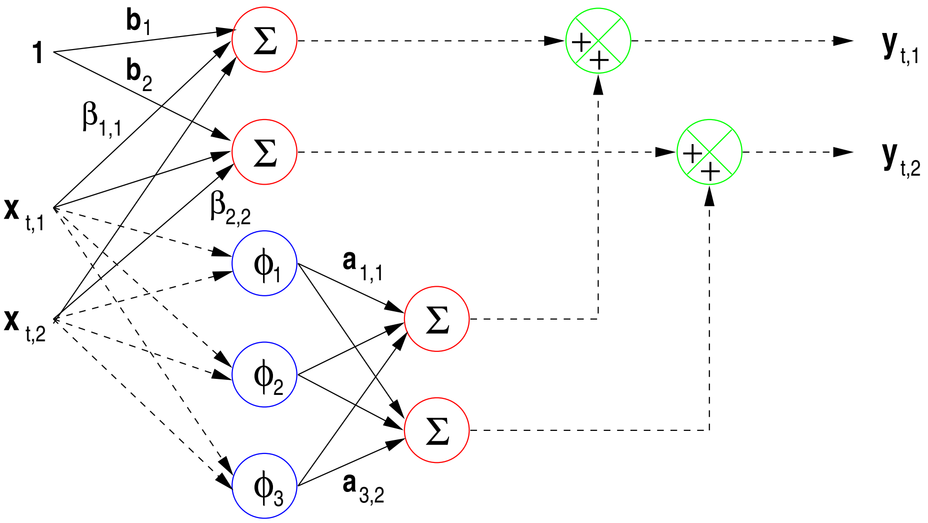

Figure 1.3 *Linear-RBF approximation model with three radial basis functions, two inputs and two outputs. The solid lines indicate weighted connections.*

- Linear: $\phi(\varrho) = \varrho$
- Cubic: $\phi(\varrho) = \varrho^3$
- Thin plate spline: $\phi(\varrho) = \varrho^2 \ln(\varrho)$
- Multi-quadric: $\phi(\varrho) = \left(\varrho^2 + \lambda^2\right)^{1/2}$
- Gaussian: $\phi(\varrho) = \exp\left(-\lambda\varrho^2\right)$

For the last two choices of basis functions, λ will be treated as a user-set parameter. Nevertheless, the Monte Carlo estimation strategies described in Chapters 5, 6 and 7 can treat the choice of basis functions as a model selection problem. It is possible to place a prior distribution on the basis functions and allow the Monte Carlo algorithms to decide which of them provide a better solution.

1.2 Scope and Contributions of this Thesis

In the example presented earlier, it was shown that the ability of a model to predict accurately with novel data depends on the amount of data, the complexity of the model and the noise in the data. It was then argued that artificial neural networks provide a general and flexible nonlinear modelling strategy. From this standpoint, the learning problem involves estimating the neural network's parameters, the number of parameters, the type of basis functions and the statistics of the noise. In addition, we might have to select the most appropriate set of input signals.

A great deal of effort has been devoted to the solution of the parameter estimation problem. The other problems have received less attention. In contrast, the issues of noise estimation and model selection will be central to the scope of this thesis. It will be possible to manage these more demanding tasks by embracing the Bayesian learning paradigm. Despite the fact that the problems of input variable selection and basis function selection are not treated explicitly, the solution to these is a natural extension of the model selection frameworks presented in Chapters 6 and 7.

Another important theme in this thesis is the issue of sequential learning and inference. Sequential training methods for neural networks are important in many applications involving real-time signal processing, where data arrival is inherently sequential. Furthermore, one might wish to adopt a sequential processing strategy to deal with non-stationarity in signals, so that information from the recent past is given greater weight than information from the distant past. Computational simplicity in the form of not having to store all the data might also constitute an additional motivating factor for sequential methods.

This thesis proposes the following:

- A novel approach to perform regularisation in sequential learning. This approach establishes theoretical links between extended Kalman filters with adaptive noise estimation, gradient descent methods with multiple adaptive learning rates and training methods with multiple smoothing regularisation coefficients.

- An expectation maximisation (EM) algorithm to estimate the parameters of an MLP, the noise statistics and the model uncertainty jointly. The method is applicable to non-stationary parameter spaces.

- A robust Bayesian method to estimate, jointly, the parameters, number of parameters, noise statistics and signal to noise ratios of an RBF network. The necessary computations are performed using a reversible jump Markov chain Monte Carlo (MCMC) simulation method. In addition, it presents an efficient reversible jump MCMC simulated annealing strategy to perform global optimisation of RBF networks. Furthermore, it proves the convergence of these algorithms rigorously[2].

- The use of particle filters and sequential Monte Carlo (SMC) methods to the neural networks field. In doing so, new SMC algorithms are devised to deal with the high dimensional parameter spaces inherent to neural network models. These algorithms are suitable for nonlinear, non-Gaussian and non-stationary modelling.
- A new and general sequential Monte Carlo approach to perform sequential noise estimation and model selection. The method is demonstrated on RBF networks.

1.3 Thesis Organisation

The chapters are, to a large extent, self contained and can be read independently. Chapter 2 is of an introductory nature. At the end of each of the main chapters, Chapters 3 to 7, the proposed methods and algorithms are demonstrated on experiments with synthetic data. In Chapter 8, further tests on a few real problems are presented. A summary of the thesis follows:

Chapter 2: Learning and Generalisation

This chapter provides a brief review of learning theory from a neural networks perspective. It addresses both the classical and Bayesian approaches. In addition, it introduces the sequential learning problem.

Chapter 3: Sequential Bayesian Learning with Gaussian Approximations

Sequential learning methods, in particular Gaussian approximation schemes, are introduced in this chapter. It is shown that an hierarchical Bayesian modelling approach enables one to perform regularisation in sequential learning. Three inference levels are identified within this hierarchy, namely model selection, parameter estimation and noise estimation. In environments where data arrives sequentially, techniques such as cross-validation to achieve regularisation or model selection are not possible. The Bayesian approach, with extended Kalman filtering at the parameter estimation level, allows for regularisation within a minimum variance framework. A multi-layer perceptron is used to generate the extended Kalman filter nonlinear measurements mapping. Several algorithms are described at the noise estimation level, thus permitting the implementation of on-line regularisation. Another contribution of this chapter is to show

[2] These contributions were strongly motivated by the work of Christophe Andrieu and Arnaud Doucet (Andrieu and Doucet, 1998b; Andrieu and Doucet, 1998a; Andrieu and Doucet, 1999).

the theoretical links between adaptive noise estimation in extended Kalman filtering, multiple adaptive learning rates and multiple smoothing regularisation coefficients.

Chapter 4: Dynamic Batch Learning with the EM Algorithm

This chapter extends the sequential Gaussian approximation framework discussed in the previous chapter to the batch learning scenario. In it, an EM algorithm for nonlinear state space models is derived. It is used to estimate jointly the neural network weights, the model uncertainty and the noise in the data. In the E-step, a forward-backward Rauch-Tung-Striebel smoother is adopted to compute the network weights. For the M-step, analytical expressions are derived to compute the model uncertainty and the measurement noise. The method is shown to be intrinsically very powerful, simple and stable.

Chapter 5: Robust Full Bayesian Learning with MCMC

This chapter begins the presentation of Monte Carlo methods, a major theme in this thesis. The reversible jump MCMC simulation algorithm is applied to RBF networks, so as to compute the joint posterior distribution of the radial basis centres and the number of basis functions. This area of research is advanced in three important directions. First, a robust prior for RBF networks is proposed. That is, the results do not depend on any heuristics or thresholds. Second, an automated growing and pruning reversible jump MCMC optimisation algorithm is designed to choose the model order according to classical AIC, BIC and MDL criteria. This MCMC algorithm estimates the maximum of the joint likelihood function of the radial basis centres and the number of bases using simulated annealing. Finally, some geometric convergence theorems for the proposed algorithms are presented.

Chapter 6: Sequential Monte Carlo Methods

Here, a novel strategy for training neural networks using sequential Monte Carlo (SMC) algorithms is discussed. Various hybrid gradient descent/sampling importance resampling algorithms are proposed. In terms of both modelling flexibility and accuracy, SMC algorithms provide a clear improvement over conventional Gaussian schemes. These algorithms may be viewed as a global learning strategy to learn the probability distributions of the network weights and outputs in a sequential framework. They are also well suited to applications involving on-line, nonlinear and non-Gaussian signal processing.

Chapter 7: Sequential Bayesian Model Selection

This chapter extends the model selection strategy discussed in Chapter 5 to the sequential learning case. This problem does not usually admit any type of closed-form analytical solutions and, as a result, one has to resort to numerical methods. The chapter proposes an original sequential simulation-based strategy to perform the necessary computations. It combines sequential importance sampling, a selection procedure and reversible jump MCMC moves. The effectiveness of the method is demonstrated by applying it to radial basis function networks.

Chapter 8: Applications

This chapter demonstrates the performance of the various methods on some interesting real data sets. It includes comprehensive comparisons between the proposed algorithms.

Chapter 9: Conclusions

This final chapter summarises the theoretical and experimental results. It discusses their relevance and suggests a few directions for further research.

Appendices

The appendices contain derivations, proofs of convergence and other ancillary information. In particular, they include a Bayesian derivation of the Kalman filter, an example on how to compute the Jacobian matrix for MLPs, the proof of an inequality used in the derivation of the EM algorithm, an introduction to MCMC simulation and a proof of convergence for the algorithms presented in Chapter 5.

2

Learning and Generalisation

The previous chapter provided a glimpse at the learning and generalisation problems. There it was hinted, by means of a simple example, that in order to obtain a good representation of the process being modelled, one needs to estimate the model complexity, parameters and noise characteristics. In addition, it was mentioned that it is beneficial to incorporate *a priori* knowledge so as to mitigate the ill-conditioned nature of the learning problem. If we follow these specifications, we can almost assuredly obtain a model that generalises well.

This chapter will briefly review the classical approaches to learning and generalisation in the neural networks field. Aside from regularisation with noise and committees of estimators, most of the standard methods fall into two broadly overlapping categories: penalised likelihood and predictive assessment methods. Penalised likelihood methods involve placing a penalty term either on the model dimension or on the smoothness of the response (Hinton, 1987; Le Cun et al., 1990; Poggio and Girosi, 1990). Predictive assessment strategies, such as the cross-validation, jacknife or bootstrap methods (Ripley, 1996; Stone, 1974; Stone, 1978; Wahba and Wold, 1969), typically entail dividing the training data set into S_N distinct subsets. The model is subsequently trained using $S_N - 1$ of the subsets and its performance is validated on the omitted subset. The procedure is repeated for each of the subsets. This predictive assessment is often used to set the penalty parameters in the penalised likelihood formulations.

These methods tend to lack a general and rigorous framework for incorporating *a priori* knowledge into the modelling process. Furthermore, they do not provide suitable foundations for the study of generalisation in sequential learning. To surmount these limitations, the Bayesian learning paradigm will be adopted in this thesis. This approach will allow us to incorporate *a priori* knowledge into the modelling process and to compute, jointly and within a probabilistic framework, the model parameters, noise characteristics, model structure and regularisation coefficients. It will also allow us to do this sequentially.

2.1 Traditional Approaches

One of the most popular approaches to train neural networks has been to compute an estimator $\widehat{\mathbf{f}}(\mathbf{x}, \widehat{\boldsymbol{\theta}})$ of the regression function $\mathbf{f}(\mathbf{x})$ by minimising the following mean square empirical risk:

$$R_e[\widehat{\mathbf{f}}(\mathbf{x}, \widehat{\boldsymbol{\theta}})] = \frac{1}{N}\sum_{t=1}^{N}(\mathbf{y}_t - \widehat{\mathbf{f}}(\mathbf{x}_t, \widehat{\boldsymbol{\theta}}_t))^2$$

The minimisation is often performed by unconstrained gradient descent methods, such as back-propagation or conjugate gradients (Bishop, 1995b). This approach causes two types of error, namely the approximation and estimation errors.

The approximation error arises because the exact nonlinear behaviour of the regression function is seldom known and, consequently, $\mathbf{f}(\mathbf{x})$ has to be approximated by a combination of parameterised basis functions $\widehat{\mathbf{f}}(\mathbf{x}, \boldsymbol{\theta})$. If the model structure has enough capacity to approximate the regression function, the approximation error will tend to zero as the number of parameters increases.

The estimation error is the result of our lack of knowledge about the conditional distribution $p(\mathbf{y}|\boldsymbol{\theta}, \mathbf{x})$. Likelihood methods do not attempt to estimate this distribution but instead minimise the empirical risk. One of the heuristic reasons for doing this is that the regression function minimises the expected risk or $L^2(p)$ norm. That is:

$$\mathbf{f}(\cdot) = \arg \min_{\mathbf{h}(\cdot) \in \boldsymbol{\mathcal{T}}} R[\mathbf{h}(\cdot)]$$

where $\mathbf{h}(\cdot)$ denotes the possible hypothesis and $\boldsymbol{\mathcal{T}}$ represents the target space where the regression function lies. The estimator $\widehat{\mathbf{f}}(\mathbf{x}, \widehat{\boldsymbol{\theta}})$ lies on a hypothesis space $\mathcal{H}$ as shown in Figure 2.1. $R[\mathbf{h}(\cdot)]$ corresponds to the mean square expected risk, given by:

$$\begin{aligned} R[\mathbf{h}(\cdot)] &= \|\mathbf{y} - \mathbf{h}(\mathbf{x}, \boldsymbol{\theta})\|^2_{L^2(p)} = \mathbb{E}[(\mathbf{y} - \mathbf{h}(\mathbf{x}, \boldsymbol{\theta}))^2] \\ &= \int_{\mathbb{R}^{c+m}} (\mathbf{y} - \mathbf{h}(\mathbf{x}, \boldsymbol{\theta}))^2 p(\mathbf{y}, \boldsymbol{\theta}|\mathbf{x}) \mathrm{d}\mathbf{y}\mathrm{d}\boldsymbol{\theta} \end{aligned} \qquad (2.1)$$

From a statistical point of view, the predictor $\widehat{\mathbf{f}}(\mathbf{x}, \widehat{\boldsymbol{\theta}})$ obtained by empirical error minimisation will approximate $\widehat{\mathbf{f}}(\mathbf{x}, \boldsymbol{\theta})$ as the number of data increases without bound. It can also be expected that as the number of parameters increases, the expression for the empirical risk becomes more complex and, therefore, the estimation error can increase (Niyogi and Girosi, 1994).

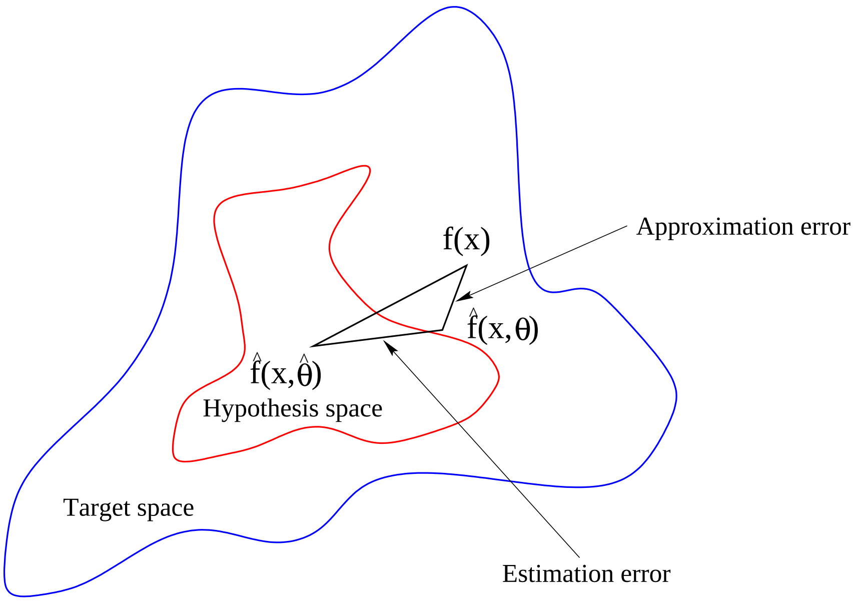

Figure 2.1 *Graphical representation of the approximation and estimation errors.*

To summarise, the approximation error is inversely related to the number of model parameters, while the estimation error is directly related to the number of parameters. This tradeoff is well known as the bias/variance tradeoff (Bishop, 1995b; de Freitas, 1997; Geman et al., 1992; Haykin, 1994; Sjöberg et al., 1995; White, 1989). The mean square error of the function $\widehat{\mathbf{f}}(\mathbf{x},\widehat{\boldsymbol{\theta}})$ as an estimator of the regression function may be decomposed into the following two terms:

$$\mathbb{E}\Big(\big(\widehat{\mathbf{f}}(\mathbf{x},\widehat{\boldsymbol{\theta}}) - \mathbf{f}(\mathbf{x})\big)^2\Big) = \underbrace{\mathbb{E}\Big(\big(\widehat{\mathbf{f}}(\mathbf{x},\widehat{\boldsymbol{\theta}}) - \mathbb{E}\big(\widehat{\mathbf{f}}(\mathbf{x},\widehat{\boldsymbol{\theta}})\big)\big)^2\Big)}_{Variance} + \underbrace{\big(\mathbb{E}\big(\widehat{\mathbf{f}}(\mathbf{x},\widehat{\boldsymbol{\theta}})\big) - \mathbf{f}(\mathbf{x})\big)^2}_{Bias}$$

The bias term measures the distance between the average estimator and the regression function, while the variance term quantifies the spread of the estimator with respect to the data distribution. To achieve good modelling performance, both the bias and the variance would have to be small. Unfortunately, with increasing model complexity, the variance term increases while the bias term decreases. This tradeoff was clearly observed in the example of Chapter 1. The second order polynomial was biased with insignificant variance error. On the other hand the sixth order polynomial was unbiased but exhibited a large variance error.

Two interesting bounds on the generalisation error for logistic MLPs and radial basis function networks have been derived by Barron and Niyogi and Girosi respectively (Barron, 1993; Niyogi and Girosi, 1994). These bounds are based on a lemma by Jones on the convergence rate of particular iterative approximation schemes (Barron, 1993; Breiman, 1993; Girosi and Anzellotti, 1995; Jones, 1992) and on the Vapnik-Chervonenkis dimension (Vapnik, 1982). The bound for MLP's is given by:

$$\mathbb{E}[(\mathbf{f}(\mathbf{x}) - \widehat{\mathbf{f}}(\mathbf{x}, \widehat{\boldsymbol{\theta}}))^2]_{\text{MLP}} \leq \mathcal{O}\left(\frac{1}{m}\right) + \mathcal{O}\left(\frac{md\ln(N)}{N}\right)$$

where $\mathcal{O}$ denotes the order of convergence, d is the dimension of the input, m is the number of model parameters and N is the number of data. The bound for radial basis functions is similar to the one for MLPs:

$$\mathbb{E}[(\mathbf{f}(\mathbf{x}) - \widehat{\mathbf{f}}(\mathbf{x}, \widehat{\boldsymbol{\theta}}))^2]_{\text{RBF}} \leq \mathcal{O}\left(\frac{1}{m}\right) + \mathcal{O}\left(\left[\frac{md\ln(mN) - \ln(\alpha)}{N}\right]^{1/2}\right)$$

where α is a small number. On the basis of these bounds, the typical dependence of the generalisation error on the number of parameters and samples can be plotted as shown in Figure 2.2.

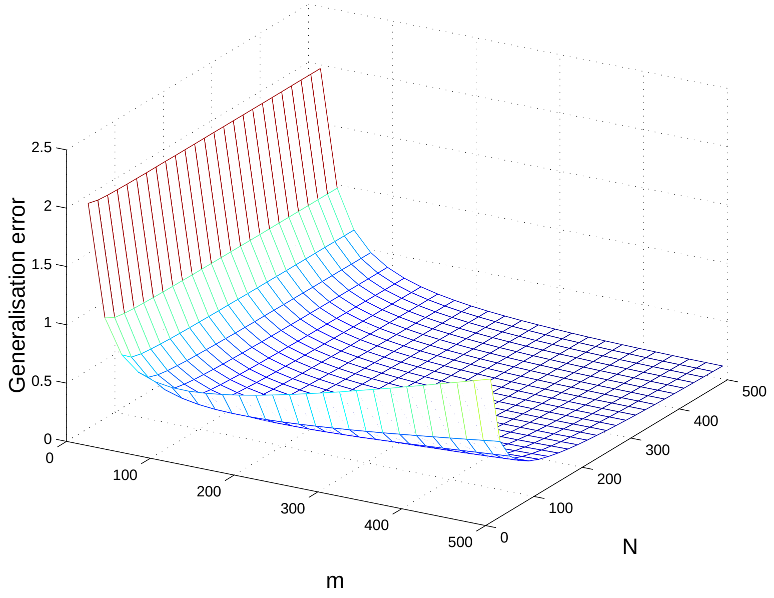

Figure 2.2 *The generalisation error dependence on the number of parameters (m) and data (N).*

Theoretically, the convergence bounds can be used to determine the optimal choice of the number of parameters. By taking their derivatives with respect to the number of parameters and equating them to zero, one can find relations for the optimal number of parameters as a function of the number of data and the input dimension:

$$m^{\star}_{\text{MLP}} \propto \left(\frac{N}{d\ln(N)}\right)^{1/2} \qquad \text{and} \qquad m^{\star}_{\text{RBF}} \propto \left(\frac{N}{d\ln(N)}\right)^{1/3}$$

It should be noted that the equations above are proportionalities and not equalities. That is, there is no knowledge about the constants of proportionality. Moreover, one should not be led erroneously to the conclusion that the estimation problem can be solved by simply using these theoretical relations of proportionality. The optimisation algorithms often converge to local minima (Saarinen et al., 1993). Typical error surfaces will, therefore, look more complex than the one depicted in Figure 2.2. A few examples on how optimisation algorithms affect the dependence of the generalisation error on the number of parameters and the size of the data set are presented in (Lawrence et al., 1996). Chapters 5 and 6 will describe global simulation methods that can mitigate the problem of local minima, in addition to being able to estimate the number of model parameters.

Within the learning paradigm discussed so far, the best one can do to obtain an acceptable generalisation performance is to balance the bias and variance terms. The only ways to reduce the bias and variance error terms simultaneously are to either increase the number of data or to model the noise characteristics and incorporate *a priori* knowledge about the form of the estimator. It will be shown later that the Bayesian learning paradigm allows us to accomplish this in a probabilistic setting.

One way, perhaps the simplest, of making use of *a priori* knowledge is to impose smoothness constraints on the model. That is, small changes in the input should lead to small changes in the output. This scheme is known as regularisation. It reduces the infinite number of possible solutions to the learning problem to one that balances the bias and variance error terms.

To obtain a function that is simultaneously close to the data and smooth, the empirical modelling error criterion may be extended as follows:

$$R_r[\widehat{\mathbf{f}}(\mathbf{x},\widehat{\boldsymbol{\theta}})] = \frac{1}{N}\sum_{t=1}^{N}(\mathbf{y}_t - \widehat{\mathbf{f}}(\mathbf{x}_t,\widehat{\boldsymbol{\theta}}_t))^2 + \nu\Omega$$

where ν is a positive parameter that serves to balance the tradeoff between smoothness and data approximation. A large value of ν places more importance on the smoothness of the model, while a small value of ν places more emphasis on fitting the data. The functional Ω penalises excessive model complexity. The regularisation parameter is often obtained by cross-validation.

Several methods have been proposed for the design of the regularisation functional. Girosi, Jones and Poggio (Girosi et al., 1995) have proposed the following functional:

$$\Omega = \int_{\mathbb{R}^d} \frac{|\tilde{\mathbf{f}}(\mathbf{s})|^2}{\widetilde{F}(\mathbf{s})}\mathrm{d}\mathbf{s}$$

where the tildes indicate Fourier transforms and $1/\widetilde{F}(\mathbf{s})$ is chosen to be a high-pass filter. In other words, the functional returns the high frequency components (oscillations) of the mapping. Therefore, a large value of ν simply indicates that any excessive oscillation will constitute a major contribution to the modelling error. This approach, in connection with one hidden layer neural networks, has led Girosi, Jones and Poggio to the formulation of generalised regularisation networks. From a unifying theoretical point of view, their work is particularly interesting since they show how different choices of $\widetilde{F}(\mathbf{s})$ may lead to various approximation schemes including radial basis, tensor splines and additive models.

Weight decay (Hinton, 1987) is another very popular choice of regularisation functional. It is given by:

$$\Omega = \sum_{i=1}^{m} \theta_i^2$$

One of the reasons for using weight decay is that superfluous parameters are forced to decay to zero. Previously, it was discussed that the generalisation performance deteriorates if the number of parameters increases excessively. Using weight decay, only a few of the parameters contribute to the mapping and hence the generalisation error decreases. From an intuitive point of view, we should notice that for MLPs with very small weights, the network outputs become approximately linear functions of the inputs. This is a consequence of the fact that logistic sigmoid functions are approximately linear for small values of their arguments.

Other common approaches to controlling the complexity of the estimates include early stopping, training with noise, committees (mixtures) of networks and growing and pruning techniques. Early stopping is a predictive assessment method that involves partitioning the data into two sets, a validation set and a training set. During training, the performance of the estimator is periodically tested on the validation set. As soon as the validation error starts increasing, training stops. Early stopping has several shortcomings. Firstly, the amount of training data is usually halved. Secondly, the estimator will be biased towards the validation set, thus requiring an extra test set. Finally, early stopping relies on the assumption that the path taken through parameter space by the optimisation algorithm passes through an acceptable solution. This is not always the case with multi-modal error surfaces.

It has been shown that training with noise added to the inputs is equivalent to regularisation (Bishop, 1995a; Leen, 1995; Wu and Moody, 1996). The minimisation of the empirical risk (R_e) with noise, of “small” amplitude, added to the input data is equivalent to minimisation of the regularised risk (R_r). Bishop (Bishop, 1995b) sheds some light on this topic by stating that the heuristic basis for training with noise is that the noise will “smear out” each data point and preclude the network from fitting individual data points precisely, and hence will reduce over-fitting.

Several researchers have argued that the performance of estimators may be considerably improved by combining several estimators of different complexity and model structure (Jacobs, 1995; Perrone, 1995; Perrone and Cooper, 1993): see (Genest and

Zidek, 1986) for a comprehensive review. If the individual models are trained so that their variance error terms are bigger than their bias error terms, then model combination may reduce the variance error component, as it involves averaging over all the estimates. It can be argued that combining models, in this way, is a brittle strategy. The resulting model still needs to be subject to the same model choice criteria as the individual models.

Finally, some effort has also been devoted to the study of growing and pruning techniques. The idea behind growing and pruning algorithms is to control the complexity of the estimator by eliminating and adding parameters to the estimator as the data is processed. Examples of this type of algorithm include the upstart algorithm (Frean, 1990), cascade correlation (Fahlman and Lebiere, 1988), optimal brain damage (Le Cun et al., 1990) and the resource allocating network (Platt, 1991). Chapter 5 will present a deeper discussion of these techniques and mention some of their shortcomings.

2.2 The Bayesian Learning Paradigm

The Bayesian learning paradigm is founded upon the premise that all forms of uncertainty should be expressed and measured by probabilities (Bernardo and Smith, 1994). Although the paradigm can be expressed in formal terms, based on mathematical abstraction and rigorous analysis, it relies upon subjective experience. That is, it offers a rationalist and coherent theory where individuals' uncertainties are described in terms of subjective probabilities. However, once the individuals' views of uncertainty are specified, and assuming they have access to the same data, the results should be unique and reproduceable.

At the centre of the Bayesian paradigm is a simple and extremely important expression known as Bayes' rule. Given some data $\mathbf{d}_{1:N} \triangleq \{\mathbf{x}_{1:N}, \mathbf{y}_{1:N}\}$ and a set of models to describe it $\mathcal{M}_k$, $k = 0, 1, 2, \ldots$, the expression for Bayes' rule is:

$$\begin{aligned} p(\mathcal{M}_i|\mathbf{d}_{1:N}) &= \frac{p(\mathbf{d}_{1:N}|\mathcal{M}_i)}{p(\mathbf{d}_{1:N})} p(\mathcal{M}_i) \\ &= \frac{p(\mathbf{d}_{1:N}|\mathcal{M}_i)}{\sum_k p(\mathbf{d}_{1:N}|\mathcal{M}_k) p(\mathcal{M}_k)} p(\mathcal{M}_i) \end{aligned}$$

The various distributions in the rule are known as the posterior, likelihood, prior and evidence (also known as the innovation or predictive distribution) in the following order:

$$\text{Posterior} = \frac{\text{Likelihood}}{\text{Evidence}} \text{Prior}$$

Our subjective beliefs and views of uncertainty are expressed in the prior. Once the data becomes available, the likelihood allows us to update these beliefs. The resulting posterior distribution incorporates both our a priori knowledge and the information conveyed by the data.

Let us assume that the parameter space for a generic neural network can be written as a finite union of subspaces $\boldsymbol{\Theta} = \left(\cup_{k=0}^{k_{\max}} \{k\} \times \boldsymbol{\Theta}_k \right)$, where k denotes the number of parameters, while $\boldsymbol{\Theta}_k$ represents the parameter space for model order k. For example, the parameters $\boldsymbol{\theta} \in \boldsymbol{\Theta}_k$ may include the network weights and the noise statistics. It is natural to assume that there is an inherent uncertainty about the values of the parameters and their number and that this uncertainty can be modelled by a prior distribution $p(k, \boldsymbol{\theta})$. Once the data is gathered, we can obtain the posterior $p(k, \boldsymbol{\theta}|\mathbf{y}_{1:N}, \mathbf{x}_{1:N})$ using Bayes' rule. The Bayesian paradigm allows us to introduce our beliefs about the noise characteristics and the complexity of the network into the modelling process via the prior and likelihood distributions. A result of modelling the uncertainty in the model complexity and the noise is that, once these quantities are properly estimated, we can obtain models that generalise well.

Since the posterior embodies all the statistical information about the parameters and their number given the measurements and the prior, one can "theoretically" obtain all features of interest by standard probability marginalisation and transformation techniques. For instance, we can estimate the predictive density:

$$p(\mathbf{y}_{N+1}|\mathbf{x}_{1:N+1}, \mathbf{y}_{1:N}) = \int_{\boldsymbol{\Theta}} p(\mathbf{y}_{N+1}|k, \boldsymbol{\theta}, \mathbf{x}_{N+1}) p(k, \boldsymbol{\theta}|\mathbf{x}_{1:N}, \mathbf{y}_{1:N}) \mathrm{d}k \mathrm{d}\boldsymbol{\theta} \tag{2.2}$$

and consequently forecast quantities of interest, such as:

$$\mathbb{E}(\mathbf{y}_{N+1}|\mathbf{x}_{1:N+1}, \mathbf{y}_{1:N}) = \int_{\boldsymbol{\Theta}} \widehat{\mathbf{f}}(k, \boldsymbol{\theta}, \mathbf{x}_{N+1}) p(k, \boldsymbol{\theta}|\mathbf{x}_{1:N}, \mathbf{y}_{1:N}) \mathrm{d}k \mathrm{d}\boldsymbol{\theta} \tag{2.3}$$

Note that the predictions must be based on all possible values of the network parameters weighted by their probability in view of the training data. The posterior distribution also enables us to evaluate posterior model probabilities $p(k|\mathbf{x}, \mathbf{y})$, which can be used to perform model selection by selecting the model order as $\underset{k \in \{0, \cdots, k_{\max}\}}{\arg\max} \; p(k|\mathbf{x}, \mathbf{y})$. In addition, we can perform parameter estimation by computing, for example, any of the following classical estimates:

MAP estimate : Maximise the probability such that the solution is the largest mode (peak) of $p(\boldsymbol{\theta}|k, \mathbf{x}, \mathbf{y})$. For uniform fixed priors, the resulting solution is the maximum likelihood estimate.

Minimum variance estimate : Minimise the error function $\int \|\boldsymbol{\theta} - \widehat{\boldsymbol{\theta}}\|^2 p(\boldsymbol{\theta}|k, \mathbf{x}, \mathbf{y}) \mathrm{d}\boldsymbol{\theta}$ so that the estimate is the expected value or conditional mean $\mathbb{E}(\boldsymbol{\theta}|k, \mathbf{x}, \mathbf{y})$.

These estimates are illustrated in Figure 2.3.

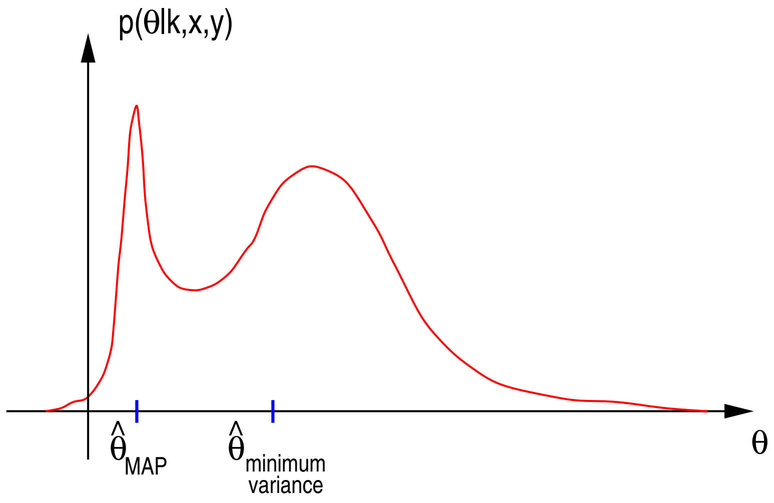

Figure 2.3 *Parameter estimation criteria based on the marginal posterior distribution.*

Within the Bayesian paradigm, learning is posed as an integration problem. The integrals appear whenever we attempt to carry out normalisation, marginalisation or expectations (Bernardo and Smith, 1994; Gelman et al., 1995). To solve these, typically high-dimensional, integrals, we can either resort to analytical integration, approximation methods, numerical integration or Monte Carlo simulation. Many real-world signal processing problems involve elements of non-Gaussianity, nonlinearity and nonstationarity, thus precluding the use of analytical integration. Approximation methods, such as Gaussian approximation and variational methods, are easy to implement. In addition, they tend to be very efficient from a computational point of view. Yet, they do not take into account all the salient statistical features of the processes under consideration, thereby often leading to poor results. Numerical integration in high dimensions is far too computationally expensive to be of any practical use. Monte Carlo methods provide the middle ground. They lead to better estimates than the approximate methods. This occurs at the expense of extra computing requirements, but the advent of cheap and massive computational power, in conjunction with some recent developments in applied statistics, means that many of these requirements can now be met. Monte Carlo methods are very flexible in that they do not require any assumptions about the probability distributions of the data. From a Bayesian perspective, Monte Carlo methods allow one to compute the full posterior probability distribution. The remaining chapters shall treat, in more detail, the problems of Gaussian approximation and Monte Carlo methods in the neural networks context.

In the past, there have been a number of attempts to apply the Bayesian learning paradigm to neural networks. In the early nineties, Buntine and Weigend (1991) and Mackay (1992) showed that a principled Bayesian learning approach to neural networks can lead to many improvements. For instance, Mackay showed that by approximating the distributions of the weights with Gaussians and adopting smoothing priors, it is possible to obtain estimates of the weights and output variances and to automatically set the regularisation coefficients.

Neal (1996) cast the net much further by introducing advanced Bayesian simulation methods, specifically the hybrid Monte Carlo method (Brass et al., 1993; Duane et al., 1987), into the analysis of neural networks. Theoretically, he also proved that certain classes of priors for neural networks, whose number or hidden neurons tends to infinity, converge to Gaussian processes.

More recently, Rios Insua and Müller (1998) , Marrs (1998) and Holmes and Mallick (1998) have addressed the issue of selecting the number of hidden neurons with growing and pruning algorithms from a Bayesian perspective. In particular, they apply the reversible jump Markov chain Monte Carlo (MCMC) algorithm of Green (Green, 1995; Richardson and Green, 1997) to feed-forward sigmoidal networks and radial basis function networks to obtain joint estimates of the number of neurons and weights. Once again, their results indicate that it is advantageous to adopt the Bayesian framework and MCMC methods to perform model order selection. There has also been some recent work on designing uninformative priors for MLPs and performing model selection with the Bayesian information criterion (Lee, 1999).

The Bayesian learning approach also has the advantage of being well-suited to the problem of sequential learning, as shown in the next section.

2.3 Sequential Learning

The representation of a dynamical system given by equation (1.1) is adequate from a functional approximation perspective. In sequential learning, however, the model parameters vary with time. A representation that reflects this behaviour would be more useful. The state space representation of a discrete stochastic dynamical system is a suitable alternative. It is given by the following two relations:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \mathbf{u}_t \tag{2.4}$$

$$\mathbf{y}_t = \widehat{\mathbf{f}}_t(\mathbf{x}_t, \boldsymbol{\theta}_t) + \mathbf{v}_t \tag{2.5}$$

where it has been assumed that the model parameters constitute the states of the system. The noise terms are often called the process noise ($\mathbf{u}_t$) and the measurement noise ($\mathbf{v}_t$). Equation (2.4) defines a first order Markov transition prior $p(\boldsymbol{\theta}_{t+1}|\boldsymbol{\theta}_t)$, while equation (2.5) defines the likelihood of the observations $p(\mathbf{y}_t|\boldsymbol{\theta}_t)$. The problem is completely defined by specifying the prior distribution $p(\boldsymbol{\theta}_0)$.

The posterior distribution $p(\boldsymbol{\theta}_{0:t}|\mathbf{x}_{1:t}, \mathbf{y}_{1:t})$, where $\mathbf{y}_{1:t} = \{\mathbf{y}_1, \ \mathbf{y}_2, \ \cdots, \ \mathbf{y}_t\}$ and $\boldsymbol{\theta}_{0:t} = \{\boldsymbol{\theta}_0, \ \boldsymbol{\theta}_2, \ \cdots, \ \boldsymbol{\theta}_t\}$, constitutes the complete solution to the sequential estimation problem. In many applications, such as tracking, it is of interest to estimate one of its marginals, namely the filtering density $p(\boldsymbol{\theta}_t|\mathbf{x}_{1:t}, \mathbf{y}_{1:t})$. By computing the filtering density recursively, we do not need to keep track of the complete history of the weights. Thus, from a storage point of view, the filtering density turns out to be more parsimonious than the full posterior density function. If we know the filtering density of the network weights, we can easily derive various estimates of the network weights, including centroids, modes, medians and confidence intervals.

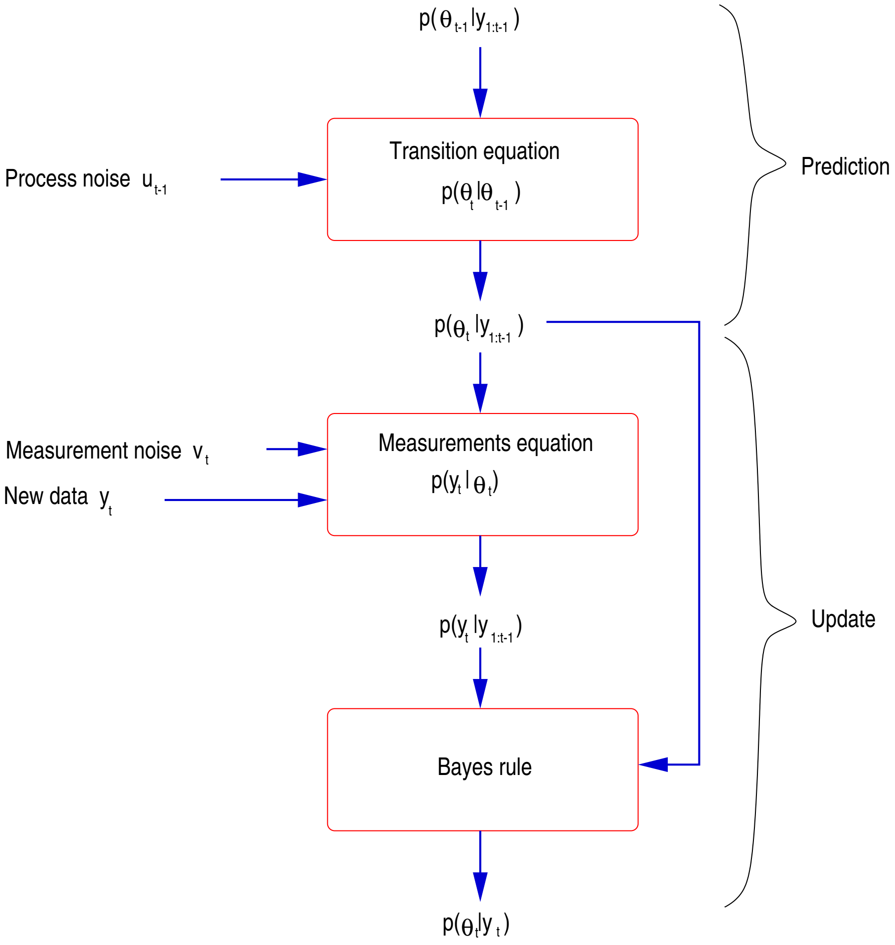

Figure 2.4 *Prediction and update stages in the recursive computation of the filtering density.*

The filtering density is estimated recursively in two stages: prediction and update, as illustrated in Figure 2.4. In the prediction step, the filtering probability density $p(\boldsymbol{\theta}_{t-1}|\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1})$ is propagated into the future via the transition density $p(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1})$

as follows:

$$p(\boldsymbol{\theta}_t|\mathbf{x}_{1:t-1},\mathbf{y}_{1:t-1}) = \int p(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1})p(\boldsymbol{\theta}_{t-1}|\mathbf{x}_{1:t-1},\mathbf{y}_{1:t-1})\mathrm{d}\boldsymbol{\theta}_{t-1} \tag{2.6}$$

The transition density is defined in terms of the probabilistic model governing the states' evolution and the process noise statistics. That is:

$$\begin{aligned} p(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1}) &= \int p(\boldsymbol{\theta}_t|\mathbf{u}_{t-1},\boldsymbol{\theta}_{t-1})p(\mathbf{u}_{t-1}|\boldsymbol{\theta}_{t-1})\mathrm{d}\mathbf{u}_{t-1} \\ &= \int \delta(\boldsymbol{\theta}_t - \mathbf{u}_{t-1} - \boldsymbol{\theta}_{t-1})p(\mathbf{u}_{t-1})\mathrm{d}\mathbf{u}_{t-1} \end{aligned}$$

where the Dirac delta function $\delta(\cdot)$ indicates that $\boldsymbol{\theta}_t$ can be computed via a purely deterministic relation when $\boldsymbol{\theta}_{t-1}$ and $\mathbf{u}_{t-1}$ are known. Note that $p(\mathbf{u}_{t-1}|\boldsymbol{\theta}_{t-1}) = p(\mathbf{u}_{t-1})$ because the process and measurement noise terms are assumed to be independent of past and present values of the states.

The update stage involves the application of Bayes' rule when new data is observed:

$$p(\boldsymbol{\theta}_t|\mathbf{x}_{1:t},\mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_t|\boldsymbol{\theta}_t,\mathbf{x}_t)p(\boldsymbol{\theta}_t|\mathbf{x}_{1:t-1},\mathbf{y}_{1:t-1})}{p(\mathbf{y}_t|\mathbf{x}_t,\mathbf{y}_{1:t-1})} \tag{2.7}$$

The likelihood density function is defined in terms of the measurements model as follows:

$$p(\mathbf{y}_t|\boldsymbol{\theta}_t,\mathbf{x}_t) = \int \delta(\mathbf{y}_t - \widehat{\mathbf{f}}(\mathbf{x}_t,\boldsymbol{\theta}_t) - \mathbf{v}_t)p(\mathbf{v}_t)\mathrm{d}\mathbf{v}_t$$

The normalising denominator of equation (2.7), that is the evidence density function, plays a key role in learning schemes that exploit Gaussian approximation (Jazwinski, 1969; Mackay, 1992a; Sibisi, 1989). It is given by:

$$p(\mathbf{y}_t|\mathbf{x}_t,\mathbf{y}_{1:t-1}) = \int p(\mathbf{y}_t|\boldsymbol{\theta}_t,\mathbf{x}_t)p(\boldsymbol{\theta}_t|\mathbf{x}_{1:t-1},\mathbf{y}_{1:t-1})\mathrm{d}\boldsymbol{\theta}_t$$

In the sequential learning scenario, the parameter estimation problem may be reformulated as having to compute an estimate $\widehat{\boldsymbol{\theta}}_t$ of the states $\boldsymbol{\theta}_t$ using the set of measurements $\{\mathbf{x}_{1:t},\mathbf{y}_{1:t}\}$. For reasons of optimality, we often want $\widehat{\boldsymbol{\theta}}_t$ to be an unbiased, minimum variance and consistent estimate (Gelb, 1974), where:

- An unbiased estimate is one whose expected value is equal to the quantity being estimated.
- A minimum variance (unbiased) estimate is one that has its variance less than or equal to that of any other unbiased estimator.
- A consistent estimate is one that converges to the true value of the quantity being estimated as the number of measurements increases.

The problem of estimating $\boldsymbol{\theta}_t$ given $\{\mathbf{x}_{1:\tau}, \mathbf{y}_{1:\tau}\}$ is called the smoothing problem if $t < \tau$; the filtering problem if $t = \tau$; or the prediction problem if $t > \tau$ (Gelb, 1974; Jazwinski, 1970). In the filtering problem, the estimate $\widehat{\boldsymbol{\theta}}_t$ can be used to predict future values of the output. Typically, one is concerned with predicting $\mathbf{y}_{t+1}$.

Note that the Bayesian sequential learning task is once again an integration problem. To solve this problem, we will usually have to perform either numerical integration, Gaussian approximation or Monte Carlo simulation.

The direct numerical integration method relies on approximating the distribution of interest by a discrete distribution on a finite grid of points. The location of the grid is a non-trivial design issue. Once the distribution is computed at the grid points, an interpolation procedure is used to approximate it in the remainder of the space. Kitagawa (Kitagawa, 1987) used this method to replace the filtering integrals by finite sums over a large set of equally spaced grid points. He chose a piece-wise linear interpolation strategy. Kramer and Sorenson (Kramer and Sorenson, 1988) adhered to the same methodology, but opted for a constant interpolating function. Pole and West (Pole and West, 1990) have attempted to mitigate the problem of choosing the grid's location by implementing a dynamic grid allocation method.

When the grid points are spaced "closely enough" and encompass the region of high probability, the method works well. However, the method is very difficult to implement in high-dimensional, multivariate problems such as neural network modelling. Here, computing at every point in a dense multi-dimensional grid becomes prohibitively expensive (Gelman et al., 1995; Gilks et al., 1996).

Until recently, the most popular approach to sequential estimation has been Gaussian approximation (Bar-Shalom and Li, 1993). In the linear Gaussian scenario, the Kalman filter provides an optimal recursive algorithm for propagating and updating the mean and covariance of the hidden states (Gelb, 1974; Jazwinski, 1970). In nonlinear scenarios, the extended Kalman filter (EKF) is a computationally efficient natural extension of the Kalman filter. It is based on a Taylor expansion of the dynamics and measurements nonlinear equations about the last predicted state. Typically, first order expansions are employed. The mean and covariance are propagated and updated by a simple set of equations, similar to the Kalman filter equations. As the number of terms in the Taylor expansion increases, the complexity of the algorithm also increases due to the computation of derivatives of increasing order. For example, a linear expansion requires the computation of the Jacobian, while a quadratic expansion involves computing the Hessian matrix.

A natural progression on sequential Gaussian approximation is to employ a mixture of Gaussian densities (Kadirkamanathan and Kadirkamanathan, 1995; Li and Bar-Shalom, 1994; Sorenson and Alspach, 1971). These mixtures can either be static or dynamic. In static mixtures, the Gaussian models assumed to be valid throughout the entire process are a subset of several hypothesised models. That is, we start with a few models and compute which of them describe the sequential process most accurately. The remaining models are then discarded. In dynamic model selection, one particular model out of a set of r operating models is selected during each estimation step. Dynamic mixtures of Gaussian models are far more general than static mixtures of models. However, in stationary environments, static mixtures are obviously more suitable. Dynamic mixtures are particularly suited to the problem of noise estimation in rapidly changing environments, such as tracking manoeuvring targets. There each model corresponds to a different hypothesis of the value of the noise covariances (Bar-Shalom and Li, 1993).

Gaussian approximation, because of its simplicity and computational efficiency, constitutes a good way of handling many problems where the density of interest has a significant and predominant mode. Many problems, however, do not fall into this category. Mixtures of Gaussians provide a better solution when there are a few dominant modes. However, they introduce extra computational requirements and complications, such as estimating the number of mixture components.

The basic idea in Monte Carlo simulation is that a set of weighted particles (samples), drawn from the posterior distribution of the model parameters, is used to map the integrations, involved in the inference process, to discrete sums. When, for simplicity, the model dimension is known and fixed, we may make use of the following Monte Carlo approximation:

$$\widehat{p}(\boldsymbol{\theta}_{0:t}|\mathbf{x}_{1:t},\mathbf{y}_{1:t}) = \frac{1}{S}\sum_{i=1}^{S}\delta_{\boldsymbol{\theta}_{0:t}^{(i)}}(\mathrm{d}\boldsymbol{\theta}_{0:t})$$

where $\boldsymbol{\theta}_{0:t}^{(i)}$ represents the particles used to describe the posterior distribution and $\delta(\mathrm{d}\cdot)$ denotes the Dirac delta function. Consequently, any expectations of the form:

$$\mathbb{E}[f_t(\boldsymbol{\theta}_{0:t})] = \int f_t(\boldsymbol{\theta}_{0:t})p(\boldsymbol{\theta}_{0:t}|\mathbf{x}_{1:t},\mathbf{y}_{1:t})\mathrm{d}\boldsymbol{\theta}_{0:t}$$

may be approximated by the following estimate:

$$\mathbb{E}[f_t(\boldsymbol{\theta}_{0:t})] \approx \frac{1}{S}\sum_{i=1}^{S} f_t(\boldsymbol{\theta}_{0:t}^{(i)})$$

where the particles $\boldsymbol{\theta}_{0:t}^{(i)}$, $i = 1, \dots, S$, are drawn from the posterior density function and assumed to be "sufficiently" independent for the approximation to hold. Monte Carlo sampling techniques are an improvement over direct numerical approximation in that they automatically select particles in regions of high probability. An extensive comparison between numerical integration methods (point mass filters) and sequential Monte Carlo methods is presented in (Bergman, 1999).

The next chapter will show how Gaussian approximations can be used to design efficient sequential training algorithms within a regularisation framework. Later, Chapters 6 and 7 will present more advanced sequential Monte Carlo simulation methods.

3

Sequential Bayesian Learning with Gaussian Approximations

As mentioned earlier, sequential training of neural networks is important in applications where data sequences either exhibit non-stationary behaviour or are difficult and expensive to obtain before the training process. Scenarios where this type of sequence arise include tracking and surveillance, control systems, fault detection, signal processing, communications, econometric systems, demographic systems, geophysical problems, operations research and automatic navigation.

This chapter uses Gaussian approximation to solve the sequential Bayesian learning problem. This choice has proved to be very popular. It has been motivated, primarily, by the optimal linear-Gaussian state space filter developed by Kalman and Bucy in 1961 (Kalman and Bucy, 1961). This filter has become an essential component of any modern tracking and time series analysis tool-box or text. Some extensions to the original work on Kalman filtering include coloured noise filters and nonlinear estimators linearised about the current estimates (Anderson and Moore, 1979). The latter are known as extended Kalman filters (EKFs). A well known limitation of Kalman estimators is the assumption of known *a priori* statistics to describe the measurement and process noise. In many applications, it is not straightforward to choose the right noise covariance matrices (Jazwinski, 1970). Unfortunately, the optimality of the Kalman filter often hinges on the designer's ability to formulate these statistics *a priori*. To circumvent this limitation and ensure optimality, it is important to design algorithms to estimate the noise covariances, without leading to a degradation in the performance of the Kalman estimator. This chapter will present a possible solution to this problem.

The problem can also be approached from a neural networks and regularisation perspective. Although there has been great interest in the topic of regularisation in batch learning tasks, this subject has not received much attention in sequential learning tasks. By adopting an hierarchical Bayesian framework in conjunction with dynamical state space models, it is possible to derive regularisation algorithms for sequential estimation. These algorithms are based on estimating the noise processes on-line, while estimating the network weights with the extended Kalman filter. In addition, this methodology will be shown to provide a unifying theoretical framework for many sequential estimation algorithms that attempt to avoid local minima in the error function. In particular, it is shown that adaptive noise covariances in extended Kalman filtering, multiple adaptive learning rates in on-line back-propagation and multiple smoothing regularisation coefficients are mathematically equivalent.

Section 3.1 describes the sequential learning task using state space models and a three-level hierarchical Bayesian structure. The three levels of inference correspond to noise estimation, parameter estimation and model selection. Section 3.2 proposes a solution to the parameter estimation level based on the application of the extended Kalman filter to neural networks. Section 3.3 is devoted to the noise estimation level and regularisation. The algorithms are tested with synthetic data in Section 3.4.

3.1 Dynamical Hierarchical Bayesian Models

As proposed in the previous chapter, the following model is adopted:

$$\begin{aligned}\boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t + \mathbf{u}_t \\ \mathbf{y}_t &= \widehat{\mathbf{f}}(\mathbf{x}_t, \boldsymbol{\theta}_t) + \mathbf{v}_t\end{aligned}$$

The measurements nonlinear mapping $\widehat{\mathbf{f}}(\mathbf{x}_t, \boldsymbol{\theta}_t)$ corresponds to a multi-layer perceptron (MLP) whose weights are the model states $\boldsymbol{\theta}_t$. The framework may be easily extended to encompass recurrent networks, radial basis networks and many other approximation techniques. The measurements are assumed to be corrupted by noise $\mathbf{v}_t$, which is assumed to be zero mean, uncorrelated Gaussian with adaptive covariance R_t. The model parameters evolve according to a deterministic component $\boldsymbol{\theta}_t$ and a stochastic component $\mathbf{u}_t$. The process noise $\mathbf{u}_t$ may represent our uncertainty on how the parameters evolve, modelling errors or unknown inputs. It is assumed to be zero mean Gaussian with adaptive covariance Q_t.

To address the problem of estimating the best model $\mathcal{M}_k$ ($k \in \{0, 1, 2, 3, \ldots, k_{\max}\}$), the parameters and noise covariances jointly, it is convenient to adopt an hierarchical Bayesian model comprising three different levels of abstraction[1]:

Level 1: Parameter estimation

$$p(\boldsymbol{\theta}_{t+1}|\mathbf{y}_{1:t+1}, \mathcal{M}_k, R_{t+1}, Q_t) = \frac{p(\mathbf{y}_{t+1}|\boldsymbol{\theta}_{t+1}, \mathcal{M}_k, R_{t+1}, Q_t)}{p(\mathbf{y}_{t+1}|\mathbf{y}_{1:t}, \mathcal{M}_k, R_{t+1}, Q_t)} p(\boldsymbol{\theta}_{t+1}|\mathbf{y}_{1:t}, \mathcal{M}_k, R_{t+1}, Q_t) \tag{3.1}$$

[1] To keep the notation simple, the input variables $\mathbf{x}$ are suppressed in the arguments of the probability distributions.

Level 2: Noise estimation

$$p(R_{t+1}, Q_t|\mathbf{y}_{1:t+1}) = \frac{p(\mathbf{y}_{t+1}|\mathbf{y}_{1:t}, \mathcal{M}_k, R_{t+1}, Q_t)}{p(\mathbf{y}_{t+1}|\mathbf{y}_{1:t}, \mathcal{M}_k)} p(R_{t+1}, Q_t|\mathbf{y}_{1:t}, \mathcal{M}_k) \tag{3.2}$$

Level 3: Model selection

$$p(\mathcal{M}_k|\mathbf{y}_{1:t+1}) = \frac{p(\mathbf{y}_{t+1}|\mathbf{y}_{1:t}, \mathcal{M}_k)}{p(\mathbf{y}_{t+1}|\mathbf{y}_{1:t})} p(\mathcal{M}_k|\mathbf{y}_{1:t}) \tag{3.3}$$

The likelihood function at a particular level constitutes the evidence function at the next higher level. Therefore, by maximising the evidence function in the parameter estimation level, we are, in fact, maximising the likelihood of the noise covariances R_t and Q_t as the new data arrives. This result provides the foundation for the noise estimation methods described later.

At the parameter estimation level, the EKF algorithm will be used to estimate the weights of the multi-layer perceptron. The EKF, however, requires knowledge of the noise covariances. To overcome this difficulty, in Section 3.3, several techniques are presented to estimate these covariances in slowly changing non-stationary environments. There, it is shown that algorithms for estimating the noise covariances allow for regularisation in a sequential framework. The treatment of sequential model selection will be delayed until Chapter 7.

3.2 Parameter Estimation

The EKF may be applied at the parameter estimation level to compute the network weights. The EKF is a minimum variance estimator based on a Taylor series expansion of the nonlinear function $\widehat{\mathbf{f}}(\mathbf{x}_t, \boldsymbol{\theta}_t)$ around the previous estimate. That is:

$$\widehat{\mathbf{f}}(\boldsymbol{\theta}_t, \mathbf{x}_t) = \widehat{\mathbf{f}}(\widehat{\boldsymbol{\theta}}_t, \mathbf{x}_t) + \frac{\partial \widehat{\mathbf{f}}(\boldsymbol{\theta}_t, \mathbf{x}_t)}{\partial \boldsymbol{\theta}_t}\Big|_{(\boldsymbol{\theta}_t = \widehat{\boldsymbol{\theta}}_t)} (\boldsymbol{\theta}_t - \widehat{\boldsymbol{\theta}}_t) + \cdots$$

Using this expansion and under the assumptions that the state space model noise processes are uncorrelated with each other and with the initial estimates of the parameters $\boldsymbol{\theta}_t$ and their covariance matrix P_t, we can model the prior, evidence and likelihood functions as follows[2]:

$$\begin{array}{lll}
\text{Prior} & = p(\boldsymbol{\theta}_{t+1}|\mathbf{y}_{1:t}, \mathcal{M}_k, R_{t+1}, Q_t) & \approx \mathcal{N}(\widehat{\boldsymbol{\theta}}_t, P_t + Q_t) \\
\text{Evidence} & = p(\mathbf{y}_{t+1}|\mathbf{y}_{1:t}, \mathcal{M}_k, R_{t+1}, Q_t) & \approx \mathcal{N}(\widehat{\mathbf{f}}(\widehat{\boldsymbol{\theta}}_t, \mathbf{x}_{t+1}), G_{t+1}(P_t + Q_t)G'_{t+1} + R_{t+1}) \\
\text{Likelihood} & = p(\mathbf{y}_{t+1}|\boldsymbol{\theta}_{t+1}, \mathcal{M}_k, R_{t+1}, Q_t) & \approx \mathcal{N}(\widehat{\mathbf{f}}(\boldsymbol{\theta}_{t+1}, \mathbf{x}_{t+1}), R_{t+1})
\end{array}$$

[2] See Appendix A for a Bayesian derivation of the Kalman filter.

where G denotes the Jacobian:

$$G = \frac{\partial \widehat{\mathbf{f}}(\boldsymbol{\theta}, \mathbf{x})}{\partial \boldsymbol{\theta}} \Bigg|_{(\boldsymbol{\theta}=\widehat{\boldsymbol{\theta}})} = \begin{bmatrix} \frac{\partial \widehat{\mathbf{f}}_1(\boldsymbol{\theta},\mathbf{x})}{\partial \boldsymbol{\theta}_1} & \frac{\partial \widehat{\mathbf{f}}_2(\boldsymbol{\theta},\mathbf{x})}{\partial \boldsymbol{\theta}_1} & \cdots & \frac{\partial \widehat{\mathbf{f}}_c(\boldsymbol{\theta},\mathbf{x})}{\partial \boldsymbol{\theta}_1} \\ \frac{\partial \widehat{\mathbf{f}}_1(\boldsymbol{\theta},\mathbf{x})}{\partial \boldsymbol{\theta}_2} & & & \\ \vdots & & & \vdots \\ \frac{\partial \widehat{\mathbf{f}}_1(\boldsymbol{\theta},\mathbf{x})}{\partial \boldsymbol{\theta}_m} & & \cdots & \frac{\partial \widehat{\mathbf{f}}_c(\boldsymbol{\theta},\mathbf{x})}{\partial \boldsymbol{\theta}_m} \end{bmatrix}'$$

Since the EKF is a suboptimal estimator based on linearisation of a nonlinear mapping, $\widehat{\boldsymbol{\theta}}_t$ is only an approximation to the expected value and, strictly speaking, P_t is an approximation to the covariance matrix:

$$\begin{aligned} \widehat{\boldsymbol{\theta}}_t &\approx \mathbb{E}(\boldsymbol{\theta}_t | \mathbf{y}_{1:t}) \\ P_t &\approx \mathbb{E}((\boldsymbol{\theta}_t - \widehat{\boldsymbol{\theta}}_t)(\boldsymbol{\theta}_t - \widehat{\boldsymbol{\theta}}_t)' | \mathbf{y}_{1:t}) \end{aligned}$$

The EKF may diverge as a result of its approximations. The consistency of the EKF may be evaluated by means of extensive Monte Carlo simulations (Bar-Shalom and Li, 1993). Substituting the expressions for the prior, likelihood and evidence into equation (3.1), yields the posterior density function:

$$\text{Posterior} = p(\boldsymbol{\theta}_{t+1} | \mathbf{y}_{1:t+1}, \mathcal{M}_k, R_{t+1}, Q_t) \approx \mathcal{N}(\widehat{\boldsymbol{\theta}}_{t+1}, P_{t+1})$$

where the updated weights, covariance and Kalman gain (K_{t+1}) are given by:

$$K_{t+1} = (P_t + Q_t) G'_{t+1} [R_{t+1} + G_{t+1}(P_t + Q_t) G'_{t+1}]^{-1} \tag{3.4}$$

$$\widehat{\boldsymbol{\theta}}_{t+1} = \widehat{\boldsymbol{\theta}}_t + K_{t+1}(\mathbf{y}_{t+1} - \widehat{\mathbf{f}}(\widehat{\boldsymbol{\theta}}_t, \mathbf{x}_{t+1})) \tag{3.5}$$

$$P_{t+1} = P_t + Q_t - K_{t+1} G_{t+1} (P_t + Q_t) \tag{3.6}$$

Figure 3.1 shows a graphical representation of the extended Kalman filter. By grouping the MLP weights into a single vector $\boldsymbol{\theta}$, we can use the EKF equations (equations (3.4) to (3.6)) to recursively compute new estimates of the weights. The entries of the Jacobian matrix are calculated by back-propagating the c output values $\{\mathbf{y}_{t,1}, \mathbf{y}_{t,2}, \cdots, \mathbf{y}_{t,c}\}$ through the network. An example of how to do this for a simple MLP is presented in Appendix B.

One of the earliest implementations of EKF trained MLPs is due to Singhal and Wu (Singhal and Wu, 1988). The algorithm's computational complexity is of the order $\mathcal{O}(cm^2)$ multiplications per time step. Shah, Palmieri and Datum (Shah et al., 1992) and Puskorius and Feldkamp (Puskorius and Feldkamp, 1991) have proposed various approximations to the weights covariance so as to simplify this problem. The EKF is an improvement over conventional MLP estimation techniques, such as on-line back-propagation, in that it makes use of second order statistics (Ruck et al., 1992; Schottky and Saad, 1999). These statistics are essential for placing error bars on the

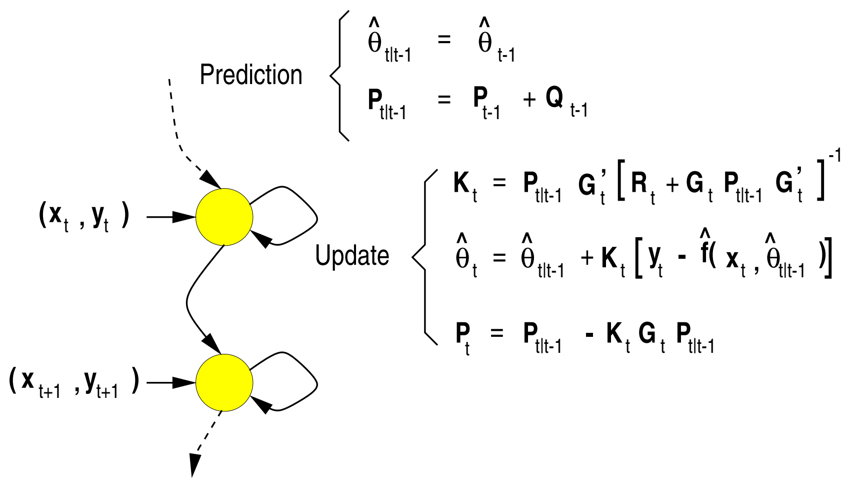
Prediction
$\hat{\theta}_{t|t-1} = \hat{\theta}_{t-1}$
$P_{t|t-1} = P_{t-1} + Q_{t-1}$
(x_t , y_t)
Update
$K_t = P_{t|t-1} G'_t [R_t + G_t P_{t|t-1} G'_t]^{-1}$
$\hat{\theta}_t = \hat{\theta}_{t|t-1} + K_t [y_t - \hat{f}(x_t , \hat{\theta}_{t|t-1})]$
$P_t = P_{t|t-1} - K_t G_t P_{t|t-1}$
(x_{t+1} , y_{t+1})

Figure 3.1 *Extended Kalman filter predictor-corrector representation.*

predictions and for combining separate networks into committees of networks when $p(\boldsymbol{\theta}_t|\mathbf{y}_{1:t})$ has multiple modes (Bar-Shalom and Li, 1993; Blom and Bar-Shalom, 1988; Kadirkamanathan and Kadirkamanathan, 1995).

3.3 Noise Estimation and Regularisation

A well known limitation of the EKF is the assumption of known *a priori* statistics to describe the measurement and process noise. Setting these noise levels appropriately often makes the difference between success and failure in the use of the EKF (Candy, 1986). In many applications, it is not straightforward to choose the noise covariances (Jazwinski, 1970; West and Harrison, 1996). In addition, in environments where the noise statistics change with time, such an approach can lead to large estimation errors and even to a divergence of errors. Several researchers in the estimation, filtering and control fields have attempted to solve this problem (Jazwinski, 1969; Mehra, 1970; Mehra, 1971; Myers and Tapley, 1976; Tenney et al., 1977). Mehra (Mehra, 1972) and Li and Bar-Shalom (Li and Bar-Shalom, 1994) give brief surveys of this topic.

It is important to note that algorithms for estimating the noise covariances within the EKF framework can lead to a degradation of the performance of the EKF. By increasing the process noise covariance Q_t, the Kalman gain also increases, thereby producing bigger changes in the weight updates (refer to equations (3.4) and (3.5)). That is, more importance is placed on the most recent measurements. Consequently, it may be asserted that filters with adaptive process noise covariances exhibit adaptive memory. Additionally, as the Kalman gain increases, the bandwidth of the filter also increases (Bar-Shalom and Li, 1993). Therefore, the filter becomes less immune to noise and outliers.

The amount of oscillation in the model prediction clearly depends on the value of the process noise covariance. As a result, this covariance can be used as a regularisation mechanism to control the smoothness of the prediction.

The following subsections derive three algorithms to estimate the noise covariances. The first two derivations serve to illustrate the fact that algorithms for adapting distributed learning rates, smoothing regularisers or stochastic noise in gradient descent methods are equivalent (Figure 3.2). The third algorithm addresses the trade-off between regularisation and tracking performance in sequential learning.

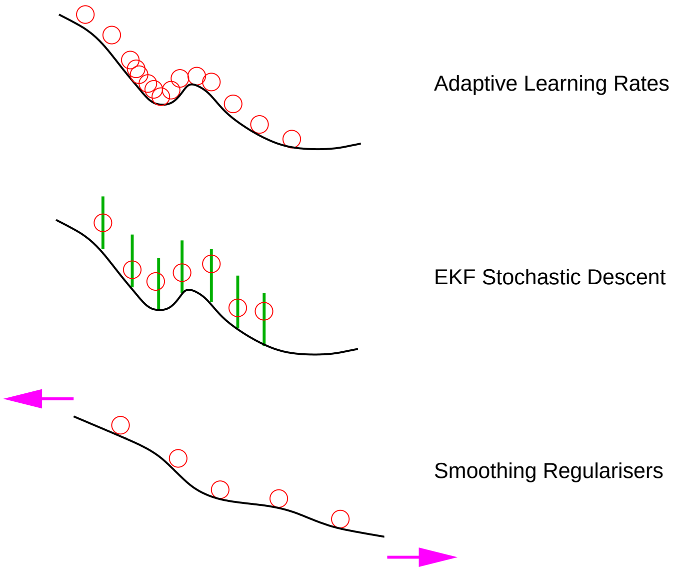

Figure 3.2 *One-dimensional example of several adaptive gradient descent methods. To escape from local minima, we can either increase the momentum of the ball as it approaches a particular local minimum, allow the ball to bounce randomly within a given vertical interval (stochastic descent) or stretch the surface with smoothing regularisers.*

3.3.1 Algorithm 1: adaptive distributed learning rates

Sutton (Sutton, 1992b) proposed an optimisation approach for linear networks, using the Kalman filter equations with P_t updated by a variation of the least-mean-square rule (Jacobs, 1988; Sutton, 1992a). The main purpose of the method was to reduce the computational time at the expense of a small deterioration in the performance of the estimator. Another important aspect of the algorithm is that it circumvents the problem of choosing the process noise covariance Q. The technique involves approximating P with a diagonal matrix, whose i-th diagonal entry is given by:

$$p_{mm} = \exp(\beta_m)$$

where β_m is updated by the least-mean-square rule modified such that the learning rates for each parameter are updated sequentially. The diagonal matrix approximation to P implies that the model parameters are uncorrelated. This assumption may, of course, degrade the performance of the EKF estimator.

To circumvent the problem of choosing the process noise covariance Q when training nonlinear neural networks, while at the same time increasing computational efficiency, Sutton's algorithm can be extended to MLPs. The network weights and Kalman gain are updated using the EKF, while the weights covariance P is updated by back-propagating the squared output errors (see Appendix B), with β as follows:

$$\beta_{t+1} = \begin{cases} \beta_t + \eta \delta_{t,i} y_{t,j} & \text{output layer} \\ \beta_t + \eta \theta_{t,i,j} \delta_{t,i} y_{t,j} (1 - y_{t,j}) x_{t,d} & \text{hidden layer} \end{cases}$$

where the index i corresponds to the i-th neuron in the output layer, j to the j-th neuron in the hidden layer, d to the d-th input variable and t to the estimation step. $\delta_{t,i}$ represents the output error for neuron i at time t. The symbols $y_{t,j}$ and η denote the output of the j-th neuron in the hidden layer and the learning rate respectively. This learning rate is a parameter that quantifies a matrix of parameters P_t. It will be referred to as a hyper-parameter.

The EKF equation used to update the weights is similar to the update equations typically used to compute the weights of neural networks by error back-propagation. The only difference is that it assumes that there is a different adaptive learning rate parameter for each weight. By comparing the EKF and gradient descent equations, we find that the mathematical relation between adaptive learning rates ($\mathcal{L}_t$) in on-line back-propagation and Kalman filtering parameters is given by:

$$\mathcal{L}_{t+1} = (P_t + Q_t)(R_{t+1} + G_{t+1}(P_t + Q_t)G'_{t+1})^{-1}$$

Thus, adapting the process noise is equivalent to adapting the learning rates.

3.3.2 Algorithm 2: evidence maximisation with weight decay priors

This section derives a sequential method for updating R and Q, based on the evidence approximation framework with weight decay priors for batch learning (see Chapter 10 of (Bishop, 1995b) and the references therein). In so doing, it is shown that algorithms for adapting the noise covariances are equivalent to algorithms that make use of smoothing regularisers.

In the evidence approximation framework for batch learning, the prior and likelihood are expressed as follows:

$$p(\boldsymbol{\theta}) = \frac{1}{(2\pi)^{m/2}\alpha^{-m/2}} \exp\Big(-\frac{\alpha}{2}\|\boldsymbol{\theta}\|^2\Big) \tag{3.7}$$

$$p(\mathbf{y}_{1:N}|\boldsymbol{\theta}) = \frac{1}{(2\pi)^{N/2}\beta^{-N/2}} \exp\Big(-\frac{\beta}{2}\sum_{t=1}^{N}(\mathbf{y}_t - \widehat{\mathbf{f}}(\boldsymbol{\theta},\mathbf{x}_t))^2\Big) \tag{3.8}$$

where the hyper-parameters α and β control the variance of the prior distribution of the weights and the variance of the measurement noise. α also plays the role of the regularisation coefficient. Using Bayes' rule (equation (3.1)) and taking into account that the evidence does not depend on the weights, the following posterior density function may be obtained:

$$p(\boldsymbol{\theta}|\mathbf{y}_{1:N}) = \frac{1}{Z_s(\alpha,\beta)}\exp(-S(\boldsymbol{\theta}))$$

where $Z_s(\alpha,\beta)$ is a normalising factor. For the prior and the likelihood of equations (3.7) and (3.8), $S(\boldsymbol{\theta})$ is given by:

$$S(\boldsymbol{\theta}) = \frac{\alpha}{2}\|\boldsymbol{\theta}\|^2 + \frac{\beta}{2}\sum_{t=1}^{N}(\mathbf{y}_t - \widehat{\mathbf{f}}(\boldsymbol{\theta},\mathbf{x}_t))^2 \tag{3.9}$$

The posterior density may be approximated by applying a Taylor series expansion of $S(\boldsymbol{\theta})$ around a local minimum ($\boldsymbol{\theta}_{MP}$) and retaining the series terms up to second order:

$$S(\boldsymbol{\theta}) = S(\boldsymbol{\theta}_{MP}) + \frac{1}{2}(\boldsymbol{\theta}-\boldsymbol{\theta}_{MP})'A(\boldsymbol{\theta}-\boldsymbol{\theta}_{MP})$$

Hence, the Gaussian approximation to the posterior density function becomes:

$$p(\boldsymbol{\theta}|\mathbf{y}_{1:N}) = \frac{1}{(2\pi)^{m/2}|A|^{-1/2}}\exp\Big(-\frac{1}{2}(\boldsymbol{\theta}-\boldsymbol{\theta}_{MP})'A(\boldsymbol{\theta}-\boldsymbol{\theta}_{MP})\Big) \tag{3.10}$$

Maximising the posterior probability density function involves minimising the error function given by equation (3.9). Equation (3.9) is a particular case of a regularised error function, as discussed in the first section of Chapter 2.

In the evidence framework, the parameters $\boldsymbol{\theta}$ are obtained by minimising equation (3.9), while the hyper-parameters α and β are obtained by maximising the evidence $p(\mathbf{y}_{1:N}|\alpha,\beta)$ after approximating the posterior density function by a Gaussian function centred at $\boldsymbol{\theta}_{MP}$. In doing so, the following recursive formulae for α and β are obtained:

$$\alpha_{t+1} = \frac{\gamma}{\sum_{i=1}^{m}\theta_i^2} \qquad \text{and} \qquad \beta_{t+1} = \frac{N-\gamma}{\sum_{t=1}^{N}(\mathbf{y}_t - \widehat{\mathbf{f}}(\boldsymbol{\theta}_t,\mathbf{x}_t))^2} \tag{3.11}$$

The quantity $\gamma = \sum_{i=1}^{m}\frac{\lambda_i}{\lambda_i+\alpha}$, represents the effective number of parameters, where the λ_i are the eigenvalues of the Hessian of the un-regularised error function. The effective number of parameters, as the name implies, is the number of parameters that effectively contributes to the neural network mapping. The remaining weights have no contribution because their magnitudes are forced to zero by the weight decay prior.

It is possible to maximise the posterior density function by performing integrations over the hyper-parameters analytically (Buntine and Weigend, 1991; Mackay, 1996; Williams, 1995; Wolpert, 1993). The latter approach is known as the MAP framework for α and β. The hyper-parameters computed by the MAP framework differ from the ones computed by the evidence framework in that the former makes use of the total number of parameters and not only the effective number of parameters. That is, α and β are updated according to:

$$\alpha_{t+1} = \frac{m}{\sum_{i=1}^{m} \theta_i^2} \qquad \text{and} \qquad \beta_{t+1} = \frac{N}{\sum_{t=1}^{N} (\mathbf{y}_t - \widehat{\mathbf{f}}(\boldsymbol{\theta}_t, \mathbf{x}_t))^2} \tag{3.12}$$

By comparing the expressions for the prior, likelihood and evidence in the EKF framework (Section 3.2) with equations (3.7), (3.8) and (3.10), the following relations can be established:

$$P = A^{-1}, \qquad Q = \alpha^{-1} I_m - A^{-1} \qquad \text{and} \qquad R = \beta^{-1} I_c \tag{3.13}$$

where I_m and I_c represent identity matrices of sizes m and c respectively. Therefore, it is possible to update Q and R sequentially by expressing them in terms of the sequential updates of α and β. That is, adapting the noise processes is equivalent to adapting the regularisation coefficients. A moving window may be implemented to estimate β. The size of the window is a parameter that requires tuning.

3.3.3 Algorithm 3: evidence maximisation with sequentially updated priors

In extended Kalman filtering, we have knowledge of the equation describing the evidence function in terms of the noise covariances. Consequently, we can compute R_t and Q_t automatically by maximising the evidence density:

$$p(y_{t+1} | \mathbf{y}_{1:t}, \mathcal{M}_k, R_{t+1}, Q_t) \approx \mathcal{N}(\widehat{\mathbf{f}}(\widehat{\boldsymbol{\theta}}_t, \mathbf{x}_{t+1}), G_{t+1}(P_t + Q_t)G'_{t+1} + R_{t+1})$$

Strictly speaking, this is not a full Bayesian solution. We are computing, solely, the likelihood of the noise covariances. That is, we are assuming no knowledge of the prior at the noise estimation level. For simplicity, we have restricted our analysis in this section to a single output. Let us now define the model residuals:

$$r_{t+1} = y_{t+1} - \mathbb{E}\big(y_{t+1} | \mathbf{y}_{1:t}, \mathcal{M}_k, R_t, Q_t\big) = y_{t+1} - \widehat{\mathbf{f}}(\widehat{\boldsymbol{\theta}}_t, \mathbf{x}_{t+1})$$

It follows that the probability of the residuals is equivalent to the evidence function at the parameter estimation level. That is:

$$p(r_{t+1}) = p(y_{t+1}|\mathbf{y}_{1:t}, \mathcal{M}_k, R_{t+1}, Q_t)$$

Let us assume, initially, that the process noise covariance may be described by a single parameter q. More specifically:

$$Q = qI_m$$

The maxima of the evidence function with respect to q may be calculated by differentiating the evidence function as follows:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}q}p(r_{t+1}) &= \frac{1}{(2\pi)^{1/2}} \exp\Big(-\frac{1}{2}\frac{r^2_{t+1}}{G_{t+1}(P_t+Q_t)G'_{t+1}+R_{t+1}}\Big) \\ & \Big[-\frac{1}{2}G_{t+1}G'_{t+1}(G_{t+1}(P_t+Q_t)G'_{t+1}+R_{t+1})^{-3/2} + \\ & \frac{1}{2}r^2_{t+1}G_{t+1}G'_{t+1}(G_{t+1}(P_t+Q_t)G'_{t+1}+R_{t+1})^{-5/2}\Big] \end{aligned}$$

Equating the derivative to zero yields:

$$r^2_{t+1} = \mathbb{E}\big(r^2_{t+1}\big) \tag{3.14}$$

It is straightforward to prove that this singularity corresponds to a global maximum on $[0, \infty)$ by computing the second derivative. This result reveals that maximising the evidence function corresponds to equating the covariance over time r^2_{t+1} to the ensemble covariance $\mathbb{E}\big(r^2_{t+1}\big)$. That is, maximising the evidence leads to a covariance matching method.

Jazwinski (Jazwinski, 1969; Jazwinski and Bailie, 1967) devised an algorithm for updating q according to equation (3.14). Since:

$$r^2_{t+1} = G_{t+1}P_tG'_{t+1} + qG_{t+1}G'_{t+1} + R_{t+1},$$

it follows that q may be recursively computed according to:

$$q = \begin{cases} \frac{r^2_{t+1}-\mathbb{E}\left(r^2_{t+1}|q=0\right)}{G_{t+1}G'_{t+1}} & \text{if } q \geq 0 \\ \\ 0 & \text{otherwise} \end{cases} \tag{3.15}$$

This estimator increases q each time the variance over time of the model residuals exceeds the ensemble variance. When q increases, the Kalman gain also increases and consequently the model parameters update also increases (equations (3.4) and (3.5)). That is, the estimator places more emphasis on the incoming data. As long as the variance over time of the residuals remains smaller than the ensemble covariance, the process noise input is zero and the filter carries on minimising the variance of the parameters (i.e. tending to a regularised solution). Section 3.4 discusses an experiment where this behaviour is illustrated.

The estimator of equation (3.15) is based on a single residual and is therefore of little statistical significance. This difficulty is overcome by employing a sliding window to compute the sample mean for N predicted residuals, instead of a single residual. Jazwinski (Jazwinski, 1969) shows that for the following sample mean:

$$m_r = \frac{1}{N}\sum_{l=1}^{N}\frac{r_{t+l}}{R_{t+l}^{1/2}},$$

we may proceed as above, by maximising $p(m_r)$, to obtain the following estimator:

$$q = \begin{cases} \frac{m_r^2 - \mathbb{E}[m_r^2|q=0]}{S} & \text{if } q \geq 0 \\ \\ 0 & \text{otherwise} \end{cases} \tag{3.16}$$

where

$$\mathbb{E}[m_r^2|q=0] = S_N P_t S_N' + 1/N,$$

$$S = S_N S_N' + S_{N-1} S_{N-1}' + \cdots + S_1 S_1'$$

and

$$S_N = \frac{1}{N}\sum_{l=1}^{N}\frac{1}{R_{t+l}^{1/2}}G_{t+l}, \quad S_{N-1} = \frac{1}{N}\sum_{l=2}^{N}\frac{1}{R_{t+l}^{1/2}}G_{t+l}, \quad \ldots, \quad S_1 = \frac{1}{N}\frac{1}{R_{t+N}^{1/2}}G_{t+N} \tag{3.17}$$

With this estimator, one has to choose the length N of the moving window used to update q. If the window size is too small, the algorithm places more emphasis on fitting the incoming data than fitting the previous data. As a result, it might never converge. On the other hand, if the window size is too large, then the algorithm will fail to adapt quickly to new environments. The problem of choosing the right window length results in a regularisation/tracking dilemma. It is a dilemma because we cannot ascertain, without *a priori* knowledge, whether the fluctuations in the data correspond to time varying components of the data or to noise.

It is possible to extend the derivation to a more general noise model by adopting the following covariance:

$$Q = \begin{bmatrix} q_1 & 0 & \cdots & 0 \\ 0 & q_2 & & 0 \\ \vdots & & \ddots & \\ 0 & 0 & & q_m \end{bmatrix} \tag{3.18}$$

By calculating the derivative of the evidence function with respect to a generic diagonal entry of Q and then equating to zero, we obtain an estimator involving the following system of equations:

$$\begin{bmatrix} (\frac{\partial y_{t+1}}{\partial \boldsymbol{\theta}_1})^2 & (\frac{\partial y_{t+1}}{\partial \boldsymbol{\theta}_2})^2 & \cdots & (\frac{\partial y_{t+1}}{\partial \boldsymbol{\theta}_m})^2 \\ (\frac{\partial y_{t}}{\partial \boldsymbol{\theta}_1})^2 & (\frac{\partial y_{t}}{\partial \boldsymbol{\theta}_2})^2 & & (\frac{\partial y_{t}}{\partial \boldsymbol{\theta}_m})^2 \\ \vdots & & \ddots & \\ (\frac{\partial y_{t-N}}{\partial \boldsymbol{\theta}_1})^2 & (\frac{\partial y_{t-N}}{\partial \boldsymbol{\theta}_2})^2 & & (\frac{\partial y_{t-N}}{\partial \boldsymbol{\theta}_m})^2 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_m \end{bmatrix} = \begin{bmatrix} \varepsilon_{t+1} \\ \varepsilon_t \\ \vdots \\ \varepsilon_{t-N} \end{bmatrix} \tag{3.19}$$

Multiple hyper-parameters are very handy when one considers distributed priors for automatic relevance determination (input and basis functions selection) (de Freitas et al., 1997; Mackay, 1994; Mackay, 1995). Estimating Q in equation (3.19) is, however, not very reliable because it involves estimating a large number of noise parameters and, in addition, it requires a long moving window of size N to avoid ill-conditioning.

We can also maximise the evidence function with respect to $R_t = \overline{r}I_m$ and obtain the following estimator for $\overline{r}$:

$$\overline{r} = \begin{cases} r_t^2 - G_t(P_t + Q_t)G_t' & \text{if } r \geq 0 \\ \\ 0 & \text{otherwise} \end{cases} \tag{3.20}$$

The hyper-parameter $\overline{r}$ is not as useful as q in controlling filter divergence. This is because $\overline{r}$ can slow down the rate of decrease of the covariance matrix P_t, but cannot cause it to increase, according to the Kalman filter equations.

3.4 Demonstrations of the Algorithms

3.4.1 Experiment 1: comparison between the noise estimation methods

To compare the performance of the various EKF training algorithms discussed in this chapter, 100 input-output data vectors were generated from the following nonlinear, non-stationary process:

$$\begin{aligned} x_t &= 0.5x_{t-1} + \frac{25x_{t-1}}{1+x_{t-1}^2} + 8\cos(1.2(t-1)) + d_t \\ y_t &= \frac{x_t^2}{20} + v_t \end{aligned}$$

where x_t denotes the input vectors and y_t the output vectors of 300 time samples each. The Gaussian process noise standard deviation was set to 0.1, while the measurement noise standard deviation was set to $3\sin(0.05t)$. The initial state x_0 was 0.1. Figure 3.3 shows the data generated with this model.

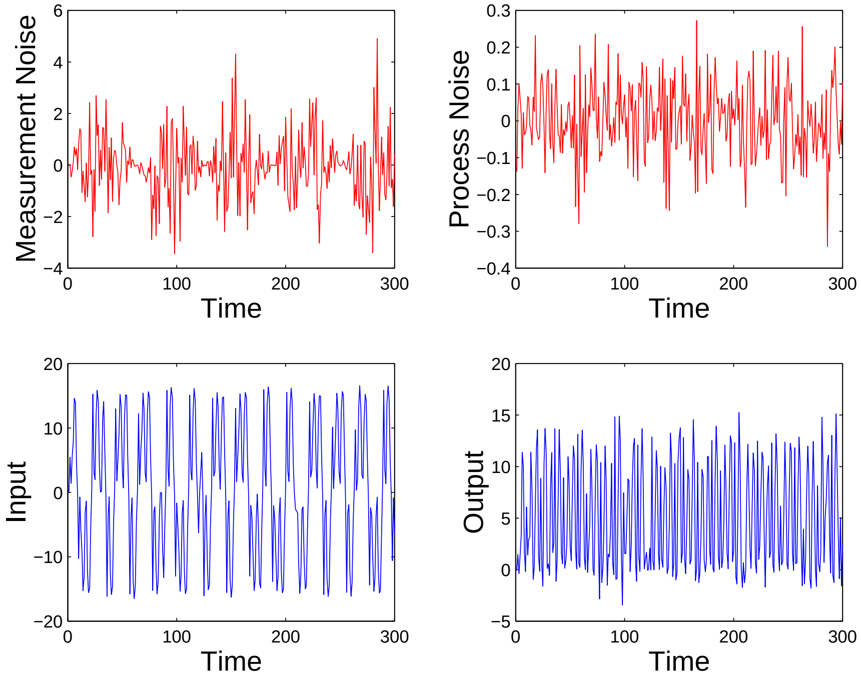

Figure 3.3 *Data generated to train MLP.*

An MLP, with 10 sigmoidal neurons in the hidden layer and 1 linear output neuron, was then trained with the following methods: the standard EKF algorithm, the EKF algorithm with P_t updated by error back-propagation (EKFBP), with evidence maximisation and weight decay priors (EKFEV), with MAP noise adaptation (EKFMAP) and with evidence maximisation and sequentially updated priors (EKFQ). The initial variance of the weights, initial weights covariance matrix entries, initial R and initial Q were set to 1, 10, 3 and 1×10^{-5} respectively. The length of the sliding window of the adaptive noise algorithms was set to 10 time steps.

The simulation results for the EKF and EKFQ algorithms are shown in Figure 3.4. Note that the EKFQ algorithm slows down the convergence of the EKF parameter estimates so as to be able to track the changing measurement variance. Table 3.1 compares the one-step-ahead root square errors (RSE) obtained with each method. The RSE are defined as follows:

$$\mathrm{RSE} = \sqrt{\sum_{t=1}^{300} (\mathbf{y}_t - \widehat{\mathbf{f}}(\widehat{\boldsymbol{\theta}}_\mathbf{t}, \mathbf{x_t}))^\mathbf{2}}$$

According to the table, it is clear that the only algorithm that provides a clear prediction improvement over the standard EKF algorithm is the evidence maximisation algorithm with sequential update priors. In terms of computational time, the EKF algorithm with

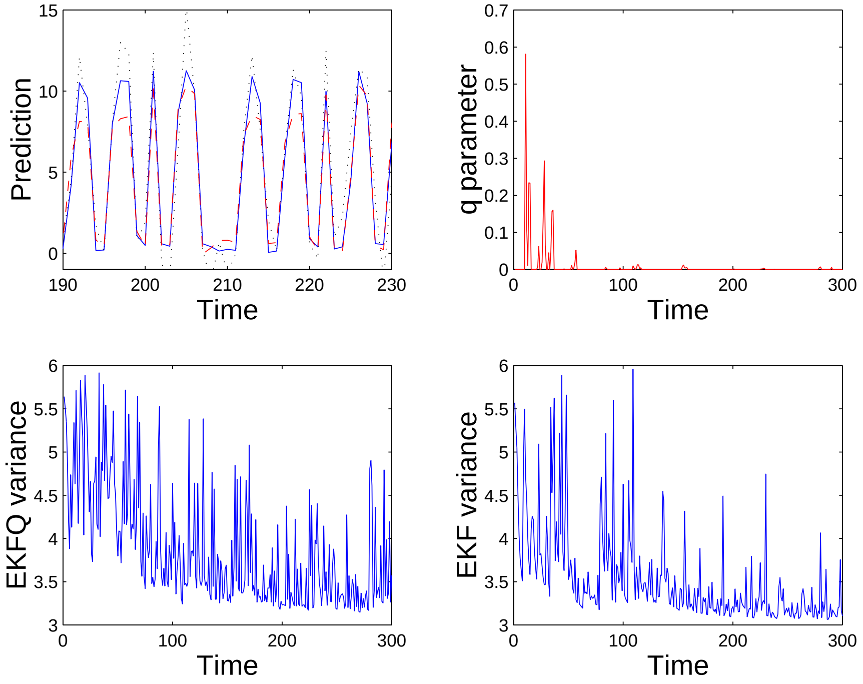

Figure 3.4 *Simulation results for the EKF and EKFQ algorithms. The top left plot shows the actual data [· · ·] and the EKF [- -] and EKFQ [—] one-step-ahead predictions. The top right plot shows the process noise parameter. The bottom plots show the innovations variance for both methods.*

P_t updated by back-propagation is faster, but its prediction is worse than the one for the standard EKF. This is not a surprising result considering the assumption of uncorrelated weights. The EKFEV and EKFMAP performed poorly because they require the network weights to converge to a good solution before the noise covariances can be updated. That is, the noise estimation algorithm does not facilitate the estimation of the weights, as it happens in the case of the EKFQ algorithm. The EKFEV and EKFMAP therefore appear to be unsuitable for sequential learning tasks.

	RSE	**Mega floating point operations**
EKF	25.95	21.9
EKFQ	23.01	24.1
EKFMAP	61.06	22.6
EKFEV	73.94	22.6
EKFBP	58.87	2.2

Table 3.1 *Simulation results for 100 runs in experiment 1.*

3.4.2 Experiment 2: Sequential evidence maximisation with sequentially updated priors

This experiment aims to describe the behaviour of the evidence maximisation algorithm (EKFQ) of equation (3.16) in a time-varying, noisy and chaotic scenario. The problem tackled is a more difficult variation of the chaotic quadratic or logistic map. 100 input (y_t) and output (y_{t+1}) data vectors were generated according to the following equation:

$$y_{t+1} = \begin{cases} 3.5y_t(1-y_t) + v_t & 1 \leq t \leq 150 \\ 3.7y_t(1-y_t) + v_t & 150 < t \leq 225 \\ 3.1y_t(1-y_t) + v_t & 225 < t \leq 300 \end{cases}$$

where v_t denotes Gaussian noise with a standard deviation of 0.01. In the interval $150 < t \leq 225$, the series exhibits chaotic behaviour. A single hidden layer MLP with 10 sigmoidal neurons in the hidden layer and a single output linear neuron was trained to approximate the mapping between (y_t) and (y_{t+1}). The initial weights, weights covariance matrix diagonal entries, R and Q were set to 1, 100, 1×10^{-4} and 0 respectively. The sliding window to estimate Q was set to 3 time steps.

As shown in Figure 3.5, during the initialisation and after each change of behaviour (samples 150 and 225), the estimator for the process noise covariance Q becomes active. That is, each time the environment undergoes a severe change more importance is given to the new data. As the environment stabilises, the minimum variance minimisation criterion of the Kalman filter leads to a decrease in the variance of the output. Therefore, it is possible to design an estimator that balances the tradeoff between regularisation and tracking. The results obtained with the EKF and EKFQ algorithms are summarised in Table 3.2.

	RSE	**Mega floating point operations**
EKF	31.76	21.8
EKFQ	1.37	23.0

Table 3.2 *Simulation results for 100 runs of the quadratic chaotic map. The EKFQ algorithm provides a large improvement over the EKF at a very small computational cost.*

3.5 Summary

This chapter presented several algorithms to perform regularisation in sequential learning tasks. The algorithms are based on Gaussian approximations and on schemes to adapt the noise processes sequentially. The experiments indicated that one of these algorithms (EKFQ) may lead to improved prediction results when either the data source

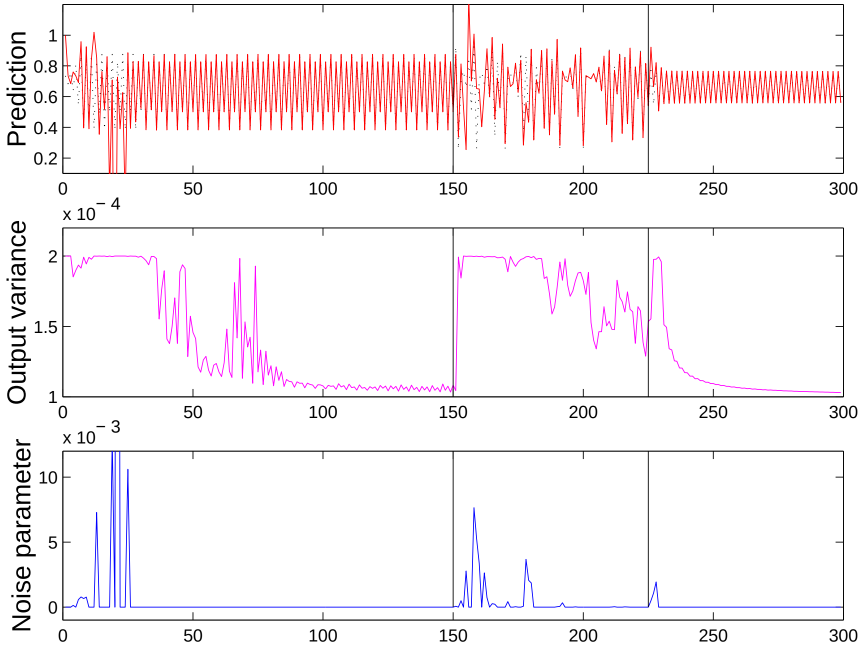

Figure 3.5 *Performance of the evidence maximisation for a non-stationary chaotic quadratic map. The top plot shows the true data [· · ·] and the prediction [—], the middle plot shows the output confidence intervals while the bottom plot shows the value of the adaptive noise parameter.*

is time varying or there is little *a priori* knowledge about how to tune the noise processes. Additional experiments are presented in Chapter 8.

It was shown that the hierarchical Bayesian inference methodology provides an elegant, unifying treatment of the sequential learning problem. Distributed learning rates, adaptive noise parameters and adaptive smoothing regularisers were shown to be mathematically equivalent. This result sheds light on many areas of the machine learning field. It places many diverse approaches to estimation and regularisation within a unified framework. Areas where further research is needed include deriving convergence bounds and implementing other model structures, such as recurrent networks.

Chapters 6 and 7 will expand the discussion on sequential learning methods. Prior to that, Chapter 4 will extend the Gaussian approximation strategy to batch learning tasks. In Chapter 5, Markov chain Monte Carlo methods will be introduced into the analysis of neural networks. This latter chapter will pave the way for the sequential algorithms introduced later.

4

Dynamic Batch Learning with the EM Algorithm

In environments where data is available in batches, it is possible to address the general problem of Bayesian learning with Gaussian approximations, in a principled way, via the expectation maximisation (EM) algorithm (Dempster et al., 1977). This chapter will focus on this learning approach and will aim to extend the current work on EM learning for dynamical linear systems to nonlinear dynamical systems. This approach will allow the weights of an MLP, the initial conditions and the noise variances to be computed jointly.

The application of the EM algorithm to learning and inference in linear dynamical systems has occupied the attention of several researchers in the past. Chen (Chen, 1981) was one of the pioneers in this field. In particular, he applied the EM algorithm to linear state space models known in the statistics literature as multiple indicators and multiple causes (MIMIC) models. In these models one observes multiple indicators and multiple causes of a single latent variable. Chen's MIMIC model was implemented in a simulation study relating social status and participation.

Watson and Engle (Watson and Engle, 1983) have suggested using the EM algorithm, in conjunction with the method of scoring, for the estimation of linear dynamic factor, MIMIC and varying coefficient regression models. They evaluated their paradigm experimentally by estimating common factors in wage rate data from several industries in Los Angeles, USA.

In 1982, Shumway and Stoffer (Shumway and Stoffer, 1982) proposed the use of the EM algorithm and linear state space models for time series smoothing and forecasting with missing observations. To demonstrate their method, they considered a health series representing total expenditures for physician services as measured by two different sources. The time series produced by each source have similar values but exhibit missing observations at different periods. In Shumway and Stoffer's approach, the two series are automatically merged into an overall expenditure series, which is then used for forecasting. Nine years later, Shumway and Stoffer (Shumway and Stoffer, 1991) extended their work to switching linear dynamic models. In essence, they derived a state space representation with measurement matrices that switch according to a time varying independent random process. They illustrate their method on an application involving the tracking of multiple targets.

The method of learning and inference in linear state space models via the EM algorithm has also played a role in the fields of speech analysis and computer vision. Digalakis, Rohlicek and Ostendorf (Digalakis et al., 1993) applied it to the speech recognition problem. They made a connection between this method and the Baum-Welch estimation algorithm for hidden Markov models (HMMs). North and Blake (North and Blake, 1998) have implemented the method to learn linear dynamic state space models used for tracking contours in images. Rao and Ballard (Rao and Ballard, 1997) have also explored the relevance of the EM algorithm together with state space estimation in the field of vision. They have developed an hierarchical network model of visual recognition that encapsulates these concepts.

Ghahramani (Ghahramani, 1997) has embedded the EM method for learning dynamic linear systems in a graphical models framework. He treats computationally intractable models, such as factorial HMMs and switching state space models, by resorting to Gibbs sampling and variational approximations. In another paper, Roweis and Ghahramani (Roweis and Ghahramani, 1999) make use of the EM algorithm and linear state space representations to present a unified view of linear Gaussian models including factor analysis, mixtures of Gaussians, standard and probabilistic versions of principal component analysis, vector quantisation, Kalman smoothing and linear hidden Markov models.

The chapter is organised as follows. Section 4.1 introduces the nonlinear state space modelling scheme adopted in this chapter. The application of extended Kalman smoothing to estimate the weights of an MLP is discussed in Section 4.2. Section 4.3 presents a brief derivation of the EM algorithm, which is used as a step towards the derivation of the EM algorithm for nonlinear state space models in Section 4.4. Section 4.5 examines some of the results obtained with synthetic data, while Section 4.6 provides a brief summary of the chapter.

4.1 Nonlinear State Space Model

To investigate the application of the EM algorithm to dynamic batch learning, the following nonlinear state space representation is used:

$$\begin{aligned} \boldsymbol{\theta}_{t+1} &= A\boldsymbol{\theta}_t + \mathbf{u}_t \\ \mathbf{y}_t &= \widehat{\mathbf{f}}(\mathbf{x}_t, \boldsymbol{\theta}_t) + \mathbf{v}_t \end{aligned}$$

Once again, the nonlinear mapping $\widehat{\mathbf{f}}(\mathbf{x}_t, \boldsymbol{\theta}_t)$ corresponds to a multi-layer perceptron. The measurement and process noise terms are assumed to be zero mean Gaussian with covariances R and Q respectively. The matrix A contains information about how the states evolve. It is particularly useful in tracking applications. However, when the above model is employed merely for parameter estimation in neural network models, the matrix A should be viewed as a mechanism to achieve directed trajectories in state space. In other words, A allows for more general jumps than the simple random walk that would result by excluding A from the model.

Despite the fact that the data is processed in batches, the model of equation (4.1) allows the weights to be time varying. It is, therefore, possible to deal with non-stationary data sets. In the event of the data being stationary, we should expect the process noise term to vanish. Consequently, if we know that the data is stationary, the estimate of the process noise can be used to determine how well the model explains the data. This is demonstrated in Section 4.5.

The objective is to estimate the model states (MLP weights) $\widehat{\boldsymbol{\theta}}_t$ and the set of parameters $\boldsymbol{\varphi} \triangleq \{R, Q, A, \boldsymbol{\mu}, \Pi\}$ given the measurements $\{\mathbf{x}_{1:N}, \mathbf{y}_{1:N}\}^1$, where $\boldsymbol{\mu}$ and Π denote the mean and covariance of the Gaussian prior $p(\boldsymbol{\theta}_0|\boldsymbol{\varphi})$.

4.2 The extended Kalman smoother

Smoothing often entails forward and backward filtering over a segment of data so as to obtain improved averaged estimates. Various techniques have been proposed to accomplish this goal (Gelb, 1974; Jazwinski, 1970). This study uses the well known Rauch-Tung-Striebel smoother (Rauch et al., 1965). The forward filtering stage involves computing the estimates $\widehat{\boldsymbol{\theta}}_t$ and P_t, over a segment of N samples, with the following EKF recursions:

$$
\begin{aligned}
\widehat{\boldsymbol{\theta}}_{t+1|t} &= A\widehat{\boldsymbol{\theta}}_t \\
P_{t+1|t} &= AP_tA' + Q \\
K_{t+1} &= P_{t+1|t}G'_{t+1}\left(R + G_{t+1}P_{t+1|t}G'_{t+1}\right)^{-1} \\
\widehat{\boldsymbol{\theta}}_{t+1} &= \widehat{\boldsymbol{\theta}}_{t+1|t} + K_{t+1}\left(\mathbf{y}_{t+1} - \widehat{\mathbf{f}}(\mathbf{x}_{t+1}, \widehat{\boldsymbol{\theta}}_{t+1|t})\right) \\
P_{t+1} &= P_{t+1|t} - K_{t+1}G_{t+1}P_{t+1|t}
\end{aligned}
$$

[1] To clarify the notation, in this chapter, the input variables $\mathbf{x}$ are suppressed in the arguments of the probability distributions.

where K denotes the Kalman gain and G the Jacobian matrix. Subsequently, the Rauch-Tung-Striebel smoother makes use of the following backward recursions:

$$\begin{aligned}
J_{t-1} &= P_{t-1} A' P_{t|t-1}^{-1} \\
\widehat{\boldsymbol{\theta}}_{t-1|N} &= \widehat{\boldsymbol{\theta}}_{t-1} J_{t-1} \big(\widehat{\boldsymbol{\theta}}_{t|N} - A \widehat{\boldsymbol{\theta}}_{t-1} \big) \\
P_{t-1|N} &= P_{t-1} + J_{t-1} \big(P_{t|N} - P_{t|t-1} \big) J_{t-1}' \\
P_{t,t-1|N} &= P_t J_{t-1}' + J_t \big(P_{t+1,t|N} - A P_t \big) J_{t-1}'
\end{aligned}$$

where the parameters, covariance and cross-covariance are defined as follows:

$$\begin{aligned}
\widehat{\boldsymbol{\theta}}_{t|N} &= \mathbb{E}\big(\boldsymbol{\theta}_t | \mathbf{y}_{1:N} \big) \\
P_{t|N} &= \mathbb{E}\big((\boldsymbol{\theta}_t - \widehat{\boldsymbol{\theta}}_t)(\boldsymbol{\theta}_t - \widehat{\boldsymbol{\theta}}_t)' | \mathbf{y}_{1:N} \big) \\
P_{t,t-1|N} &= \mathbb{E}\big((\boldsymbol{\theta}_t - \widehat{\boldsymbol{\theta}}_t)(\boldsymbol{\theta}_{t-1} - \widehat{\boldsymbol{\theta}}_{t-1})' | \mathbf{y}_{1:N} \big)
\end{aligned}$$

They may be initialised with the following values:

$$\begin{aligned}
\widehat{\boldsymbol{\theta}}_{N|N} &= \widehat{\boldsymbol{\theta}}_N \\
P_{N|N} &= P_N \\
P_{N,N-1|N} &= \big(I - K_N G_N' \big) A P_{N-1}
\end{aligned}$$

4.3 The EM Algorithm

So far, it has been shown that given a set of parameters $\boldsymbol{\varphi} = \{R, Q, A, \boldsymbol{\mu}, \Pi\}$ and a matrix of N measurements $\mathbf{y}_{1:N}$, it is possible to compute the expected values of the states with an extended Kalman smoother. This section presents an EM algorithm to learn the parameters $\boldsymbol{\varphi}$.

The EM algorithm is an iterative method for finding a mode of the likelihood function $p(\mathbf{y}_{1:N}|\boldsymbol{\varphi})$. It proceeds as follows: (E-step 1) estimate the states $\boldsymbol{\theta}_{0:N}$ given a set of parameters $\boldsymbol{\varphi}$, (M-step 1) estimate the parameters given the new states, (E-step 2) re-estimate the states with the new parameters, and so forth. The most remarkable attribute of the EM algorithm is that it ensures an increase in the likelihood function at each iteration. However, as the EKF can only provide an approximation to the true states $\boldsymbol{\theta}_{0:N}$ in the E step, the EM algorithm to train MLPs is not necessarily guaranteed to converge.

It is convenient to think of $\boldsymbol{\theta}_{0:N}$ either as missing observations or as latent variables. EM is particularly useful because many models, such as mixtures and hierarchical models, may be re-expressed in augmented parameter spaces, where the extra parameters $\boldsymbol{\theta}_{0:N}$ can be thought of as missing data. That is, in situations where it is hard to maximise $p(\mathbf{y}_{1:N}|\boldsymbol{\varphi})$, EM will allow us to accomplish this by working with $p(\mathbf{y}_{1:N}, \boldsymbol{\theta}_{0:N}|\boldsymbol{\varphi})$.

To gain more insight into the EM method, let us express the likelihood function as follows:

$$p(\mathbf{y}_{1:N}|\boldsymbol{\varphi}) = p(\mathbf{y}_{1:N}|\boldsymbol{\varphi})\frac{p(\boldsymbol{\theta}_{0:N}|\mathbf{y}_{1:N},\boldsymbol{\varphi})}{p(\boldsymbol{\theta}_{0:N}|\mathbf{y}_{1:N},\boldsymbol{\varphi})} = \frac{p(\boldsymbol{\theta}_{0:N},\mathbf{y}_{1:N}|\boldsymbol{\varphi})}{p(\boldsymbol{\theta}_{0:N}|\mathbf{y}_{1:N},\boldsymbol{\varphi})}$$

Taking the logarithms of both sides yields the following identity:

$$\ln p(\mathbf{y}_{1:N}|\boldsymbol{\varphi}) = \ln p(\boldsymbol{\theta}_{0:N},\mathbf{y}_{1:N}|\boldsymbol{\varphi}) - \ln p(\boldsymbol{\theta}_{0:N}|\mathbf{y}_{1:N},\boldsymbol{\varphi})$$

Let us treat $\boldsymbol{\theta}_{0:N}$ as a random variable with distribution $p(\boldsymbol{\theta}_{0:N}|\mathbf{y}_{1:N},\boldsymbol{\varphi}^{\text{old}})$, where $\boldsymbol{\varphi}^{\text{old}}$ is the current guess. If we then take expectations on both sides of the previous identity, while remembering that the left hand side does not depend on $\boldsymbol{\theta}_{0:N}$, we get:

$$\ln p(\mathbf{y}_{1:N}|\boldsymbol{\varphi}) = \mathbb{E}\big(\ln p(\boldsymbol{\theta}_{0:N},\mathbf{y}_{1:N}|\boldsymbol{\varphi})\big) - \mathbb{E}\big(\ln p(\boldsymbol{\theta}_{0:N}|\mathbf{y}_{1:N},\boldsymbol{\varphi})\big) \tag{4.1}$$

where the expectations involve averaging over the matrix $\boldsymbol{\theta}_{0:N}$ under the distribution $p(\boldsymbol{\theta}_{0:N}|\mathbf{y}_{1:N},\boldsymbol{\varphi}^{\text{old}})$. For example:

$$\mathbb{E}\big(\ln p(\boldsymbol{\theta}_{0:N},\mathbf{y}_{1:N}|\boldsymbol{\varphi})\big) = \int \big(\ln p(\boldsymbol{\theta}_{0:N},\mathbf{y}_{1:N}|\boldsymbol{\varphi})\big)p(\boldsymbol{\theta}_{0:N}|\mathbf{y}_{1:N},\boldsymbol{\varphi}^{\text{old}})\mathrm{d}\boldsymbol{\theta}_{0:N}$$

A key result for the EM algorithm (see Appendix C for a proof) is that the second term on the right side of equation (4.1) is maximised for $\boldsymbol{\varphi}^{\text{old}}$. That is:

$$\mathbb{E}\big(\ln p(\boldsymbol{\theta}_{0:N}|\mathbf{y}_{1:N},\boldsymbol{\varphi}^{\text{old}})\big) \geq \mathbb{E}\big(\ln p(\boldsymbol{\theta}_{0:N}|\mathbf{y}_{1:N},\boldsymbol{\varphi})\big)$$

for any $\boldsymbol{\varphi}$.

To apply the EM algorithm, we need to compute the first term on the right hand side of equation (4.1) repeatedly. The aim is to maximise this term at each iteration. One method of maximising it is discussed in detail in the next section. For the time being, let us assume that we can maximise it, that is:

$$\mathbb{E}\big(\ln p(\boldsymbol{\theta}_{0:N},\mathbf{y}_{1:N}|\boldsymbol{\varphi}^{\text{new}})\big) \geq \mathbb{E}\big(\ln p(\boldsymbol{\theta}_{0:N},\mathbf{y}_{1:N}|\boldsymbol{\varphi}^{\text{old}})\big)$$

Then, it follows that the likelihood function also increases at every iteration. To demonstrate this important result, consider the change in likelihood for a single iteration:

$$\begin{aligned}\ln p(\mathbf{y}_{1:N}|\boldsymbol{\varphi}^{\text{new}}) - \ln p(\mathbf{y}_{1:N}|\boldsymbol{\varphi}^{\text{old}}) = &\Big(\mathbb{E}\big(\ln p(\boldsymbol{\theta}_{0:N},\mathbf{y}_{1:N}|\boldsymbol{\varphi}^{\text{new}})\big) - \mathbb{E}\big(\ln p(\boldsymbol{\theta}_{0:N},\mathbf{y}_{1:N}|\boldsymbol{\varphi}^{\text{old}})\big)\Big)\\ &- \Big(\mathbb{E}\big(\ln p(\boldsymbol{\theta}_{0:N}|\mathbf{y}_{1:N},\boldsymbol{\varphi}^{\text{new}})\big) - \mathbb{E}\big(\ln p(\boldsymbol{\theta}_{0:N}|\mathbf{y}_{1:N},\boldsymbol{\varphi}^{\text{old}})\big)\Big)\end{aligned}$$

The right hand side of the above equation is positive because we are averaging under $p(\boldsymbol{\theta}_{0:N}|\mathbf{y}_{1:N},\boldsymbol{\varphi}^{\text{old}})$. Consequently, the likelihood function is guaranteed to increase at each iteration.

The EM algorithm's name originates from the steps that are required to increase $\mathbb{E}\big(\ln p(\boldsymbol{\theta}_{0:N},\mathbf{y}_{1:N}|\boldsymbol{\varphi})\big)$, namely compute the Expectation and then Maximise it. The EM algorithm thus involves the following steps:

Initialisation : Start with a guess for $\boldsymbol{\varphi}^0$.

E-step : Determine the expected log-likelihood density function of the complete data given the current estimate $\boldsymbol{\varphi}^{\text{old}}$:

$$\mathbb{E}\big(\ln p(\boldsymbol{\theta}_{0:N}, \mathbf{y}_{1:N}|\boldsymbol{\varphi})\big) = \int \big(\ln p(\boldsymbol{\theta}_{0:N}, \mathbf{y}_{1:N}|\boldsymbol{\varphi})\big) p(\boldsymbol{\theta}_{0:N}|\mathbf{y}_{1:N}, \boldsymbol{\varphi}^{\text{old}}) \mathrm{d}\boldsymbol{\theta}_{0:N}$$

M-step : Compute a new value of $\boldsymbol{\varphi}$ that maximises the expected log-likelihood of the complete data. The maximum can be found by simple differentiation of the expected log-likelihood with respect to $\boldsymbol{\varphi}$.

Note that at the M-step, we only require an increase in the expected log-likelihood of the complete data. That is, we do not need to find the maximum. This is the basis for generalised EM (GEM) algorithms (Dempster et al., 1977; Gelman et al., 1995).

4.4 The EM algorithm for nonlinear state space models

To derive the EM algorithm for nonlinear state space models, we need to develop an expression for the likelihood of the completed data. The likelihood of the data given the states, the initial conditions and the evolution of the states may be approximated by Gaussian distributions. In particular, if the initial mean and covariance of the states is given by $\boldsymbol{\mu}$ and Π, then:

$$\begin{aligned} p(\boldsymbol{\theta}_0|\boldsymbol{\varphi}) &= \frac{1}{(2\pi)^{m/2}|\Pi|^{1/2}} \exp\Big[-\frac{1}{2}(\boldsymbol{\theta}_0 - \boldsymbol{\mu})'\Pi^{-1}(\boldsymbol{\theta}_0 - \boldsymbol{\mu})\Big] \\ p(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1}, \boldsymbol{\varphi}) &= \frac{1}{(2\pi)^{m/2}|Q|^{1/2}} \exp\Big[-\frac{1}{2}(\boldsymbol{\theta}_t - A\boldsymbol{\theta}_{t-1})'Q^{-1}(\boldsymbol{\theta}_t - A\boldsymbol{\theta}_{t-1})\Big] \\ p(\mathbf{y}_t|\boldsymbol{\theta}_t, \boldsymbol{\varphi}) &= \frac{1}{(2\pi)^{c/2}|R|^{1/2}} \exp\Big[-\frac{1}{2}(\mathbf{y}_t - \widehat{\mathbf{f}}(\boldsymbol{\theta}_t, \mathbf{x}_t))'R^{-1}(\mathbf{y}_t - \widehat{\mathbf{f}}(\boldsymbol{\theta}_t, \mathbf{x}_t))\Big] \end{aligned}$$

Under the model assumptions of uncorrelated noise sources and Markov state evolution, the likelihood of the complete data is given by:

$$p(\boldsymbol{\theta}_{0:N}, \mathbf{y}_{1:N}|\boldsymbol{\varphi}) = p(\boldsymbol{\theta}_0|\boldsymbol{\varphi}) \prod_{t=1}^{N} p(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1}, \boldsymbol{\varphi}) \prod_{t=1}^{N} p(\mathbf{y}_t|\boldsymbol{\theta}_t, \boldsymbol{\varphi})$$

Hence, the log-likelihood of the complete data is given by the following expression:

$$\begin{aligned} \ln p(\boldsymbol{\theta}_{0:N}, \mathbf{y}_{1:N}|\boldsymbol{\varphi}) \;=\; & -\sum_{t=1}^{N}\Big[\frac{1}{2}(\mathbf{y}_t - \widehat{\mathbf{f}}(\boldsymbol{\theta}_t, \mathbf{x}_t))'R^{-1}(\mathbf{y}_t - \widehat{\mathbf{f}}(\boldsymbol{\theta}_t, \mathbf{x}_t))\Big] - \frac{N}{2}\ln|R| \\ & -\sum_{t=1}^{N}\Big[\frac{1}{2}(\boldsymbol{\theta}_t - A\boldsymbol{\theta}_{t-1})'Q^{-1}(\boldsymbol{\theta}_t - A\boldsymbol{\theta}_{t-1})\Big] - \frac{N}{2}\ln|Q| \\ & -\frac{1}{2}(\boldsymbol{\theta}_0 - \boldsymbol{\mu})'\Pi^{-1}(\boldsymbol{\theta}_0 - \boldsymbol{\mu}) - \frac{1}{2}\ln|\Pi| - \frac{Nc + (N+1)m}{2}\ln(2\pi) \end{aligned} \tag{4.2}$$

As discussed in the previous section, all we need to do now is to compute the expectation of $\ln p(\boldsymbol{\theta}_{0:N}, \mathbf{y}_{1:N}|\boldsymbol{\varphi})$ and then differentiate the result with respect to the parameters $\boldsymbol{\varphi}$ so as to maximise it. The EM algorithm for nonlinear state space models will thus involve computing the expected values of the states and covariances with the extended Kalman smoother and then maximising the parameters $\boldsymbol{\varphi}$ with the formulae obtained by differentiating the expected log-likelihood.

4.4.1 Computing the expectation of the log-likelihood

The derivation requires the following sufficient statistics:

$$\begin{aligned}
\mathbb{E}\big(\boldsymbol{\theta}_t|\mathbf{y}_{1:N}\big) &= \widehat{\boldsymbol{\theta}}_{t|N} \\
\mathbb{E}\big(\boldsymbol{\theta}_t\boldsymbol{\theta}_t'|\mathbf{y}_{1:N}\big) &= P_{t|N} + \widehat{\boldsymbol{\theta}}_{t|N}\widehat{\boldsymbol{\theta}}'_{t|N} \\
\mathbb{E}\big(\boldsymbol{\theta}_t\boldsymbol{\theta}_{t-1}'|\mathbf{y}_{1:N}\big) &= P_{t,t-1|N} + \widehat{\boldsymbol{\theta}}_{t|N}\widehat{\boldsymbol{\theta}}'_{t-1|N}
\end{aligned}$$

Now, taking the expectation of the log-likelihood for the complete data, by averaging over $\boldsymbol{\theta}_{0:N}$ under the distribution $p(\boldsymbol{\theta}_{0:N}|\mathbf{y}_{1:N}, \boldsymbol{\varphi}^{\text{old}})$, one gets the following expression:

$$\begin{aligned}
\mathbb{E}\big(\ln p(\boldsymbol{\theta}_{0:N}, \mathbf{y}_{1:N}|\boldsymbol{\varphi})\big) = & -\frac{N}{2}\ln|R| - \frac{N}{2}\ln|Q| - \frac{1}{2}\ln|\Pi| - \frac{Nc+(N+1)m}{2}\ln(2\pi) \\
& -\sum_{t=1}^{N}\frac{1}{2}\mathbb{E}\Big[\,\mathbf{y}_t'R^{-1}\mathbf{y}_t - \mathbf{y}_t'R^{-1}\widehat{\mathbf{f}}(\boldsymbol{\theta}_t,\mathbf{x}_t) \\
& \qquad - \widehat{\mathbf{f}}(\boldsymbol{\theta}_t,\mathbf{x}_t)'R^{-1}\mathbf{y}_t + \widehat{\mathbf{f}}(\boldsymbol{\theta}_t,\mathbf{x}_t)'R^{-1}\widehat{\mathbf{f}}(\boldsymbol{\theta}_t,\mathbf{x}_t)\Big] \\
& -\sum_{t=1}^{N}\frac{1}{2}\mathbb{E}\Big[\,\boldsymbol{\theta}_t'Q^{-1}\boldsymbol{\theta}_t - \boldsymbol{\theta}_t'Q^{-1}A\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}_{t-1}'A'Q^{-1}\boldsymbol{\theta}_t \\
& \qquad + \boldsymbol{\theta}_{t-1}'A'Q^{-1}A\boldsymbol{\theta}_{t-1}\Big] \\
& -\frac{1}{2}\mathbb{E}\Big[\,\boldsymbol{\theta}_0'\Pi^{-1}\boldsymbol{\theta}_0 - \boldsymbol{\theta}_0'\Pi^{-1}\boldsymbol{\mu} - \boldsymbol{\mu}'\Pi^{-1}\boldsymbol{\theta}_0 + \boldsymbol{\mu}'\Pi^{-1}\boldsymbol{\mu}\Big]
\end{aligned}$$

We need to digress briefly to compute the expectation of the measurements mapping $\widehat{\mathbf{f}}(\boldsymbol{\theta}_t, \mathbf{x}_t)$. We should recall that the EKF approximation to this mapping is given by:

$$\widehat{\mathbf{f}}(\boldsymbol{\theta}_t,\mathbf{x}_t) = \widehat{\mathbf{f}}(\widehat{\boldsymbol{\theta}}_{t|N},\mathbf{x}_t) + \frac{\partial\widehat{\mathbf{f}}(\boldsymbol{\theta}_t,\mathbf{x}_t)}{\partial\boldsymbol{\theta}_t}\Big|_{(\boldsymbol{\theta}_t=\widehat{\boldsymbol{\theta}}_{t|N})}(\boldsymbol{\theta}_t - \widehat{\boldsymbol{\theta}}_{t|N}) + \cdots$$

Consequently, if we take expectations on both sides of the equation, we get:

$$\mathbb{E}\big(\widehat{\mathbf{f}}(\boldsymbol{\theta}_t,\mathbf{x}_t)\big) \approx \widehat{\mathbf{f}}(\widehat{\boldsymbol{\theta}}_{t|N},\mathbf{x}_t)$$

and

$$\begin{aligned}
&\mathbb{E}\big((\widehat{\mathbf{f}}(\boldsymbol{\theta}_t,\mathbf{x}_t)-\widehat{\mathbf{f}}(\widehat{\boldsymbol{\theta}}_{t|N},\mathbf{x}_t))(\widehat{\mathbf{f}}(\boldsymbol{\theta}_t,\mathbf{x}_t)-\widehat{\mathbf{f}}(\widehat{\boldsymbol{\theta}}_{t|N},\mathbf{x}_t))'\big)\\
&\quad\approx\mathbb{E}\Big[\,\Big(\widehat{\mathbf{f}}(\widehat{\boldsymbol{\theta}}_{t|N},\mathbf{x}_t)+\frac{\partial\widehat{\mathbf{f}}(\boldsymbol{\theta}_t,\mathbf{x}_t)}{\partial\boldsymbol{\theta}_t}\Big|_{(\boldsymbol{\theta}_t=\widehat{\boldsymbol{\theta}}_{t|N})}(\boldsymbol{\theta}_t-\widehat{\boldsymbol{\theta}}_{t|N})-\widehat{\mathbf{f}}(\widehat{\boldsymbol{\theta}}_{t|N},\mathbf{x}_t)\Big)\\
&\qquad\Big(\widehat{\mathbf{f}}(\widehat{\boldsymbol{\theta}}_{t|N},\mathbf{x}_t)+\frac{\partial\widehat{\mathbf{f}}(\boldsymbol{\theta}_t,\mathbf{x}_t)}{\partial\boldsymbol{\theta}_t}\Big|_{(\boldsymbol{\theta}_t=\widehat{\boldsymbol{\theta}}_{t|N})}(\boldsymbol{\theta}_t-\widehat{\boldsymbol{\theta}}_{t|N})-\widehat{\mathbf{f}}(\widehat{\boldsymbol{\theta}}_{t|N},\mathbf{x}_t)\Big)'\,\Big]\\
&\quad=\mathbb{E}\Big[\,G_t(\boldsymbol{\theta}_t-\widehat{\boldsymbol{\theta}}_{t|N})(\boldsymbol{\theta}_t-\widehat{\boldsymbol{\theta}}_{t|N})'G_t'\,\Big]\\
&\quad=G_tP_{t|N}G_t'
\end{aligned}$$

Hence, under the distribution $p(\boldsymbol{\theta}_{0:N}|\mathbf{y}_{1:N},\boldsymbol{\varphi}^{\text{old}})$, it follows that:

$$\mathbb{E}\big(\widehat{\mathbf{f}}(\boldsymbol{\theta}_t,\mathbf{x}_t)\widehat{\mathbf{f}}(\boldsymbol{\theta}_t,\mathbf{x}_t)'\big)\approx G_tP_{t|N}G_t'+\widehat{\mathbf{f}}(\widehat{\boldsymbol{\theta}}_{t|N},\mathbf{x}_t)\widehat{\mathbf{f}}(\widehat{\boldsymbol{\theta}}_{t|N},\mathbf{x}_t)'$$

Using this approximation and the fact that the trace and expectation operators are linear, the expectation of the log-likelihood becomes:

$$\begin{aligned}
\mathbb{E}\big(\ln p(\boldsymbol{\theta}_{0:N},\mathbf{y}_{1:N}|\boldsymbol{\varphi})\big)\approx{}&-\frac{N}{2}\ln|R|-\frac{N}{2}\ln|Q|-\frac{1}{2}\ln|\Pi|-\frac{Nc+(N+1)m)}{2}\ln(2\pi)\\
&-\sum_{t=1}^{N}\frac{1}{2}\text{tr}\Big(R^{-1}\Big[\mathbf{y}_t\mathbf{y}_t'-\widehat{\mathbf{f}}(\widehat{\boldsymbol{\theta}}_{t|N},\mathbf{x}_t)\mathbf{y}_t'-\mathbf{y}_t\widehat{\mathbf{f}}(\widehat{\boldsymbol{\theta}}_{t|N},\mathbf{x}_t)'\\
&\qquad+\widehat{\mathbf{f}}(\widehat{\boldsymbol{\theta}}_{t|N},\mathbf{x}_t)\widehat{\mathbf{f}}(\widehat{\boldsymbol{\theta}}_{t|N},\mathbf{x}_t)'+G_tP_{t|N}G_t'\Big]\Big)\\
&-\sum_{t=1}^{N}\frac{1}{2}\text{tr}\Big(Q^{-1}\Big[\widehat{\boldsymbol{\theta}}_{t|N}\widehat{\boldsymbol{\theta}}_{t|N}'+P_{t|N}-2A(\widehat{\boldsymbol{\theta}}_{t|N}\widehat{\boldsymbol{\theta}}_{t-1|N}'+P_{t,t-1|N})'\\
&\qquad+A(\widehat{\boldsymbol{\theta}}_{t-1}\widehat{\boldsymbol{\theta}}_{t-1|N}'+P_{t-1|N})A'\Big]\Big)\\
&-\frac{1}{2}\text{tr}\Big(\Pi^{-1}\Big[\widehat{\boldsymbol{\theta}}_{0|N}\widehat{\boldsymbol{\theta}}_{0|N}'+P_{0|N}-2\widehat{\boldsymbol{\theta}}_{0|N}\boldsymbol{\mu}'+\boldsymbol{\mu}\boldsymbol{\mu}'\Big]\Big)
\end{aligned}$$

Completing squares and using the following abbreviations:

$$\begin{aligned}
\Gamma&=\sum_{t=1}^{N}\widehat{\boldsymbol{\theta}}_{t|N}\widehat{\boldsymbol{\theta}}_{t|N}'+P_{t|N}\\
\Delta&=\sum_{t=1}^{N}\widehat{\boldsymbol{\theta}}_{t-1|N}\widehat{\boldsymbol{\theta}}_{t-1|N}'+P_{t-1|N}\\
\Upsilon&=\sum_{t=1}^{N}\widehat{\boldsymbol{\theta}}_{t|N}\widehat{\boldsymbol{\theta}}_{t-1|N}'+P_{t,t-1|N}
\end{aligned}$$

we get our final expression for the approximate expectation of the log-likelihood:

$$\begin{aligned}
\mathbb{E}\big(\ln p(\boldsymbol{\theta}_{0:N}, \mathbf{y}_{1:N}|\boldsymbol{\varphi})\big) \approx\; & -\frac{N}{2}\ln|R| - \frac{N}{2}\ln|Q| - \frac{1}{2}\ln|\Pi| - \frac{Nc+(N+1)m}{2}\ln(2\pi) \\
& -\sum_{t=1}^{N}\frac{1}{2}\mathrm{tr}\Big(R^{-1}\Big[\big(\mathbf{y}_t - \widehat{\mathbf{f}}(\widehat{\boldsymbol{\theta}}_{t|N}, \mathbf{x}_t)\big)\big(\mathbf{y}_t - \widehat{\mathbf{f}}(\widehat{\boldsymbol{\theta}}_{t|N}, \mathbf{x}_t)\big)' \\
& \qquad + G_t P_{t|N} G_t'\Big]\Big) \\
& -\frac{1}{2}\mathrm{tr}\Big(Q^{-1}\Big[\Gamma - 2A\Upsilon' + A\Delta A'\Big]\Big) \\
& -\frac{1}{2}\mathrm{tr}\Big(\Pi^{-1}\Big[(\widehat{\boldsymbol{\theta}}_{0|N} - \boldsymbol{\mu})(\widehat{\boldsymbol{\theta}}_{0|N} - \boldsymbol{\mu})' + P_{0|N}\Big]\Big)
\end{aligned} \tag{4.3}$$

4.4.2 Differentiating the expected log-likelihood

To maximise the expected value of the log-likelihood with respect to the parameters $\boldsymbol{\varphi}$, we need to compute the derivatives with respect to each parameter individually. The following results for matrix differentiation (see for example (Graham, 1981)):

$$\begin{aligned}
\frac{\partial \ln|A|}{\partial A} &= (A^{-1})' \\
\frac{\partial \mathrm{tr}(BA)}{\partial A} &= B' \\
\frac{\partial \mathrm{tr}(A'BA)}{\partial A} &= BA + B'A
\end{aligned}$$

are used in the subsequent sections.

4.4.2.1 Maximum with respect to A

Differentiating the expected log-likelihood with respect to A yields:

$$\begin{aligned}
\frac{\partial}{\partial A}\mathbb{E}\big(\ln p(\boldsymbol{\theta}_{0:N}, \mathbf{y}_{1:N}|\boldsymbol{\varphi})\big) &\approx -\frac{1}{2}\frac{\partial}{\partial A}\mathrm{tr}\Big(Q^{-1}\Big[\Gamma - 2A\Upsilon' + A\Delta A'\Big]\Big) \\
&= -\frac{1}{2}\Big(-2Q^{-1}\Upsilon + 2Q^{-1}A\Delta\Big)
\end{aligned}$$

Equating this result to zero yields the value of A that maximises the approximate log-likelihood:

$$A = \Upsilon\Delta^{-1} \tag{4.4}$$

4.4.2.2 Maximum with respect to R

Differentiating the expected log-likelihood with respect to R^{-1} gives:

$$\begin{aligned}\frac{\partial}{\partial R^{-1}}\mathbb{E}\big(\ln p(\boldsymbol{\theta}_{0:N},\mathbf{y}_{1:N}|\boldsymbol{\varphi})\big) &\approx \frac{\partial}{\partial R^{-1}}\Big(\frac{N}{2}\ln|R^{-1}| - \sum_{t=1}^{N}\frac{1}{2}\mathrm{tr}\Big(R^{-1}\Big[G_tP_{t|N}G_t' \\ &\qquad + (\mathbf{y}_t - \widehat{\mathbf{f}}(\widehat{\boldsymbol{\theta}}_{t|N},\mathbf{x}_t))(\mathbf{y}_t - \widehat{\mathbf{f}}(\widehat{\boldsymbol{\theta}}_{t|N},\mathbf{x}_t))'\Big]\Big)\Big) \\ &= \frac{N}{2}R - \sum_{t=1}^{N}\frac{1}{2}\Big(G_tP_{t|N}G_t' \\ &\qquad + (\mathbf{y}_t - \widehat{\mathbf{f}}(\widehat{\boldsymbol{\theta}}_{t|N},\mathbf{x}_t))(\mathbf{y}_t - \widehat{\mathbf{f}}(\widehat{\boldsymbol{\theta}}_{t|N},\mathbf{x}_t))'\Big)\end{aligned}$$

Hence, by equating the above result to zero, the maximum of the approximate log-likelihood with respect to R is given by:

$$R = \frac{1}{N}\sum_{t=1}^{N}\Big(G_tP_{t|N}G_t' + (\mathbf{y}_t - \widehat{\mathbf{f}}(\widehat{\boldsymbol{\theta}}_{t|N},\mathbf{x}_t))(\mathbf{y}_t - \widehat{\mathbf{f}}(\widehat{\boldsymbol{\theta}}_{t|N},\mathbf{x}_t))'\Big) \tag{4.5}$$

4.4.2.3 Maximum with respect to Q

Following the same steps, the derivative of the expected log-likelihood with respect to Q^{-1} is given by:

$$\frac{\partial}{\partial Q^{-1}}\mathbb{E}\big(\ln p(\boldsymbol{\theta}_{0:N},\mathbf{y}_{1:N}|\boldsymbol{\varphi})\big) \approx \frac{N}{2}Q - \frac{1}{2}\Big(\Gamma - 2A\Upsilon' + A\Delta A'\Big)$$

Hence, equating to zero and using the result that $A = \Upsilon\Delta^{-1}$, the maximum of the approximate log-likelihood with respect to Q is given by:

$$Q = \frac{1}{N}\Big(\Gamma - \Upsilon\Delta^{-1}\Upsilon'\Big) \tag{4.6}$$

4.4.2.4 Maximum with respect to $\boldsymbol{\mu}$

It is also possible to treat the initial conditions as parameters and improve their estimates in the M-step of the EM algorithm. Finding the derivative of the expected log-likelihood with respect to the initial states gives:

$$\frac{\partial}{\partial\boldsymbol{\mu}}\mathbb{E}\big(\ln p(\boldsymbol{\theta}_{0:N},\mathbf{y}_{1:N}|\boldsymbol{\varphi})\big) \approx \frac{1}{2}\Pi^{-1}\Big(-2\widehat{\boldsymbol{\theta}}_{0|N} + 2\boldsymbol{\mu}\Big)$$

Hence, the initial value for the states should be:

$$\boldsymbol{\mu} = \widehat{\boldsymbol{\theta}}_{0|N} \tag{4.7}$$

4.4.2.5 Maximum with respect to Π

The derivative of the expected log-likelihood with respect to the inverse of the initial covariance gives:

$$\frac{\partial}{\partial \Pi^{-1}} \mathbb{E}\big(\ln p(\boldsymbol{\theta}_{0:N}, \mathbf{y}_{1:N}|\boldsymbol{\varphi})\big) \approx \frac{\Pi}{2} - \frac{1}{2}\left((\widehat{\boldsymbol{\theta}}_{0|N} - \boldsymbol{\mu})(\widehat{\boldsymbol{\theta}}_{0|N} - \boldsymbol{\mu})' + P_{0|N}\right)$$

Therefore, the initial covariance should be updated as follows:

$$\Pi = P_{0|N} \tag{4.8}$$

4.4.3 The E and M steps for neural network models

The EM algorithm for MLPs is as follows:

Initialisation : Start with a guess for $\boldsymbol{\varphi} = \{R, Q, A, \boldsymbol{\mu}, \Pi\}$.

E-step : Determine the expected values $\widehat{\boldsymbol{\theta}}_{t|N}$, $P_{t|N}$ and $P_{t,t-1|N}$, given the current parameter estimate $\boldsymbol{\varphi}^{\text{old}}$, using the extended Kalman smoothing equations described in Section 4.2.

M-step : Compute new values of the parameters $\boldsymbol{\varphi} = \{R, Q, A, \boldsymbol{\mu}, \Pi\}$ using equations (4.4) to (4.8).

The complexity of this algorithm is $\mathcal{O}(m^3N)$ operations per iteration.

4.5 Simple Regression Example

For demonstration purposes, the EM method is applied to the problem of learning the following nonlinear mapping from (x_1, x_2) to y:

$$y = 4\sin(x_1 - 2) + 2x_2 + 5 + \eta$$

where x_1 and x_2 were chosen to be two normal random sequences of 700 samples each. The noise process η was sampled from a zero mean Gaussian distribution with variance $R = 0.5$. An MLP with four sigmoidal neurons in the hidden layer and a linear neuron in the output layer was used to approximate the measurements mapping. After 50 iterations, as shown in Figure 4.1, the estimate of observation variance R converges to the true value. In addition, the trace of the process noise covariance Q goes to zero. Note that since the data is stationary, the trace of Q should tend to zero. That is, the trace of Q can be used to provide an estimate of how well the model fits the data. The innovations covariance (variance of the evidence function $p(\mathbf{y}_t|\mathbf{y}_{1:t-1}, \widehat{\boldsymbol{\theta}}_{t|t-1}, Q_{t-1}, R_{t-1})$) tends to R over the entire data set, as shown in Figure 4.2. The top plot of this figure shows that the MLP approximates the true function without fitting the noise. That is, it generalises well. Figure 4.1 also shows how the log-likelihood increases at each step, thereby demonstrating that the algorithm converges well.

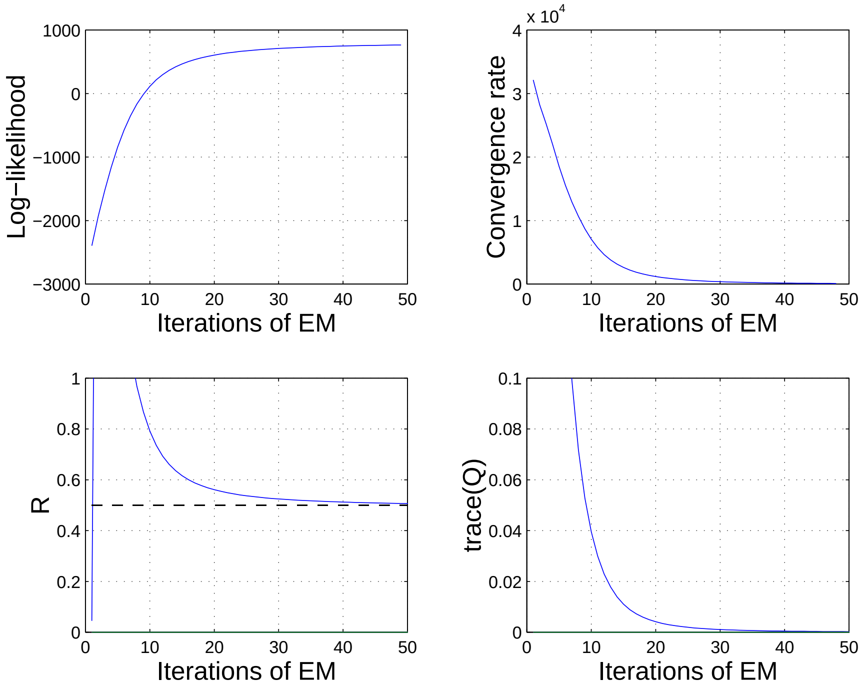

Figure 4.1 *The top plots show the log-likelihood function and the convergence rate (log-likelihood slope) for the simple regression problem. The bottom plots show the convergence of the measurements noise covariance R and the trace of the process noise covariance Q.*

4.6 Summary

In this chapter, an EM algorithm was derived to estimate the neural network weights, measurement noise and model uncertainty jointly. A simple experiment indicates that it performs well in terms of model accuracy and generalisation ability. Further research avenues include extending the method to other types of noise processes, establishing theoretical convergence bounds and investigating ways of efficiently initialising the algorithm so as to avoid local minima. The latter problem is to a large extent surmounted by the algorithms proposed in the following chapter.

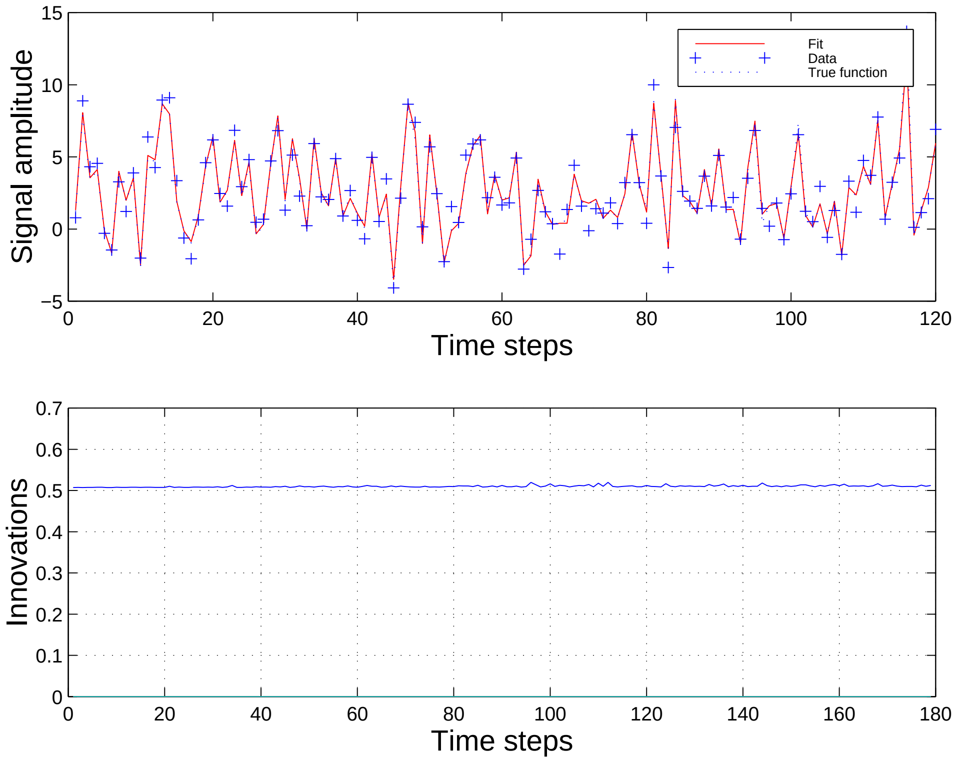

Figure 4.2 *The top plot shows that the MLP fit, for the regression example, approximates the true function (the former is almost exactly on top of the latter); it does not fit the noise. As a result the network exhibits good generalisation performance. The bottom plot shows that the uncertainty in the predictions (innovations) converges to the uncertainty engendered by the measurement noise.*

5

Robust Full Bayesian Learning with MCMC

The previous chapter described an EM batch learning strategy to estimate the noise statistics and parameters of an MLP network. This chapter moves beyond the confines of Gaussian approximation to a more flexible class of learning algorithms: Markov chain Monte Carlo. Readers without a background in MCMC simulation are encouraged to consult Appendix D for an introduction to the field. This learning strategy, albeit more computationally expensive, can perform full Bayesian inference. In other words, it allows one to obtain robust and joint estimates of the signal to noise ratios, noise statistics, network parameters and number of parameters.

The correct selection of the number of neurons is an essential requirement of neural network training. There have been three main approaches to this problem, namely penalised likelihood, predictive assessment and growing and pruning techniques. In the penalised likelihood context, a penalty term is added to the likelihood function so as to limit the number of neurons; thereby avoiding over-fitting. Classical examples of penalty terms include the well known Akaike information criterion (AIC), Bayesian information criterion (BIC) and minimum description length (MDL) (Akaike, 1974; Schwarz, 1985; Rissanen, 1987). As mentioned in Chapter 2, penalised likelihood has also been used extensively to impose smoothing constraints either via weight decay priors (Hinton, 1987; Mackay, 1992b) or functional regularisers that penalise for high frequency signal components (Girosi et al., 1995).

In the predictive assessment approach, the data is split into a training set, a validation set and possibly a test set. The key idea is to balance the bias and variance of the predictor by choosing the number of neurons so that the errors in each data set are of the same magnitude.

The problem with the previous approaches, known as the model adequacy problem, is that they assume one knows which models to test. To overcome this difficulty, various authors have proposed model selection methods, whereby the number of neurons is set by growing and pruning algorithms. Examples of this class of algorithm include the upstart algorithm (Frean, 1990), cascade correlation (Fahlman and Lebiere, 1988), optimal brain damage (Le Cun et al., 1990) and the resource allocating network (RAN) (Platt, 1991). A major shortcoming of these methods is that they lack robustness in that the results depend on several heuristically set thresholds. For argument's sake, let us consider the case of the RAN algorithm. A new radial basis function is added to the hidden layer each time an input in a novel region of the input space is found. Unfortunately, novelty is assessed in terms of two heuristically set thresholds. The centre of the Gaussian basis function is then placed at the location of the novel input, while its width depends on the distance between the novel input and the stored patterns. For improved efficiency, the amplitudes of the Gaussians may be estimated with an extended Kalman filter (Kadirkamanathan and Niranjan, 1993). Yingwei, Sundararajan and Saratchandran (Yingwei et al., 1997) have extended the approach by proposing a simple pruning technique. Their strategy is to monitor the outputs of the Gaussian basis functions continuously and compare them to a threshold. If a particular output remains below the threshold over a number of consecutive inputs, then the corresponding basis function is removed.

Recently, Rios Insua and Müller (1998) , Marrs (1998) and Holmes and Mallick (1998) have addressed the issue of selecting the number of hidden neurons with growing and pruning algorithms from a Bayesian perspective. In particular, they apply the reversible jump Markov chain Monte Carlo (MCMC) algorithm of Green (Green, 1995; Richardson and Green, 1997) to feed-forward sigmoidal networks and radial basis function (RBF) networks to obtain joint estimates of the number of neurons and weights.

This chapter also applies the reversible jump MCMC simulation algorithm to RBF networks so as to compute the joint posterior distribution of the radial basis parameters and the number of basis functions. It advances this area of research in three important directions. Firstly, it proposes an hierarchical prior for RBF networks. That is, it uses a full Bayesian model, which accounts for model order uncertainty and regularisation, and shows that the results appear to be robust with respect to the prior specification. Secondly, it proposes an automated growing and pruning reversible jump MCMC optimisation algorithm to choose the model order using the classical AIC, BIC and MDL criteria. This algorithm estimates the maximum of the joint likelihood function of the radial basis parameters and the number of bases using a reversible jump MCMC simulated annealing approach. It has the advantage of being more computationally efficient than the reversible jump MCMC algorithm used to perform the integrations with the hierarchical full Bayesian model. Finally, a geometric convergence theorem for the homogeneous reversible jump MCMC algorithm and a convergence theorem for the annealed reversible jump MCMC algorithm are presented.

The chapter is organised as follows. Section 5.1 discusses the approximation model. Section 5.2 formalises the Bayesian model and specifies the prior distributions. Section 5.3 is devoted to Bayesian computation. It first proposes an MCMC sampler to perform Bayesian inference when the number of basis functions is given. Subsequently, a reversible jump MCMC algorithm is derived to deal with the case where the number of basis functions is unknown. A reversible jump MCMC simulated annealing algorithm to perform stochastic optimisation using the AIC, BIC and MDL criteria is proposed in Section 5.4. The convergence of the algorithms is established in Section 5.5. The performance of the proposed algorithms is illustrated by computer simulations in Section 5.6. Finally, some conclusions are drawn in Section 5.7. The proofs of convergence are presented in Appendix E.

5.1 Problem Statement

This chapter adopts the approximation scheme of Holmes and Mallick (1998), consisting of a mixture of k RBFs and a linear regression term. However, the work can be straightforwardly extended to other regression models. This model, as described in Chapter 1, is given by:

$$\begin{array}{lll} \mathcal{M}_0: & \mathbf{y}_t = \boldsymbol{b} + \boldsymbol{\beta}'\mathbf{x}_t + \mathbf{n}_t & k = 0 \\ \mathcal{M}_k: & \mathbf{y}_t = \sum_{j=1}^{k} \mathbf{a}_j \phi(\|\mathbf{x}_t - \boldsymbol{\mu}_j\|) + \boldsymbol{b} + \boldsymbol{\beta}'\mathbf{x}_t + \mathbf{n}_t & k \geq 1 \end{array} \tag{5.1}$$

where $\|\cdot\|$ denotes a distance metric (usually Euclidean or Mahalanobis), $\boldsymbol{\mu}_j \in \mathbb{R}^d$ denotes the j-th RBF centre for a model with k RBFs, $\mathbf{a}_j \in \mathbb{R}^c$ denotes the j-th RBF amplitude and $\boldsymbol{b} \in \mathbb{R}^c$ and $\boldsymbol{\beta} \in \mathbb{R}^d \times \mathbb{R}^c$ denote the linear regression parameters. The noise sequence $\mathbf{n}_t \in \mathbb{R}^c$ is assumed to be zero-mean white Gaussian. It is important to mention that although the dependency of $\boldsymbol{b}$, $\boldsymbol{\beta}$ and $\mathbf{n}_t$ on k has not been made explicit, these parameters are indeed affected by the value of k.

As mentioned earlier, depending on our *a priori* knowledge about the smoothness of the mapping, we can choose different types of basis functions (Girosi et al., 1995). The most common choices are:

- Linear: $\phi(\varrho) = \varrho$
- Cubic: $\phi(\varrho) = \varrho^3$
- Thin plate spline: $\phi(\varrho) = \varrho^2 \ln(\varrho)$
- Multi-quadric: $\phi(\varrho) = \left(\varrho^2 + \lambda^2\right)^{1/2}$

- Gaussian: $\phi(\varrho) = \exp\Big(-\lambda\varrho^2\Big)$

For the last two choices of basis functions, λ is a user-set parameter. For convenience, the approximation model is expressed in vector-matrix form[1]:

$$\mathbf{y} = \mathbf{D}\big(\boldsymbol{\mu}_{1:k,1:d}, \mathbf{x}_{1:N,1:d}\big)\boldsymbol{\alpha}_{1:1+d+k,1:c} + \mathbf{n}_t \tag{5.2}$$

That is:

$$\begin{bmatrix} \mathbf{y}_{1,1}\cdots\mathbf{y}_{1,c} \\ \mathbf{y}_{2,1}\cdots\mathbf{y}_{2,c} \\ \\ \vdots \\ \\ \mathbf{y}_{N,1}\cdots\mathbf{y}_{N,c} \end{bmatrix} = \begin{bmatrix} 1 & \mathbf{x}_{1,1}\cdots\mathbf{x}_{1,d} & \phi(\mathbf{x}_1,\boldsymbol{\mu}_1)\cdots\phi(\mathbf{x}_1,\boldsymbol{\mu}_k) \\ 1 & \mathbf{x}_{2,1}\cdots\mathbf{x}_{2,d} & \phi(\mathbf{x}_2,\boldsymbol{\mu}_1)\cdots\phi(\mathbf{x}_2,\boldsymbol{\mu}_k) \\ \\ \vdots & \vdots & \vdots \\ \\ 1 & \mathbf{x}_{N,1}\cdots\mathbf{x}_{N,d} & \phi(\mathbf{x}_N,\boldsymbol{\mu}_1)\cdots\phi(\mathbf{x}_N,\boldsymbol{\mu}_k) \end{bmatrix} \begin{bmatrix} \boldsymbol{b}_1\cdots\boldsymbol{b}_c \\ \boldsymbol{\beta}_{1,1}\cdots\boldsymbol{\beta}_{1,c} \\ \vdots \\ \boldsymbol{\beta}_{d,1}\cdots\boldsymbol{\beta}_{d,c} \\ \mathbf{a}_{1,1}\cdots\mathbf{a}_{1,c} \\ \vdots \\ \mathbf{a}_{k,1}\cdots\mathbf{a}_{k,c} \end{bmatrix} + \mathbf{n}_{1:N}$$

where the noise process is assumed to be normally distributed as follows:

$$\mathbf{n}_t \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} \boldsymbol{\sigma}_1^2 & 0 & \cdots & 0 \\ 0 & \boldsymbol{\sigma}_2^2 & \cdots & 0 \\ & & \ddots & \\ 0 & 0 & \cdots & \boldsymbol{\sigma}_c^2 \end{bmatrix}\right)$$

Once again, it should be stressed that $\boldsymbol{\sigma}^2$ depends implicitly on the model order k. The number k of RBFs and their parameters $\boldsymbol{\theta} \triangleq \{\boldsymbol{\alpha}_{1:m,1:c}, \boldsymbol{\mu}_{1:k,1:d}, \boldsymbol{\sigma}^2_{1:c}\}$, with $m = 1+d+k$, are unknown. Given the data set $\{\mathbf{x}, \mathbf{y}\}$, the objective is to estimate k and $\boldsymbol{\theta} \in \boldsymbol{\Theta}_k$.

5.2 Bayesian Model and Aims

This chapter follows a Bayesian approach where the unknowns k and $\boldsymbol{\theta}$ are regarded as being drawn from appropriate prior distributions. These priors reflect our degree of belief in the relevant values of these quantities (Bernardo and Smith, 1994). An hierarchical prior structure is used to treat the priors' parameters (hyper-parameters) as random variables drawn from suitable distributions (hyper-priors). That is, instead

[1] As mentioned earlier, the notation $\mathbf{y}_{1:N,1:c}$ is used to denote an N by c matrix, where N is the number of data and c the number of outputs. That is, $\mathbf{y}_{1:N,j} \triangleq (\mathbf{y}_{1,j}, \mathbf{y}_{2,j}, \ldots, \mathbf{y}_{N,j})'$ denotes all the observations corresponding to the j-th output (j-th column of $\mathbf{y}$). To simplify the notation, $\mathbf{y}_t$ is equivalent to $\mathbf{y}_{t,1:c}$. That is, if one index does not appear, it is implied that we are referring to all of its possible values. Similarly, $\mathbf{y}$ is equivalent to $\mathbf{y}_{1:N,1:c}$. The shorter notation will be favoured, while the longer notation will be invoked to avoid ambiguities and emphasise certain dependencies.

of fixing the hyper-parameters arbitrarily, we acknowledge that there is an inherent uncertainty in what we think their values should be. By devising probabilistic models that deal with this uncertainty, it is possible to implement estimation techniques that are robust to the specification of the hyper-priors.

The remainder of the section is organised as follows. Firstly, an hierarchical model prior, which defines a probability distribution over the space of possible structures of the data, is proposed. Subsequently, the estimation and inference aims are specified. Finally, the analytical properties of the model are exploited to obtain an expression, up to a normalising constant, of the joint posterior distribution of the basis centres and their number.

5.2.1 Prior distributions

The overall parameter space $\mathbf{\Theta} \times \mathbf{\Psi}$ can be written as a finite union of subspaces $\mathbf{\Theta} \times \mathbf{\Psi} = \left(\cup_{k=0}^{k_{\max}} \{k\} \times \mathbf{\Theta}_k\right) \times \mathbf{\Psi}$ where $\mathbf{\Theta}_0 \triangleq (\mathbb{R}^{d+1})^c \times (\mathbb{R}^+)^c$ and $\mathbf{\Theta}_k \triangleq (\mathbb{R}^{d+1+k})^c \times (\mathbb{R}^+)^c \times \mathbf{\Omega}_k$ for $k \in \{1, \ldots, k_{\max}\}$. That is, $\boldsymbol{\alpha} \in (\mathbb{R}^{d+1+k})^c$, $\boldsymbol{\sigma} \in (\mathbb{R}^+)^c$ and $\boldsymbol{\mu} \in \mathbf{\Omega}_k$. The hyper-parameter space $\mathbf{\Psi} \triangleq (\mathbb{R}^+)^{c+1}$, with elements $\boldsymbol{\psi} \triangleq \{\Lambda, \boldsymbol{\delta}^2\}$, will be discussed at the end of this section.

The space of the radial basis centres $\mathbf{\Omega}_k$ is defined as a compact set that encompasses the input data: $\mathbf{\Omega}_k \triangleq \{\boldsymbol{\mu}; \boldsymbol{\mu}_{1:k,i} \in [\min(\mathbf{x}_{1:N,i}) - \iota\Xi_i, \max(\mathbf{x}_{1:N,i}) + \iota\Xi_i]^k$ for $i = 1, \ldots, d$ with $\boldsymbol{\mu}_{j,i} \neq \boldsymbol{\mu}_{l,i}$ for $j \neq l\}$. $\Xi_i = \| \max(\mathbf{x}_{1:N,i}) - \min(\mathbf{x}_{1:N,i}) \|$ denotes the Euclidean distance for the i-th dimension of the input and ι is a user specified parameter that we only need to consider if we wish to place basis functions outside the region where the input data lie. That is, $\mathbf{\Omega}_k$ is allowed to include the space of the input data and is extended by a factor which is proportional to the spread of the input data. Typically, researchers either set ι to zero and choose the basis centres from the input data (Holmes and Mallick, 1998; Kadirkamanathan and Niranjan, 1993) or compute the basis centres using clustering algorithms (Moody and Darken, 1988). The premise here is that it is better to place the basis functions where the data is dense; not in regions of extrapolation. In this case, the basis centres are sampled from the space $\mathbf{\Omega}_k$, whose hyper-volume is $\Im^k \triangleq \left(\prod_{i=1}^{d}(1 + 2\iota)\Xi_i\right)^k$. Figure 5.1 shows this space for a two-dimensional input.

The maximum number of basis functions is defined as $k_{\max} \triangleq (N - (d+1))$ [2]. The following definition is also needed $\mathbf{\Omega} \triangleq \cup_{k=0}^{k_{\max}} \{k\} \times \mathbf{\Omega}_k$ with $\mathbf{\Omega}_0 \triangleq \emptyset$. There is a natural hierarchical structure to this setup (Richardson and Green, 1997), which can

[2] The constraint $k \leq N - (d+1)$ is added because otherwise the columns of $\mathbf{D}(\boldsymbol{\mu}_{1:k}, \mathbf{x})$ are linearly dependent and the parameters $\boldsymbol{\theta}$ may not be uniquely estimated from the data (see equation (5.2)).

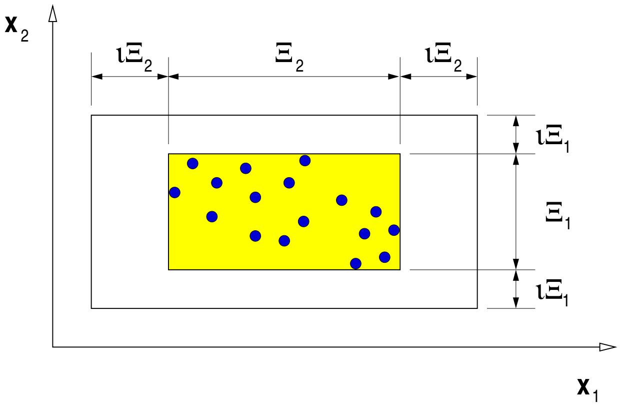

Figure 5.1 *RBF centres space $\mathbf{\Omega}$ for a two-dimensional input. The circles represent the input data.*

be formalised by modelling the joint distribution of all variables as:

$$p(k,\boldsymbol{\theta},\boldsymbol{\psi},\mathbf{y}|\mathbf{x}) = p(\mathbf{y}|k,\boldsymbol{\theta},\boldsymbol{\psi},\mathbf{x})p(\boldsymbol{\theta}|k,\boldsymbol{\psi},\mathbf{x})p(k,\boldsymbol{\psi}|\mathbf{x}) \tag{5.3}$$

where $p(k,\boldsymbol{\psi}|\mathbf{x})$ is the joint model order and hyper-parameters probability, $p(\boldsymbol{\theta}|k,\boldsymbol{\psi},\mathbf{x})$ is the parameters' prior and $p(\mathbf{y}|k,\boldsymbol{\theta},\boldsymbol{\psi},\mathbf{x})$ is the likelihood. Under the assumption of independent outputs given $(k,\boldsymbol{\theta})$, the likelihood for the approximation model described in the previous section is:

$$\begin{aligned} p(\mathbf{y}|k,\boldsymbol{\theta},\boldsymbol{\psi},\mathbf{x}) &= \prod_{i=1}^{c} p(\mathbf{y}_{1:N,i}|k,\boldsymbol{\alpha}_{1:m,i},\boldsymbol{\mu}_{1:k},\boldsymbol{\sigma}_i^2,\mathbf{x}) \\ &= \prod_{i=1}^{c}(2\pi\boldsymbol{\sigma}_i^2)^{-N/2}\exp\left(-\frac{1}{2\boldsymbol{\sigma}_i^2}\left(\mathbf{y}_{1:N,i}-\mathbf{D}(\boldsymbol{\mu}_{1:k},\mathbf{x})\boldsymbol{\alpha}_{1:m,i}\right)'\left(\mathbf{y}_{1:N,i}-\mathbf{D}(\boldsymbol{\mu}_{1:k},\mathbf{x})\boldsymbol{\alpha}_{1:m,i}\right)\right) \end{aligned} \tag{5.4}$$

The following structure is assumed for the prior distribution:

$$\begin{aligned} p(k,\boldsymbol{\theta},\boldsymbol{\psi}) &= p(\boldsymbol{\alpha}_{1:m}|k,\boldsymbol{\mu}_{1:k},\boldsymbol{\sigma}^2,\Lambda,\boldsymbol{\delta}^2)p(\boldsymbol{\mu}_{1:k}|k,\boldsymbol{\sigma}^2,\Lambda,\boldsymbol{\delta}^2)p(k|\boldsymbol{\sigma}^2,\Lambda,\boldsymbol{\delta}^2)p(\boldsymbol{\sigma}^2|\Lambda,\boldsymbol{\delta}^2)p(\Lambda,\boldsymbol{\delta}^2) \\ &= p(\boldsymbol{\alpha}_{1:m}|k,\boldsymbol{\sigma}^2,\boldsymbol{\delta}^2)p(\boldsymbol{\mu}_{1:k}|k)p(k|\Lambda)p(\boldsymbol{\sigma}^2)p(\Lambda)p(\boldsymbol{\delta}^2) \end{aligned} \tag{5.5}$$

where the scale parameters $\boldsymbol{\sigma}_i^2$, $i = 1,\ldots,c$, are assumed to be independent of the hyper-parameters (that is, $p(\boldsymbol{\sigma}^2|\Lambda,\boldsymbol{\delta}^2) = p(\boldsymbol{\sigma}^2)$), independent of each other ($p(\boldsymbol{\sigma}^2) = \prod_{i=1}^{c} p(\boldsymbol{\sigma}_i^2)$) and distributed according to conjugate inverse-Gamma prior distributions:

$$\boldsymbol{\sigma}_i^2 \sim \mathcal{IG}\left(\frac{\upsilon_0}{2},\frac{\gamma_0}{2}\right)$$

When $\upsilon_0 = 0$ and $\gamma_0 = 0$, we obtain Jeffreys' uninformative prior $p\left(\boldsymbol{\sigma}_i^2\right) \propto 1/\boldsymbol{\sigma}_i^2$

(Bernardo and Smith, 1994). Given $\boldsymbol{\sigma}^2$, the following prior distribution is introduced:

$$\begin{aligned} p(k, \boldsymbol{\alpha}_{1:m}, \boldsymbol{\mu}_{1:k} | \boldsymbol{\sigma}^2, \Lambda, \boldsymbol{\delta}^2) &= p(\boldsymbol{\alpha}_{1:m} | k, \boldsymbol{\mu}_{1:k}, \boldsymbol{\sigma}^2, \Lambda, \boldsymbol{\delta}^2) p(\boldsymbol{\mu}_{1:k} | k, \boldsymbol{\sigma}^2) p(k | \boldsymbol{\sigma}^2, \Lambda, \boldsymbol{\delta}^2) \\ &= \left[\prod_{i=1}^{c} |2\pi \boldsymbol{\sigma}_i^2 \boldsymbol{\Sigma}_i|^{-1/2} \exp\left(-\frac{1}{2\boldsymbol{\sigma}_i^2} \boldsymbol{\alpha}'_{1:m,i} \boldsymbol{\Sigma_i^{-1}} \boldsymbol{\alpha}_{1:m,i} \right) \right] \left[\frac{\mathbb{I}_{\boldsymbol{\Omega}}(k, \boldsymbol{\mu}_{1:k})}{\Im^k} \right] \left[\frac{\Lambda^k / k!}{\sum_{j=0}^{k_{\max}} \Lambda^j / j!} \right] \end{aligned} \tag{5.6}$$

where $\boldsymbol{\Sigma}_i^{-1} = \boldsymbol{\delta}_i^{-2} \mathbf{D}'(\boldsymbol{\mu}_{1:k}, \mathbf{x}) \mathbf{D}(\boldsymbol{\mu}_{1:k}, \mathbf{x})$ and $\mathbb{I}_{\boldsymbol{\Omega}}(k, \boldsymbol{\mu}_{1:k})$ is the indicator function of the set $\boldsymbol{\Omega}$ (1 if $(k, \boldsymbol{\mu}_{1:k}) \in \boldsymbol{\Omega}$, 0 otherwise).

The prior model order distribution $p(k|\Lambda)$ is a truncated Poisson distribution. Conditional upon k, the RBF centres are uniformly distributed. Finally, conditional upon $(k, \boldsymbol{\mu}_{1:k})$, the coefficients $\boldsymbol{\alpha}_{1:m,i}$ are assumed to be zero-mean Gaussian with variance $\boldsymbol{\sigma}_i^2 \boldsymbol{\Sigma}_i$. The terms $\boldsymbol{\delta}^2 \in (\mathbb{R}^+)^c$ and $\Lambda \in \mathbb{R}^+$ can be respectively interpreted as the expected signal to noise ratios and the expected number of radial bases. The prior for the coefficients has been previously advocated by various authors (George and Foster, 1997; Smith and Kohn, 1996). It corresponds to the popular g-prior distribution (Zellner, 1986) and can be derived using a maximum entropy approach (Andrieu, 1998). An important property of this prior is that it penalises basis functions being too close as, in this situation, the determinant of $\boldsymbol{\Sigma}_i^{-1}$ tends to zero.

The hyper-parameters allow us to accomplish the goal of designing robust model selection schemes. They are assumed to be independent of each other, hence $p(\Lambda, \boldsymbol{\delta}^2) = p(\Lambda) p(\boldsymbol{\delta}^2)$. Moreover, $p(\boldsymbol{\delta}^2) = \prod_{i=1}^{c} p(\boldsymbol{\delta}_i^2)$. As $\boldsymbol{\delta}^2$ is a scale parameter, vague conjugate prior density is ascribed to it: $\boldsymbol{\delta}_i^2 \sim \mathcal{IG}(\alpha_{\delta^2}, \beta_{\delta^2})$ for $i = 1, \ldots, c$, with $\alpha_{\delta^2} = 2$ and $\beta_{\delta^2} > 0$. The variance of this hyper-prior with $\alpha_{\delta^2} = 2$ is infinite. The same method is applied to Λ by setting an uninformative conjugate prior (Bernardo and Smith, 1994): $\Lambda \sim \mathcal{G}a\left(1/2 + \varepsilon_1, \varepsilon_2\right)$ $(\varepsilon_i \ll 1 \; i = 1, 2)$. The hierarchical prior (equation (5.5)) can be visualised with a directed acyclic graphical model (DAG) as shown in Figure 5.2.

5.2.2 Estimation and inference aims

The Bayesian inference of k, $\boldsymbol{\theta}$ and $\boldsymbol{\psi}$ is based on the joint posterior distribution $p(k, \boldsymbol{\theta}, \boldsymbol{\psi} | \mathbf{x}, \mathbf{y})$ obtained from Bayes' theorem. Our aim is to estimate this joint distribution from which, by standard probability marginalisation and transformation techniques, one can "theoretically" obtain all posterior features of interest. For instance, one might wish to perform inference with the predictive density:

$$p(\mathbf{y}_{N+1} | \mathbf{x}_{1:N+1}, \mathbf{y}_{1:N}) = \int_{\boldsymbol{\Theta} \times \boldsymbol{\Psi}} p(\mathbf{y}_{N+1} | k, \boldsymbol{\theta}, \boldsymbol{\psi}, \mathbf{x}_{N+1}) p(k, \boldsymbol{\theta}, \boldsymbol{\psi} | \mathbf{x}_{1:N}, \mathbf{y}_{1:N}) \mathrm{d}k \mathrm{d}\boldsymbol{\theta} \mathrm{d}\boldsymbol{\psi} \tag{5.7}$$

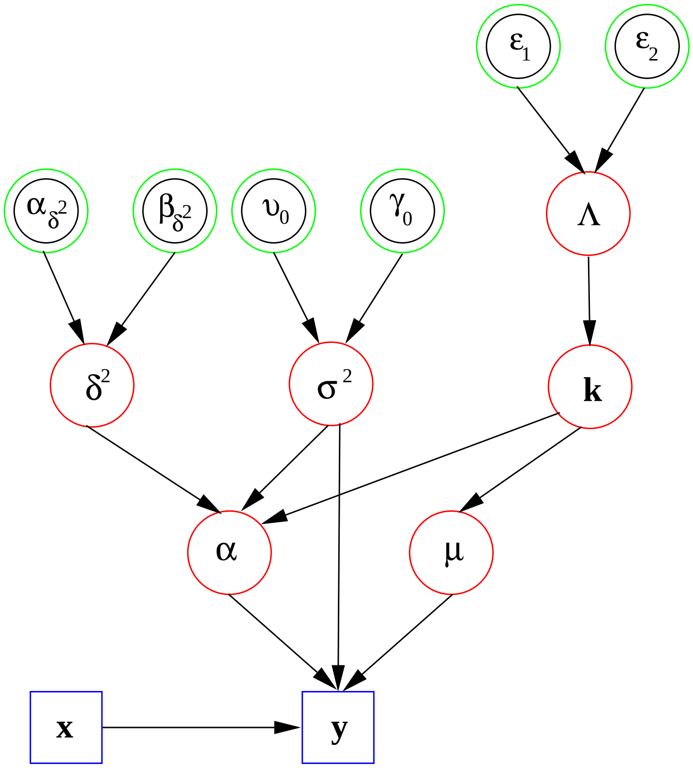

Figure 5.2 *Directed acyclic graphical model for our prior.*

and consequently make predictions, such as:

$$\mathbb{E}(\mathbf{y}_{N+1}|\mathbf{x}_{1:N+1},\mathbf{y}_{1:N}) = \int_{\mathbf{\Theta}\times\mathbf{\Psi}} \mathbf{D}(\boldsymbol{\mu}_{1:k},\mathbf{x}_{N+1})\boldsymbol{\alpha}_{1:m}p(k,\boldsymbol{\theta},\boldsymbol{\psi}|\mathbf{x}_{1:N},\mathbf{y}_{1:N})\mathrm{d}k\mathrm{d}\boldsymbol{\theta}\mathrm{d}\boldsymbol{\psi} \quad (5.8)$$

One might also be interested in evaluating the posterior model probabilities $p(k|\mathbf{x},\mathbf{y})$, which can be used to perform model selection by selecting the model order according to the criterion $\underset{k\in\{0,\ldots,k_{\max}\}}{\arg\max}\ p(k|\mathbf{x},\mathbf{y})$. In addition, we can perform parameter estimation by computing, for example, the conditional expectation $\mathbb{E}(\boldsymbol{\theta}|k,\mathbf{x},\mathbf{y})$.

It is not possible to obtain these quantities analytically, as it requires the evaluation of high dimensional integrals of nonlinear functions in the parameters, as shown in the following subsection. Here, an MCMC method is proposed to perform the necessary Bayesian computations. MCMC techniques were introduced in the mid 1950's in statistical physics and started appearing in the fields of applied statistics, signal processing and neural networks in the 1980's and 1990's (Besag et al., 1995; Holmes and Mallick, 1998; Müller and Rios Insua, 1998; Neal, 1996; Rios Insua and Müller, 1998; Tierney, 1994). The key idea is to build an ergodic Markov chain $(k^{(i)},\boldsymbol{\theta}^{(i)},\boldsymbol{\psi}^{(i)})_{i\in\mathbb{N}}$ whose equilibrium distribution is the desired posterior distribution. Under weak additional assumptions, the $P \gg 1$ samples generated by the Markov chain are asymptotically distributed according to the posterior distribution and thus allow easy evaluation of all posterior features of interest. For example:

$$\widehat{p}(k=j|\mathbf{x},\mathbf{y}) = \frac{1}{P}\sum_{i=1}^{P}\mathbb{I}_{\{j\}}(k^{(i)}) \text{ and } \widehat{\mathbb{E}}(\boldsymbol{\theta}|k=j,\mathbf{x},\mathbf{y}) = \frac{\sum_{i=1}^{P}\boldsymbol{\theta}^{(i)}\mathbb{I}_{\{j\}}(k^{(i)})}{\sum_{i=1}^{P}\mathbb{I}_{\{j\}}(k^{(i)})} \tag{5.9}$$

In addition, we can obtain predictions, such as:

$$\widehat{\mathbb{E}}(\mathbf{y}_{N+1}|\mathbf{x}_{1:N+1},\mathbf{y}_{1:N}) = \frac{1}{P}\sum_{i=1}^{P}\mathbf{D}(\boldsymbol{\mu}_{1:k}^{(i)},\mathbf{x}_{N+1})\boldsymbol{\alpha}_{1:m}^{(i)} \tag{5.10}$$

As shown in the next subsection, $\boldsymbol{\alpha}_{1:m}$ can be integrated analytically. Consequently, the variance of the predictions can be reduced by employing the following Rao Blackwellised estimate (Liu et al., 1994):

$$\widehat{\mathbb{E}}(\mathbf{y}_{N+1}|\mathbf{x}_{1:N+1},\mathbf{y}_{1:N}) = \frac{1}{P}\sum_{i=1}^{P}\mathbf{D}(\boldsymbol{\mu}_{1:k}^{(i)},\mathbf{x}_{N+1})\mathbb{E}(\boldsymbol{\alpha}_{1:m}|k^{(i)},\boldsymbol{\mu}_{1:k}^{(i)},\sigma_k^{2(i)},\boldsymbol{\delta}^{2(i)},\mathbf{x},\mathbf{y})$$

5.2.3 Integration of the nuisance parameters

The proposed Bayesian model allows for the integration of the so-called nuisance parameters, $\boldsymbol{\alpha}_{1:m}$ and $\boldsymbol{\sigma}^2$, leading to an expression for the distribution $p(k,\boldsymbol{\mu}_{1:k},\Lambda,\boldsymbol{\delta}^2|\mathbf{x},\mathbf{y})$ up to a normalising constant. According to Bayes theorem:

$$\begin{aligned}
p(k,\boldsymbol{\alpha}_{1:m},\boldsymbol{\mu}_{1:k},\boldsymbol{\sigma}^2,\Lambda,\boldsymbol{\delta}^2|\mathbf{x},\mathbf{y}) &\propto p(\mathbf{y}|k,\boldsymbol{\alpha}_{1:m},\boldsymbol{\mu}_{1:k},\boldsymbol{\sigma}^2,\Lambda,\boldsymbol{\delta}^2,\mathbf{x})p(k,\boldsymbol{\alpha}_{1:m},\boldsymbol{\mu}_{1:k},\boldsymbol{\sigma}^2,\Lambda,\boldsymbol{\delta}^2)\\
&\propto\left[\prod_{i=1}^{c}(2\pi\boldsymbol{\sigma}_i^2)^{-\frac{N}{2}}\exp\Big(-\frac{1}{2\boldsymbol{\sigma}_i^2}\big(\mathbf{y}_{1:N,i}-\mathbf{D}(\boldsymbol{\mu}_{1:k},\mathbf{x})\boldsymbol{\alpha}_{1:m,i}\big)'\big(\mathbf{y}_{1:N,i}-\mathbf{D}(\boldsymbol{\mu}_{1:k},\mathbf{x})\boldsymbol{\alpha}_{1:m,i}\big)\Big)\right]\\
&\times\left[\prod_{i=1}^{c}|2\pi\boldsymbol{\sigma}_i^2\boldsymbol{\Sigma}_i|^{-1/2}\exp\Big(-\frac{1}{2\boldsymbol{\sigma}_i^2}\boldsymbol{\alpha}'_{1:m,i}\boldsymbol{\Sigma}_i^{-1}\boldsymbol{\alpha}_{1:m,i}\Big)\right]\left[\frac{\mathbb{I}_{\boldsymbol{\Omega}}(k,\boldsymbol{\mu}_{1:k})}{\Im^k}\right]\left[\frac{\Lambda^k/k!}{\sum_{j=0}^{k_{\max}}\Lambda^j/j!}\right]\\
&\times\left[\prod_{i=1}^{c}(\boldsymbol{\sigma}_i^2)^{-(v_0/2+1)}\exp\Big(-\frac{\gamma_0}{2\boldsymbol{\sigma}_i^2}\Big)\right]\left[\prod_{i=1}^{c}(\boldsymbol{\delta}_i^2)^{-(\alpha_{\delta^2}+1)}\exp\Big(-\frac{\beta_{\delta^2}}{\boldsymbol{\delta}_i^2}\Big)\right]\\
&\times\left[(\Lambda)^{(\varepsilon_1-1/2)}\exp\Big(-\varepsilon_2\Lambda\Big)\right]
\end{aligned} \tag{5.11}$$

We can then proceed to multiply the exponential terms of the likelihood and coefficients prior, and complete squares to obtain:

$$\begin{aligned}
p(k, \boldsymbol{\alpha}_{1:m}, \boldsymbol{\mu}_{1:k}, \boldsymbol{\sigma}^2, \Lambda, \boldsymbol{\delta}^2 | \mathbf{x}, \mathbf{y}) \propto & \left[\prod_{i=1}^{c} (2\pi\boldsymbol{\sigma}_i^2)^{-N/2} \exp\Big(-\frac{1}{2\boldsymbol{\sigma}_i^2} \mathbf{y}'_{1:N,i} \mathbf{P}_{i,k} \mathbf{y}_{1:N,i} \Big) \right] \\
& \times \left[\prod_{i=1}^{c} |2\pi\boldsymbol{\sigma}_i^2 \boldsymbol{\Sigma}_i|^{-1/2} \exp\Big(-\frac{1}{2\boldsymbol{\sigma}_i^2} (\boldsymbol{\alpha}_{1:m,i} - \mathbf{h}_{i,k})' \mathbf{M}_{i,k}^{-1} (\boldsymbol{\alpha}_{1:m,i} - \mathbf{h}_{i,k}) \Big) \right] \\
& \times \left[\frac{\mathbb{I}_{\boldsymbol{\Omega}}(k, \boldsymbol{\mu}_{1:k})}{\Im^k} \right] \left[\frac{\Lambda^k / k!}{\sum_{j=0}^{k_{\max}} \Lambda^j / j!} \right] \left[\prod_{i=1}^{c} (\boldsymbol{\sigma}_i^2)^{-(\upsilon_0/2+1)} \exp\Big(-\frac{\gamma_0}{2\boldsymbol{\sigma}_i^2} \Big) \right] \\
& \times \left[\prod_{i=1}^{c} (\boldsymbol{\delta}_i^2)^{-(\alpha_{\delta^2}+1)} \exp\Big(-\frac{\beta_{\delta^2}}{\boldsymbol{\delta}_i^2} \Big) \right] \left[(\Lambda)^{(\varepsilon_1 - 1/2)} \exp\Big(-\varepsilon_2 \Lambda \Big) \right]
\end{aligned}$$

where:

$$\begin{aligned}
\mathbf{M}_{i,k}^{-1} &= \mathbf{D}'(\boldsymbol{\mu}_{1:k}, \mathbf{x}) \mathbf{D}(\boldsymbol{\mu}_{1:k}, \mathbf{x}) + \boldsymbol{\Sigma}_i^{-1} \\
\mathbf{h}_{i,k} &= \mathbf{M}_{i,k} \mathbf{D}'(\boldsymbol{\mu}_{1:k}, \mathbf{x}) \mathbf{y}_{1:N,i} \\
\mathbf{P}_{i,k} &= \mathbf{I}_N - \mathbf{D}(\boldsymbol{\mu}_{1:k}, \mathbf{x}) \mathbf{M}_{i,k} \mathbf{D}'(\boldsymbol{\mu}_{1:k}, \mathbf{x})
\end{aligned}$$

We can now integrate with respect to $\boldsymbol{\alpha}_{1:m}$ (Gaussian distribution) and with respect to $\boldsymbol{\sigma}_i^2$ (inverse Gamma distribution) to obtain the following expression for the posterior:

$$\begin{aligned}
p(k, \boldsymbol{\mu}_{1:k}, \Lambda, \boldsymbol{\delta}^2 | \mathbf{x}, \mathbf{y}) \propto & \left[\prod_{i=1}^{c} \big(1 + \boldsymbol{\delta}_i^2\big)^{-m/2} \Big(\frac{\gamma_0 + \mathbf{y}'_{1:N,i} \mathbf{P}_{i,k} \mathbf{y}_{1:N,i}}{2} \Big)^{(-\frac{N+\upsilon_0}{2})} \right] \\
& \times \left[\frac{\mathbb{I}_{\boldsymbol{\Omega}}(k, \boldsymbol{\mu}_k)}{\Im^k} \right] \left[\frac{\Lambda^k / k!}{\sum_{j=0}^{k_{\max}} \Lambda^j / j!} \right] \left[\prod_{i=1}^{c} (\boldsymbol{\delta}_i^2)^{-(\alpha_{\delta^2}+1)} \exp\Big(-\frac{\beta_{\delta^2}}{\boldsymbol{\delta}_i^2} \Big) \right] \\
& \times \left[(\Lambda)^{(\varepsilon_1 - 1/2)} \exp\Big(-\varepsilon_2 \Lambda \Big) \right] \qquad (5.12)
\end{aligned}$$

It is worth noticing that the posterior distribution is highly non-linear in the RBF centres $\boldsymbol{\mu}_k$ and that an expression of $p(k|\mathbf{x}, \mathbf{y})$ cannot be obtained in closed-form.

5.3 Bayesian Computation

For the sake of clarity, k is initially assumed to be given. After dealing with this fixed dimension scenario, an algorithm where k is treated as an unknown random variable is presented.

5.3.1 MCMC sampler for fixed dimension

The following hybrid MCMC sampler, which combines Gibbs steps and Metropolis-Hastings (MH) steps (Besag et al., 1995; Gilks et al., 1996; Tierney, 1994), is proposed:

Fixed dimension MCMC algorithm

```
1. Initialisation. Fix the value of $k$ and set $(\boldsymbol{\theta}^{(0)}, \boldsymbol{\psi}^{(0)})$ and $i = 1$.
2. Iteration $i$
   • For $j = 1, ..., k$
       – Sample $u \sim \mathcal{U}_{[0,1]}$.
       – If $u < \varpi$, perform a Metropolis-Hastings step admitting $p(\boldsymbol{\mu}_{j,1:d}|\mathbf{x}, \mathbf{y}, \boldsymbol{\mu}_{-j,1:d}^{(i)})$ as
         invariant distribution and $q_1(\boldsymbol{\mu}_{j,1:d}^{\star}|\boldsymbol{\mu}_{j,1:d}^{(i)})$ as proposal distribution.
       – Else perform a Metropolis-Hastings step using $p(\boldsymbol{\mu}_{j,1:d}|\mathbf{x}, \mathbf{y}, \boldsymbol{\mu}_{-j,1:d}^{(i)})$ as invariant
         distribution and $q_2(\boldsymbol{\mu}_{j,1:d}^{\star}|\boldsymbol{\mu}_{j,1:d}^{(i)})$ as proposal distribution.
     End For.
   • Sample the nuisance parameters $(\boldsymbol{\alpha}_{1:m}^{(i)}, \boldsymbol{\sigma}^{2(i)})$ using equations (5.16) and (5.17).
   • Sample the hyper-parameters $(\Lambda^{(i)}, \boldsymbol{\delta}^{2(i)})$ using equations (5.18) and (5.19) .
3. $i \leftarrow i + 1$ and go to 2.
                                                                                ■
```

The simulation parameter ϖ is a real number satisfying $0 < \varpi < 1$. Its value indicates our belief about which proposal distribution leads to faster convergence. If we have no preference for a particular proposal, we can set it to 0.5. The various steps of the algorithm are detailed in the following subsections. In order to simplify the notation, the superscript $\cdot^{(i)}$ is dropped from all variables at iteration i.

5.3.1.1 Updating the RBF centres

Sampling the RBF centres is difficult because the distribution is nonlinear in these parameters. Here, they are sampled one-at-a-time using a mixture of MH steps. An MH step of invariant distribution $\pi(\mathbf{z})$ and proposal distribution $q(\mathbf{z}^{\star}|\,\mathbf{z})$ involves sampling a candidate value $\mathbf{z}^{\star}$ given the current value $\mathbf{z}$ according to $q(\mathbf{z}^{\star}|\,\mathbf{z})$. The Markov chain then moves towards $\mathbf{z}^{\star}$ with probability $\mathcal{A}(\mathbf{z}, \mathbf{z}^{\star}) \triangleq \min\{1, (\pi(\mathbf{z})q(\mathbf{z}^{\star}|\mathbf{z}))^{-1}\pi(\mathbf{z}^{\star})$ $q(\mathbf{z}|\mathbf{z}^{\star})\}$, otherwise it remains equal to $\mathbf{z}$. This algorithm is very general, but for good performance in practice, it is necessary to use "clever" proposal distributions to avoid rejecting too many candidates.

According to equation (5.12), the target distribution is the full conditional distribution of a basis centre:

$$p(\boldsymbol{\mu}_{j,1:d}|\mathbf{x},\mathbf{y},\boldsymbol{\mu}_{-j,1:d}) \propto \left[\prod_{i=1}^{c}\left(\frac{\gamma_0+\mathbf{y}'_{1:N,i}\mathbf{P}_{i,k}\mathbf{y}_{1:N,i}}{2}\right)^{(-\frac{N+v_0}{2})}\right]\mathbb{I}_{\boldsymbol{\Omega}}(k,\boldsymbol{\mu}_{1:k}) \tag{5.13}$$

where $\boldsymbol{\mu}_{-j,1:d}$ denotes $\{\boldsymbol{\mu}_{1,1:d},\boldsymbol{\mu}_{2,1:d},\ldots,\boldsymbol{\mu}_{j-1,1:d},\boldsymbol{\mu}_{j+1,1:d},\ldots,\boldsymbol{\mu}_{k,1:d}\}$.

With probability $0<\varpi<1$, the proposal $q_1(\boldsymbol{\mu}^{\star}_{j,1:d}|\boldsymbol{\mu}_{j,1:d})$ corresponds to randomly sampling a basis centre from the interval $[\min(\mathbf{x}_{1:N,i})-\iota\Xi_i,\max(\mathbf{x}_{1:N,i})+\iota\Xi_i]^k$ for $i=1,\ldots,d$. The motivation for using such a proposal distribution is that the regions where the data is dense are reached quickly. Subsequently, with probability $1-\varpi$, we can perform an MH step with proposal distribution $q_2(\boldsymbol{\mu}^{\star}_{j,1:d}|\boldsymbol{\mu}_{j,1:d})$:

$$\boldsymbol{\mu}^{\star}_{j,1:d}|\boldsymbol{\mu}_{j,1:d}\sim\mathcal{N}(\boldsymbol{\mu}_{j,1:d},\sigma^2_{RW}\mathbf{I}_d) \tag{5.14}$$

This proposal distribution yields a candidate $\boldsymbol{\mu}^{\star}_{j,1:d}$ which is a perturbation of the current centre. The perturbation is a zero-mean Gaussian random variable with variance $\sigma^2_{RW}\mathbf{I}_d$. This random walk is introduced to perform a local exploration of the posterior distribution. In both cases, the acceptance probability is given by:

$$\mathcal{A}(\boldsymbol{\mu}_{j,1:d},\boldsymbol{\mu}^{\star}_{j,1:d})=\min\left\{1,\left[\prod_{i=1}^{c}\left(\frac{\gamma_0+\mathbf{y}'_{1:N,i}\mathbf{P}_{i,k}\mathbf{y}_{1:N,i}}{\gamma_0+\mathbf{y}'_{1:N,i}\mathbf{P}^{\star}_{i,k}\mathbf{y}_{1:N,i}}\right)^{(\frac{N+v_0}{2})}\right]\mathbb{I}_{\boldsymbol{\Omega}}(k,\boldsymbol{\mu}^{\star}_{1:k})\right\} \tag{5.15}$$

where $\mathbf{P}^{\star}_{i,k}$ and $\mathbf{M}^{\star}_{i,k}$ are similar to $\mathbf{P}_{i,k}$ and $\mathbf{M}_{i,k}$ with $\boldsymbol{\mu}_{1:k,1:d}$ replaced by $\{\boldsymbol{\mu}_{1,1:d},\boldsymbol{\mu}_{2,1:d},\ldots,\boldsymbol{\mu}_{j-1,1:d},\boldsymbol{\mu}^{\star}_{j,1:d},\boldsymbol{\mu}_{j+1,1:d},\ldots,\boldsymbol{\mu}_{k,1:d}\}$. The combination of these proposal distributions has been found to work well in practice.

5.3.1.2 Sampling the nuisance parameters

Section 5.2.3 derived an expression for $p(k,\boldsymbol{\mu}_{1:k},\Lambda,\boldsymbol{\delta}^2|\mathbf{x},\mathbf{y})$ from the full posterior distribution $p(k,\boldsymbol{\alpha}_{1:m},\boldsymbol{\mu}_{1:k},\boldsymbol{\sigma}^2,\Lambda,\boldsymbol{\delta}^2|\mathbf{x},\mathbf{y})$ by performing some algebraic manipulations and integrating with respect to $\boldsymbol{\alpha}_{1:m}$ (Gaussian distribution) and $\boldsymbol{\sigma}^2$ (inverse Gamma distribution). As a result, if we take into consideration that:

$$\begin{aligned}p(k,\boldsymbol{\alpha}_{1:m},\boldsymbol{\mu}_{1:k},\boldsymbol{\sigma}^2,\Lambda,\boldsymbol{\delta}^2|\mathbf{x},\mathbf{y}) &= p(\boldsymbol{\alpha}_{1:m}|k,\boldsymbol{\mu}_{1:k},\boldsymbol{\sigma}^2,\Lambda,\boldsymbol{\delta}^2,\mathbf{x},\mathbf{y})p(k,\boldsymbol{\mu}_{1:k},\boldsymbol{\sigma}^2,\Lambda,\boldsymbol{\delta}^2|\mathbf{x},\mathbf{y})\\ &= p(\boldsymbol{\alpha}_{1:m}|k,\boldsymbol{\mu}_{1:k},\boldsymbol{\sigma}^2,\Lambda,\boldsymbol{\delta}^2,\mathbf{x},\mathbf{y})p(\boldsymbol{\sigma}^2|k,\boldsymbol{\mu}_{1:k},\Lambda,\boldsymbol{\delta}^2,\mathbf{x},\mathbf{y})p(k,\boldsymbol{\mu}_{1:k},\Lambda,\boldsymbol{\delta}^2|\mathbf{x},\mathbf{y})\end{aligned}$$

it follows that, for $i=1,\ldots,c$, $\boldsymbol{\alpha}_{1:m,i}$ and $\boldsymbol{\sigma}^2_{i,k}$ are distributed according to:

$$\begin{aligned}\boldsymbol{\sigma}^2_i|(k,\boldsymbol{\mu}_{1:k},\boldsymbol{\delta}^2,\mathbf{x},\mathbf{y}) &\sim \mathcal{IG}\left(\frac{v_0+N}{2},\frac{\gamma_0+\mathbf{y}'_{1:N,i}\mathbf{P}_{i,k}\mathbf{y}_{1:N,i}}{2}\right) &(5.16)\\ \boldsymbol{\alpha}_{1:m,i}|(k,\boldsymbol{\mu}_{1:k},\boldsymbol{\sigma}^2,\boldsymbol{\delta}^2,\mathbf{x},\mathbf{y}) &\sim \mathcal{N}(\mathbf{h}_{i,k},\boldsymbol{\sigma}^2_i\mathbf{M}_{i,k}) &(5.17)\end{aligned}$$

5.3.1.3 Sampling the hyper-parameters

By considering $p(k, \boldsymbol{\alpha}_{1:m}, \boldsymbol{\mu}_{1:k}, \boldsymbol{\sigma}^2, \Lambda, \boldsymbol{\delta}^2|\mathbf{x}, \mathbf{y})$, we can clearly see that the hyper-parameters $\boldsymbol{\delta}_i$ (for $i = 1, \ldots, c$) can be simulated from the full conditional distribution:

$$\boldsymbol{\delta}_i^2|(k, \boldsymbol{\alpha}_{1:m}, \boldsymbol{\mu}_{1:k}, \boldsymbol{\sigma}_i^2, \mathbf{x}, \mathbf{y}) \sim \mathcal{IG}\left(\alpha_{\delta^2} + \frac{m}{2}, \beta_{\delta^2} + \frac{1}{2\boldsymbol{\sigma}_i^2}\boldsymbol{\alpha}'_{1:m,i}\mathbf{D}'(\boldsymbol{\mu}_{1:k}, \mathbf{x})\mathbf{D}(\boldsymbol{\mu}_{1:k}, \mathbf{x})\boldsymbol{\alpha}_{1:m,i}\right) \tag{5.18}$$

On the other hand, an expression for the posterior distribution of Λ is not so straightforward, because the prior for k is a truncated Poisson distribution. Λ can be simulated using the MH algorithm with a proposal corresponding to the full conditional that would be obtained if the prior for k was an infinite Poisson distribution. That is, we can use the following Gamma proposal for Λ:

$$q(\Lambda^\star) \propto \Lambda^{\star(1/2+\varepsilon_1+k)} \exp\big(-(1+\varepsilon_2)\Lambda^\star\big) \tag{5.19}$$

and subsequently perform a Metropolis-Hastings step with the full conditional distribution $p(\Lambda|k, \boldsymbol{\mu}_{1:k}, \boldsymbol{\delta}^2, \mathbf{x}, \mathbf{y})$ as invariant distribution.

5.3.2 MCMC sampler for unknown dimension

Now let us consider the case where k is unknown. Here, the Bayesian computation for the estimation of the joint posterior distribution $p(k, \boldsymbol{\theta}, \boldsymbol{\psi}|\mathbf{x}, \mathbf{y})$ is even more complex. One obvious solution would consist of upper bounding k by say $k_{\max}$ and running $k_{\max} + 1$ independent MCMC samplers, each being associated with a fixed number $k = 0, ..., k_{\max}$. However, this approach suffers from severe drawbacks. Firstly, it is computationally very expensive since $k_{\max}$ can be large. Secondly, the same computational effort is attributed to each value of k. In fact, some of these values are of no interest in practice because they have a very weak posterior model probability $p(k|\mathbf{x}, \mathbf{y})$.

Another solution would be to construct an MCMC sampler that would be able to sample directly from the joint distribution on $\boldsymbol{\Theta} \times \boldsymbol{\Psi} = \left(\cup_{k=0}^{k_{\max}} \{k\} \times \boldsymbol{\Theta}_k\right) \times \boldsymbol{\Psi}$. Standard MCMC methods are not able to "jump" between subspaces $\boldsymbol{\Theta}_k$ of different dimensions. However, recently, Green has introduced a new flexible class of MCMC samplers, the so-called reversible jump MCMC, that are capable of jumping between subspaces of different dimensions (Green, 1995). This is a general state-space MH algorithm (see (Andrieu et al., 1999f) for an introduction). One proposes candidates according to a set of proposal distributions. These candidates are randomly accepted according to an acceptance ratio which ensures reversibility and thus invariance of the Markov chain with respect to the posterior distribution. Here, the chain must move across subspaces of different dimensions, and therefore the proposal distributions are more complex: see

(Green, 1995; Richardson and Green, 1997) for details. For our problem, the following moves have been selected:

1. Birth of a new basis by proposing its location in the interval $[\min(\mathbf{x}_{1:N,i}) - \iota\Xi_i, \max(\mathbf{x}_{1:N,i}) + \iota\Xi_i]^k$ for $i = 1, \ldots, d$.
2. Death of an existing basis by removing it at random.
3. Merge a randomly chosen basis function and its closest neighbour into a single basis function.
4. Split a randomly chosen basis function into two neighbour basis functions, such that the distance between them is shorter than the distance between the proposed basis function and any other existing basis function. This distance constraint ensures reversibility.
5. Update the RBF centres.

These moves are defined by heuristic considerations, the only condition to be fulfilled being to maintain the correct invariant distribution. A particular choice will only have influence on the convergence rate of the algorithm. The birth and death moves allow the network to grow from k to $k+1$ and decrease from k to $k-1$ respectively. The split and merge moves also perform dimension changes from k to $k+1$ and k to $k-1$. The merge move serves to avoid the problem of placing too many basis functions in the same neighbourhood. That is, when amplitudes of many basis functions, in a close neighbourhood, add to the amplitude that would be obtained by using fewer basis functions, the merge move combines some of these basis functions. On the other hand, the split move is useful in regions of the data where there are close components. Other moves may be proposed, but the ones suggested here have been found to lead to satisfactory results.

The resulting transition kernel of the simulated Markov chain is then a mixture of the different transition kernels associated with the moves described above. This means that at each iteration one of the candidate moves, birth, death, merge, split or update, is randomly chosen. The probabilities for choosing these moves are b_k, d_k, m_k, s_k and u_k respectively, such that $b_k + d_k + m_k + s_k + u_k = 1$ for all $0 \leq k \leq k_{\max}$. A move is performed if the algorithm accepts it. For $k = 0$ the death, split and merge moves are impossible, so that $d_0 \triangleq 0$, $s_0 \triangleq 0$ and $m_0 \triangleq 0$. The merge move is also not permitted for $k = 1$, that is $m_1 \triangleq 0$. For $k = k_{\max}$, the birth and split moves are not allowed and therefore $b_{k_{\max}} \triangleq 0$ and $s_{k_{\max}} \triangleq 0$. Except in the cases described above, the following probabilities are adopted:

$$b_k \triangleq c^\star \min\left\{1, \frac{p\,(k+1)}{p\,(k)}\right\}, \; d_{k+1} \triangleq c^\star \min\left\{1, \frac{p\,(k)}{p\,(k+1)}\right\} \tag{5.20}$$

where $p(k)$ is the prior probability of model $\mathcal{M}_k$ and $c^\star$ is a parameter which tunes the proportion of dimension/update moves. As pointed out in (Green, 1995), this choice ensures that $b_k p(k) \left[d_{k+1} p(k+1)\right]^{-1} = 1$, which means that an MH algorithm in a single dimension, with no observations, would have 1 as acceptance probability. The parameter $c^\star$ is set to 0.25 and then $b_k + d_k + m_k + s_k \in [0.25, 1]$ for all k (Green, 1995). In addition, $m_k = d_k$ and $s_k = b_k$. The main steps of the algorithm can now be described as follows:

Reversible Jump MCMC algorithm

```
1. Initialisation: set $(k^{(0)}, \boldsymbol{\theta}^{(0)}, \boldsymbol{\psi}^{(0)}) \in \boldsymbol{\Theta} \times \boldsymbol{\Psi}$.
2. Iteration $i$.
    • Sample $u \sim \mathcal{U}_{[0,1]}$.
    • If $(u \leq b_{k^{(i)}})$
        – then "birth" move (See Section 5.3.2.1).
        – else if $(u \leq b_{k^{(i)}} + d_{k^{(i)}})$ then "death" move (See Section 5.3.2.1).
        – else if $(u \leq b_{k^{(i)}} + d_{k^{(i)}} + s_{k^{(i)}})$ then "split" move (See Section 5.3.2.2).
        – else if $(u \leq b_{k^{(i)}} + d_{k^{(i)}} + s_{k^{(i)}} + m_{k^{(i)}})$ then "merge" move (See Section 5.3.2.2).
        – else update the RBF centres (See Section 5.3.2.3).
      End If.
    • Sample the nuisance parameters $\left(\boldsymbol{\sigma}_k^{2(i)}, \boldsymbol{\alpha}_k^{(i)}\right)$ using equations (5.16) and (5.17).
    • Simulate the hyper-parameters $(\Lambda^{(i)}, \boldsymbol{\delta}^{2(i)})$ using equations (5.18) and (5.19) .
3. $i \leftarrow i + 1$ and go to 2.
```

∎

These different moves are explained in the following subsections. Once again, in order to simplify the notation, superscript $\cdot^{(i)}$ is dropped from all variables at iteration i.

5.3.2.1 Birth and death moves

Suppose that the current state of the Markov chain is in $\{k\} \times \mathbf{\Theta}_k \times \mathbf{\Psi}$, then:

Birth move

- Propose a new RBF centre at random from the interval $[\min(\mathbf{x}_{1:N,i}) - \iota\Xi_i, \max(\mathbf{x}_{1:N,i}) + \iota\Xi_i]$ for $i = 1, \ldots, d$.
- Evaluate $\mathcal{A}_{birth}$, see equation(5.24), and sample $u \sim \mathcal{U}_{[0,1]}$.
- If $u \leq \mathcal{A}_{birth}$ then the state of the Markov chain becomes $\left(k+1, \boldsymbol{\mu}_{1:k+1}\right)$, else it remains equal to $(k, \boldsymbol{\mu}_{1:k})$.

Now, assume that the current state of the Markov chain is in $\{k\} \times \mathbf{\Theta}_k \times \mathbf{\Psi}$, then:

Death move

- Choose the basis centre, to be deleted, at random among the k existing bases.
- Evaluate $\mathcal{A}_{death}$, see equation (5.24), and sample $u \sim \mathcal{U}_{[0,1]}$.
- If $u \leq \mathcal{A}_{death}$ then the state of the Markov chain becomes $\left(k-1, \boldsymbol{\mu}_{1:k-1}\right)$, else it remains equal to $(k, \boldsymbol{\mu}_{1:k})$.

The acceptance ratio for the proposed birth move is deduced from the following expression (Green, 1995):

$$r_{birth} \triangleq (posterior\ distributions\ ratio) \times (proposal\ ratio) \times (Jacobian) \tag{5.21}$$

That is:

$$r_{birth} = \frac{p(k+1, \boldsymbol{\mu}_{1:k+1}, \Lambda, \boldsymbol{\delta}^2|\mathbf{x}, \mathbf{y})}{p(k, \boldsymbol{\mu}_{1:k}, \Lambda, \boldsymbol{\delta}^2|\mathbf{x}, \mathbf{y})} \times \frac{d_{k+1}/(k+1)}{b_k/\Im} \times \left|\frac{\partial(\boldsymbol{\mu}_{1:k+1})}{\partial(\boldsymbol{\mu}_{1:k}, \boldsymbol{\mu}^{\star})}\right|$$

Clearly, the Jacobian is equal to 1 and after simplification we obtain:

$$r_{birth} = \left[\prod_{i=1}^{c} \frac{1}{(1+\boldsymbol{\delta}_i^2)^{1/2}} \left(\frac{\gamma_0 + \mathbf{y}'_{1:N,i}\mathbf{P}_{i,k}\mathbf{y}_{1:N,i}}{\gamma_0 + \mathbf{y}'_{1:N,i}\mathbf{P}_{i,k+1}\mathbf{y}_{1:N,i}}\right)^{\left(\frac{N+v_0}{2}\right)}\right] \frac{1}{(k+1)} \tag{5.22}$$

Similarly, for the death move:

$$r_{death} = \frac{p(k-1, \boldsymbol{\mu}_{1:k-1}, \Lambda, \boldsymbol{\delta}^2|\mathbf{x},\mathbf{y})}{p(k, \boldsymbol{\mu}_{1:k}, \Lambda, \boldsymbol{\delta}^2|\mathbf{x},\mathbf{y})} \times \frac{b_{k-1}/\Im}{d_k/k} \times \left| \frac{\partial(\boldsymbol{\mu}_{1:k-1}, \boldsymbol{\mu}^\star)}{\partial(\boldsymbol{\mu}_{1:k})} \right|$$

and consequently:

$$r_{death} = \left[\prod_{i=1}^{c} (1+\boldsymbol{\delta}_i^2)^{1/2} \left(\frac{\gamma_0 + \mathbf{y}'_{1:N,i}\mathbf{P}_{i,k}\mathbf{y}_{1:N,i}}{\gamma_0 + \mathbf{y}'_{1:N,i}\mathbf{P}_{i,k-1}\mathbf{y}_{1:N,i}} \right)^{(\frac{N+v_0}{2})} \right] k \tag{5.23}$$

The acceptance probabilities corresponding to the described moves are:

$$\mathcal{A}_{birth} = \min\{1, r_{birth}\}, \mathcal{A}_{death} = \min\{1, r_{death}\} \tag{5.24}$$

5.3.2.2 Split and merge moves

The merge move involves randomly selecting a basis function ($\boldsymbol{\mu}_1$) and then combining it with its closest neighbour ($\boldsymbol{\mu}_2$) into a single basis function $\boldsymbol{\mu}$, whose new location is:

$$\boldsymbol{\mu} = \frac{\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2}{2} \tag{5.25}$$

The corresponding split move that guarantees reversibility is:

$$\begin{aligned} \boldsymbol{\mu}_1 &= \boldsymbol{\mu} - u_{ms}\varsigma^\star \\ \boldsymbol{\mu}_2 &= \boldsymbol{\mu} + u_{ms}\varsigma^\star \end{aligned} \tag{5.26}$$

where $\varsigma^\star$ is a simulation parameter and $u_{ms} \sim \mathcal{U}_{[0,1]}$. Note that to ensure reversibility, the merge move is only performed if $\|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\| < 2\varsigma^\star$. Suppose now that the current state of the Markov chain is in $\{k\} \times \boldsymbol{\Theta}_k \times \boldsymbol{\Psi}$, then:

Split move

- Randomly choose an existing RBF centre.
- Substitute it for two neighbour basis functions, whose centres are obtained using equation (5.26). The new centres must be bound to lie in the space $\boldsymbol{\Omega}_k$ and the distance (typically Euclidean) between them has to be shorter than the distance between the proposed basis function and any other existing basis function.
- Evaluate $\mathcal{A}_{split}$, see equation(5.29), and sample $u \sim \mathcal{U}_{[0,1]}$.
- If $u \leq \mathcal{A}_{split}$ then the state of the Markov chain becomes $\left(k+1, \boldsymbol{\mu}_{1:k+1}\right)$, else it remains equal to $(k, \boldsymbol{\mu}_{1:k})$.

■

Now, assume that the current state of the Markov chain is in $\{k\} \times \boldsymbol{\Theta}_k \times \boldsymbol{\Psi}$, then:

Merge move

- Choose a basis centre at random among the k existing bases. Then find the closest basis function to it applying some distance metric, *e.g.* Euclidean.
- If $\|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\| < 2\varsigma^\star$, substitute the two basis functions for a single basis function in accordance with equation (5.25).
- Evaluate $\mathcal{A}_{merge}$, see equation (5.29), and sample $u \sim \mathcal{U}_{[0,1]}$.
- If $u \leq \mathcal{A}_{merge}$ then the state of the Markov chain becomes $\left(k-1, \boldsymbol{\mu}_{1:k-1}\right)$, else it remains equal to $(k, \boldsymbol{\mu}_{1:k})$.

■

The acceptance ratio for the proposed split move is given by:

$$r_{split} = \frac{p(k+1, \boldsymbol{\mu}_{1:k+1}, \Lambda, \boldsymbol{\delta}^2|\mathbf{x},\mathbf{y})}{p(k, \boldsymbol{\mu}_{1:k}, k, \Lambda, \boldsymbol{\delta}^2|\mathbf{x},\mathbf{y})} \times \frac{m_{k+1}/(k+1)}{p(u_{ms})s_k/k} \times \left|\frac{\partial(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2)}{\partial(\boldsymbol{\mu}, u_{ms})}\right|$$

In this case, the Jacobian is equal to:

$$J_{split} = \left|\frac{\partial(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2)}{\partial(\boldsymbol{\mu}, u_{ms})}\right| = \begin{vmatrix} 1 & 1 \\ -\varsigma^\star & \varsigma^\star \end{vmatrix} = 2\varsigma^\star$$

and, after simplification, we obtain:

$$r_{split} = \left[\prod_{i=1}^{c} \frac{1}{(1+\boldsymbol{\delta}_i^2)^{1/2}} \left(\frac{\gamma_0 + \mathbf{y}'_{1:N,i}\mathbf{P}_{i,k}\mathbf{y}_{1:N,i}}{\gamma_0 + \mathbf{y}'_{1:N,i}\mathbf{P}_{i,k+1}\mathbf{y}_{1:N,i}}\right)^{\left(\frac{N+v_0}{2}\right)} l\right] \frac{k\varsigma^\star}{\Im(k+1)} \tag{5.27}$$

Similarly, for the merge move:

$$r_{merge} = \frac{p(k-1, \boldsymbol{\mu}_{1:k-1}, \Lambda, \boldsymbol{\delta}^2|\mathbf{x},\mathbf{y})}{p(k, \boldsymbol{\mu}_{1:k}, \Lambda, \boldsymbol{\delta}^2|\mathbf{x},\mathbf{y})} \times \frac{s_{k-1}/(k-1)}{m_k/k} \times \left|\frac{\partial(\boldsymbol{\mu}, u_{ms})}{\partial(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2)}\right|$$

and, since $J_{merge} = 1/2\varsigma^\star$, it follows that:

$$r_{merge} = \left[\prod_{i=1}^{c} (1+\boldsymbol{\delta}_i^2)^{1/2} \left(\frac{\gamma_0 + \mathbf{y}'_{1:N,i}\mathbf{P}_{i,k}\mathbf{y}_{1:N,i}}{\gamma_0 + \mathbf{y}'_{1:N,i}\mathbf{P}_{i,k-1}\mathbf{y}_{1:N,i}}\right)^{\left(\frac{N+v_0}{2}\right)}\right] \frac{k\Im}{\varsigma^\star(k-1)} \tag{5.28}$$

The acceptance probabilities for the split and merge moves are:

$$\mathcal{A}_{split} = \min\{1, r_{split}\}, \mathcal{A}_{merge} = \min\{1, r_{merge}\} \tag{5.29}$$

5.3.2.3 Update move

The update move does not involve changing the dimension of the model. It requires an iteration of the fixed dimension MCMC sampler presented in Section 5.3.1.

The method presented so far can be very accurate, yet it can be computationally demanding. The following section presents a method that requires optimisation instead of integration to obtain estimates of the parameters and model dimension. This method, although less accurate, as shown in Section 5.6, is less computationally demanding. The choice of one method over the other should ultimately depend on the modelling constraints and specifications.

5.4 Reversible Jump Simulated Annealing

This section shows that traditional model selection criteria within a penalised likelihood framework, such as AIC, BIC and MDL (Akaike, 1974; Schwarz, 1985; Rissanen, 1987), can be shown to correspond to particular hyper-parameter choices in our hierarchical Bayesian formulation. That is, it is possible to calibrate the prior choices so that the problem of model selection within the penalised likelihood context can be mapped exactly to a problem of model selection via posterior probabilities. This technique has been previously applied in the areas of variable selection (George and Foster, 1997) and the detection of harmonics in noisy signals (Andrieu and Doucet, 1998b).

After resolving the calibration problem, maximum likelihood estimation, with the aforementioned model selection criteria, is performed by maximising the calibrated posterior distribution. To accomplish this goal, an MCMC simulated annealing algorithm, which makes use of the homogeneous reversible jump MCMC kernel as proposal, is proposed. This approach has the advantage that we can start with an arbitrary model order and the algorithm will perform dimension jumps until it finds the “true” model order. That is, one does not have to resort to the more expensive task of running a fixed dimension algorithm for each possible model order and subsequently selecting the best model.

5.4.1 Penalised likelihood model selection

Traditionally, penalised likelihood model order selection strategies, based on standard information criteria, require the evaluation of the maximum likelihood (ML) estimates for each model order. The number of required evaluations can be prohibitively expensive unless appropriate heuristics are available. Subsequently, a particular model $\mathcal{M}_s$ is selected if it is the one that minimises the sum of the log-likelihood and a penalty term that depends on the model dimension (Djurić, 1998; Gelfand and Dey, 1997). In mathematical terms, this estimate is given by:

$$\mathcal{M}_s = \underset{\mathcal{M}_k : k \in \{0,\dots,k_{\max}\}}{\arg\min} \left\{ -\log(p(\mathbf{y}|k, \widehat{\boldsymbol{\theta}}, \mathbf{x})) + \mathcal{P} \right\} \tag{5.30}$$

where $\widehat{\boldsymbol{\theta}} = (\widehat{\boldsymbol{\alpha}}_{1:m}, \widehat{\boldsymbol{\mu}}_{1:k}, \widehat{\boldsymbol{\sigma}}_k^2)$ is the ML estimate of $\boldsymbol{\theta}$ for model $\mathcal{M}_k$. $\mathcal{P}$ is a penalty term that depends on the model order. Examples of ML penalties include the well known AIC, BIC and MDL information criteria (Akaike, 1974; Schwarz, 1985; Rissanen, 1987). The expressions for these in the case of Gaussian observation noise are:

$$\mathcal{P}_{\text{AIC}} = \xi \qquad \text{and} \qquad \mathcal{P}_{\text{BIC}} = \mathcal{P}_{\text{MDL}} = \frac{\xi}{2}\log(N) \tag{5.31}$$

where ξ denotes the number of model parameters ($k(c+1) + c(1+d)$ in the case of an RBF network). These criteria are motivated by different factors: AIC is based on expected information, BIC is an asymptotic Bayes factor and MDL involves evaluating the minimum information required to transmit some data and a model, which describes the data, over a communications channel.

Using the conventional estimate of the variance for Gaussian distributions:

$$\widehat{\boldsymbol{\sigma}}_i^2 = \frac{1}{N}\left(\mathbf{y}_{1:N,i} - \mathbf{D}(\widehat{\boldsymbol{\mu}}_{1:k},\mathbf{x})\widehat{\boldsymbol{\alpha}}_{1:m,i}\right)'\left(\mathbf{y}_{1:N,i} - \mathbf{D}(\widehat{\boldsymbol{\mu}}_{1:k},\mathbf{x})\widehat{\boldsymbol{\alpha}}_{1:m,i}\right) = \frac{1}{N}\mathbf{y}'_{1:N,i}\mathbf{P}^*_{i,k}\mathbf{y}_{1:N,i}$$

where $\mathbf{P}^*_{i,k}$ is the least squares orthogonal projection matrix:

$$\mathbf{P}^*_{i,k} = \mathbf{I}_N - \mathbf{D}(\widehat{\boldsymbol{\mu}}_{1:k},\mathbf{x})\left[\mathbf{D}'(\widehat{\boldsymbol{\mu}}_{1:k},\mathbf{x})\mathbf{D}(\widehat{\boldsymbol{\mu}}_{1:k},\mathbf{x})\right]^{-1}\mathbf{D}'(\widehat{\boldsymbol{\mu}}_{1:k},\mathbf{x})$$

we can expand equation (5.30) as follows:

$$\begin{aligned}
\mathcal{M}_s &= \underset{\mathcal{M}_k : k \in \{0,\dots,k_{\max}\}}{\arg\min} \left\{ -\log\left[\prod_{i=1}^{c}(2\pi\widehat{\boldsymbol{\sigma}}_i^2)^{-N/2}\exp\left(-\frac{1}{2\widehat{\boldsymbol{\sigma}}_i^2}\left(\mathbf{y}_{1:N,i} - \mathbf{D}(\widehat{\boldsymbol{\mu}}_{1:k},\mathbf{x})\widehat{\boldsymbol{\alpha}}_{1:m,i}\right)'\right.\right.\right. \\
&\qquad \left.\left.\left.\left(\mathbf{y}_{1:N,i} - \mathbf{D}(\widehat{\boldsymbol{\mu}}_{1:k},\mathbf{x})\widehat{\boldsymbol{\alpha}}_{1:m,i}\right)\right)\right] + \mathcal{P}\right\} \\
&= \underset{\mathcal{M}_k : k \in \{0,\dots,k_{\max}\}}{\arg\min} \left\{ \frac{N}{2}\sum_{i=1}^{c}\log\left(2\pi\widehat{\boldsymbol{\sigma}}_i^2\right) + \sum_{i=1}^{c}\frac{1}{2\widehat{\boldsymbol{\sigma}}_i^2}\left(\mathbf{y}_{1:N,i} - \mathbf{D}(\widehat{\boldsymbol{\mu}}_{1:k},\mathbf{x})\widehat{\boldsymbol{\alpha}}_{1:m,i}\right)'\right. \\
&\qquad \left.\left(\mathbf{y}_{1:N,i} - \mathbf{D}(\widehat{\boldsymbol{\mu}}_{1:k},\mathbf{x})\widehat{\boldsymbol{\alpha}}_{1:m,i}\right) + \mathcal{P}\right\} \\
&= \underset{\mathcal{M}_k : k \in \{0,\dots,k_{\max}\}}{\arg\min} \left\{ \frac{N}{2}\sum_{i=1}^{c}\log\left(\widehat{\boldsymbol{\sigma}}_i^2\right) + \mathcal{P}\right\} \\
&= \underset{\mathcal{M}_k : k \in \{0,\dots,k_{\max}\}}{\arg\min} \left\{ \frac{N}{2}\sum_{i=1}^{c}\log\left(\mathbf{y}'_{1:N,i}\mathbf{P}^*_{i,k}\mathbf{y}_{1:N,i}\right) + \mathcal{P}\right\} \\
&= \underset{\mathcal{M}_k : k \in \{0,\dots,k_{\max}\}}{\arg\max} \left\{ \left[\prod_{i=1}^{c}(\mathbf{y}'_{1:N,i}\mathbf{P}^*_{i,k}\mathbf{y}_{1:N,i})^{-N/2}\right]\exp(-\mathcal{P})\right\} \qquad (5.32)
\end{aligned}$$

The following subsection shows that calibrating the priors in the hierarchical Bayes model leads to the expression given by equation (5.32).

5.4.2 Calibration

It is useful and elucidating to impose some restrictions on the Bayesian hierarchical prior (equation (5.12)) to obtain the AIC and MDL criteria. We may begin by assuming that the hyper-parameter $\boldsymbol{\delta}$ is fixed to a particular value, say $\overline{\boldsymbol{\delta}}$, and that we no longer have a definite expression for the model prior $p(k)$, so that:

$$p(k, \boldsymbol{\mu}_{1:k}|\mathbf{x}, \mathbf{y}) \propto \left[\prod_{i=1}^{c}(1+\overline{\boldsymbol{\delta}}_i^2)^{-m/2}\left(\frac{\gamma_0 + \mathbf{y}'_{1:N,i}\mathbf{P}_{i,k}\mathbf{y}_{1:N,i}}{2}\right)^{(-\frac{N+v_0}{2})}\right]\left[\frac{\mathbb{I}_{\boldsymbol{\Omega}}(k, \boldsymbol{\mu}_k)}{\Im^k}\right]p(k)$$

Furthermore, setting $v_0 = 0$ and $\gamma_0 = 0$ to obtain Jeffreys' uninformative prior $p(\boldsymbol{\sigma}_i^2) \propto 1/\boldsymbol{\sigma}_i^2$ results in the following expression:

$$p(k, \boldsymbol{\mu}_{1:k}|\mathbf{x}, \mathbf{y}) \propto \left[\prod_{i=1}^{c}\left(1+\overline{\boldsymbol{\delta}}_i^2\right)^{-k/2}\left(\mathbf{y}'_{1:N,i}\mathbf{P}_{i,k}\mathbf{y}_{1:N,i}\right)^{-\frac{N}{2}}\right]\left[\frac{\mathbb{I}_{\boldsymbol{\Omega}}(k, \boldsymbol{\mu}_k)}{\Im^k}\right]p(k)$$

where:

$$\begin{aligned}
\mathbf{M}_{i,k}^{-1} &= \left(1+\overline{\boldsymbol{\delta}}_i^{-2}\right)\mathbf{D}'(\boldsymbol{\mu}_{1:k}, \mathbf{x})\mathbf{D}(\boldsymbol{\mu}_{1:k}, \mathbf{x}) \\
\mathbf{h}_{i,k} &= \mathbf{M}_{i,k}\mathbf{D}'(\boldsymbol{\mu}_{1:k}, \mathbf{x})\mathbf{y}_{1:N,i} \\
\mathbf{P}_{i,k} &= \mathbf{I}_N - \mathbf{D}(\boldsymbol{\mu}_{1:k}, \mathbf{x})\mathbf{M}_{i,k}\mathbf{D}'(\boldsymbol{\mu}_{1:k}, \mathbf{x})
\end{aligned}$$

Finally, we can select $\overline{\boldsymbol{\delta}}_i^2$ and $p(k)$ such that:

$$\left[\prod_{i=1}^{c}\left(1+\overline{\boldsymbol{\delta}}_i^2\right)^{-k/2}\right]\left[\frac{\mathbb{I}_{\boldsymbol{\Omega}}(k, \boldsymbol{\mu}_k)}{\Im^k}\right]p(k) = \exp(-\mathcal{P}) \propto \exp(-\mathcal{C}k) \tag{5.33}$$

thereby ensuring that the expression for the calibrated posterior $p(k, \boldsymbol{\mu}_{1:k}|\mathbf{x}, \mathbf{y})$ corresponds to the term that needs to be maximised in the penalised likelihood framework (equation (5.32)). Note that for the purposes of optimisation, we only need the proportionality condition with $\mathcal{C} = c + 1$ for the AIC criterion and $\mathcal{C} = (c+1)\log(N)/2$ for the MDL and BIC criteria. We could, for example, satisfy the proportionality by remaining in the compact set $\boldsymbol{\Omega}$ and choosing the prior:

$$p(k) = \frac{\Lambda^k}{\sum_{j=0}^{k_{\max}} \Lambda^j}$$

with the following fixed value for Λ:

$$\overline{\Lambda} = \left[\prod_{i=1}^{c}\left(1+\overline{\boldsymbol{\delta}}_i^2\right)^{\frac{1}{2}}\right]\Im\exp(-\mathcal{C}) \tag{5.34}$$

In addition, we have to let $\overline{\boldsymbol{\delta}} \rightarrow \infty$ so that $\mathbf{P}_{i,k} \rightarrow \mathbf{P}_{i,k}^*$.

It has thus been shown that by calibrating the priors in the hierarchical Bayesian formulation, in particular by treating Λ and $\boldsymbol{\delta}^2$ as fixed quantities instead of as random variables, letting $\overline{\boldsymbol{\delta}} \rightarrow \infty$, choosing an uninformative Jeffreys' prior for $\boldsymbol{\sigma}^2$ and setting Λ as in equation (5.34), one can obtain the expression that needs to be maximised in the classical penalised likelihood formulation with AIC, MDL and BIC model selection criteria. Consequently, the penalised likelihood framework can be interpreted as a problem of maximising the joint posterior posterior distribution $p(k, \boldsymbol{\mu}_{1:k}|\mathbf{x}, \mathbf{y})$. Effectively, this MAP estimate can be obtained as follows:

$$\begin{aligned}(k, \boldsymbol{\mu}_{1:k})_{\text{MAP}} &= \underset{k,\mu_{1:k}\in\boldsymbol{\Omega}}{\arg\max}\; p(k, \boldsymbol{\mu}_{1:k}|\mathbf{x}, \mathbf{y}) \\ &= \underset{k,\mu_{1:k}\in\boldsymbol{\Omega}}{\arg\max} \left\{ \left[\prod_{i=1}^{c} (\mathbf{y}'_{1:N,i}\mathbf{P}^*_{i,k}\mathbf{y}_{1:N,i})^{-N/2} \right] \exp(-\mathcal{P}) \right\} \end{aligned} \tag{5.35}$$

The sufficient conditions that need to be satisfied so that the distribution $p(k, \boldsymbol{\mu}_{1:k}|\mathbf{x}, \mathbf{y})$ is proper are not overly restrictive. Firstly, $\boldsymbol{\Omega}$ has to be a compact set, which is not a problem in our setting. Secondly, $\mathbf{y}'_{1:N,i}\mathbf{P}^*_{i,k}\mathbf{y}_{1:N,i}$ has to be larger than zero for $i = 1, \ldots, c$. In Appendix E, Lemma 1, it is shown that this is the case unless $\mathbf{y}_{1:N,i}$ spans the space of the columns of $\mathbf{D}(\boldsymbol{\mu}_{1:k}, \mathbf{x})$, in which case $\mathbf{y}'_{1:N,i}\mathbf{P}^*_{i,k}\mathbf{y}_{1:N,i} = 0$. This event is rather unlikely to occur in our approximation framework, yet we can safeguard against it happening by choosing a very large value for $\overline{\boldsymbol{\delta}}$ in the simulations. This is a standard least squares trick known as ridge regression (Marquardt and Snee, 1975; Wetherill, 1986).

5.4.3 Reversible jump simulated annealing

From an MCMC perspective, we can solve the stochastic optimisation problem posed in the previous subsection by adopting a simulated annealing strategy (Geman and Geman, 1984; Van Laarhoven and Arts, 1987). The simulated annealing method involves simulating a non-homogeneous Markov chain whose invariant distribution at iteration i is no longer equal to $\pi(\mathbf{z})$, but to:

$$\pi_i(\mathbf{z}) \propto \pi^{1/T_i}(\mathbf{z})$$

where T_i is a decreasing cooling schedule with $\lim_{i\rightarrow+\infty} T_i = 0$. The reason for doing this is that, under weak regularity assumptions on $\pi(\mathbf{z})$, $\pi^{\infty}(\mathbf{z})$ is a probability density that concentrates itself on the set of global maxima of $\pi(\mathbf{z})$.

As with the MH method, the simulated annealing method with distribution $\pi(\mathbf{z})$ and proposal distribution $q(\mathbf{z}^\star|\,\mathbf{z})$ involves sampling a candidate value $\mathbf{z}^\star$ given the current value $\mathbf{z}$ according to $q(\mathbf{z}^\star|\,\mathbf{z})$. The Markov chain moves towards $\mathbf{z}^\star$ with probability $\mathcal{A}_{\text{SA}}(\mathbf{z}, \mathbf{z}^\star) = \min\left\{1, \left(\pi^{1/T_i}(\mathbf{z})\, q(\mathbf{z}^\star|\,\mathbf{z})\right)^{-1} \pi^{1/T_i}(\mathbf{z}^\star)\, q(\mathbf{z}|\,\mathbf{z}^\star)\right\}$, otherwise it remains equal to $\mathbf{z}$. If we choose the homogeneous transition kernel $\mathcal{K}(\mathbf{z}, \mathbf{z}^\star)$ of the reversible jump algorithm as the proposal distribution and use the reversibility property:

$$\pi(\mathbf{z}^\star)\mathcal{K}(\mathbf{z}^\star, \mathbf{z}) = \pi(\mathbf{z})\mathcal{K}(\mathbf{z}, \mathbf{z}^\star)$$

it follows that:

$$\mathcal{A}_{\text{RJSA}} = \min\left\{1, \frac{\pi^{(1/T_i - 1)}(\mathbf{z}^\star)}{\pi^{(1/T_i - 1)}(\mathbf{z})}\right\} \tag{5.36}$$

Consequently, the following algorithm, with $b_k = d_k = m_k = s_k = u_k = 0.2$, can find the joint MAP estimate of $\boldsymbol{\mu}_{1:k}$ and k:

Reversible Jump Simulated Annealing

```
1. Initialisation: set $\left(k^{(0)}, \theta^{(0)}\right) \in \boldsymbol{\Theta}$.
2. Iteration $i$.
   • Sample $u \sim \mathcal{U}_{[0,1]}$ and set the temperature with a cooling schedule.
   • If $(u \leq b_{k^{(i)}})$
       – then "birth" move (See Section 5.4.5).
       – else if $(u \leq b_{k^{(i)}} + d_{k^{(i)}})$ then "death" move (See Section 5.4.5).
       – else if $(u \leq b_{k^{(i)}} + d_{k^{(i)}} + s_{k^{(i)}})$ then "split" move (See Section 5.4.6).
       – else if $(u \leq b_{k^{(i)}} + d_{k^{(i)}} + s_{k^{(i)}} + m_{k^{(i)}})$ then "merge" move (See Section 5.4.6).
       – else update the RBF centres (See Section 5.4.4).
     End If.
   • Perform an MH step with the annealed acceptance ratio (equation (5.36)).
3. $i \leftarrow i + 1$ and go to 2.
4. Compute the coefficients $\boldsymbol{\alpha}_{1:m}$ by least squares (optimal in this case):
     $\widehat{\boldsymbol{\alpha}}_{1:m,i} = [\mathbf{D}'(\boldsymbol{\mu}_{1:k}, \mathbf{x})\mathbf{D}(\boldsymbol{\mu}_{1:k}, \mathbf{x})]^{-1}\mathbf{D}'(\boldsymbol{\mu}_{1:k}, \mathbf{x})\mathbf{y}_{1:N,i}$
```

■

The simulated annealing moves are explained in the following subsections.

5.4.4 Update move

The radial basis centres are sampled in the same way as explained in Section 5.3.1.1. However, the target distribution is given by:

$$p(\boldsymbol{\mu}_{j,1:d}|\mathbf{x},\mathbf{y},\boldsymbol{\mu}_{-j,1:d}) \propto \left[\prod_{i=1}^{c}(\mathbf{y}'_{1:N,i}\mathbf{P}^{*}_{i,k}\mathbf{y}_{1:N,i})^{(-\frac{N}{2})}\right]\exp(-\mathcal{P}) \tag{5.37}$$

and, consequently, the acceptance probability is:

$$\mathcal{A}_{\text{RJSA}}(\boldsymbol{\mu}_{j,1:d},\boldsymbol{\mu}^{\star}_{j,1:d}) = \min\left\{1,\left[\prod_{i=1}^{c}\left(\frac{\mathbf{y}'_{1:N,i}\mathbf{P}^{*}_{i,k}\mathbf{y}_{1:N,i}}{\mathbf{y}'_{1:N,i}\mathbf{P}^{\star}_{i,k}\mathbf{y}_{1:N,i}}\right)^{(\frac{N}{2})}\right]\right\} \tag{5.38}$$

where $\mathbf{P}^{\star}_{i,k}$ is similar to $\mathbf{P}^{*}_{i,k}$ with $\boldsymbol{\mu}_{1:k,1:d}$ replaced by $\{\boldsymbol{\mu}_{1,1:d},\boldsymbol{\mu}_{2,1:d},\cdots,\boldsymbol{\mu}_{j-1,1:d},\boldsymbol{\mu}^{\star}_{j,1:d},\boldsymbol{\mu}_{j+1,1:d},\cdots,\boldsymbol{\mu}_{k,1:d}\}$.

5.4.5 Birth and death moves

The birth and death moves are similar to the ones proposed in Section 5.3.2.1, except that the expressions for r_{birth} and r_{death} (with $b_k = d_k = 0.2$) become:

$$r_{birth} = \left[\prod_{i=1}^{c}\left(\frac{\mathbf{y}'_{1:N,i}\mathbf{P}^{*}_{i,k}\mathbf{y}_{1:N,i}}{\mathbf{y}'_{1:N,i}\mathbf{P}^{*}_{i,k+1}\mathbf{y}_{1:N,i}}\right)^{(\frac{N}{2})}\right]\frac{\Im\exp(-\mathcal{C})}{k+1} \tag{5.39}$$

Similarly,

$$r_{death} = \left[\prod_{i=1}^{c}\left(\frac{\mathbf{y}'_{1:N,i}\mathbf{P}^{*}_{i,k}\mathbf{y}_{1:N,i}}{\mathbf{y}'_{1:N,i}\mathbf{P}^{*}_{i,k-1}\mathbf{y}_{1:N,i}}\right)^{(\frac{N}{2})}\right]\frac{k\exp(\mathcal{C})}{\Im} \tag{5.40}$$

Hence, the acceptance probabilities corresponding to the described moves are:

$$\mathcal{A}_{birth} = \min\{1, r_{birth}\}, \mathcal{A}_{death} = \min\{1, r_{death}\} \tag{5.41}$$

5.4.6 Split and merge moves

Again the split and merge moves are analogous to the ones proposed in Section 5.3.2.2, except that the expressions for r_{split} and r_{merge} (with $m_k = s_k = 0.2$) become:

$$r_{split} = \left[\prod_{i=1}^{c}\left(\frac{\mathbf{y}'_{1:N,i}\mathbf{P}^{*}_{i,k}\mathbf{y}_{1:N,i}}{\mathbf{y}'_{1:N,i}\mathbf{P}^{*}_{i,k+1}\mathbf{y}_{1:N,i}}\right)^{(\frac{N}{2})}\right]\frac{k\varsigma^{\star}\exp(-\mathcal{C})}{k+1} \tag{5.42}$$

and

$$r_{merge} = \left[\prod_{i=1}^{c}\left(\frac{\mathbf{y}'_{1:N,i}\mathbf{P}^{*}_{i,k}\mathbf{y}_{1:N,i}}{\mathbf{y}'_{1:N,i}\mathbf{P}^{*}_{i,k-1}\mathbf{y}_{1:N,i}}\right)^{(\frac{N}{2})}\right]\frac{k\exp(\mathcal{C})}{\varsigma^{\star}(k-1)} \tag{5.43}$$

The acceptance probabilities for these moves are:

$$\mathcal{A}_{split} = \min\{1, r_{split}\}, \mathcal{A}_{merge} = \min\{1, r_{merge}\} \tag{5.44}$$

5.5 Convergence Results

It is easy to prove that the reversible jump MCMC algorithm applied to the full Bayesian model converges: in other words, that the Markov chain $\left(k^{(i)}, \boldsymbol{\mu}_{1:k}^{(i)}, \Lambda^{(i)}, \boldsymbol{\delta}^{2(i)}\right)_{i\in\mathbb{N}}$ is ergodic. Here, a stronger result is proved by showing that $\left(k^{(i)}, \boldsymbol{\mu}_{1:k}^{(i)}, \Lambda^{(i)}, \boldsymbol{\delta}^{2(i)}\right)_{i\in\mathbb{N}}$ converges to the required posterior distribution at a geometric rate.

For the homogeneous kernel, the following result holds:

Theorem 1 *Let* $\left(k^{(i)}, \boldsymbol{\mu}_{1:k}^{(i)}, \Lambda^{(i)}, \boldsymbol{\delta}^{2(i)}\right)_{i\in\mathbb{N}}$ *be the Markov chain whose transition kernel has been described in Section 5.3. This Markov chain converges to the probability distribution* $p\left(k, \boldsymbol{\mu}_{1:k}, \Lambda, \boldsymbol{\delta}^2 \middle| \mathbf{x}, \mathbf{y}\right)$. *Furthermore this convergence occurs at a geometric rate, that is, for almost every initial point* $\left(k^{(0)}, \boldsymbol{\mu}_{1:k}^{(0)}, \Lambda^{(0)}, \boldsymbol{\delta}^{2(0)}\right) \in \boldsymbol{\Omega} \times \boldsymbol{\Psi}$ *there exists a function of the initial states* $C\left(k^{(0)}, \boldsymbol{\mu}_{1:k}^{(0)}, \Lambda^{(0)}, \boldsymbol{\delta}^{2(0)}\right) > 0$ *and a constant* $\rho \in [0,1)$ *such that:*

$$\left\| p^{(i)}\left(k, \boldsymbol{\mu}_{1:k}, \Lambda, \boldsymbol{\delta}^2\right) - p\left(k, \boldsymbol{\mu}_{1:k}, \Lambda, \boldsymbol{\delta}^2 \middle| \mathbf{x}, \mathbf{y}\right) \right\|_{TV} \leq C\left(k^{(0)}, \boldsymbol{\mu}_{1:k}^{(0)}, \Lambda^{(0)}, \boldsymbol{\delta}^{2(0)}\right) \rho^{\lfloor i/k_{\max} \rfloor} \tag{5.45}$$

where $p^{(i)}\left(k, \boldsymbol{\mu}_{1:k}, \Lambda, \boldsymbol{\delta}^2\right)$ *is the distribution of* $\left(k^{(i)}, \boldsymbol{\mu}_{1:k}^{(i)}, \Lambda^{(i)}, \boldsymbol{\delta}^{2(i)}\right)$ *and* $\|\cdot\|_{TV}$ *is the total variation norm (Tierney, 1994).*

Proof. See Appendix B ■

Corollary 1 *Since at each iteration* i *one simulates the nuisance parameters* $(\boldsymbol{\alpha}_{1:m}, \boldsymbol{\sigma}_k^2)$, *then the distribution of the series* $(k^{(i)}, \boldsymbol{\alpha}_{1:m}^{(i)}, \boldsymbol{\mu}_{1:k}^{(i)}, \boldsymbol{\sigma}_k^{2(i)}, \Lambda^{(i)}, \boldsymbol{\delta}^{2(i)})_{i\in\mathbb{N}}$ *converges geometrically towards* $p(k, \boldsymbol{\alpha}_{1:m}, \boldsymbol{\mu}_{1:k}, \boldsymbol{\sigma}_k^2, \Lambda, \boldsymbol{\delta}^2|\mathbf{x}, \mathbf{y})$ *at the same rate* ρ.

In other words, the distribution of the Markov chain converges at least at a geometric rate, dependent on the initial state, to the required equilibrium distribution $p(k, \boldsymbol{\theta}, \psi|\mathbf{x}, \mathbf{y})$.

Remark 1 *In practice, one cannot evaluate* ρ *but Theorem 1 proves its existence. This type of convergence ensures that a central limit theorem for ergodic averages is valid (Meyn and Tweedie, 1993; Tierney, 1994). Moreover, in practice there is empirical evidence that the Markov chain converges quickly.*

The following convergence theorem for the reversible jump MCMC simulated annealing algorithm applies:

Theorem 2 *Under certain assumptions found in* (Andrieu et al., 1999b)*, the series of* $(\boldsymbol{\theta}^{(i)}, k^{(i)})$ *converges in probability to the set of global maxima* $(\boldsymbol{\theta}^{\max}, k^{\max})$*, that is for any* $\epsilon > 0$*, it follows that:*

$$\lim_{i\to\infty} P\left(\frac{p(\boldsymbol{\theta}^{(i)}, k^{(i)})}{p(\boldsymbol{\theta}^{\max}, k^{\max})} \geq 1 - \epsilon\right) = 1$$

Proof. If we follow the same steps as in Proposition 1 of Appendix E, with $\boldsymbol{\delta}^2$ and Λ fixed, it is easy to show that the transition kernels for each temperature are uniformly geometrically ergodic. Hence, as a corollary of (Andrieu et al., 1999b, Theorem 1), the convergence result for the simulated annealing MCMC algorithm follows ■

5.6 Experiments

When implementing the reversible jump MCMC algorithm, discussed in Section 5.4, one might encounter problems of ill-conditioning, in particular for high dimensional parameter spaces. There are two satisfactory ways of overcoming this problem. Firstly, we can introduce a ridge regression component so that the expression for $\mathbf{M}_{i,k}^{-1}$ in Section 5.2.3 becomes:

$$\mathbf{M}_{i,k}^{-1} = \mathbf{D}'(\boldsymbol{\mu}_{1:k}, \mathbf{x})\mathbf{D}(\boldsymbol{\mu}_{1:k}, \mathbf{x}) + \boldsymbol{\Sigma}_i^{-1} + \hbar\mathbf{I}_m$$

where $\hbar$ is a small number. Alternatively, we can introduce a slight modification of the prior for $\boldsymbol{\alpha}_{1:m}$:

$$p(\boldsymbol{\alpha}_{1:m}|k, \boldsymbol{\mu}_{1:k}, \boldsymbol{\sigma}^2, \Lambda, \boldsymbol{\delta}^2) = \left[\prod_{i=1}^{c} |2\pi\boldsymbol{\sigma}_i^2\boldsymbol{\delta}_i^2\mathbf{I}_m|^{-1/2} \exp\left(-\frac{1}{2\boldsymbol{\sigma}_i^2\boldsymbol{\delta}_i^2}\boldsymbol{\alpha}'_{1:m,i}\boldsymbol{\alpha}_{1:m,i}\right)\right] \tag{5.46}$$

In doing so, the marginal posterior distribution becomes:

$$\begin{aligned} p(k, \boldsymbol{\mu}_{1:k}, \Lambda, \boldsymbol{\delta}^2|\mathbf{x}, \mathbf{y}) \propto & \left[\prod_{i=1}^{c} (\boldsymbol{\delta}_i^2)^{-m/2} |\mathbf{M}_{i,k}|^{1/2} \left(\frac{\gamma_0 + \mathbf{y}'_{1:N,i}\mathbf{P}_{i,k}\mathbf{y}_{1:N,i}}{2}\right)^{(-\frac{N+v_0}{2})}\right] \\ & \times \left[\frac{\mathbb{I}_{\boldsymbol{\Omega}}(k, \boldsymbol{\mu}_k)}{\Im^k}\right]\left[\frac{\Lambda^k/k!}{\sum_{j=0}^{k_{\max}} \Lambda^j/j!}\right]\left[\prod_{i=1}^{c} (\boldsymbol{\delta}_i^2)^{-(\alpha_{\delta^2}+1)} \exp\left(-\frac{\beta_{\boldsymbol{\delta}^2}}{\boldsymbol{\delta}_i^2}\right)\right] \\ & \times \left[(\Lambda)^{(\varepsilon_1 - 1/2)} \exp\left(-\varepsilon_2\Lambda\right)\right] \end{aligned} \tag{5.47}$$

where:

$$\begin{aligned} \mathbf{M}_{i,k}^{-1} &= \mathbf{D}'(\boldsymbol{\mu}_{1:k}, \mathbf{x})\mathbf{D}(\boldsymbol{\mu}_{1:k}, \mathbf{x}) + \boldsymbol{\delta}_i^{-2}\mathbf{I}_m \\ \mathbf{h}_{i,k} &= \mathbf{M}_{i,k}\mathbf{D}'(\boldsymbol{\mu}_{1:k}, \mathbf{x})\mathbf{y}_{1:N,i} \\ \mathbf{P}_{i,k} &= \mathbf{I}_N - \mathbf{D}(\boldsymbol{\mu}_{1:k}, \mathbf{x})\mathbf{M}_{i,k}\mathbf{D}'(\boldsymbol{\mu}_{1:k}, \mathbf{x}) \end{aligned}$$

It has been found that although both strategies can deal with the problem of limited numerical precision, the second approach seems to be more stable. In addition, the second approach does not oblige us to select a value for the simulation parameter $\hbar$. The results presented henceforth were obtained using this approach.

5.6.1 Signal detection example

The problem of detecting signal components in noisy signals has occupied the minds of many researchers for a long time (Djurić, 1996; Fisher, 1929; Hannan, 1961). Here, the rather simple toy problem of detecting Gaussian components in a noisy signal is considered. The aim is to compare the performance of the hierarchical Bayesian model selection scheme and the penalised likelihood model selection criteria (AIC, MDL) when the amount of noise in the signal varies.

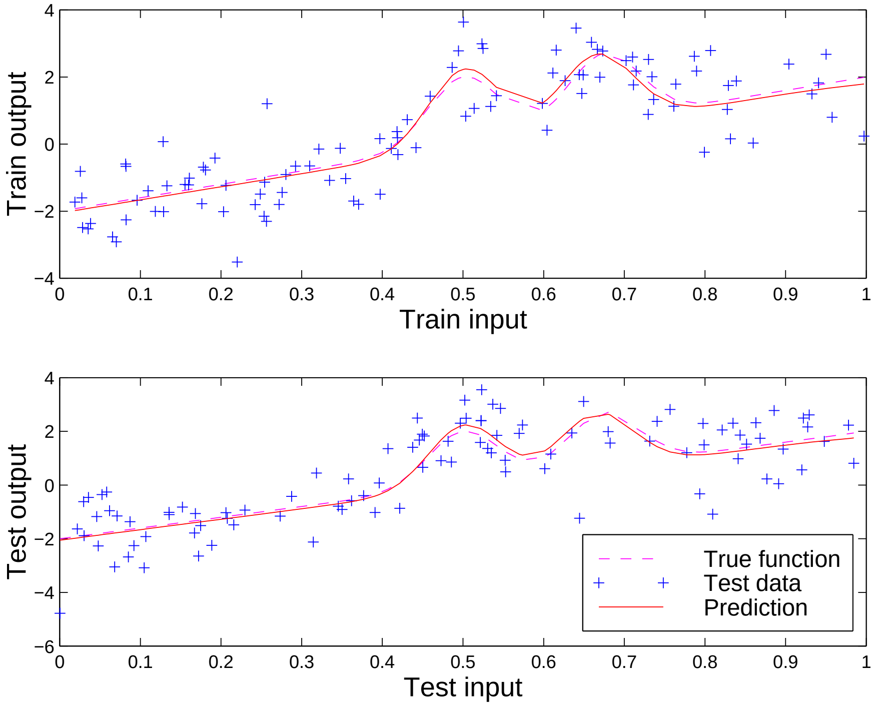

Figure 5.3 *Performance of the reversible jump MCMC algorithm on the signal detection problem. Despite the large noise variance, the estimates of the true function and noise process are very accurate, thereby leading to good generalisation (no over-fitting).*

The data was generated from the following univariate function using 50 covariate points uniformly on $[-2, 2]$:

$$y = x + 2\exp(-16x^2) + 2\exp(-16(x-0.7)^2 + n$$

where $n \sim \mathcal{N}(0, \sigma^2)$. The data was then rescaled to make the input data lie in the interval $[0, 1]$. The full Bayesian and simulated annealing algorithms were used to estimate the number of components in the signal for different levels of noise. The experiment was repeated 100 times for each noise level. Gaussian radial basis functions with the same variance as the Gaussian signal components were chosen. For the simulated annealing method, a linear cooling schedule was adopted: $T_i = a - bi$, where

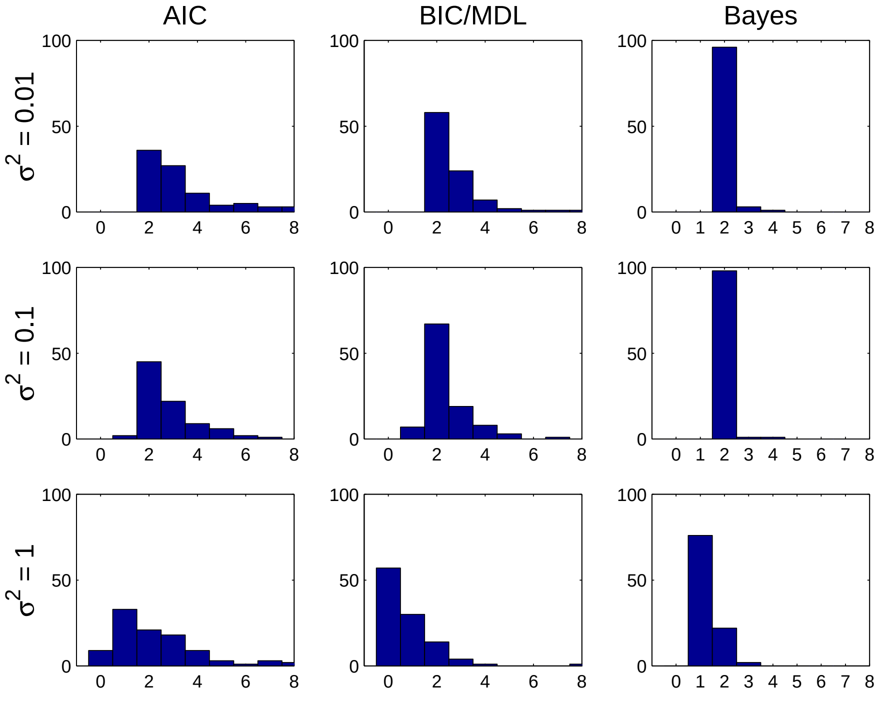

Figure 5.4 *Histograms of the main mode $\widehat{p}(k|\mathbf{x},\mathbf{y})$ for 100 trials of each noise level. The Bayes solution provides a better estimate of the true number of bases (2) than the MDL/BIC and AIC criteria.*

$a, b \in \mathbb{R}^+$ and $T_i > 0$ for $i = 1, 2, 3, \ldots$. In particular, the initial and final temperatures were set to 1 and 1×10^{-5}. For the Bayesian model, diffuse priors ($\alpha_{\delta^2} = 2, \beta_{\delta^2} = 10$ (see experiments in Chapter 7), $\upsilon_0 = 0, \gamma_0 = 0, \varepsilon_1 = 0.001$ and $\varepsilon_2 = 0.0001$) were selected. Finally, the simulation parameters $k_{\max}$, ι, σ^2_{RW} and $\varsigma^\star$ were set to 20, 0.1, 0.001 and 0.1.

Figure 5.3 shows the typical fits that were obtained for training and validation data sets. By varying the variance of the noise σ^2, the main mode and fractions of unexplained variance were estimated. For the AIC and BIC/MDL criteria, the main mode corresponds to the one for which the posterior is maximised, while for the Bayesian approach, the main mode corresponds to the MAP of the model order probabilities $\widehat{p}(k|\mathbf{x},\mathbf{y})$, computed as suggested in Section 5.2.2.

The fractions of unexplained variance (fv) were computed as follows:

$$\text{fv} = \frac{1}{100} \sum_{i=1}^{100} \frac{\sum_{t=1}^{50} (y_{t,i} - \widehat{y}_{t,i})^2}{\sum_{t=1}^{50} (y_{t,i} - \overline{y}_i)^2}$$

where $\widehat{y}_{t,i}$ denotes the t-th prediction for the i-th trial and $\overline{y}_i$ is the estimated mean of y_i. The normalisation in the fv error measure makes it independent of the size of the data set. If the estimated mean was to be used as the predictor of the data, the fv would be equal to 1. The results obtained are shown in Figure 5.4 and Table 5.1. The fv's for each model selection approach are very similar. This result is expected since the problem under consideration is rather simple and the error variations could possibly be attributed to the fact that only 100 realisations of the noise process for each σ^2 are used. What is important is that, even in this scenario, it is clear that the full Bayesian model provides more accurate estimates of the model order.

σ^2	AIC	BIC/MDL	Bayes
0.01	0.0070	0.0076	0.0069
0.1	0.0690	0.0732	0.0657
1	0.6083	0.4846	0.5105

Table 5.1 *Fraction of unexplained variance for different values of the noise variance, averaged over* 100 *test sets.*

5.7 Summary

This chapter presented a general methodology for estimating, jointly, the noise variance, parameters and number of parameters of an RBF model. In adopting a Bayesian model and a reversible jump MCMC algorithm to perform the necessary integrations, it was demonstrated that the method is very accurate and generalises well. Chapter 8 will show the robustness of the algorithm to the specification of the prior's parameters. The chapter also considered the problem of stochastic optimisation for model order selection and proposed a solution that makes use of a reversible jump simulated annealing algorithm and classical information criteria. Moreover, it presented theorems of geometric convergence for the reversible jump algorithm with the full Bayesian model and convergence for the simulated annealing algorithm.

6

Sequential Monte Carlo Methods

The Taylor series approximations in the EKF algorithm can lead to poor representations of the probability distributions of interest. With nonlinear models these distributions tend to be multi-modal. Gaussian approximation in such cases could miss the rich structure brought in by the nonlinear model. In addition, if the noise model is not Gaussian the EKF can lead to erroneous results.

Sequential Monte Carlo (SMC) methods provide a route for overcoming these problems. Moreover, they allow for a complete representation of the posterior distribution, so any statistical estimates, such as the mean, modes, kurtosis and variance, can be easily computed. Tracking in computer vision (Isard and Blake, 1996) is a good example of an application area where the Kalman filter approximation is shown to fail to capture the multi modalities, while SMC methods perform well. SMC algorithms, under the names of particle filters, sequential sampling-importance resampling (SIR), bootstrap filters and condensation trackers have also been applied to a wide range of problems, including target tracking (Gordon et al., 1993), financial analysis (Liu and West, 2000; Müller, 1992; Pitt and Shephard, 1999), diagnostic measures of fit (Pitt and Shephard, 1999), sequential imputation in missing data problems (Kong et al., 1994), blind deconvolution (Liu and Chen, 1995), non-stationary independent component analysis (Everson and Roberts, 1999), aircraft navigation (Bergman, 1999), communications and audio engineering (Clapp and Godsill, 1999), change point detection (Fearnhead, 1998), population biology (Bolviken and Storvic, 2000), in-situ ellipsometry (Marrs, 2000), electricity load forecasting (Djurić, 2000), control (Stavropoulos, 1998) and medical prognosis (Berzuini et al., 1997).

This chapter applies SMC techniques in a state space setting to show how a multilayer perceptron may be trained in environments where data arrives one at a time. It places a degree of emphasis on reviewing the existing state-of-the-art SMC tools as the field is new to the neural networks and machine learning communities. It considers sequential importance sampling and resampling algorithms and points out some of their limitations. Subsequently, it presents more sophisticated algorithms and illustrates their performance on some simple regression and classification problems. The problem of model selection with SMC methods will be covered in the next chapter.

Section 6.1 formulates the problem of training neural networks in terms of a state space representation. A Monte Carlo solution is subsequently proposed in Section 6.2. In Sections 6.3 and 6.4, a generic sequential importance sampling methodology is derived and some of its limitations are mentioned. To circumvent these, selection methods are introduced in Section 6.5. This leads to the presentation of the SIR algorithm and some improvements in Sections 6.6 and 6.7. Section 6.8 discusses an advanced SMC algorithm for neural networks. Section 6.9 introduces the HySIR algorithm. Sections 6.10 and 6.11 delve briefly into the topics of smoothing and convergence. Finally, Some experiments are described in Section 6.12.

6.1 State Space Neural Network Model

As before, it is convenient to start from a state space representation to model the neural network's evolution in time:

$$\begin{aligned}\boldsymbol{\theta}_{t+1} &= \mathbf{h}(\boldsymbol{\theta}_t, \mathbf{u}_t) \\ \mathbf{y}_t &= \widehat{\mathbf{f}}_t(\mathbf{x}_t, \boldsymbol{\theta}_t, \mathbf{v}_t)\end{aligned}$$

where $\widehat{\mathbf{f}}_t(\mathbf{x}_t, \boldsymbol{\theta}_t, \mathbf{v}_t)$ denotes an MLP with weights $\boldsymbol{\theta}_t$. In the SMC framework, the probability distributions of the noise terms are specified by the user. The process noise is modelled as an additive Gaussian process $\mathbf{u}_t \sim \mathcal{N}(\mathbf{0}, Q)$, so that $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \mathbf{u}_t$. For regression tasks, the measurement noise may be modelled as an additive Gaussian process $\mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, R)$ and $\mathbf{y}_t = \widehat{\mathbf{f}}_t(\mathbf{x}_t, \boldsymbol{\theta}_t) + \mathbf{v}_t$. Note that this framework also allows us to model the noise in the input $\mathbf{x}_t$. When concerned with binary classification, the measurement's distribution is modelled as a Bernoulli process with likelihood $p(\mathbf{y}_t|\mathbf{x}_t, \boldsymbol{\theta}_t) = \widehat{\mathbf{f}}_t(\mathbf{x}_t, \boldsymbol{\theta}_t)^{\mathbf{y}_t}\big(1 - \widehat{\mathbf{f}}_t(\mathbf{x}_t, \boldsymbol{\theta}_t)\big)^{1-\mathbf{y}_t}$, where $\widehat{\mathbf{f}}_t(\mathbf{x}_t, \boldsymbol{\theta}_t)$ is the probability of class membership obtained with a logistic output neuron. The SMC framework is very flexible in that it allows for any other noise distributions. For instance, one could adopt heavy-tailed measurement noise distributions to deal with outliers and impulsive noise. This is done in Chapter 8.

The noise terms are assumed to be uncorrelated with the network weights and the initial conditions. The evolution of the states (network weights) corresponds to a Markov process with initial distribution $p(\boldsymbol{\theta}_0)$ and transition prior $p(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1})$. The observations are assumed to be conditionally independent given the states. These are standard assumptions in a large class of tracking and time series problems (Harvey, 1989).

Our goal will be to approximate the posterior distribution $p(\boldsymbol{\theta}_{0:t}|\mathbf{d}_{1:t})$ and one of its marginals, the filtering density $p(\boldsymbol{\theta}_t|\mathbf{d}_{1:t})$, where $\mathbf{d}_{1:t} = \{\mathbf{x}_{1:t}, \mathbf{y}_{1:t}\}$. By computing the filtering density recursively, we do not need to keep track of the complete history of the parameters. Towards the end, this chapter considers ways of approximating $p(\boldsymbol{\theta}_{0:t}, R, Q|\mathbf{d}_{1:t})$. Since the distribution of the input data is not being modelled, $\mathbf{x}_{1:t}$ will be dropped from the arguments of the probability distributions to simplify the notation.

6.2 Monte Carlo Simulation

Many signal processing problems, such as equalisation of communication signals, financial time series prediction, medical prognosis, target tracking and geophysical data analysis, involve elements of non-Gaussianity, nonlinearity and non-stationarity. It is, therefore, not often possible to derive exact closed form estimators based upon the standard criteria of maximum likelihood, maximum *a posteriori* or minimum mean-squared error. Analytical approximations to the true distribution of the data do not take into account all the salient statistical features of the processes under consideration. Monte Carlo simulation methods, on the other hand, provide a complete description of the probability distribution of the data, thus leading to improvements in the accuracy of the analysis.

In Monte Carlo simulation, a set of weighted particles (samples), drawn from the posterior distribution of the model parameters, is used to map integrals to discrete sums. More precisely, the posterior can be approximated by the following empirical estimate:

$$\widehat{p}(\boldsymbol{\theta}_{0:t}|\mathbf{y}_{0:t}) = \frac{1}{N}\sum_{i=1}^{N} \delta_{\boldsymbol{\theta}_{0:t}^{(i)}}(\mathrm{d}\boldsymbol{\theta}_{0:t})$$

where the random samples $\{\boldsymbol{\theta}_{0:t}^{(i)}; i = 1, \ldots, N\}$, are drawn from the posterior distribution and $\delta(\mathrm{d}\cdot)$ denotes the Dirac delta function. Consequently, any expectations of the form:

$$\mathbb{E}\big(g_t(\boldsymbol{\theta}_{0:t})\big) = \int g_t(\boldsymbol{\theta}_{0:t}) p(\boldsymbol{\theta}_{0:t}|\mathbf{y}_{1:t}) \mathrm{d}\boldsymbol{\theta}_{0:t}$$

may be approximated by the following estimate:

$$\overline{\mathbb{E}\big(g_t(\boldsymbol{\theta}_{0:t})\big)} = \frac{1}{N}\sum_{i=1}^{N} g_t(\boldsymbol{\theta}_{0:t}^{(i)})$$

where the particles $\boldsymbol{\theta}_{0:t}^{(i)}$ are assumed to be independent and identically distributed (i.i.d.) for the approximation to hold. Indeed, according to the law of large numbers, we have $\overline{\mathbb{E}\big(g_t(\boldsymbol{\theta}_{0:t})\big)} \xrightarrow[N\rightarrow\infty]{a.s.} \mathbb{E}\big(g_t(\boldsymbol{\theta}_{0:t})\big)$, where $\xrightarrow[N\rightarrow\infty]{a.s.}$ denotes almost sure convergence. Moreover, if the posterior variance of $g_t(\boldsymbol{\theta}_{0:t})$ is bounded, that is $var_{p(\cdot|\mathbf{d}_{1:t})}\big(g_t(\boldsymbol{\theta}_{0:t})\big) < \infty$, then the following central limit theorem holds:

$$\sqrt{N}\left(\overline{\mathbb{E}\big(g_t(\boldsymbol{\theta}_{0:t})\big)} - \mathbb{E}\big(g_t(\boldsymbol{\theta}_{0:t})\big)\right) \underset{N\rightarrow\infty}{\Longrightarrow} \mathcal{N}\left(0, var_{p(\cdot|\mathbf{d}_{1:t})}\big(g_t(\boldsymbol{\theta}_{0:t})\big)\right)$$

where $\underset{N\rightarrow\infty}{\Longrightarrow}$ denotes convergence in distribution.

In recent years, many researchers in the statistical and signal processing communities have, almost simultaneously, proposed several variations of sequential Monte Carlo algorithms. These algorithms have been applied to a wide range of problems, as mentioned in the introduction to this chapter. However, basic sequential Monte Carlo methods had already been introduced in the automatic control field in the late sixties. For instance, Handschin and Mayne (Handschin and Mayne, 1969) tackled the problem

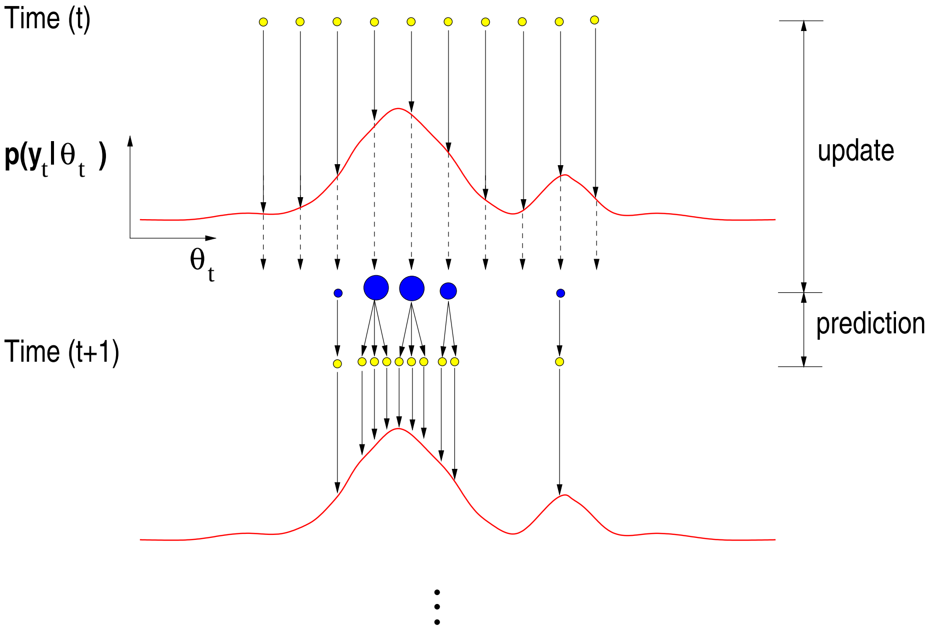

Figure 6.1 *Update and prediction stages in a generic sequential Monte Carlo algorithm with the transition prior as proposal. In the update stage, the likelihood of each particle is evaluated. The size of the circles indicates the likelihood of a particular particle. The particles are then selected according to their respective likelihoods. In this process, the particles with higher likelihood are allowed to have more "children". Subsequently, the algorithm computes the predicted values of the particles by evaluating the transition equations. The end result is that the surviving particles provide a better weighted description of the likelihood function.*

of nonlinear filtering with a sequential importance sampling approach. They combined analytical and numerical techniques, using the Control Variate Method (Hammersley and Handscomb, 1968), to reduce the variance of the estimates. In the seventies, various researchers continued working on these ideas (Akashi and Kumamoto, 1977; Handschin, 1970; Zaritskii et al., 1975). The paper of Zaritskii, Svetnik and Shimelevich (Zaritskii et al., 1975) is particularly interesting. They treated the problems of continuous and discrete time filtering and introduced several novel ideas. With the exception of Doucet (Doucet, 1997), most authors have overlooked the Monte Carlo sampling ideas proposed in the seventies. Figure 6.1 illustrates the operation of a generic SMC method. Only the fittest particles, that is the ones with the highest likelihood, are selected in the update stage. These then proceed to be multiplied, according to their likelihood, in the prediction stage.

It is instructive to approach the problem from an optimisation perspective. Figures 6.2 and 6.3 show the windowed global descent in the error function that is typically observed. The diagrams shed light on the roles played by the noise covariances R and Q. Q dictates by how much the cloud of samples is expanded in the prediction stage. By increasing Q, the density of the cloud of samples is reduced. Consequently, the algorithm will take longer to converge. A very small Q, on the other hand, will not allow the algorithm to explore new regions of the parameter space. Ideally, one needs to implement an algorithm that adapts Q automatically. This is explored in Sections 6.7 to 6.9. R controls the resolution of the update stage, as shown in Figure 6.4. A small value of R will cause the likelihood to be too narrow. Consequently, only a few trajectories will be able to propagate to the next time step. If R is too small, the optimisation scheme might fail to detect some of the important modes of the likelihood function. By increasing R, we broaden the likelihood function, thereby increasing the number of surviving trajectories. If we choose R to be too large, all the trajectories become equally likely and the algorithm might not converge. As shown in Figures 6.2 and 6.3, R gives rise to a threshold $T(R)$ in the error function, which determines the width of the updated clouds of samples.

The following sections will introduce SMC methods, more formally, from a Bayesian sampling-importance resampling perspective.

6.3 Bayesian Importance Sampling

As mentioned in the previous section, one can approximate the posterior distribution with a function on a finite discrete support. Consequently, it follows from the strong law of large numbers that as the number or samples N increases, expectations can be mapped into sums. Unfortunately, it is often impossible to sample directly from

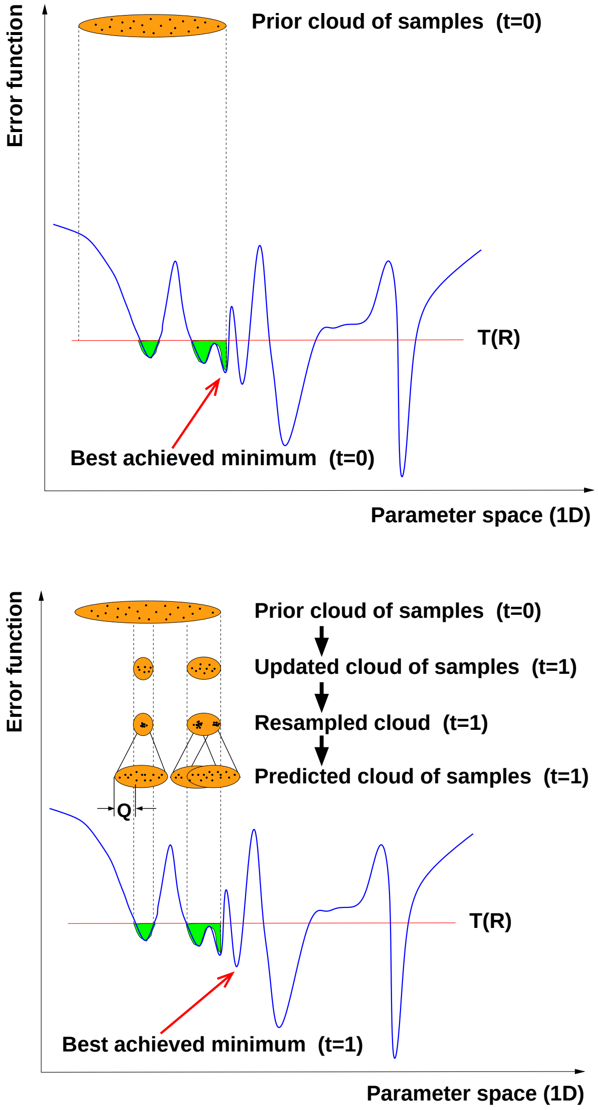
Error function
Prior cloud of samples (t=0)
T(R)
Best achieved minimum (t=0)
Parameter space (1D)
Error function
Prior cloud of samples (t=0)
Updated cloud of samples (t=1)
Resampled cloud (t=1)
Predicted cloud of samples (t=1)
Q
T(R)
Best achieved minimum (t=1)
Parameter space (1D)

Figure 6.2 *First and second steps of a one-dimensional sequential Monte Carlo optimisation problem. The size of the initial cloud of samples determines the search region in parameter space. The goal is to find the best possible minimum of the error function. It is clear that as the number of samples increases, our chances of reaching lower minima increase. In the second step, the updated clouds of samples are generated. The width of these clouds is obtained from the intersection between the width of the prior cloud of samples, the error function and a threshold determined by the measurements noise covariance R. The updated clouds of samples are denser than the prior cloud of samples. Next, the samples are grouped in regions of higher likelihood in a resampling step. Finally, the clouds of samples are expanded by a factor determined by the process noise covariance Q. This expansion allows the clouds to reach regions of the parameter function where the error function is lower.*

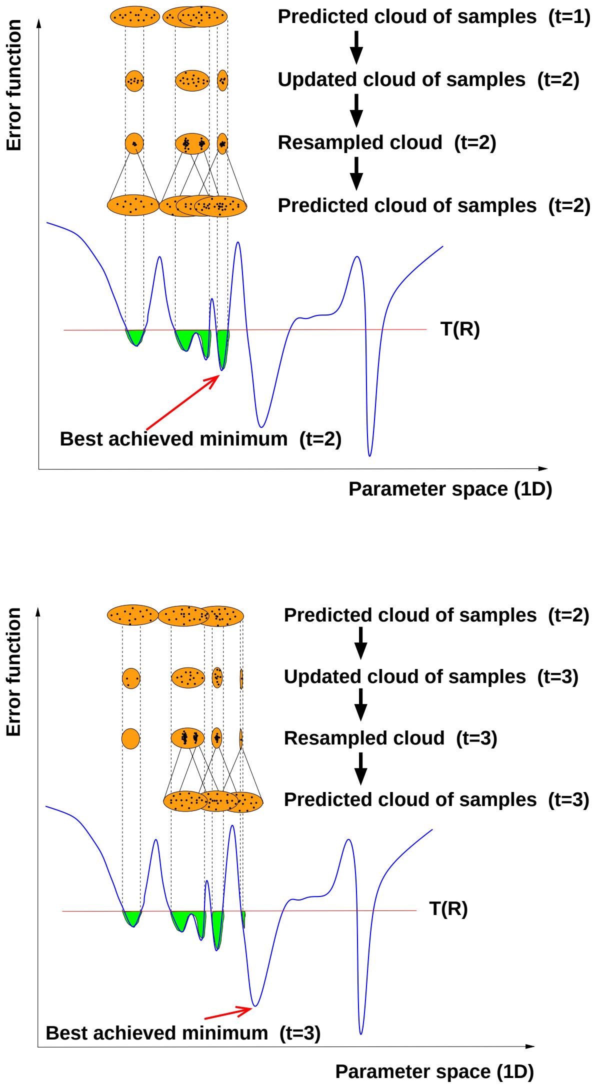
Error function
Predicted cloud of samples (t=1)
Updated cloud of samples (t=2)
Resampled cloud (t=2)
Predicted cloud of samples (t=2)
T(R)
Best achieved minimum (t=2)
Parameter space (1D)
Error function
Predicted cloud of samples (t=2)
Updated cloud of samples (t=3)
Resampled cloud (t=3)
Predicted cloud of samples (t=3)
T(R)
Best achieved minimum (t=3)
Parameter space (1D)

Figure 6.3 *Third and fourth step of a one-dimensional sequential Monte Carlo optimisation problem. To reach the global minimum on the right, the number of samples has to be increased.*

the posterior density function. However, we can circumvent this difficulty by sampling from a known, easy-to-sample, proposal distribution $q(\boldsymbol{\theta}_{0:t}|\mathbf{y}_{1:t})$ and making use of the following substitution:

$$\begin{aligned}
\mathbb{E}\big(g_t(\boldsymbol{\theta}_{0:t})\big) &= \int g_t(\boldsymbol{\theta}_{0:t})\frac{p(\boldsymbol{\theta}_{0:t}|\mathbf{y}_{1:t})}{q(\boldsymbol{\theta}_{0:t}|\mathbf{y}_{1:t})}q(\boldsymbol{\theta}_{0:t}|\mathbf{y}_{1:t})\mathrm{d}\boldsymbol{\theta}_{0:t} \\
&= \int g_t(\boldsymbol{\theta}_{0:t})\frac{p(\mathbf{y}_{1:t}|\boldsymbol{\theta}_{0:t})p(\boldsymbol{\theta}_{0:t})}{p(\mathbf{y}_{1:t})q(\boldsymbol{\theta}_{0:t}|\mathbf{y}_{1:t})}q(\boldsymbol{\theta}_{0:t}|\mathbf{y}_{1:t})\mathrm{d}\boldsymbol{\theta}_{0:t} \\
&= \int g_t(\boldsymbol{\theta}_{0:t})\frac{w_t(\boldsymbol{\theta}_{0:t})}{p(\mathbf{y}_{1:t})}q(\boldsymbol{\theta}_{0:t}|\mathbf{y}_{1:t})\mathrm{d}\boldsymbol{\theta}_{0:t}
\end{aligned}$$

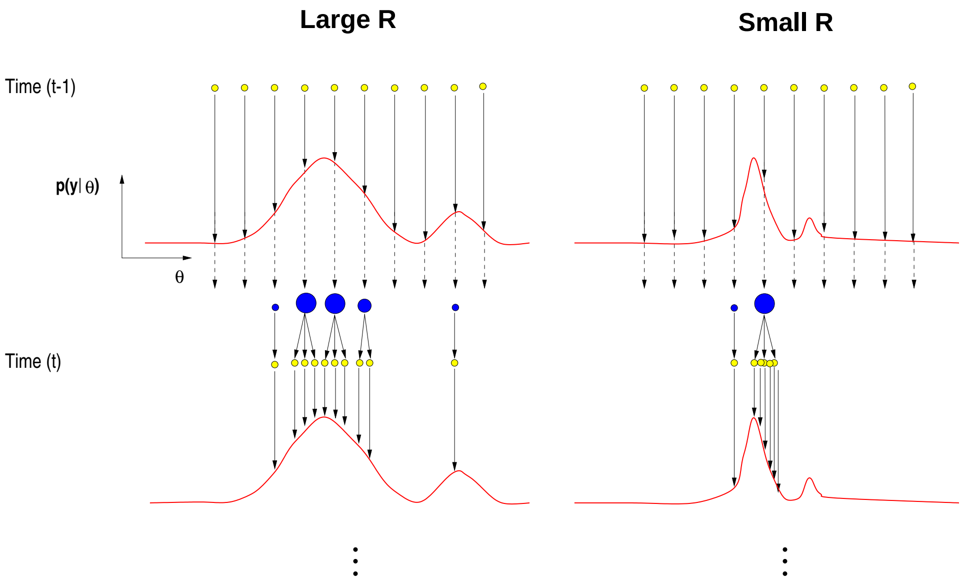

Figure 6.4 *Role played by the measurements variance R. Small values of R result in a narrow likelihood, with only a few trajectories being able to propagate to the next time step. If R is too small, we might miss some of the important modes. On the other hand, if R is too large, we are not giving preference to any of the trajectories and the algorithm might not converge.*

where the variables $w_t(\boldsymbol{\theta}_{0:t})$ are known as the unnormalised importance weights:

$$w_t = \frac{p(\mathbf{y}_{1:t}|\boldsymbol{\theta}_{0:t})p(\boldsymbol{\theta}_{0:t})}{q(\boldsymbol{\theta}_{0:t}|\mathbf{y}_{1:t})} \tag{6.1}$$

We can get rid of the unknown normalising density $p(\mathbf{y}_{1:t})$ as follows:

$$\begin{aligned}
\mathbb{E}\big(g_t(\boldsymbol{\theta}_{0:t})\big) &= \frac{1}{p(\mathbf{y}_{1:t})}\int g_t(\boldsymbol{\theta}_{0:t})w_t(\boldsymbol{\theta}_{0:t})q(\boldsymbol{\theta}_{0:t}|\mathbf{y}_{1:t})\mathrm{d}\boldsymbol{\theta}_{0:t} \\
&= \frac{\int g_t(\boldsymbol{\theta}_{0:t})w_t(\boldsymbol{\theta}_{0:t})q(\boldsymbol{\theta}_{0:t}|\mathbf{y}_{1:t})\mathrm{d}\boldsymbol{\theta}_{0:t}}{\int p(\mathbf{y}_{1:t}|\boldsymbol{\theta}_{0:t})p(\boldsymbol{\theta}_{0:t})\frac{q(\boldsymbol{\theta}_{0:t}|\mathbf{y}_{1:t})}{q(\boldsymbol{\theta}_{0:t}|\mathbf{y}_{1:t})}\mathrm{d}\boldsymbol{\theta}_{0:t}} \\
&= \frac{\int g_t(\boldsymbol{\theta}_{0:t})w_t(\boldsymbol{\theta}_{0:t})q(\boldsymbol{\theta}_{0:t}|\mathbf{y}_{1:t})\mathrm{d}\boldsymbol{\theta}_{0:t}}{\int w_t(\boldsymbol{\theta}_{0:t})q(\boldsymbol{\theta}_{0:t}|\mathbf{y}_{1:t})\mathrm{d}\boldsymbol{\theta}_{0:t}} \\
&= \frac{\mathbb{E}_q\left(w_t(\boldsymbol{\theta}_{0:t})g_t(\boldsymbol{\theta}_{0:t})\right)}{\mathbb{E}_q\left(w_t(\boldsymbol{\theta}_{0:t})\right)}
\end{aligned}$$

where the notation $\mathbb{E}_q$ has been used to emphasise that the expectations are taken over the proposal distribution $q(\cdot|\mathbf{y}_{1:t})$. Hence, by drawing samples from the proposal function $q(\cdot|\mathbf{y}_{1:t})$, we can approximate the expectations of interest by the following estimate:

$$\begin{aligned}\overline{\mathbb{E}\big(g_t(\boldsymbol{\theta}_{0:t})\big)} &= \frac{1/N\sum_{i=1}^{N} g_t(\boldsymbol{\theta}_{0:t}^{(i)})w_t(\boldsymbol{\theta}_{0:t}^{(i)})}{1/N\sum_{i=1}^{N} w_t(\boldsymbol{\theta}_{0:t}^{(i)})} \\ &= \sum_{i=1}^{N} g_t(\boldsymbol{\theta}_{0:t}^{(i)})\widetilde{w}_t(\boldsymbol{\theta}_{0:t}^{(i)}) \qquad (6.2)\end{aligned}$$

where the normalised importance weights $\widetilde{w}_t^{(i)}$ are given by:

$$\widetilde{w}_t^{(i)} = \frac{w_t^{(i)}}{\sum_{j=1}^{N} w_t^{(j)}}$$

The estimate of equation (6.2) is biased as it involves a ratio of estimates. However, it is possible to obtain asymptotic convergence and a central limit theorem for $\overline{\mathbb{E}\big(g_t(\boldsymbol{\theta}_{0:t})\big)}$ under the following assumptions (Doucet, 1998; Geweke, 1989):

1. $\boldsymbol{\theta}_{0:t}^{(i)}$ corresponds to a set of i.i.d. samples drawn from the proposal distribution, the support of the proposal distribution includes the support of the posterior distribution and $\mathbb{E}\big(g_t(\boldsymbol{\theta}_{0:t})\big)$ exists and is finite.

2. The expectations of w_t and $w_t g_t^2(\boldsymbol{\theta}_{0:t})$ over the posterior distribution exist and are finite.

A sufficient condition to verify the second assumption is to have bounds on the variance of $g_t(\boldsymbol{\theta}_{0:t})$ and on the importance weights. Thus, as N tends to infinity, the posterior density function can be approximated arbitrarily well by the point-mass estimate:

$$\widehat{p}(\boldsymbol{\theta}_{0:t}|\mathbf{y}_{1:t}) = \sum_{i=1}^{N} \widetilde{w}_t^{(i)} \delta_{\boldsymbol{\theta}_{0:t}^{(i)}}(\mathrm{d}\boldsymbol{\theta}_{0:t})$$

6.4 Sequential Importance Sampling

The proposal distribution can be expanded as follows:

$$q(\boldsymbol{\theta}_{0:t}|\mathbf{y}_{1:t}) = q(\boldsymbol{\theta}_0|\mathbf{y}_{1:t}) \prod_{j=1}^{t} q(\boldsymbol{\theta}_j|\boldsymbol{\theta}_{1:j-1}, \mathbf{y}_{1:t})$$

However, in order to compute a sequential estimate of the posterior distribution at time t without modifying the previously simulated states $\boldsymbol{\theta}_{0:t-1}$, the following proposal distribution may be adopted:

$$\begin{aligned}q(\boldsymbol{\theta}_{0:t}|\mathbf{y}_{1:t}) &= q(\boldsymbol{\theta}_0) \prod_{j=1}^{t} q(\boldsymbol{\theta}_j|\boldsymbol{\theta}_{1:j-1}, \mathbf{y}_{1:j}) \\ &= q(\boldsymbol{\theta}_{0:t-1}|\mathbf{y}_{1:t-1})q(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{0:t-1}, \mathbf{y}_{1:t}) \qquad (6.3)\end{aligned}$$

At this stage, we need to recall that we assumed that the states correspond to a Markov process and that the observations are conditionally independent given the states. That is:

$$p(\boldsymbol{\theta}_{0:t}) = p(\boldsymbol{\theta}_0)\prod_{j=1}^{t} p(\boldsymbol{\theta}_j|\boldsymbol{\theta}_{j-1}) \qquad \text{and} \qquad p(\mathbf{y}_{1:t}|\boldsymbol{\theta}_{0:t}) = \prod_{j=1}^{t} p(\mathbf{y}_j|\boldsymbol{\theta}_j) \tag{6.4}$$

By substituting equations (6.3) and (6.4) into equation (6.1), a recursive estimate for the importance weights can be derived as follows:

$$\begin{aligned} w_t &= \frac{p(\mathbf{y}_{1:t}|\boldsymbol{\theta}_{0:t})p(\boldsymbol{\theta}_{0:t})}{q(\boldsymbol{\theta}_{0:t-1}|\mathbf{y}_{1:t-1})q(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{0:t-1},\mathbf{y}_{1:t})} \\ &= w_{t-1}\frac{p(\mathbf{y}_{1:t}|\boldsymbol{\theta}_{0:t})p(\boldsymbol{\theta}_{0:t})}{p(\mathbf{y}_{1:t-1}|\boldsymbol{\theta}_{0:t-1})p(\boldsymbol{\theta}_{0:t-1})}\frac{1}{q(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{0:t-1},\mathbf{y}_{1:t})} \\ &= w_{t-1}\frac{p(\mathbf{y}_t|\boldsymbol{\theta}_t)p(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1})}{q(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{0:t-1},\mathbf{y}_{1:t})} \end{aligned} \tag{6.5}$$

Equation (6.5) provides a mechanism to sequentially update the importance weights. Since we can sample from the proposal function and evaluate the likelihood and transition probabilities, all we need to do is generate a prior set of samples and iteratively compute the importance weights. This procedure, known as sequential importance sampling (SIS), allows us to obtain the type of estimates described by equation (6.2).

6.4.1 Choice of proposal distribution

The choice of proposal function is one of the most critical design issues in importance sampling algorithms. The preference for proposal functions that minimise the variance of the importance weights is advocated by (Doucet, 1997). The following result has been proved:

Proposition 1 [Proposition 3 of (Doucet et al., 1999)] *The importance distribution $q(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{0:t-1},\mathbf{y}_{1:t}) = p(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{0:t-1},\mathbf{y}_{1:t})$ minimises the variance of the importance ratios conditional on $\boldsymbol{\theta}_{0:t-1}$ and $\mathbf{y}_{1:t}$.*

This choice of proposal function has also been advocated by other researchers (Kong et al., 1994; Liu and Chen, 1995; Zaritskii et al., 1975). Nonetheless, the distribution:

$$q(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{0:t-1},\mathbf{y}_{1:t}) = p(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1}) \tag{6.6}$$

is the most popular choice of proposal function (Avitzour, 1995; Beadle and Djurić, 1997; Gordon et al., 1993; Isard and Blake, 1996; Kitagawa, 1996). Although it results in higher Monte Carlo variation than the optimal proposal $p(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{0:t-1},\mathbf{y}_{1:t})$, as a result of it not incorporating the most recent observations, it is usually easier to implement

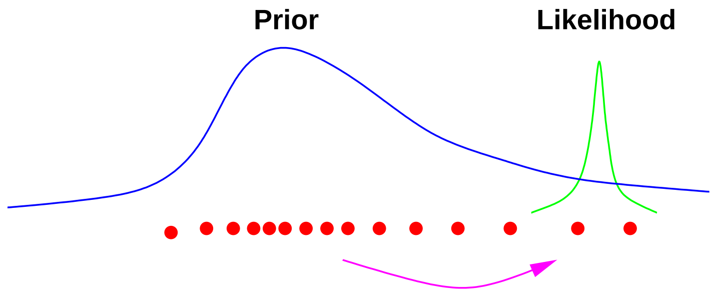

Figure 6.5 *The optimal importance distribution allows us to move the samples in the prior to regions of high likelihood. This is of paramount importance if the likelihood happens to lie in one of the tails of the prior distribution.*

(Berzuini et al., 1997; Doucet, 1998; Liu and Chen, 1998). As illustrated in Figure 6.5, if we fail to use the latest available information to propose new values for the states, only a few particles survive. It is therefore of paramount importance to move the particles towards the regions of high likelihood. Section 6.7 describes several algorithms to implement the optimal importance function.

6.4.2 Degeneracy of the SIS algorithm

The SIS algorithm discussed so far has a serious limitation: the variance of the importance ratios ($\overline{w}_t \triangleq p(\boldsymbol{\theta}_{0:t}|\mathbf{y}_{1:t})/q(\boldsymbol{\theta}_{0:t}|\mathbf{y}_{1:t})$) increases stochastically over time.

Proposition 2 [Page 285 of (Kong et al., 1994), proposition 4 of (Doucet et al., 1999)] *The unconditional variance (that is, when the observations are regarded as random) of the importance ratios increases over time.*

To understand why the variance increase poses a problem, suppose that we want to sample from the posterior. In that case, we want the proposal density to be very close to the posterior density. When this happens, we obtain the following results for the mean and variance:

$$\mathbb{E}_q\left(\frac{p(\boldsymbol{\theta}_{0:t}|\mathbf{y}_{1:t})}{q(\boldsymbol{\theta}_{0:t}|\mathbf{y}_{1:t})}\right) = 1$$

and

$$var_q\left(\frac{p(\boldsymbol{\theta}_{0:t}|\mathbf{y}_{1:t})}{q(\boldsymbol{\theta}_{0:t}|\mathbf{y}_{1:t})}\right) = \mathbb{E}_q\left(\left(\frac{p(\boldsymbol{\theta}_{0:t}|\mathbf{y}_{1:t})}{q(\boldsymbol{\theta}_{0:t}|\mathbf{y}_{1:t})} - 1\right)^2\right) = 0$$

In other words, we expect the variance to be close to zero in order to obtain reasonable estimates. Therefore, a variance increase has a harmful effect on the accuracy of the simulations. In practice, the degeneracy caused by the variance increase can be observed by monitoring the importance ratios. Typically, what we observe is that, after a few iterations, one of the normalised importance ratios tends to 1, while the remaining ratios tend to zero. The next section presents a strategy to reduce this depletion of samples.

6.5 Selection

To avoid the degeneracy of the SIS simulation method, a selection (resampling) stage may be used to eliminate samples with low importance ratios and multiply samples with high importance ratios. It is possible to see an analogy to the steps in genetic algorithms (Higuchi, 1997).

A selection scheme associates to each particle $\boldsymbol{\theta}_{0:t}^{(i)}$ a number of "children", say $N_i \in \mathbb{N}$, such that $\sum_{i=1}^{N} N_i = N$. Several selection schemes have been proposed in the literature. These schemes satisfy $\mathbb{E}\big(N_i\big) = N\widetilde{w}_t^{(i)}$ but their performance varies in terms of the variance of the particles $var\big(N_i\big)$. Recent theoretical results in (Crisan et al., 1999) indicate that the restriction $\mathbb{E}\big(N_i\big) = N\widetilde{w}_t^{(i)}$ is unnecessary to obtain convergence results (Doucet et al., 1999). So it is possible to design biased but very quick selection schemes.

6.5.1 Sampling-importance resampling and multinomial sampling

Many of the ideas on resampling have stemmed from the work of Efron (Efron, 1982), Rubin (Rubin, 1988) and Smith and Gelfand (Smith and Gelfand, 1992). Resampling involves mapping the Dirac random measure $\{\boldsymbol{\theta}_{0:t}^{(i)}, \widetilde{w}_t^{(i)}\}$ into an equally weighted random measure $\{\boldsymbol{\theta}_{0:t}^{(j)}, N^{-1}\}$. This can be accomplished by sampling uniformly from the discrete set $\{\boldsymbol{\theta}_{0:t}^{(i)}; i = 1, \ldots, N\}$ with probabilities $\{\widetilde{w}_t^{(i)}; i = 1, \ldots, N\}$ as proposed in the seminal paper of Gordon, Salmond and Smith (1993). A mathematical proof of this can be found on pages 111–112 of (Gordon, 1994). Figure 6.6 shows a way of sampling from this discrete set. After constructing the cumulative distribution of the discrete set, a uniformly drawn sampling index i is projected onto the distribution range and then onto the distribution domain. The intersection with the domain constitutes the new sample index j. That is, the vector $\boldsymbol{\theta}_{0:t}^{(j)}$ is accepted as the new sample. Clearly, the vectors with the larger sampling ratios will end up with more copies after the resampling process.

Sampling N times from the cumulative discrete distribution $\sum_{i=1}^{N} \widetilde{w}_t^{(i)} \delta_{\boldsymbol{\theta}_{0:t}^{(i)}}(\mathrm{d}\boldsymbol{\theta}_{0:t})$ is equivalent to drawing $(N_i; i = 1, \ldots, N)$ from a multinomial distribution with parameters N and $\widetilde{w}_t^{(i)}$. This procedure can be implemented in $\mathcal{O}\left(N\right)$ operations (Doucet, 1998; Pitt and Shephard, 1999) following the work of (Ripley, 1987, pp. 96). As we are sampling from a multinomial distribution, the variance is $var(N_i) = N\widetilde{w}_t^{(i)}\big(1 - \widetilde{w}_t^{(i)}\big)$.

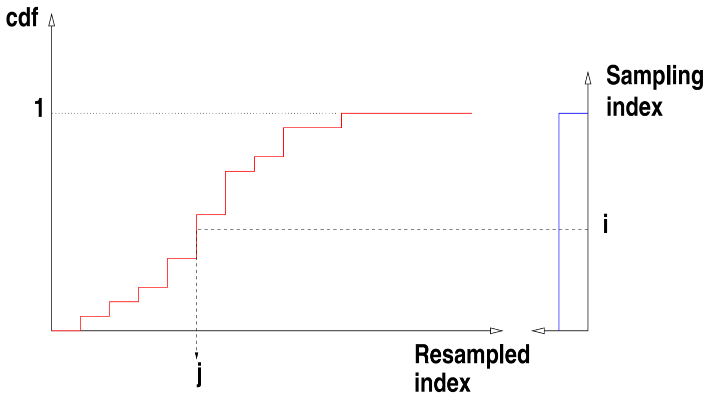

Figure 6.6 *Resampling process, whereby a random measure* $\{\boldsymbol{\theta}_{1:t}^{(i)}, \widetilde{w}_t^{(i)}\}$ *is mapped into an equally weighted random measure* $\{\boldsymbol{\theta}_{1:t}^{(j)}, N^{-1}\}$*. The index* i *is drawn from a uniform distribution.*

As pointed out in (Carpenter et al., 1999) and (Liu and Chen, 1998), it is possible to design selection schemes with lower variance.

6.5.2 Residual resampling

This procedure was introduced by (Liu and Chen, 1998). It involves the following steps. Firstly, set $\widetilde{N}_i = \left\lfloor N\widetilde{w}_t^{(i)} \right\rfloor$. Secondly, perform an SIR procedure to select the remaining $\overline{N}_t = N - \sum_{i=1}^{N} \widetilde{N}_i$ samples with new weights $w_t'^{(i)} = \overline{N}_t^{-1}\left(\widetilde{w}_t^{(i)}N - \widetilde{N}_i\right)$. Finally, add the results to the current $\widetilde{N}_i$. For this scheme, the variance $\left(var(N_i) = \overline{N}_t w_t'^{(i)}\left(1 - w_t'^{(i)}\right)\right)$ is smaller than the one given by the SIR scheme. Moreover, this procedure is computationally cheaper.

6.5.3 Systematic sampling

This method, proposed in (Carpenter et al., 1999), is explained in great detail in (Fearnhead, 1998, Chapter 5). It introduces a variance on N_i even smaller than the residual resampling scheme, namely $var(N_i) = \overline{N}_t w_t'^{(i)}\left(1 - \overline{N}_t w_t'^{(i)}\right)$. Its computational complexity is $\mathcal{O}(N)$.

6.5.4 When to resample

It has been argued in (Liu and Chen, 1995; Liu and Chen, 1998) that when all the importance weights are nearly equal, resampling only reduces the number of distinctive streams and introduces extra variation in the simulations. Although resampling increases the variance of the estimate of $\mathbb{E}\left(g_t(\boldsymbol{\theta}_{0:t})\right)$ at time t, it usually leads to a decrease in the variance of future estimates (Doucet et al., 1999; Liu and Chen, 1995).

Using a result from (Kong et al., 1994), Liu and Chen suggest that one should resample only when the effective sample size N_{eff} is below a fixed heuristic threshold, where the effective sample size is defined as:

$$N_{eff} = \frac{1}{\sum_{i=1}^{N}\left(\widetilde{w}_t^{(i)}\right)^2}$$

This result is intuitively reasonable. One should resample only when the empirical variance of the normalised importance weights increases excessively. However, it is not entirely satisfactory as it depends on a fixed threshold.

6.6 The SIR Algorithm

It has been explained, so far, how to compute the importance weights sequentially and how to improve the sample set by resampling. A complete and detailed SIR algorithm for MLPs can now be presented:

SIR Algorithm for MLPs

1. At $t = 0$

   - For $i = 1, \ldots, N$, draw the weights $\boldsymbol{\theta}_0^{(i)}$ from the first layer prior $p_1(\boldsymbol{\theta}_0)$ and the second layer prior $p_2(\boldsymbol{\theta}_0)$.

2. For $t = 1, 2, \ldots$

   (a) Bayesian importance sampling step

   - For $i = 1, \ldots, N$, sample:

$$\widehat{\boldsymbol{\theta}}_t^{(i)} \sim q(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{0:t-1}^{(i)}, \mathbf{y}_{1:t}) \tag{6.7}$$

   and set:

$$\widehat{\boldsymbol{\theta}}_{0:t}^{(i)} \triangleq \left(\boldsymbol{\theta}_{0:t-1}^{(i)}, \widehat{\boldsymbol{\theta}}_t^{(i)}\right)$$

   - For $i = 1, \ldots, N$, evaluate the importance weights up to a normalising constant:

$$w_t^{(i)} = w_{t-1}^{(i)} \frac{p(\mathbf{y}_t|\widehat{\boldsymbol{\theta}}_t^{(i)})p(\widehat{\boldsymbol{\theta}}_t^{(i)}|\boldsymbol{\theta}_{t-1}^{(i)})}{q(\widehat{\boldsymbol{\theta}}_t^{(i)}|\boldsymbol{\theta}_{0:t-1}^{(i)}, \mathbf{y}_{1:t})} \tag{6.8}$$

```
        • For $i = 1, \ldots, N$, normalise the importance weights:
              $\widetilde{w}_t^{(i)} = w_t^{(i)} \left[ \sum_{j=1}^{N} w_t^{(j)} \right]^{-1}$
    (b) If $N_{eff} \geq$ Threshold:
        • For $i = 1, \ldots, N$, set $\boldsymbol{\theta}_{0:t}^{(i)} = \widehat{\boldsymbol{\theta}}_{0:t}^{(i)}$
        Else
        • Selection step
            – Multiply/Suppress samples $\widehat{\boldsymbol{\theta}}_{0:t}^{(i)}$ with high/low importance weights $\widetilde{w}_t^{(i)}$,
              respectively, to obtain $N$ random samples $\boldsymbol{\theta}_{0:t}^{(i)}$ approximately distributed
              according to $p(\boldsymbol{\theta}_{0:t}^{(i)}|\mathbf{y}_{1:t})$.
            – For $i = 1, \ldots, N$, set $w_t^{(i)} = \widetilde{w}_t^{(i)} = \frac{1}{N}$
                                                                                        ■
```

The generic SIR algorithm is rather straightforward to implement. If we choose to sample from the transition prior $p(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{0:t-1})$, then:

$$\widehat{\boldsymbol{\theta}}_{t+1}^{(i)} = \boldsymbol{\theta}_t^{(i)} + \mathbf{u}_t^{(i)}$$

where $\mathbf{u}_t^{(i)} \sim \mathcal{N}(\mathbf{0}, Q)$ in our case. For this choice of proposal, the importance weights become:

$$w_t^{(i)} = w_{t-1}^{(i)} p(\mathbf{y}_t|\widehat{\boldsymbol{\theta}}_t^{(i)})$$

For simple regression tasks, we can choose a Gaussian likelihood:

$$p(\mathbf{y}_t|\boldsymbol{\theta}_t) \propto \exp\left(-\frac{1}{2} \big(\mathbf{y}_t - \widehat{\mathbf{f}}_t(\mathbf{x}_t, \boldsymbol{\theta}_t)\big)' R^{-1} \big(\mathbf{y}_t - \widehat{\mathbf{f}}_t(\mathbf{x}_t, \boldsymbol{\theta}_t)\big) \right)$$

If it is suspected that the data set may contain several outliers, likelihood functions with heavy tails should be employed. For binary classification tasks, with output logistic functions, we should choose Bernoulli likelihoods:

$$p(\mathbf{y}_t|\boldsymbol{\theta}_t) = \widehat{\mathbf{f}}_t(\mathbf{x}_t, \boldsymbol{\theta}_t)^{\mathbf{y}_t} \big(1 - \widehat{\mathbf{f}}_t(\mathbf{x}_t, \boldsymbol{\theta}_t)\big)^{1-\mathbf{y}_t}$$

Note that in this case there is no observation noise term because the targets correspond to a discrete set of noiseless class labels. For multiple classes, one can adopt multinomial distributions.

A different initial prior is used for each layer because the functional form of the neurons varies with layer (Müller and Rios Insua, 1998). In addition, we can select the prior's hyper-parameters for each layer in accordance with the magnitude of the input and output signals.

Section 6.8 will propose a more complex algorithm that uses an approximation of the optimal importance distribution and allows us to estimate the noise covariance R. It also incorporates an MCMC step to reduce the variance introduced by the selection step.

6.7 Improvements on SIR Simulation

The success of the SIR algorithm depends on the validity of the following underlying assumptions:

- The Dirac point-mass approximation provides an adequate representation of the posterior distribution.
- It is possible to obtain samples from the posterior by sampling from a suitable proposal distribution and applying importance sampling corrections.

If any of these conditions are not met, the SIR algorithm will inexorably fail. The discreteness of the approximation poses a resolution problem. In the resampling stage, any particular sample with a high importance weight will be duplicated many times. As a result, the cloud of samples may eventually collapse to a single sample. This degeneracy will limit the ability of the algorithm to search for lower minima in other regions of the error surface. In other words, the number of samples used to describe the posterior density function will become too small and inadequate. A brute force strategy to overcome this problem is to increase the number of particles. More refined approaches to surmount this problem include roughening, kernel smoothing and MCMC move steps. These are discussed subsequently.

The importance sampling approximation depends on how close the proposal distribution is to the posterior distribution. As illustrated in Figure 6.5, if the likelihood is too peaked or if there is little overlap between the prior and the likelihood, one needs to move the samples to regions of high likelihood. This is the basis for local Monte Carlo methods discussed in Section 6.7.2. Recently, Neal (1998) proposed a powerful annealing strategy to design importance distributions. However, it is too computationally demanding for it to be of practical use in the sequential context.

6.7.1 Roughening and prior boosting

In (Gordon et al., 1993), Gordon, Salmond and Smith propose a roughening procedure, whereby an independent jitter, with zero mean and standard deviation σ, is added to each sample drawn in the resampling stage. The standard deviation of the jitter is given by:

$$\sigma = K_s \mathcal{I} N^{-1/m} \tag{6.9}$$

where $\mathcal{I}$ denotes the length of the interval between the maximum and minimum samples of each specific component, K_s is a tuning parameter and, as before, m represents the number of weights. Large values of K_s blur the distribution, while very small values produce tight clusters of points around the original samples. The simulations in this study indicate that this technique can work very well and that tuning K_s requires little effort. Note that this strategy, is nothing but a simple Kernel smoothing technique.

Another strategy proposed by Gordon to solve the discreteness problem, is to increase the number of particles in the prediction stage by sampling M times from the proposal distribution, with $M > N$ (Gordon, 1994). Subsequently, in the resampling stage, only N particles are drawn. A drawback of this approach is that the variance of the boosted filter is greater than the variance of a simple SIR filter with M particles (Fearnhead, 1998). Nonetheless, boosting can lead to a small computational gain.

6.7.2 Local Monte Carlo methods

Local Monte Carlo methods allow us to mitigate the importance sampling problem. They allow us to move samples from the prior to regions of high likelihood. Of course, if we do this, we need to take cautious steps to deal with outliers. For example, one could introduce heavy tailed data models or use some form of smoothing. The following subsections, summarily, run through some techniques proposed in the literature to deal with the importance sampling problem.

6.7.2.1 Prior editing

Prior editing (Gordon et al., 1993) is an *ad-hoc* acceptance test for proposing particles in regions of high likelihood. After the prediction step, the residual error $\mathbf{e}_t = \mathbf{y}_t - \widehat{\mathbf{f}}_t(\mathbf{x}_t, \widehat{\boldsymbol{\theta}}_t^{(i)})$ is computed. If $|\mathbf{e}_t| > K_l\sqrt{r}$, where r is the scale of the measurement error model and K_l is a constant chosen to indicate the region of non-negligible likelihood, then the sample $\widehat{\boldsymbol{\theta}}_t^{(i)}$ is rejected. The procedure is repeated until a specified number of particles is accepted. The problem with this approach is that it is too heuristic and can be computationally intensive unless the rejection rate is small. In addition, it introduces a bias on the distribution of the particles.

6.7.2.2 Local linearisation

This is a popular method for devising proposal distributions that approximate the optimal importance distribution: see (Doucet, 1998; Pitt and Shephard, 1999) for example. It relies on Taylor series expansions of the likelihood and transition prior. For example, for a first order expansion of the measurements equation, one could use the EKF to propose from the importance distribution $q(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{0:t-1}^{(i)}, \mathbf{y}_{1:t})$; $i = 1, \ldots, N$. That is, at time $t-1$ one uses the EKF equations, with the new data, to compute the mean $\overline{\widehat{\boldsymbol{\theta}}}_t^{(i)}$ and covariance $\widehat{P}^{(i)}$ of the importance distribution for each particle. Next, we sample the i-th particle from this distribution. The method requires that we propagate the covariance $\widehat{P}^{(i)}$ and specify the EKF noise covariances.

Since the EKF is a minimum variance estimator, this local linearisation method leads to an improved annealed sampling algorithm, whereby the variance of each sampling trajectory changes with time. Ideally, we start searching over a large region of the error surface and as time progresses, we concentrate on the regions of lower error. Section 6.8 presents a more detailed description of the algorithm.

6.7.2.3 Rejection methods

If the likelihood is bounded, say $p(\mathbf{y}_t|\boldsymbol{\theta}_t) < M_t$, it is possible to sample from the optimal importance distribution $p(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1}, \mathbf{y}_t)$ using an accept/reject procedure. Firstly, we obtain a sample from the prior $\widehat{\boldsymbol{\theta}} \sim p(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1})$ and a uniform variable $u \sim \mathcal{U}_{[0,1]}$. Subsequently, the sample from the prior is accepted if $u \leq p(\mathbf{y}_t|\widehat{\boldsymbol{\theta}}_t)/M_t$. Otherwise, we reject the proposed sample and repeat the process until N samples are accepted. Unfortunately, the rejection sampler requires a random number of iterations at each time step. This proves to be computationally expensive in high-dimensional spaces (Doucet, 1998; Müller, 1991; Pitt and Shephard, 1999).

6.7.2.4 MCMC methods

The fundamental idea behind MCMC methods is to construct a Markov chain that allows us to sample from the proposal $p(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{0:t-1}, \mathbf{y}_{1:t})$ and eventually from the required posterior distribution $p(\boldsymbol{\theta}_{0:t}|\mathbf{y}_{1:t})$ (Berzuini et al., 1997; Gordon and Whitby, 1995; Müller, 1992). Generally, it can take a large number of iterations before this happens. Moreover, it is difficult to even assess when the chain has converged. For most real-time sequential processing applications, this MCMC strategy can be too computationally demanding.

6.7.2.5 Auxiliary particle filters

The auxiliary particle filter allows us to obtain approximate samples from the optimal importance distribution by introducing an auxiliary variable k. Specifically, the aim of the algorithm is to draw samples from the joint distribution:

$$q(\boldsymbol{\theta}_t, k|\boldsymbol{\theta}_{0:t-1}, \mathbf{y}_{1:t}) \propto p(\mathbf{y}_t|\mu_t^{(k)})p(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1}^{(k)})p(\boldsymbol{\theta}_{1:t-1}^{(k)}|\mathbf{y}_{1:t-1})$$

where $\mu_t^{(k)}, k = 1, \ldots, N$ is the mean, mode, draw, or some other value associated with the transition prior. One way to accomplish this objective is to evaluate the marginal

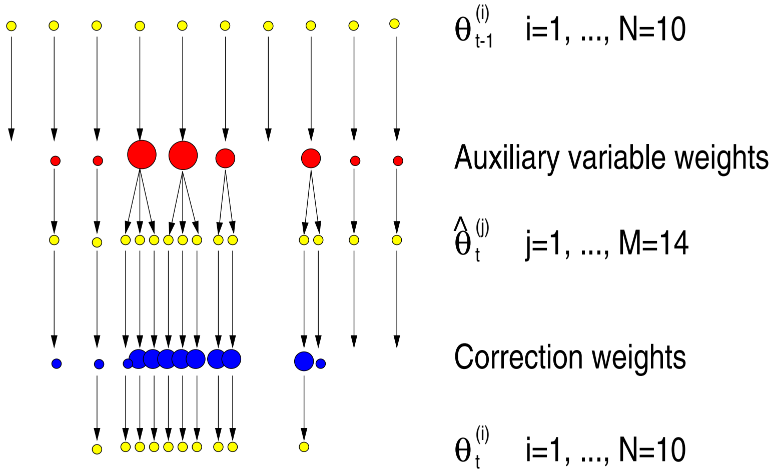
$\theta_{t-1}^{(i)}$ i=1, ..., N=10
Auxiliary variable weights
$\hat{\theta}_{t}^{(j)}$ j=1, ..., M=14
Correction weights
$\theta_{t}^{(i)}$ i=1, ..., N=10

Figure 6.7 *Auxiliary particle filter. The sizes of the weights are proportional to the likelihood of the related particle.*

auxiliary variable weights $g(k|\boldsymbol{\theta}_{0:t-1}, \mathbf{y}_{1:t}) \propto p(\mathbf{y}_t|\mu_t^{(k)})p(\boldsymbol{\theta}_{1:t-1}^{(k)}|\mathbf{y}_{1:t-1})$ and use them to select M particles from the transition prior. Typically, one boosts the sample set so that $M > N$. This resampling stage is illustrated in Figure 6.7. The particle filter then proceeds to evaluate the correction weights:

$$w_t = \frac{p(\mathbf{y}_t|\boldsymbol{\theta}_t^{(j)})}{p(\mathbf{y}_t|\mu_t^{(k_j)})}$$

where $j = 1, \ldots, M$ and k_j denotes the k-th "parent" of particle j. Finally, the correction weights are used to perform a second selection step to obtain N particles approximately distributed according to the posterior distribution.

In comparison to the SIR filter, the auxiliary particle filter can generate better estimates of the posterior whenever the likelihood is situated in one of the prior's tails. On the other hand, if the likelihood and prior coincide, the SIR filter can produce more accurate estimates. The latter behaviour is a consequence of the extra variance introduced by the additional selection step.

One alternative way of viewing the auxiliary particle filter is to interpret the distribution $q(\boldsymbol{\theta}_t, k|\boldsymbol{\theta}_{0:t-1}, \mathbf{y}_{1:t})$ as the importance proposal. In doing so, the following importance weights are obtained:

$$\begin{aligned} w_t & \propto \frac{p(\boldsymbol{\theta}_{0:t}^{(k)}|\mathbf{y}_{1:t})}{p(\mathbf{y}_t|\mu_t^{(k)})p(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1}^{(k)})p(\boldsymbol{\theta}_{1:t-1}^{(k)}|\mathbf{y}_{1:t-1})} \\ & \propto \frac{p(\mathbf{y}_t|\boldsymbol{\theta}_t^{(k)})p(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1}^{(k)})p(\boldsymbol{\theta}_{1:t-1}^{(k)}|\mathbf{y}_{1:t-1})}{p(\mathbf{y}_t|\mu_t^{(k)})p(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1}^{(k)})p(\boldsymbol{\theta}_{1:t-1}^{(k)}|\mathbf{y}_{1:t-1})} \\ & = \frac{p(\mathbf{y}_t|\boldsymbol{\theta}_t^{(k)})}{p(\mathbf{y}_t|\mu_t^{(k)})} \end{aligned}$$

That is, aside from boosting, auxiliary particle filters are very similar to the local Monte Carlo methods discussed so far.

6.7.3 Kernel density estimation

The idea of using kernel methods to produce smooth estimates of the posterior density has been espoused by many researchers in the field (Gordon, 1994; Hürzeler and Künsch, 1998; Liu and West, 2000; Musso et al., 2000). Kernel methods allow us to replace the Dirac point-mass approximation:

$$\widehat{p}(\boldsymbol{\theta}_{0:t}|\mathbf{y}_{1:t}) = \sum_{i=1}^{N} \widetilde{w}_t^{(i)} \delta_{\boldsymbol{\theta}_{0:t}^{(i)}}(\mathrm{d}\boldsymbol{\theta}_{0:t})$$

by the following estimate:

$$\widehat{p}(\boldsymbol{\theta}_{0:t}|\mathbf{y}_{1:t}) = \sum_{i=1}^{N} \widetilde{w}_t^{(i)} \frac{1}{\delta} K\left(\frac{\boldsymbol{\theta}_{0:t} - \widehat{\boldsymbol{\theta}}_{0:t}^{(i)}}{\delta}\right)$$

where $K(\cdot)$ is usually assumed to be a unimodal symmetric density function, centred at $\widehat{\boldsymbol{\theta}}_{0:t}^{(i)}$, with smoothing parameter δ. Figure 6.8 illustrates the type of approximations obtained using Dirac and kernel approximations.

Kernel methods can mitigate the problem of sample depletion in the resampling step. This is accomplished by sampling with replacement from the discrete set $\{\widehat{\boldsymbol{\theta}}_{0:t}^{(i)}, \widetilde{w}_t^{(i)}\}$ to obtain $\{\widehat{\boldsymbol{\theta}}_{0:t}^{(j)}, N^{-1}\}$ and, thereafter, adding a smoothing term. That is:

$$\boldsymbol{\theta}_{0:t}^{(j)} = \widehat{\boldsymbol{\theta}}_{0:t}^{(j)} + \varepsilon_j$$

where $p(\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_N) = \prod_{j=1}^{N} \delta^{-1} K(\delta^{-1}\varepsilon_j)$ The critical issue in the kernel smoothing approach is to estimate the smoothing parameter δ. This estimation is typically carried out by minimising the mean integrated square error (Silverman, 1986). Unfortunately, in high dimensions ($m \geq 4$), it is virtually impossible to perform kernel density estimation. Nevertheless, heuristic methods, such as discount factors, can be used to obtain rough estimates of δ (Liu and West, 2000).

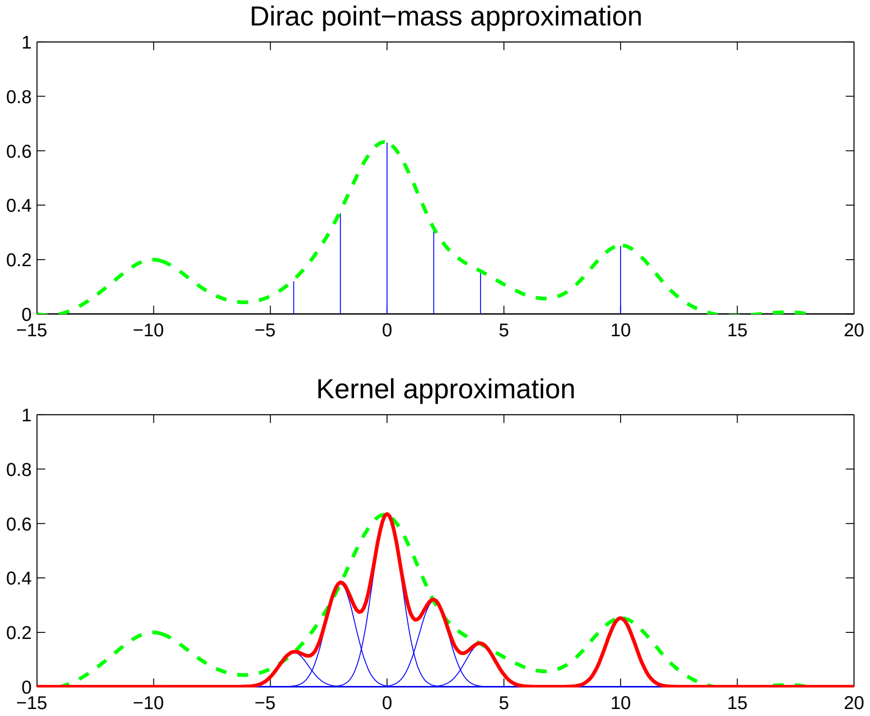

Figure 6.8 *Dirac and kernel smoothing approximations to the posterior distribution.*

Kernel methods, in conjunction with accept/reject procedures, have also been applied to draw approximate samples from the optimal importance distribution: see Algorithm 4.3 in (Hürzeler, 1998). This idea has been extended in (Musso et al., 2000), where particular emphasis is placed on reducing the computational cost of the algorithm.

Kernel methods, although in a new guise, have clear affinities with the alternative methods described in this section. For instance, choosing the variance of the linear approximations to the importance distribution is to a certain extent equivalent to choosing the smoothing parameter δ.

6.7.4 MCMC move step

After the selection scheme at time t, we obtain N particles distributed marginally approximately according to $p(\boldsymbol{\theta}_{0:t}|\mathbf{y}_{1:t})$. As discussed earlier, the discrete nature of the approximation can lead to a skewed importance weights distribution. That is, many particles have no children ($N_i = 0$), whereas others have a large number of children, the extreme case being $N_i = N$ for a particular value i. In this case, there is a severe depletion of samples. A strategy for improving the results involves introducing MCMC steps of invariant distribution $p(\boldsymbol{\theta}_{0:t}|\mathbf{y}_{1:t})$ on each particle (Andrieu et al., 1999e; Carpenter et al., 1999; Doucet and Gordon, 1999; Gilks and Berzuini, 1998; MacEachern et al., 1999). The basic idea is that, by applying a Markov transition kernel, the total variation of the current distribution with respect to the invariant distribution can only decrease. Note, however, that we do not require this kernel to be ergodic. Convergence results for this type of algorithm are presented in (Gilks and Berzuini, 1998).

It is possible to generalise this idea and to introduce MCMC steps on the product space with invariant distribution $\prod_{i=1}^{N} p(\boldsymbol{\theta}_{0:t}^{(i)}|\mathbf{y}_{1:t})$, that is to apply MCMC steps on the entire population of particles. It should be noted that independent MCMC steps spread out the particles in a particular mode more evenly, but do not explore modes devoid of particles, unless "clever" proposal distributions are available. By adopting MCMC steps on the whole population, we can draw upon many of the ideas developed in parallel MCMC computation. In this work, however, we limit ourselves to the simpler case of using independent MCMC transitions steps on each particle. In particular, we propose to use mixtures of Metropolis-Hastings steps to ensure an efficient exploration of the sample space. Later, in Chapter 7, we adopt reversible jump MCMC steps (Green, 1995) so as to allow the particles to move from one subspace to other subspaces of, possibly, different dimension.

6.8 SMC Algorithm for MLPs

We are now in a position to present an SMC algorithm for MLPs that samples approximately from the optimal importance distribution and applies a mixture of Metropolis-Hastings steps to improve the exploration of the parameter space. The approximate samples from the optimal importance distribution are obtained by performing a local linearisation with the EKF.

If we choose the prior $p(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1})$ as proposal, we rely solely on probing the error surface at several points as shown in Figures 6.2 and 6.3. This strategy fails to make use of all the information implicit in the error surface. For instance, the method could be enhanced by evaluating the gradient and other higher derivatives of the error surface. This is evident in Figure 6.9, where it is shown that by incorporating gradient information in the proposal distribution, the algorithm can reach lower minima.

In the neural networks context, the success of sub-optimal gradient descent methods motivates the inclusion of a gradient descent step in the proposal distribution. For instance, one could propose from a Gaussian proposal with mean:

$$\begin{aligned}\overline{\overline{\boldsymbol{\theta}}}_t &= \boldsymbol{\theta}_{t-1} - \frac{\alpha}{2}\frac{\partial}{\partial \boldsymbol{\theta}_{t-1}}(\mathbf{y}_t - \widehat{\mathbf{f}}_{\mathbf{t}}(\mathbf{x}_{\mathbf{t}}, \boldsymbol{\theta}_{\mathbf{t-1}}))^{\mathbf{2}} \\ &= \boldsymbol{\theta}_{t-1} + \alpha(\mathbf{y}_t - \widehat{\mathbf{f}}_{\mathbf{t}}(\mathbf{x}_{\mathbf{t}}, \boldsymbol{\theta}_{\mathbf{t-1}}))\frac{\partial \widehat{\mathbf{f}}_{\mathbf{t}}(\mathbf{x}_{\mathbf{t}}, \boldsymbol{\theta}_{\mathbf{t-1}})}{\partial \boldsymbol{\theta}_{\mathbf{t-1}}} \end{aligned} \tag{6.10}$$

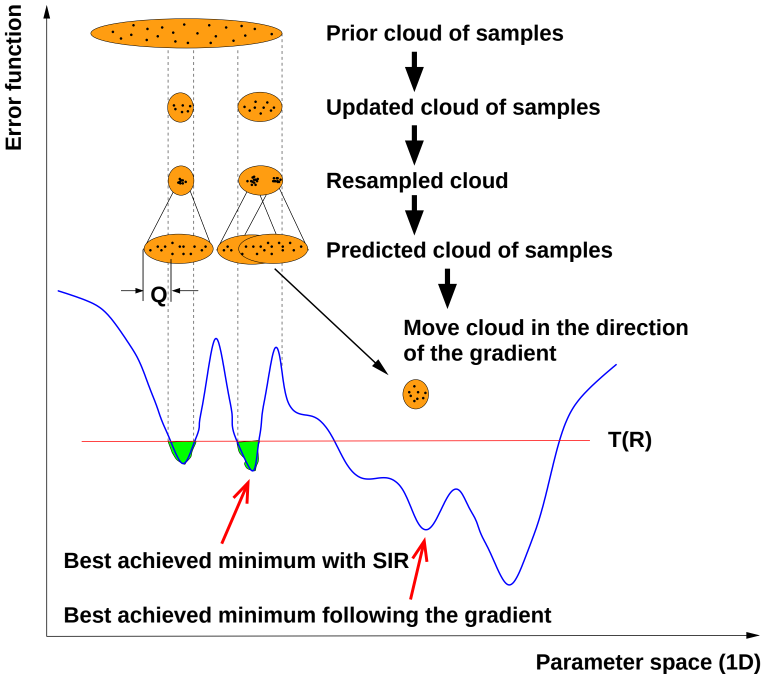
Error function
Prior cloud of samples
Updated cloud of samples
Resampled cloud
Predicted cloud of samples
Q
Move cloud in the direction of the gradient
T(R)
Best achieved minimum with SIR
Best achieved minimum following the gradient
Parameter space (1D)

Figure 6.9 *By following the gradient of the error function, it is possible to reach lower minima.*

The term $\partial\widehat{\mathbf{f}}_\mathbf{t}(\mathbf{x}_\mathbf{t}, \boldsymbol{\theta}_{\mathbf{t}-\mathbf{1}})/\partial\boldsymbol{\theta}_{\mathbf{t}-\mathbf{1}}$ is the Jacobian. It can be easily computed by back-propagating derivatives through the network as explained in Appendix B. The gradient descent learning rate α is a tuning parameter that controls how fast we should descend. Too large a parameter would cause the trajectory in the error surface to over-fluctuate, thereby never converging. A very small value, on the other hand, would not contribute towards speeding the algorithm's convergence.

The plain gradient descent algorithm can get trapped at shallow local minima. One way of overcoming this problem is to define the error surface in terms of a Hamiltonian that accounts for the approximation errors and the momentum of each trajectory. This is the basis of the Hybrid Monte Carlo algorithm (Brass et al., 1993; Duane et al., 1987). In this algorithm, each trajectory is updated by approximating the Hamiltonian differential equations by a leapfrog discretisation scheme. The discretisation may, however, introduce biases. In our work, we favour a stochastic gradient descent approach based on the extended Kalman filter. This technique avoids shallow local minima by adaptively adding noise to the network weights, as discussed in Chapter 3.

Another advantage of this method is that the covariance of the weights changes over time, and since the extended Kalman filter is a minimum variance estimator, it should decrease with time. Consequently, as the tracking improves, the variation of the network weights is reduced. This annealing process improves the efficiency of the

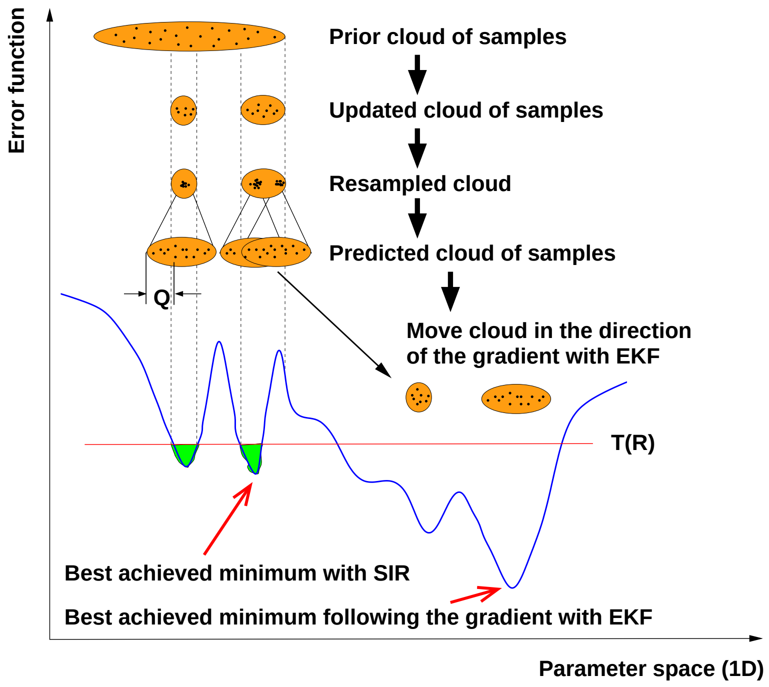
Error function
Prior cloud of samples
Updated cloud of samples
Resampled cloud
Predicted cloud of samples
Q
Move cloud in the direction
of the gradient with EKF
T(R)
Best achieved minimum with SIR
Best achieved minimum following the gradient with EKF
Parameter space (1D)

Figure 6.10 *Using the EKF to propose particles. By adding noise to the network weights and adapting the weights covariance, it is possible to reach better minima with the EKF than with plain gradient descent techniques.*

search for global minima in parameter space and reduces the variance of the estimates.

By adopting an EKF proposal, we can present an SMC algorithm to estimate the network weights and measurement noise covariance R_t jointly. R_t is assumed to obey the following diffusion process:

$$\log(R_t) = \log(R_{t-1}) + \epsilon_R$$

where $\epsilon_R \sim \mathcal{N}(\mathbf{0}, \delta_R^2 \mathbf{I}_c)$ and δ_R is a tuning hyper-parameter. We are assuming that the measurement noise is Gaussian. If this were not the case, we could still use a similar method to update the minimum statistics of the noise process. For instance, if the noise can be described by two Gaussian components, then the algorithm can be easily extended by replacing R_t with $\{R_{1,t}, R_{2,t}\}$ and the weights of the mixture. At time t, the algorithm proceeds as follows:

SMC algorithm for Regression with MLPs

1. Bayesian importance sampling step

   - For $i = 1, \ldots, N$:

  - Compute the Jacobian G_t.
  - Sample $\widehat{R}_t^{(i)} \sim p(R_t | R_{t-1}^{(i)})$
  - Update the MLP weights in the direction of the gradient with the EKF:

$$\begin{aligned} K_t &= (P_{t-1}^{(i)} + Q_{t-1}^*)G_t'[R_t^* + G_t(P_{t-1}^{(i)} + Q_{t-1}^*)G_t']^{-1} \\ \overline{\widehat{\boldsymbol{\theta}}}_t^{(i)} &= \boldsymbol{\theta}_{t-1}^{(i)} + K_t(\mathbf{y}_t - \widehat{\mathbf{f}}(\boldsymbol{\theta}_{t-1}^{(i)}, \mathbf{x}_t)) \\ \widehat{P}_t^{(i)} &= P_{t-1}^{(i)} + Q_{t-1}^* - K_t G_t(P_{t-1}^{(i)} + Q_{t-1}^*) \end{aligned}$$

  - Sample $\widehat{\boldsymbol{\theta}}_t^{(i)} \sim q(\overline{\widehat{\boldsymbol{\theta}}}_t^{(i)} | \boldsymbol{\theta}_{0:t-1}^{(i)}, , \mathbf{y}_{1:t}) = \mathcal{N}\big(\overline{\widehat{\boldsymbol{\theta}}}_t^{(i)}, \widehat{P}_t^{(i)}\big)$
  - Set:

$$\begin{aligned} \widehat{\boldsymbol{\theta}}_{0:t}^{(i)} &\triangleq \big(\boldsymbol{\theta}_{0:t-1}^{(i)}, \widehat{\boldsymbol{\theta}}_t^{(i)}\big) \\ \widehat{R}_{0:t}^{(i)} &\triangleq \big(R_{0:t-1}^{(i)}, \widehat{R}_t^{(i)}\big) \\ \widehat{P}_{0:t}^{(i)} &\triangleq \big(P_{0:t-1}^{(i)}, \widehat{P}_t^{(i)}\big) \end{aligned}$$

- For $i = 1, \ldots, N$, evaluate the importance weights up to a normalising constant:

$$w_t^{(i)} \propto \frac{p(\mathbf{y}_t | \widehat{\boldsymbol{\theta}}_t^{(i)}, \widehat{R}_t^{(i)}) p(\widehat{\boldsymbol{\theta}}_t^{(i)} | \boldsymbol{\theta}_{t-1}^{(i)})}{q(\widehat{\boldsymbol{\theta}}_t^{(i)} | \boldsymbol{\theta}_{0:t-1}^{(i)}, \mathbf{y}_{1:t})}$$

- For $i = 1, \ldots, N$, normalise the importance weights:

$$\widetilde{w}_t^{(i)} = w_t^{(i)} \left[\sum_{j=1}^{N} w_t^{(j)}\right]^{-1}$$

2. <u>Selection step</u>

   - Multiply/Suppress samples $(\widehat{\boldsymbol{\theta}}_{0:t}^{(i)}, \widehat{P}_{0:t}^{(i)}, \widehat{R}_{0:t}^{(i)})$ with high/low importance weights $\widetilde{w}_t^{(i)}$, respectively, to obtain N random samples $(\widetilde{\boldsymbol{\theta}}_{0:t}^{(i)}, \widetilde{P}_{0:t}^{(i)}, \widetilde{R}_{0:t}^{(i)})$.

3. <u>MCMC step</u>

   For $i = 1, \ldots, N$:

   - $R_{0:t}^{(i)} = \widetilde{R}_{0:t}^{(i)}$
   - Sample $u \sim \mathcal{U}_{[0,1]}$.
   - If $(u \leq 0.5)$
     - then perform a local MCMC step:
       * Sample $v \sim \mathcal{U}_{[0,1]}$.
       * Sample the proposal candidate $\boldsymbol{\theta}_t^{\star(i)} \sim p(\boldsymbol{\theta}_t | \boldsymbol{\theta}_{t-1}^{(i)}) = \mathcal{N}\big(\widetilde{\boldsymbol{\theta}}_{t-1}^{(i)}, Q_{t-1}\big)$

```
        * If $v \leq \min\left\{1, \frac{p(\mathbf{y}_t|\boldsymbol{\theta}_t^{\star(i)}, R_t^{(i)})}{p(\mathbf{y}_t|\widetilde{\boldsymbol{\theta}}_t^{(i)}, R_t^{(i)})}\right\}$
            · then accept move:
                  $\boldsymbol{\theta}_{0:t}^{(i)} = (\widetilde{\boldsymbol{\theta}}_{0:t-1}^{(i)}, \boldsymbol{\theta}_t^{\star(i)})$
                  $P_{0:t}^{(i)} = \widetilde{P}_{0:t}^{(i)}$
            · else reject move:
                  $\boldsymbol{\theta}_{0:t}^{(i)} = \widetilde{\boldsymbol{\theta}}_{0:t}^{(i)}$
                  $P_{0:t}^{(i)} = \widetilde{P}_{0:t}^{(i)}$
          End If.
    – else perform a gradient MCMC step:
        * Sample $v \sim \mathcal{U}_{[0,1]}$.
        * Sample the candidate $\boldsymbol{\theta}_t^{\star(i)} \sim q(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{0:t-1}^{(i)}, \mathbf{y}_{1:t}) = \mathcal{N}(\overline{\widetilde{\boldsymbol{\theta}}}_{t-1}^{(i)}, \overline{\widetilde{P}}_{t-1}^{(i)})$,
          where:
                  $K_t = (\widetilde{P}_{t-1}^{(i)} + Q_{t-1}^*)G_t'[R_t^* + G_t(\widetilde{P}_{t-1}^{(i)} + Q_{t-1}^*)G_t']^{-1}$
                  $\overline{\widetilde{\boldsymbol{\theta}}}_t^{(i)} = \widetilde{\boldsymbol{\theta}}_{t-1}^{(i)} + K_t(\mathbf{y}_t - \widehat{\mathbf{f}}(\widetilde{\boldsymbol{\theta}}_{t-1}^{(i)}, \mathbf{x}_t))$
                  $\overline{\widetilde{P}}_t^{(i)} = \widetilde{P}_{t-1}^{(i)} + Q_{t-1}^* - K_t G_t(\widetilde{P}_{t-1}^{(i)} + Q_{t-1}^*)$
          and $G_t$ is evaluated at $\widetilde{\boldsymbol{\theta}}_{t-1}^{(i)}$.
        * If $v \leq \min\left\{1, \frac{p(\mathbf{y}_t|\boldsymbol{\theta}_t^{\star(i)}, R_t^{(i)})p(\boldsymbol{\theta}_t^{\star(i)}|\widetilde{\boldsymbol{\theta}}_{t-1}^{(i)})q(\widetilde{\boldsymbol{\theta}}_t^{(i)}|\widetilde{\boldsymbol{\theta}}_{0:t-1}^{(i)}, \mathbf{y}_{1:t})}{p(\mathbf{y}_t|\widetilde{\boldsymbol{\theta}}_t^{(i)}, R_t^{(i)})p(\widetilde{\boldsymbol{\theta}}_t^{(i)}|\widetilde{\boldsymbol{\theta}}_{t-1}^{(i)})q(\boldsymbol{\theta}_t^{\star(i)}|\widetilde{\boldsymbol{\theta}}_{0:t-1}^{(i)}, \mathbf{y}_{1:t})}\right\}$
            · then accept move:
                  $\boldsymbol{\theta}_{0:t}^{(i)} = (\widetilde{\boldsymbol{\theta}}_{0:t-1}^{(i)}, \boldsymbol{\theta}_t^{\star(i)})$
                  $P_{0:t}^{(i)} = (\widetilde{P}_{0:t-1}^{(i)}, \overline{\widetilde{P}}_t^{(i)})$
            · else reject move:
                  $\boldsymbol{\theta}_{0:t}^{(i)} = \widetilde{\boldsymbol{\theta}}_{0:t}^{(i)}$
                  $P_{0:t}^{(i)} = \widetilde{P}_{0:t}^{(i)}$
          End If.
  End If.
```

This algorithm is similar to the SIR algorithm, with the exception that it introduces an MCMC mixture of two steps. The first step allows the Markov chain to explore local neighbourhoods. Therefore, the samples in each mode are spread out more evenly. The same result could be obtained with kernel smoothing methods. However, the second step allows for larger stochastic jumps in the direction of the gradient. This is one of the great advantages of MCMC methods over other techniques. MCMC mixtures allow us to incorporate many proposals and heuristics within a rigorous stochastic framework.

Note that we included the word regression in the title of the previous algorithm. We did this because, when dealing with classification, the EKF does not constitute a suitable proposal. Since the class labels are noiseless, and the normalising factor of the EKF's posterior is proportional to the measurement noise covariance, the EKF algorithm degenerates. This problem can be circumvented by adding artificial measurement noise. Yet, a better strategy is to simply propose from the transition prior and use a Bernoulli likelihood. In this case, we shall also adopt a random walk MCMC move.

6.9 HySIR: an Optimisation Strategy

It is possible to design suboptimal SMC algorithms to perform maximisation. The main purpose of such methods is to find the peaks of the posterior instead of attempting to provide a complete description of it. Here, great emphasis is placed on reducing the computational cost. One possible strategy is to modify the algorithm presented in the previous section as follows:

HySIR algorithm for Regression with MLPs

```
1. Pseudo-importance sampling step
    • For $i = 1, \dots, N$:
        – Compute the Jacobian $G_t$.
        – Sample $\widehat{R}_t^{(i)} \sim p(R_t|R_{t-1}^{(i)})$ and $\widehat{\boldsymbol{\theta}}_t^{(i)} \sim p(\boldsymbol{\theta}_t^{(i)}|\boldsymbol{\theta}_{0:t-1}^{(i)})$.
        – Update the MLP weights in the direction of the gradient with the EKF:
              $K_t = (P_{t-1}^{(i)} + Q_{t-1}^*)G_t'[R_t^* + G_t(P_{t-1}^{(i)} + Q_{t-1}^*)G_t']^{-1}$
              $\overline{\widehat{\boldsymbol{\theta}}}_t^{(i)} = \boldsymbol{\theta}_{t-1}^{(i)} + K_t(\mathbf{y}_t - \widehat{\mathbf{f}}(\boldsymbol{\theta}_{t-1}^{(i)}, \mathbf{x}_t))$
              $\widehat{P}_t^{(i)} = P_{t-1}^{(i)} + Q_{t-1}^* - K_t G_t(P_{t-1}^{(i)} + Q_{t-1}^*)$
        – Set:
              $\widehat{\boldsymbol{\theta}}_{0:t}^{(i)} \triangleq (\boldsymbol{\theta}_{0:t-1}^{(i)}, \overline{\widehat{\boldsymbol{\theta}}}_t^{(i)})$
              $\widehat{R}_{0:t}^{(i)} \triangleq (R_{0:t-1}^{(i)}, \widehat{R}_t^{(i)})$
              $\widehat{P}_{0:t}^{(i)} \triangleq (P_{0:t-1}^{(i)}, \widehat{P}_t^{(i)})$
```

- For $i = 1, \ldots, N$, evaluate the selection weights up to a normalising constant:

$$w_t^{(i)} \propto p(\mathbf{y}_t | \widehat{\boldsymbol{\theta}}_t^{(i)}, \widehat{R}_t^{(i)})$$

- For $i = 1, \ldots, N$, normalise the importance weights.

2. Selection step

- Multiply/Suppress samples $(\widehat{\boldsymbol{\theta}}_{0:t}^{(i)}, \widehat{P}_{0:t}^{(i)}, \widehat{R}_{0:t}^{(i)})$ with high/low importance weights $\widetilde{w}_t^{(i)}$, respectively, to obtain N random samples $(\boldsymbol{\theta}_{0:t}^{(i)}, P_{0:t}^{(i)}, R_{0:t}^{(i)})$.

■

The HySIR (hybrid EKF-SIR) may be interpreted as a dynamical mixture of EKFs. That is, the estimates depend on a few modes of the posterior density function. Note that before the EKF step, we sample the weights from the transition prior. The purpose of this is to introduce random variations in the cloud of particles. This issue will be further explained by means of an experiment in Section 6.12.3.

In addition to having to compute the normalised importance ratios ($\mathcal{O}(N(cS_1^2 + d^2 S_1))$) operations, where S_1 is the number of hidden neurons, N the number of samples, d the input dimension and c the number of outputs) and resampling ($\mathcal{O}(N)$ operations), the HySIR has to perform the extended Kalman filter updates ($\mathcal{O}(Ncm^2)$ operations, where m is the number of network weights). That is, the computational complexity of the HySIR increases approximately linearly with the number of EKFs in the mixture. For problems with dominant modes, we only need to use a few trajectories and, consequently, the HySIR provides accurate and computationally efficient solutions.

The HySIR approach makes use of the likelihood function twice per time step. It therefore concentrates the particles in regions of high likelihood. Consequently, if we make the noise distributions too peaked, the algorithm will only give a representation of a narrow region of the posterior density function. Typically, the algorithm will converge to a few modes of the posterior density function. This is illustrated later in Section 6.12.3.

6.10 Interval and Fixed Lag Smoothing

According to the theory and algorithms presented thus far, we can marginalise the distribution:

$$\widehat{p}(\boldsymbol{\theta}_{0:t} | \mathbf{y}_{1:t}) = \sum_{i=1}^{N} \widetilde{w}_t^{(i)} \delta_{\boldsymbol{\theta}_{0:t}^{(i)}}(\mathrm{d}\boldsymbol{\theta}_{0:t})$$

to obtain fixed interval smoothed estimates $p(\boldsymbol{\theta}_\tau|\mathbf{y}_{1:t})$, where $\tau \leq t$. To accomplish this, at each time step one needs to resample the previous state trajectories using the current importance weights. In doing this, we encounter two difficulties. Firstly, resampling the past trajectories at each time step leads to a significant depletion of samples. Secondly, one has to store in memory all the past trajectories. The first problem can be solved to a certain degree by applying MCMC moves to the past trajectories. This approach also allows us to introduce Gibbs sampling steps to estimate the noise statistics (Andrieu et al., 1999e; Gilks and Berzuini, 1998). It is also possible to solve the second problem if we can find a way of expressing the past history of the particles in terms of a set of sufficient statistics (Andrieu et al., 1999e; Fearnhead, 1998). If we are interested in filtering instead of smoothing, there is no need to store the past trajectories.

An alternative smoothing strategy, known as fixed-lag smoothing, involves estimating the distribution $p(\boldsymbol{\theta}_t|\mathbf{y}_{1:t+p})$ where $p > 0$ is the length of the lag. Approaches to solve this problem, using forward sampling/backward filtering algorithms over the lag of data, are proposed in (Doucet et al., 1999; Kitagawa, 1996).

6.11 A Brief Note on the Convergence of SMC Algorithms

The theoretical convergence of SMC algorithms is an area of active research. Here we present some results derived from (Crisan et al., 1999; Del Moral and Guionnet, 1999)[1]. To make the material accessible to a wide audience, we have simplified the presentation. Readers are encouraged to consult the original references should any confusion or ambiguities arise. In (Crisan et al., 1999), the following result is proved:

Proposition 3 [Theorem 3.2 of (Crisan et al., 1999)] *If the function $g_t(\cdot)$ and the unnormalised importance weights w_t are bounded, then, under a few regularity conditions, for all $t > 0$ we have the following convergence result:*

$$\lim_{N_0\to\infty} \mathbb{E}\left[\left(\frac{1}{N}\sum_{i=1}^{N} g_t(\boldsymbol{\theta}_t)\delta_{\boldsymbol{\theta}_t^{(i)}}(d\boldsymbol{\theta}_t) - \int g_t(\boldsymbol{\theta}_t)p(\boldsymbol{\theta}_t|\mathbf{y}_{1:t})d\boldsymbol{\theta}_t\right)^2\right] = 0$$

That is, it is possible to obtain convergence in time if we have an infinite number of particles at the beginning. A more interesting goal is to prove that the error is bounded uniformly in time. In this regard, the following stronger result has been proved recently:

Proposition 4 [Theorem 3.7 of (Del Moral and Guionnet, 1999)] *Assume that the measurements mapping $\widehat{\mathbf{f}}_t(\cdot)$ is upper bounded and that the distribution of the measurement noise is lower and upper bounded, then for any bounded function $g_t(\cdot)$, such that*

[1] I would like to acknowledge Arnaud Doucet for having helped me to understand these results.

$\|g_t(\cdot)\| \leq 1$, *we have:*

$$\sup_{t\geq 0} \log \mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N} g_t(\boldsymbol{\theta}_t)\delta_{\boldsymbol{\theta}_t^{(i)}}(d\boldsymbol{\theta}_t) - \int g_t(\boldsymbol{\theta}_t)p(\boldsymbol{\theta}_t|\mathbf{y}_{1:t})d\boldsymbol{\theta}_t\right] \leq \frac{5\exp(2\gamma)}{N^{\alpha/2}}$$

where α is positive and γ is a positive bound. Note that, at this point in time, the assumptions are still very restrictive. Yet the results are very powerful and should soon lead to stronger theorems.

There are also empirical methods that enable us to assess the convergence of SMC algorithms using Cramér-Rao and posterior Cramér-Rao bounds: see (Bergman, 1999) for a very lucid presentation. For example, for our state space model we have the following posterior Cramér-Rao bound:

Proposition 5 [Theorem 4.2 of (Bergman, 1999)] *If the estimator's bias is bounded, then the mean square error of the estimator is bounded from below:*

$$\mathbb{E}\big[(\widehat{\boldsymbol{\theta}}_t - \boldsymbol{\theta}_t)(\widehat{\boldsymbol{\theta}}_t - \boldsymbol{\theta}_t)'\big] \geq \big(J(\boldsymbol{\theta}_t)\big)^{-1}$$

where $J(\boldsymbol{\theta}_t) = \mathbb{E}\big[\frac{\partial \log p(\mathbf{y}_t|\boldsymbol{\theta}_t)}{\partial \boldsymbol{\theta}}\frac{\partial \log p(\mathbf{y}_t|\boldsymbol{\theta}_t)}{\partial \boldsymbol{\theta}}'\big]$ *is the Fisher information matrix (FIM).*

For example, if we assume that the initial, process noise and measurement noise distributions are $\mathcal{N}(\mathbf{0}, P_0)$, $\mathcal{N}(\mathbf{0}, Q)$ and $\mathcal{N}(0, R)$ respectively, then for M random walk realisations of the states, we have the following empirical bound:

$$\sqrt{\frac{1}{M}\sum_{i=1}^{M}\|\widehat{\boldsymbol{\theta}}_t^i - \boldsymbol{\theta}_t^i\|^2} \gtrsim \sqrt{\mathrm{tr}P_t}$$

where tr denotes the trace of a matrix and

$$P_{t+1}^{-1} = Q^{-1} - Q^{-1}\big(P_t^{-1} + Z_t + Q_t^{-1}\big)^{-1}Q^{-1}$$

with

$$Z_t = R^{-1}\mathbb{E}\left[\left(\frac{\partial\widehat{\mathbf{f}}_t(\mathbf{x}_t, \boldsymbol{\theta}_t)}{\partial\boldsymbol{\theta}_t}\right)\left(\frac{\partial\widehat{\mathbf{f}}_t(\mathbf{x}_t, \boldsymbol{\theta}_t)}{\partial\boldsymbol{\theta}_t}\right)'\right]$$

Z_t can be approximated by the Monte Carlo sample average over the M independent realisations of the states. For a fixed value of the states, the Cramér-Rao bound is simply given by the Riccati equation (equation (3.6)) with the Jacobian evaluated at the true states (Bergman, 1999).

6.12 Experiments

This section presents three experiments using synthetic data. The first experiment provides a comparison between the various algorithms discussed in the chapter. It addresses the trade-off between accuracy and computational efficiency. The second experiment illustrates how SMC methods can be used for time-varying probabilistic classification. In this scenario, the EKF performs poorly as the target distribution is Bernoulli. The third experiment shows how the HySIR algorithm can be used to estimate time-varying latent variables. It also serves the purpose of illustrating the effect of the process noise covariance Q on the simulations.

6.12.1 Experiment 1: time-varying function approximation

Input-output data was generated using the following time-varying function:

$$y(x_1(k), x_2(k)) = 4\sin(x_1(k) - 2) + 2x_2(k)^2 + 5\cos(0.02k) + 5 + \nu$$

where the inputs $x_1(k)$ and $x_2(k)$ were simulated from a Gaussian distribution with zero mean and unit variance. The noise ν was drawn from a Gaussian distribution with zero mean and variance equal to 0.1. Subsequently, the data was approximated using an MLP with 5 hidden sigmoidal neurons and a linear output neuron. The MLP was trained sequentially using the HySIR, SIR, SIS, EKF and SMC algorithms. The difference between the SIR and SMC algorithms and the SIS algorithm is that SIR and SMC resample every time, while SIS only resamples if the effective sample set is below a threshold. The threshold was set to $N/3$.

The number of samples (N) was set to 100 for the SIS, SIR and SMC methods and to 10 for the HySIR method. The values 100 and 1 were assigned to the initial variance of the weights and the diagonal entries of the weights covariance matrix. The diagonal entries of Q and R were given the values 0.5 and 2 respectively. Finally, the extended Kalman filter noise hyper-parameters R^* and Q^* were set to 0.1 and 0.01.

Table 6.1 shows the average one-step-ahead prediction errors obtained for 100 runs, of 200 time steps each, on a Silicon Graphics R10000 workstation. On average, there was a 50% occurrence of resampling steps using the SIS algorithm. It is clear from the results that avoiding the resampling stage does not yield a great reduction in computational time. This is because one needs to evaluate neural network mappings for each trajectory in the sampling stage. As a result, the sampling stage tends to be much more computationally expensive than the resampling stage. The results also show that the HySIR performs better than the conventional SIR and EKF algorithms. It is, however, computationally more expensive, and fails to provide a complete description of the posterior distribution. Neglecting computing time constraints, the results for the SIR and SMC algorithm can be improved by increasing the number of particles.

It is interesting to note that the HySIR, despite requiring more floating point operations, is faster than the SIR algorithm. This shows the effect of memory access on the execution time: efficient memory access through cache systems requires temporal

	EKF	SIS	SIR	HySIR	SMC
RMS Error	6.51	3.87	3.27	1.17	2.89
Mega flops	4.8	5.6	5.8	48.6	2670.5
Time (seconds)	1.2	27.4	28.9	24.9	269.2

Table 6.1 *RMS errors and floating point operations for the algorithms used in the function approximation problem. It should be noted that the RMS errors include the convergence period. This explains why they are higher than the variance of the additive noise.*

and spatial locality of reference. The design of efficient software and hardware platforms is an important avenue for further research in the field of sequential Monte Carlo simulation.

Figure 6.11 shows two plots of the prediction errors for the EKF and HySIR. The left plot shows the best and worst results obtained with the EKF out of 100 trials, each trial involving a different initial weights vector. The 100 initial weights vectors were then used to initialise the HySIR algorithm with 100 sampling trajectories. As shown in the right plot, the global search nature of the HySIR algorithm allows it to converge much faster than the EKF algorithm. This is a clear example of how dynamic mixtures of models, with model selection at each time step, perform better than static mixtures (choosing the best EKF out of the 100 trials).

6.12.2 Experiment 2: sequential classification

Sequential classification problems arise in a few areas of technology (Melvin, 1996; Penny et al., 1999). Often, it is necessary to monitor a set of signals and decide, online, to which of two classes they belong. For example, when monitoring patients, we might wish to decide whether they require an increase in drug intake at several intervals in time.

Here, it is shown that particle filters provide an efficient and elegant probabilistic solution to this problem. A synthetic problem, where the target classes change with time, was considered. Specifically, the data was generated from two two-dimensional overlapping classes ($\mathcal{N}([0;0], 0.15\mathbf{I}_2)$ and $\mathcal{N}([1;1], 0.15\mathbf{I}_2)$) between $t = 1$ and $t = 100$. The variances and means of the two classes were changed to $\mathcal{N}([0.5;0], 0.1\mathbf{I}_2)$ and $\mathcal{N}([0.5;1.5], 0.15\mathbf{I}_2)$, between $t = 101$ and $t = 200$. The data is depicted in Figure 6.12.

An MLP with 3 hidden logistic functions and an output logistic function was applied to classify the data. The network was trained, sequentially, with an SMC algorithm by proposing from the transition prior. An MCMC step was used to reduce the resampling variance. The number of particles was set to 200, the prior network weights were

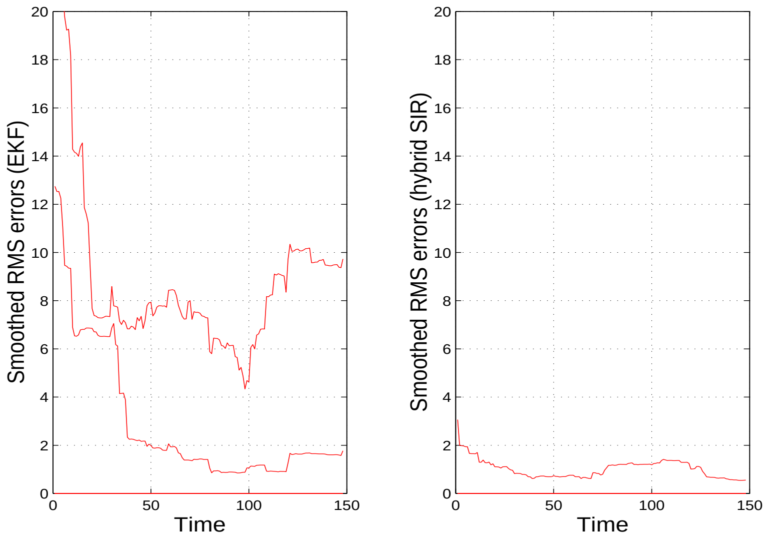

Figure 6.11 *Convergence of the EKF and HySIR algorithms for Experiment 1. The left plot shows the best and worst EKF convergence results out of 100 runs using different initial conditions. The initial conditions for each EKF were used to initialise the EKF filters within the HySIR algorithm. At each time step, the HySIR evaluates a weighted combination of several EKFs. By performing this type of dynamic model selection, it is able to search over large regions of parameter space and, hence, converge much faster than the standard EKF algorithm.*

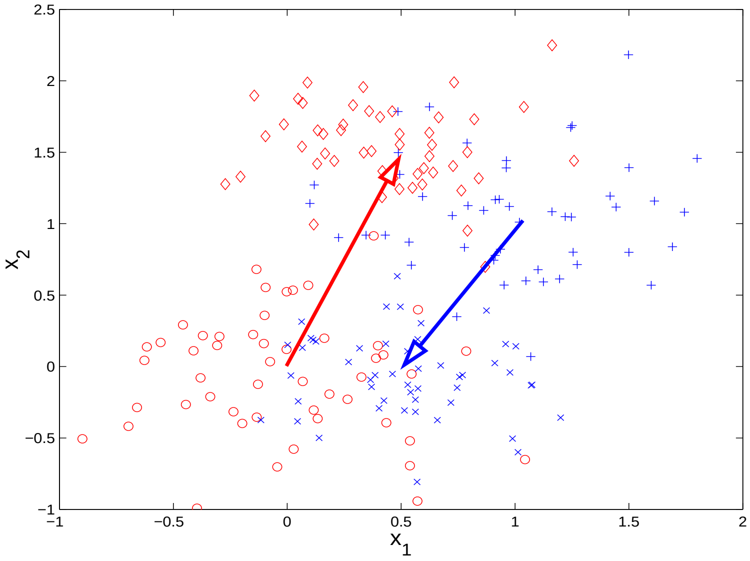

Figure 6.12 *Data generated for the classification problem. The arrows indicate the shift in the means at* $t = 100$.

sampled from $\mathcal{N}(\mathbf{0}, 10\mathbf{I}_2)$, while their diffusion parameter was set to 0.2. Figure 6.13 shows the one-step-ahead predicted class probabilities, the output labels obtained by thresholding the output at 0.5 and the cumulative mis-classification errors. Figure 6.14 depicts the evolution of the probabilities of class membership. Despite the change in the target distributions at $t = 100$, the algorithm recovers very quickly.

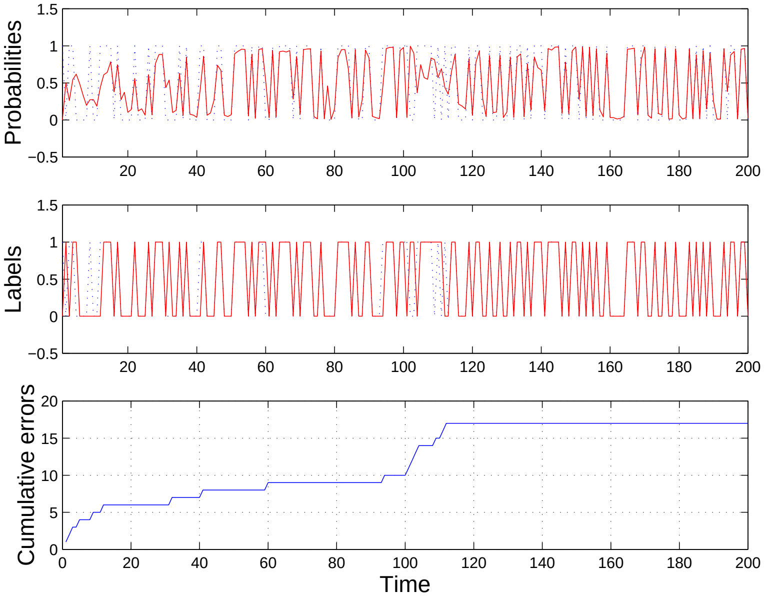

Figure 6.13 *The top plot shows the true labels [$\cdots$] and one-step-ahead predicted probabilities of class membership [—]. The middle plot shows the predicted labels using a threshold of 0.5. The bottom plot shows the cumulative number of mis-classification errors.*

6.12.3 Experiment 3: latent states tracking

To assess the ability of the hybrid algorithm to estimate time-varying hidden parameters, input-output data was generated from a logistic function followed by a linear scaling and a displacement as shown in Figure 6.15. Two Gaussian input sequences were applied to the model. This simple model is equivalent to an MLP with one hidden neuron and an output linear neuron. A Second model was then trained with the same structure using the input-output data generated by the first model.

In the training phase, of 200 time steps, the model weights were allowed to vary with time. During this phase, the HySIR algorithm was used to track the input-output training data and estimate the latent model weights. After the 200-th time step, the values of the weights were fixed and another 200 input-output data test sets were generated from the original model. The input test data was then fed to the trained model, using the weights values estimated at the 200-th time step. Subsequently, the

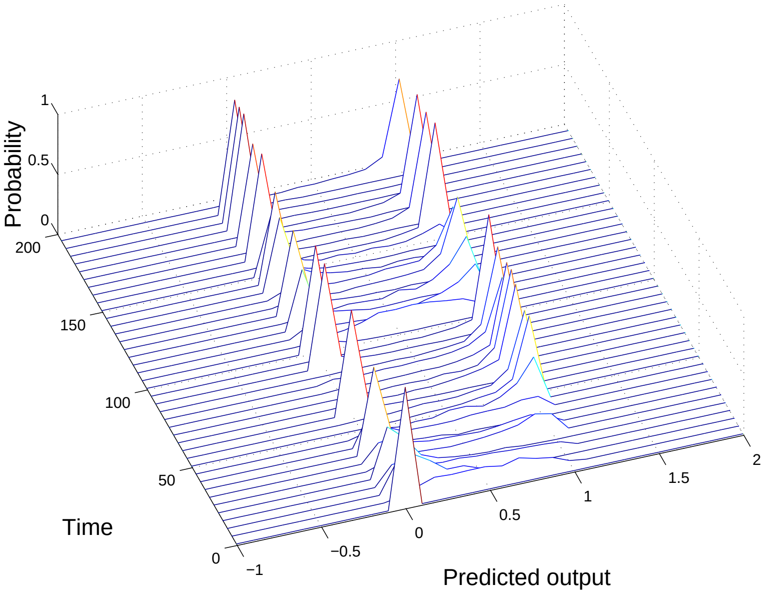

Figure 6.14 *Probabilities of class membership for the classification problem.*

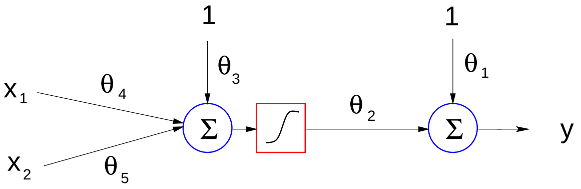

Figure 6.15 *Logistic function with linear scaling and displacement used in Experiment 3. The weights were chosen as follows:* $\boldsymbol{\theta}_{1,t} = 1 + t/100$, $\boldsymbol{\theta}_{2,t} = \sin(0.06t) - 2$, $\boldsymbol{\theta}_{3,t} = 0.1$, $\boldsymbol{\theta}_{4,t} = 1$, $\boldsymbol{\theta}_{5,t} = -0.5$.

output prediction of the trained model was compared to the output data from the original model so as to assess the generalisation performance of the training process.

The experiment was repeated using two different values of $Q = q\mathbf{I}_5$, as shown in Figures 6.16 and 6.17. With a very small value of q (0.0001), the algorithm was able to estimate the time-varying latent weights. After 100 time steps, the algorithm became very confident, as indicated by the narrow error bars. By increasing q to 0.01, it was found that the algorithm became less confident. Yet, it was able to explore wider regions of parameter space, as shown in the histogram of Figure 6.17.

The variance (Q) of the noise added in the prediction stage implies a trade-off between the algorithm's confidence and the size of the search region in parameter space. Too large a value of q will allow the algorithm to explore a large region of the space, but will never allow the algorithm to converge to a specific error minimum. A very small

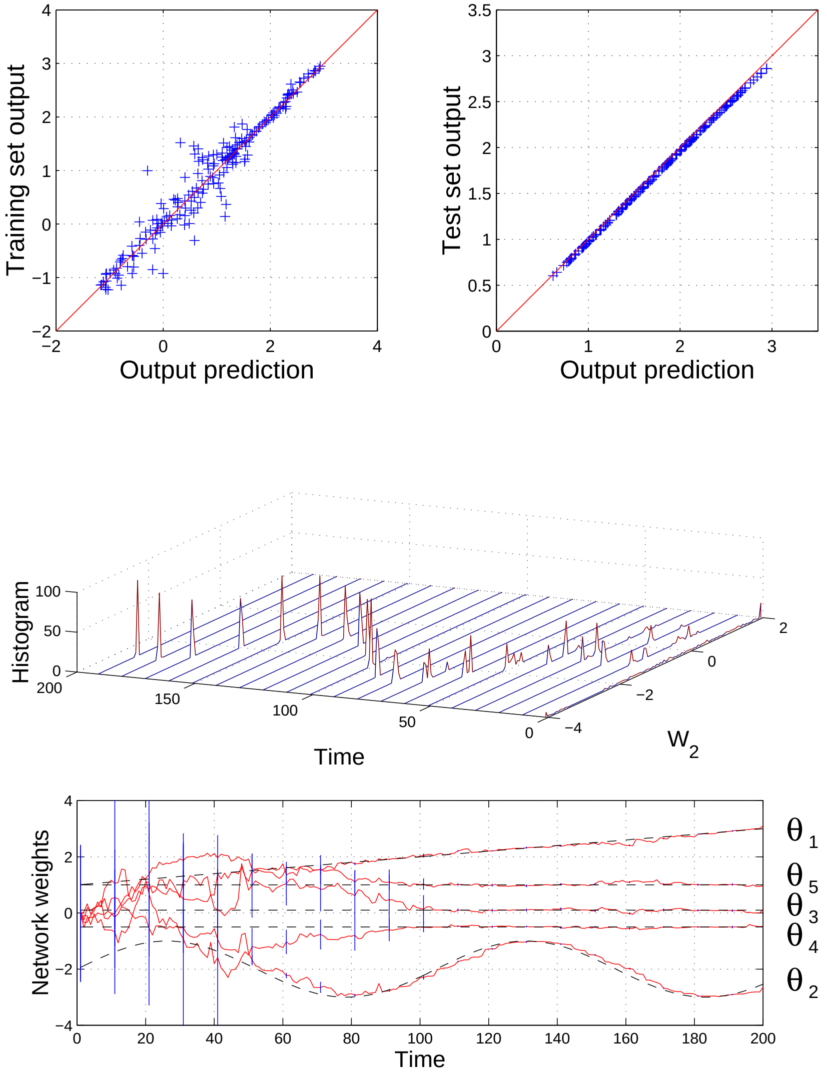

Figure 6.16 *Weights tracking with $q = 0.0001$. The top left plot shows the one-step-ahead prediction errors in the training set. The top right plots shows the prediction errors with validation data, assuming that after the 200-th time step the weights stop varying with time. The middle plot shows the histogram of $\boldsymbol{\theta}_2$. Finally, the bottom plot shows the tracking of the weights with the posterior mean estimate computed by the HySIR. The bottom plot also shows the one standard deviation error bars of the estimator. For this particular value of q, the estimator becomes very confident after 100 iterations and, consequently, the histogram for $\boldsymbol{\theta}_2$ converges to a narrow Gaussian distribution.*

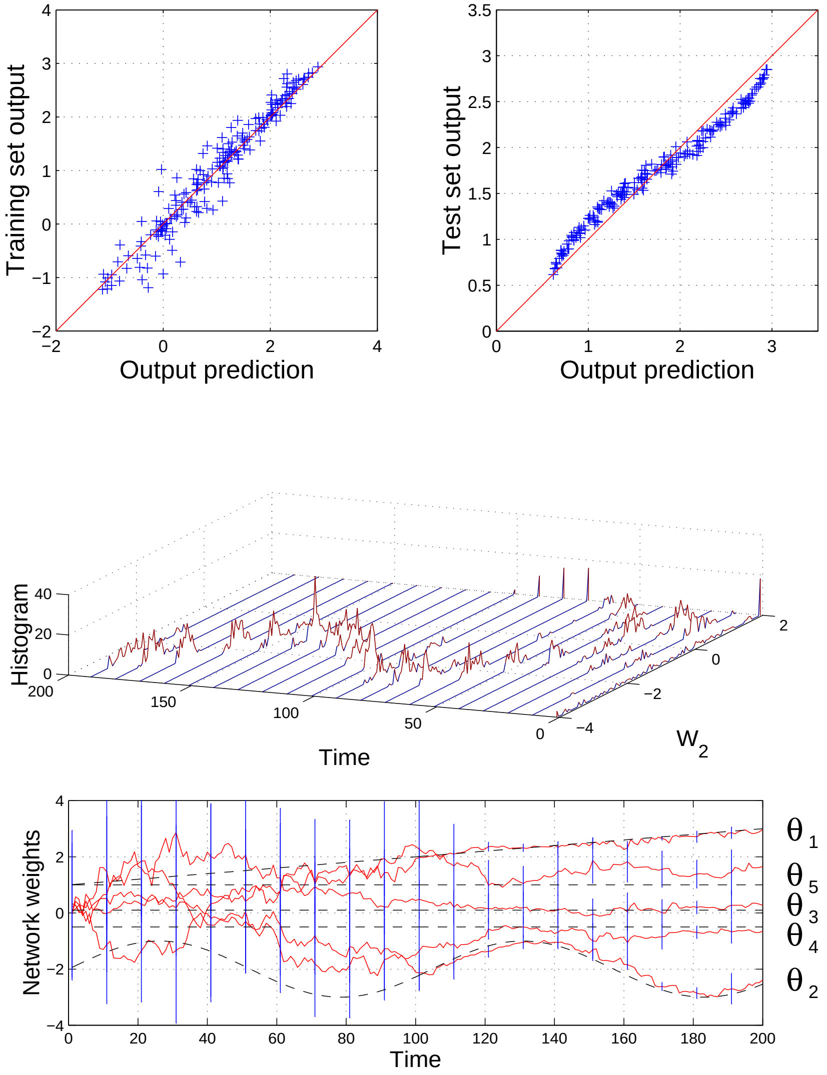

Figure 6.17 *Weights tracking with $q = 0.01$. The top left plot shows the one-step-ahead prediction errors in the training set. The top right plots shows the prediction errors with validation data, assuming that after the 200-th time step the weights stop varying with time. The middle plot shows the histogram of $\boldsymbol{\theta}_2$. Finally, the bottom plot shows the tracking of the weights with the posterior mean estimate computed by the HySIR. The bottom plot also shows the one standard deviation error bars of the estimator. For this value of q, the search region is particularly large (wide histograms), but the algorithm lacks confidence (wide error bars).*

value, on the other hand, will cause all the sampling trajectories to converge to a very narrow region. The problem with the latter is that if the process generating the data

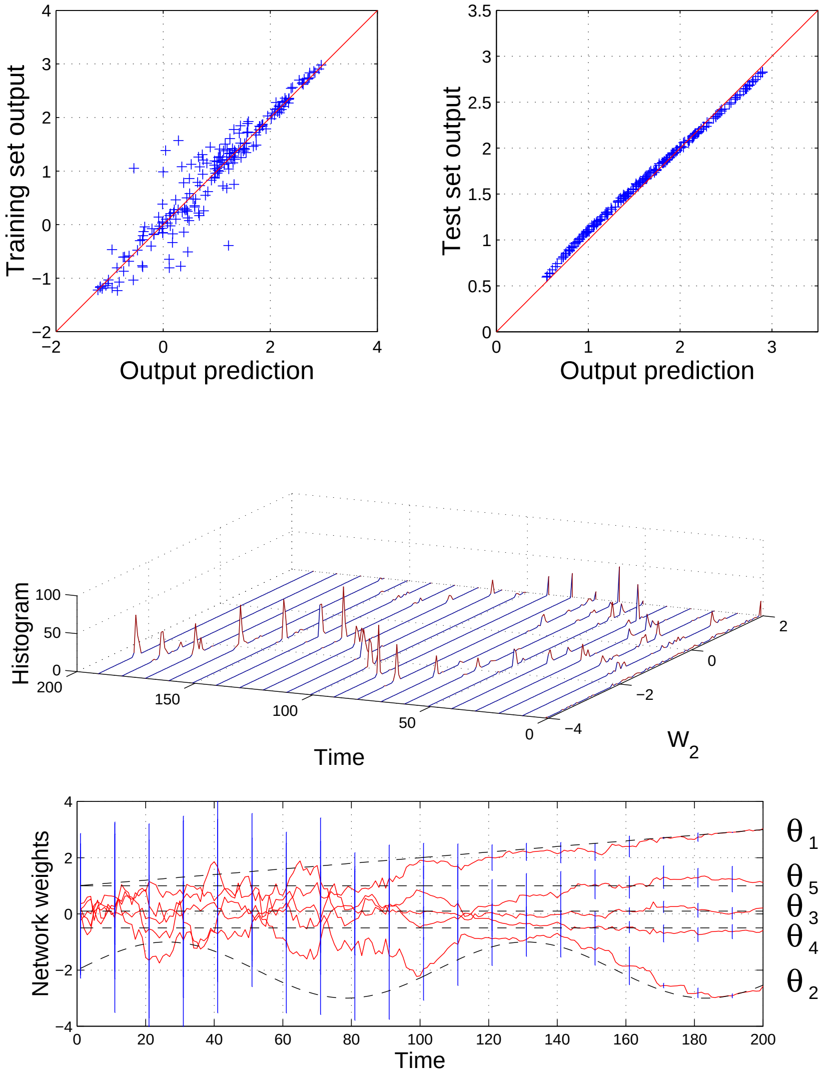

Figure 6.18 *Weights tracking with* $q = 0.0001$ *and two random mutations per time step with* $q = 0.1$. *The top left plot shows the one-step-ahead prediction errors in the training set. The top right plots shows the prediction errors with validation data, assuming that after the 200-th time step the weights stop varying with time. The middle plot shows the histogram of* $\boldsymbol{\theta}_2$. *Finally, the bottom plot shows the tracking of the weights with the posterior mean estimate computed by the HySIR. The bottom plot also shows the one standard deviation error bars of the estimator. The histogram shows that the mutations expand the search region considerably.*

changes with time, we want to keep the search region large enough so as to find the new minimum quickly and efficiently. To address this problem, a very simple heuristic has been employed. Specifically, when Gaussian noise is added in the prediction stage, a small set of the trajectories is selected randomly. Subsequently, Gaussian noise with a covariance parameter much larger than q is added to the selected trajectories. In theory, this should allow the algorithm to converge and yet allow it to keep exploring other regions of parameter space. From an evolutionary computing perspective, we can think of these few randomly selected trajectories as mutations. The results of applying this idea are shown in Figure 6.18. They suggest that this heuristic reduces the trade-off between the algorithm's confidence and the size of the search region. Yet, further research on algorithms for adapting Q is necessary.

6.13 Summary

This chapter presented a sequential Monte Carlo approach for training neural networks in a Bayesian setting. This approach enables accurate characterisation of the probability distributions. Propagating a number of samples in a state space dynamical systems formulation of the problem is the key idea in this framework. As the framework is new to the neural networks community, emphasis has been placed on describing the latest available SMC tools.

Experimental illustrations presented here suggest that it is possible to effectively apply sampling methods to tackle difficult problems involving nonlinear, non-Gaussian and time varying processes, typical of many sequential signal processing problems. For instance, it was shown that SMC methods provide an elegant solution to the logistic classification problem, where the target distributions cannot be characterised appropriately using Gaussian approximation.

The chapter also proposed an efficient optimisation algorithm, which combines gradient information of the error function with probabilistic sampling information. If the goal is to find the modes of the posterior, the simulations seem to indicate that the HySIR performs better than EKF and conventional SIR approaches. HySIR can be interpreted as a Gaussian mixture filter, in that only a few sampling trajectories need to be employed. Yet, as the number of trajectories increases, the computational requirements increase only linearly. Therefore, the method is also suitable as a sampling strategy for approximating multi-modal distributions.

The importance of the MCMC step was emphasised, not only because it reduces the resampling variance, but also because it allows one to explore the state space efficiently. The following chapter will show how this step can be used to design variable dimension SMC algorithms.

7

Sequential Bayesian Model Selection

This chapter[1] develops a sequential Monte Carlo method to perform Bayesian model selection. This on-line simulation-based estimation scheme combines sequential importance sampling, a selection scheme and reversible jump MCMC moves. It makes it possible to run a single algorithm which concentrates itself on the model orders of interest. This method is applied to the problem of estimating, on-line, the parameters, noise variance, and number of neurons of a hybrid model comprising a linear model and a radial basis function (RBF) network. A full Bayesian prior that allows one to integrate the basis coefficients and linear regression parameters is proposed. As a result of this, an efficient Rao-Blackewellised algorithm, that computes the coefficients and linear regression parameters using a stochastic bank of Kalman filters, is obtained.

The chapter is organised as follows. Section 7.1 introduces the model selection problem and describes the approximation and probabilistic models. It also presents the estimation objectives. Section 7.2 proposes a generic simulation-based method to perform sequential Bayesian estimation and model order selection simultaneously and discusses the related implementation issues. Section 7.3 applies the proposed algorithm to sequential Bayesian model order selection of hybrid linear-RBF models observed in Gaussian noise. The experimental results are presented in Section 7.4. Finally, some conclusions are drawn in Section 7.5.

7.1 Problem Statement

A sequential data processing scenario, where the model can change at each time step, is considered. In particular, the model active at time t will be denoted $\mathcal{M}_{t,k_t}$, where k_t corresponds to its current dimension. Examples include neural network models

[1] This chapter would not have been possible without the help of Christophe Andrieu and Arnaud Doucet. I am very thankful to them.

where the number of neurons changes over time and autoregressive (AR) processes where the number of lags also varies with time (Andrieu et al., 1999e). However, the framework is more general than the one of model order selection in embedded models (see Appendix D). As the subscript t indicates, one can have different varieties of models in this formulation. For instance, one could easily treat the problem of neural networks where the type of basis functions changes over time.

For each model, at time t, there is a corresponding set of unknown parameters $\boldsymbol{\varphi}_{k_t,t} \in \boldsymbol{\Omega}_{k_t,t}$. For example, these parameters could correspond to the weights of a neural network or a set of AR coefficients. The subscript in k_t is used to emphasise that the dimension of the parameter vector (k_t) changes with time.

If the dimension k_t takes an integer value between 0 and $k_{\max}$ at each time step, then there are $(k_{max}+1)^{t+1}$ possible model trajectories between time 0 and time t. The finite set of candidate models for this period can be described with $\mathcal{M}_{0:t,k_{0:t}}$, $k_{0:t} \in \{0,\ldots,k_{max}\}^{t+1}$, where $k_{0:t} \triangleq \{k_0,k_1,\ldots,k_t\}$. The corresponding set of parameters is given by $\boldsymbol{\varphi}_{k_{0:t},0:t} \in \boldsymbol{\Omega}_{k_{0:t},0:t}$.

It is assumed that these parameters and their number are random quantities with given prior distributions. It is possible as a result to define the following joint distribution for these quantities given the data $\mathbf{d}_{1:t}$:

$$p(k_{0:t},\boldsymbol{\varphi}_{k_{0:t},0:t},\mathbf{d}_{1:t}) = p(\mathbf{d}_{1:t}|k_{0:t},\boldsymbol{\varphi}_{k_{0:t},0:t})p(\boldsymbol{\varphi}_{k_{0:t},0:t}|k_{0:t})p(k_{0:t})$$

where $p(\mathbf{d}_{1:t}|k_{0:t},\boldsymbol{\varphi}_{k_{0:t},0:t})$ is the likelihood function, $p(\boldsymbol{\varphi}_{k_{0:t},0:t}|k_{0:t})$ is the prior parameters distribution and $p(k_{0:t})$ is the prior model distribution. This general framework encompasses many types of signal models. For demonstration purposes, the problem of RBF models is considered.

7.1.1 Example: radial basis function networks

The approximation scheme consisting of a mixture of k radial basis functions and a linear regression term is adopted again in this chapter. The approximation model $\mathcal{M}$ for the observations at time t is:

$$\begin{array}{lcl} \mathcal{M}_{t,0}: & \mathbf{y}_t = \boldsymbol{b}_t + \boldsymbol{\beta}_t'\mathbf{x}_t + \mathbf{n}_t & k_t = 0 \\ \mathcal{M}_{t,k_t}: & \mathbf{y}_t = \sum_{j=1}^{k_t} \mathbf{a}_{j,t}\phi(\|\mathbf{x}_t - \boldsymbol{\mu}_{j,t}\|) + \boldsymbol{b}_t + \boldsymbol{\beta}_t'\mathbf{x}_t + \mathbf{n}_t & k_t \geq 1 \end{array} \quad (7.1)$$

where $\|\cdot\|$ denotes a distance metric (usually Euclidean or Mahalanobis), $\boldsymbol{\mu}_{j,t} \in \mathbb{R}^d$ denotes the j-th RBF centre for a model with k RBFs, $\mathbf{a}_{j,t} \in \mathbb{R}^c$ denotes the j-th RBF amplitude and $\boldsymbol{b}_t \in \mathbb{R}^c$ and $\boldsymbol{\beta}_t \in \mathbb{R}^{d\times c}$ denote the linear regression parameters. The noise sequence $\mathbf{n}_t \in \mathbb{R}^c$ is assumed to be zero-mean Gaussian; its variance changes over time. Note that the parameters $\boldsymbol{b}_t$, $\boldsymbol{\beta}_t$ and $\mathbf{n}_t$ on k_t are affected by the value of k_t.

For convenience, the approximation model is expressed in vector-matrix form[2]:

$$\mathbf{y}_t = \mathbf{D}(\boldsymbol{\mu}_{1:k_t,t}, \mathbf{x}_t)\boldsymbol{\alpha}_{1:1+d+k_t,t} + \mathbf{n}_t$$

where:

$$\begin{aligned}
\mathbf{y}_t &= [\mathbf{y}_{1,t} \cdots \mathbf{y}_{c,t}] \\
\mathbf{D}(\boldsymbol{\mu}_{1:k_t,t}, \mathbf{x}_t) &= [1 \quad \mathbf{x}_{1,t} \cdots \mathbf{x}_{d,t} \quad \phi(\mathbf{x}_{1:d,t}, \boldsymbol{\mu}_{1:d,1,t}) \cdots \phi(\mathbf{x}_{1:d,t}, \boldsymbol{\mu}_{1:d,k_t,t})] \\
\boldsymbol{\alpha}_{1:c,1:1+d+k_t,t} &= \begin{bmatrix} \boldsymbol{b}_{1,t} & \boldsymbol{\beta}_{1,1,t} \cdots \boldsymbol{\beta}_{d,1,t} & \mathbf{a}_{1,1,t} \cdots \mathbf{a}_{1,k_t,t} \\ \boldsymbol{b}_{2,t} & \boldsymbol{\beta}_{1,2,t} \cdots \boldsymbol{\beta}_{d,2,t} & \mathbf{a}_{2,1,t} \cdots \mathbf{a}_{2,k_t,t} \\ & \vdots & \\ \boldsymbol{b}_{c,t} & \boldsymbol{\beta}_{1,c,t} \cdots \boldsymbol{\beta}_{d,c,t} & \mathbf{a}_{c,1,t} \cdots \mathbf{a}_{c,k_t,t} \end{bmatrix}'
\end{aligned}$$

and, finally, the noise process is assumed to be normally distributed as follows:

$$\mathbf{n}_t \sim \mathcal{N}\left(\mathbf{0}_{c\times 1}, diag\left(\boldsymbol{\sigma}^2_{1,t}, \ldots, \boldsymbol{\sigma}^2_{c,t}\right)\right)$$

The number k_t of RBFs and their parameters $\boldsymbol{\theta}_{k_t,t} \triangleq \{\boldsymbol{\alpha}_{1:c,1:m_t,t}, \boldsymbol{\mu}_{1:d,1:k_t,t}, \boldsymbol{\sigma}^2_{1:c,t}\}$, with $m_t = 1 + d + k_t$, are assumed to be unknown and need to be estimated sequentially.

7.1.2 Bayesian model

A state space representation is adopted to describe the network's evolution over time:

$$\begin{aligned}
\mathbf{y}_t &= \mathbf{D}(\boldsymbol{\mu}_{1:k_t,t}, \mathbf{x}_t)\boldsymbol{\alpha}_{1:m_t,t} + \mathbf{n}_t && (7.2) \\
k_{t+1} &= k_t + \epsilon_k && (7.3) \\
\boldsymbol{\mu}_{1:k_{t+1},t+1} &= \boldsymbol{\mu}_{1:k_{t+1},t} + \epsilon_{\boldsymbol{\mu}} && (7.4) \\
\boldsymbol{\alpha}_{1:m_{t+1},t+1} &= \boldsymbol{\alpha}_{1:m_{t+1},t} + \epsilon_{\boldsymbol{\alpha}} && (7.5) \\
\log(\boldsymbol{\sigma}^2_{t+1}) &= \log(\boldsymbol{\sigma}^2_t) + \epsilon_{\boldsymbol{\sigma}} && (7.6)
\end{aligned}$$

where the diffusion process ϵ_k is sampled from a discrete distribution:

$$\epsilon_k \sim p(k_{t+1}|k_t)$$

The other processes are sampled from normal distributions:

$$\begin{aligned}
\epsilon_{\boldsymbol{\mu}} &\sim \mathcal{N}(0, \delta^2_{\boldsymbol{\mu}}\mathbf{I}_d) \\
\epsilon_{\boldsymbol{\alpha}} &\sim \mathcal{N}(0, \delta^2_{\boldsymbol{\alpha}}\mathbf{I}_{m_t}) \\
\epsilon_{\boldsymbol{\sigma}} &\sim \mathcal{N}(0, \delta^2_{\boldsymbol{\sigma}}\mathbf{I}_c)
\end{aligned}$$

[2] The notation $\mathbf{y}_{1:c,t} \triangleq (\mathbf{y}_{1,t}, \mathbf{y}_{2,t}, \ldots, \mathbf{y}_{c,t})'$ is once again used to denote all the output observations at time t. To simplify this notation, $\mathbf{y}_t$ is equivalent to $\mathbf{y}_{1:c,t}$. That is, if one index does not appear, it is implied that we are referring to all of its possible values.

where $m_t = k_t + d + 1$ and $\mathbf{I}_m$ denotes the identity matrix of size $m \times m$. Note that if k_t changes, the dimensions of $\boldsymbol{\alpha}_{1:m_t,t}$ and $\boldsymbol{\mu}_{1:k_t,t}$ also change accordingly. In the example of RBFs, if k_t increases, new basis centres $\boldsymbol{\mu}_t$ are sampled from a Gaussian distribution centred at the data $\mathbf{x}_t$. That is, one stochastically places basis functions where there is more information. On the other hand, if k_t decreases, one samples uniformly from the existing set of basis functions to decide which bases should be deleted.

The joint distribution of the proposed model, with $\boldsymbol{\theta}_{k_{0:t},0:t} \triangleq \{\boldsymbol{\alpha}_{1:m_{0:t},0:t}, \boldsymbol{\mu}_{1:k_{0:t},0:t}, \boldsymbol{\sigma}^2_{0:t}\}$, is:

$$p(k_{0:t}, \boldsymbol{\theta}_{k_{0:t},0:t}, \mathbf{x}_{1:t}, \mathbf{y}_{1:t}) \propto p(\mathbf{y}_{1:t}|k_{0:t}, \boldsymbol{\theta}_{k_{0:t},0:t}, \mathbf{x}_{1:t}) p(\boldsymbol{\mu}_{1:k_{0:t},0:t}) p(\boldsymbol{\alpha}_{1:m_{0:t},0:t}) p(\boldsymbol{\sigma}^2_{0:t}) p(k_{0:t})$$

$$= \left[\prod_{l=1}^{t} p(\mathbf{y}_l|k_l, \boldsymbol{\theta}_{k_l,l}, \mathbf{x}_l) p(\boldsymbol{\mu}_{1:k_l,l}|\boldsymbol{\mu}_{1:k_{l-1},l-1}) p(\boldsymbol{\alpha}_{1:m_l,l}|\boldsymbol{\alpha}_{1:m_{l-1},l-1}) p(\boldsymbol{\sigma}^2_l|\boldsymbol{\sigma}^2_{l-1}) p(k_l|k_{l-1})\right]$$

$$\times\, p(\boldsymbol{\mu}_{k_0,0}) p(\boldsymbol{\alpha}_{m_0,0}) p(\boldsymbol{\sigma}^2_0) p(k_0) \tag{7.7}$$

Multivariate normal distributions are adopted to represent the priors $p(\boldsymbol{\mu}_{k_0,0})$ and $p(\boldsymbol{\alpha}_{k_0,0})$, a uniform distribution for $p(\boldsymbol{\sigma}^2_{k_0,0})$ and a discrete uniform distribution for $p(k_0)$. The parameters of these priors should reflect our degree of belief in the initial values of the various quantities of interest (Bernardo and Smith, 1994).

In our model, it is clear that $\boldsymbol{\varphi}_{k_{0:t},0:t} = \boldsymbol{\theta}_{k_{0:t},0:t}$ and:

$$\boldsymbol{\Omega}_{k_{0:t},0:t} \triangleq (\mathbb{R}^{d+1+S_{k_t}})^c \times (\mathbb{R}^{S_{k_t}})^d \times \mathbb{R}^{c(t+1)}$$

where $\boldsymbol{\alpha}_{1:c,1:m_{0:t},0:t} \in (\mathbb{R}^{d+1+S_{k_t}})^c$, $\boldsymbol{\mu}_{1:d,1:k_{0:t},0:t} \in (\mathbb{R}^{S_{k_t}})^d$, $\boldsymbol{\sigma}^2_{1:c,0:t} \in \mathbb{R}^{c(t+1)}$ and $S_{k_t} = \sum_{l=0}^{t} k_l$. Figure 7.1 shows the directed acyclic graphical model representation of the joint distribution for the first four time steps.

7.1.3 Estimation objectives

Our aim is to estimate the joint posterior distribution $p(k_{0:t}, \boldsymbol{\varphi}_{k_{0:t},0:t}|\mathbf{d}_{1:t})$, defined on the space $\boldsymbol{\Omega}_{0:t} = \bigcup_{l=1}^{(k_{max}+1)^{t+1}} \boldsymbol{\Omega}_{k^{[l]}_{0:t},0:t} \times \{k^{[l]}_{0:t}\}$, recursively in time. Note that $k^{[l]}_{0:t}$ represents the l-th possible model trajectory for an arbitrary ordering. According to Bayes' rule, the posterior distribution $p(k_{0:t}, \boldsymbol{\varphi}_{k_{0:t},0:t}|\mathbf{d}_{1:t})$ is given by:

$$\frac{p(\mathbf{d}_t|\mathbf{d}_{1:t-1}, \boldsymbol{\varphi}_{k_{0:t},0:t}, k_{0:t}) p(k_{0:t}, \boldsymbol{\varphi}_{k_{0:t},0:t}|\mathbf{d}_{1:t-1})}{\sum_{l=1}^{(k_{max}+1)^{t+1}} \int_{\boldsymbol{\Omega}_{k^{[l]}_{0:t},0:t}} p(\mathbf{d}_t|\mathbf{d}_{1:t-1}, \boldsymbol{\varphi}_{k^{[l]}_{0:t},0:t}, k^{[l]}_{0:t}) p(k^{[l]}_{0:t}, \boldsymbol{\varphi}_{k^{[l]}_{0:t},0:t}|\mathbf{d}_{1:t-1}) d\boldsymbol{\varphi}_{k^{[l]}_{0:t},0:t}}$$

It is, in most occasions, impossible to derive closed-form analytical expressions for this distribution and its features of interest, such as the posterior model probabilities $p(k_{0:t}|\mathbf{d}_{1:t})$. As a result, one has to resort to computational methods in most situations.

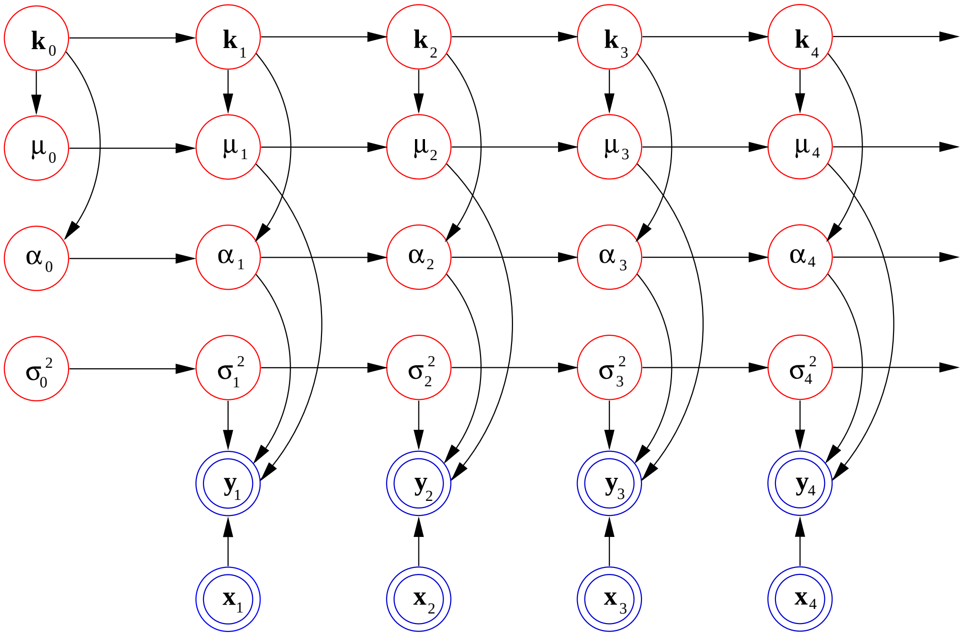

Figure 7.1 *Dynamical directed acyclic graphical model. The circles represent unknowns, while the double circles correspond to the data. The hyper-parameters $\delta_{\boldsymbol{\mu}}^2$, $\delta_{\boldsymbol{\alpha}}^2$, $\delta_{\boldsymbol{\sigma}}^2$ and δ_k have been omitted for clarity.*

7.2 Sequential Monte Carlo

With a few exceptions (Andrieu et al., 1999e; Djurić, 1999), most existing SMC methods belong to a framework where the model order is assumed to be known. Here, a Bayesian importance sampling method is presented to deal with the more general case of unknown model dimension. To accomplish this goal in an efficient way, reversible jump MCMC steps are introduced into the algorithm. The next subsection presents a generic version of the algorithm.

7.2.1 Generic SMC algorithm

Given N particles $\{(\boldsymbol{\varphi}_{k_{0:t-1},0:t-1}^{(i)}, k_{0:t-1}^{(i)}); i \in \{1, \ldots, N\}\}$ at time $t-1$, approximately distributed according to $p(\boldsymbol{\varphi}_{k_{0:t-1},0:t-1}, k_{0:t-1}|\mathbf{d}_{1:t-1})$, the algorithm allows us to compute N particles $(\boldsymbol{\varphi}_{k_{0:t},0:t}^{(i)}, k_{0:t}^{(i)})$, approximately distributed according to the posterior $p(\boldsymbol{\varphi}_{k_{0:t},0:t}, k_{0:t}|\mathbf{d}_{1:t})$, at time t. Before presenting the algorithm, let us first introduce the quantities $(\boldsymbol{\varphi}_{k_t,-t}, k_t)$ to denote the set of variables in $(\boldsymbol{\varphi}_{k_{0:t},0:t}, k_t)$, but not in $(\boldsymbol{\varphi}_{k_{0:t-1},0:t-1}, k_{t-1})$ (note that this set can be empty for some problems). The importance function for this set will be denoted $q(\varphi_{k_t,-t}, k_t|\varphi_{k_{0:t-1},0:t-1}, k_{0:t-1}, \mathbf{d}_{1:t})$. This function is such that the support of the distribution $p(\boldsymbol{\varphi}_{k_{0:t},0:t}, k_{0:t}|\mathbf{d}_{1:t})$ is included in the support of the product $p(\boldsymbol{\varphi}_{k_{0:t-1},0:t-1}, k_{0:t-1}|\mathbf{d}_{1:t-1})\; q(\boldsymbol{\varphi}_{k_t,-t}, k_t|\boldsymbol{\varphi}_{k_{0:t-1},0:t-1}, k_{0:t-1}, \mathbf{d}_{1:t})$.

The algorithm proceeds as follows at time t:

SMC Algorithm for Bayesian Model Selection

1. Bayesian importance sampling step

   - For $i = 1, \ldots, N$, sample:

   $$(\widehat{\boldsymbol{\varphi}}^{(i)}_{\widehat{k}_t,-t}, \widehat{k}^{(i)}_t) \sim q(\boldsymbol{\varphi}_{k_t,-t}, k_t | \boldsymbol{\varphi}^{(i)}_{k_{0:t-1},0:t-1}, k^{(i)}_{0:t-1}, \mathbf{d}_{1:t})$$

   and set:

   $$(\widehat{\boldsymbol{\varphi}}^{(i)}_{\widehat{k}_{0:t},0:t}, \widehat{k}^{(i)}_{0:t}) \triangleq \left((\boldsymbol{\varphi}^{(i)}_{k_{0:t-1},0:t-1}, k^{(i)}_{0:t-1}), (\widehat{\boldsymbol{\varphi}}^{(i)}_{\widehat{k}_t,-t}, \widehat{k}^{(i)}_t)\right)$$

   - For $i = 1, \ldots, N$, evaluate the importance weights up to a normalising constant:

   $$w^{(i)}_t = \frac{p(\widehat{\boldsymbol{\varphi}}^{(i)}_{\widehat{k}_{0:t},0:t}, \widehat{k}^{(i)}_{0:t} | \mathbf{d}_{1:t})}{p(\widehat{\boldsymbol{\varphi}}^{(i)}_{\widehat{k}_{0:t-1},0:t-1}, \widehat{k}^{(i)}_{0:t-1} | \mathbf{d}_{1:t-1}) q(\widehat{\boldsymbol{\varphi}}^{(i)}_{\widehat{k}_t,-t}, \widehat{k}^{(i)}_t | \boldsymbol{\varphi}^{(i)}_{k_{0:t-1},0:t-1}, k^{(i)}_{0:t-1}, \mathbf{d}_{1:t})} \tag{7.8}$$

   - For $i = 1, \ldots, N$, normalise the importance weights:

   $$\widetilde{w}^{(i)}_t = w^{(i)}_t \left[\sum_{j=1}^{N} w^{(j)}_t\right]^{-1} \tag{7.9}$$

2. Selection step

   - Multiply/Suppress samples $(\widehat{\boldsymbol{\varphi}}^{(i)}_{\widehat{k}_{0:t},0:t}, \widehat{k}^{(i)}_{0:t})$ with high/low importance weights $\widetilde{w}^{(i)}_t$, respectively, to obtain N random samples $(\widetilde{\boldsymbol{\varphi}}^{(i)}_{\widetilde{k}_{0:t},0:t}, \widetilde{k}^{(i)}_{0:t})$ approximately distributed according to $p(\widetilde{\boldsymbol{\varphi}}^{(i)}_{\widetilde{k}_{0:t},0:t}, \widetilde{k}^{(i)}_{0:t} | \mathbf{d}_{1:t})$.

3. MCMC step

   - Apply a Markov transition kernel with invariant distribution given by the product $\prod_{i=1}^{N} p(\boldsymbol{\varphi}^{(i)}_{k_{0:t},0:t}, k^{(i)}_{0:t} | \mathbf{d}_{1:t})$ to obtain $(\boldsymbol{\varphi}^{(i)}_{k_{0:t},0:t}, k^{(i)}_{0:t})$.

■

7.2.2 Implementation issues

7.2.2.1 Bayesian importance sampling step

As mentioned earlier, Bayesian importance sampling has been formulated mainly in scenarios where the distribution from which one wishes to sample is defined on a simple set with a unique dominating measure (Geweke, 1989). For the sake of clarity, this section describes the more general case that arises when the target distribution is defined on a union of distinct subspaces, that is:

$$\pi(\boldsymbol{\varphi}_{k_{0:t},0:t}, k_{0:t}) = \sum_{l=1}^{(k_{max}+1)^{t+1}} \pi_{k_{0:t}^{[l]}}(\boldsymbol{\varphi}_{k_{0:t}^{[l]},0:t}, k_{0:t}^{[l]}) \mathbb{I}_{\boldsymbol{\Omega}_{k_{0:t}^{[l]},0:t} \times \{k_{0:t}^{[l]}\}}(\boldsymbol{\varphi}_{k_{0:t},0:t}, k_{0:t})$$

Let us introduce the following importance distribution:

$$q(\boldsymbol{\varphi}_{k_{0:t},0:t}, k_{0:t}) = \sum_{l=1}^{(k_{max}+1)^{t+1}} q_{k_{0:t}^{[l]}}(\boldsymbol{\varphi}_{k_{0:t}^{[l]},0:t}, k_{0:t}^{[l]}) \mathbb{I}_{\boldsymbol{\Omega}_{k_{0:t}^{[l]},0:t} \times \{k_{0:t}^{[l]}\}}(\boldsymbol{\varphi}_{k_{0:t},0:t}, k_{0:t})$$

where the support of $q_{k_{0:t}^{[l]}}(\boldsymbol{\varphi}_{k_{0:t}^{[l]},0:t}, k_{0:t}^{[l]})$ includes the support of $\pi_{k_{0:t}^{[l]}}(\boldsymbol{\varphi}_{k_{0:t}^{[l]},0:t}, k_{0:t}^{[l]})$ for $l = 1, \dots, (k_{max}+1)^{t+1}$. Hence, $\pi(\boldsymbol{\varphi}_{k_{0:t},0:t}, k_{0:t})$ can be rewritten as:

$$\frac{\sum_{l=1}^{(k_{max}+1)^{t+1}} w_{k_{0:t}^{[l]}}(\boldsymbol{\varphi}_{k_{0:t}^{[l]},0:t}, k_{0:t}^{[l]}) q_{k_{0:t}^{[l]}}(\boldsymbol{\varphi}_{k_{0:t}^{[l]},0:t}, k_{0:t}^{[l]}) \mathbb{I}_{\boldsymbol{\Omega}_{k_{0:t}^{[l]},0:t} \times \{k_{0:t}^{[l]}\}}(\boldsymbol{\varphi}_{k_{0:t},0:t}, k_{0:t})}{\sum_{l=1}^{(k_{max}+1)^{t+1}} \int_{\boldsymbol{\Omega}_{k_{0:t}^{[l]},0:t}} w_{k_{0:t}^{[l]}}(\boldsymbol{\varphi}_{k_{0:t}^{[l]},0:t}, k_{0:t}^{[l]}) q_{k_{0:t}^{[l]}}(\boldsymbol{\varphi}_{k_{0:t}^{[l]},0:t}, k_{0:t}^{[l]}) \mathrm{d}\boldsymbol{\varphi}_{k_{0:t}^{[l]},0:t}}$$

where

$$w_{k_{0:t}^{[l]}}(\boldsymbol{\varphi}_{k_{0:t}^{[l]},0:t}, k_{0:t}^{[l]}) = \frac{\pi_{k_{0:t}^{[l]}}(\boldsymbol{\varphi}_{k_{0:t}^{[l]},0:t}, k_{0:t}^{[l]})}{q_{k_{0:t}^{[l]}}(\boldsymbol{\varphi}_{k_{0:t}^{[l]},0:t}, k_{0:t}^{[l]})}$$

represents the unnormalised importance ratios. Let us assume that we have N samples $(\boldsymbol{\varphi}_{k_{0:t},0:t}^{(i)}, k_{0:t}^{(i)})$ distributed according to $q(\boldsymbol{\varphi}_{k_{0:t},0:t}, k_{0:t})$, then the Bayesian importance sampling method gives us the following point mass approximation of $\pi(\boldsymbol{\varphi}_{k_{0:t},0:t}, k_{0:t})$:

$$\widehat{\pi}(d\boldsymbol{\varphi}_{k_{0:t},0:t}, k_{0:t}) = \frac{\sum_{l=1}^{(k_{max}+1)^{t+1}} \sum_{I^{\star}_{k_{0:t}^{[l]}}} w_{k_{0:t}^{[l],(i)}}(\boldsymbol{\varphi}^{(i)}_{k_{0:t}^{[l],(i)},0:t}, k_{0:t}^{[l],(i)}) \delta_{\boldsymbol{\varphi}^{(i)}_{k_{0:t}^{(i)},0:t}, k_{0:t}^{(i)}}(\mathrm{d}\boldsymbol{\varphi}_{k_{0:t},0:t}, k_{0:t})}{\sum_{l=1}^{(k_{max}+1)^{t+1}} \sum_{I^{\star}_{k_{0:t}^{[l]}}} w_{k_{0:t}^{[l],(i)}}(\boldsymbol{\varphi}^{(i)}_{k_{0:t}^{[l],(i)},0:t}, k_{0:t}^{[l],(i)})} \tag{7.10}$$

with $I^{\star}_{k_{0:t}^{[l]}} = \{i \in \{1, \dots, N\} / (\boldsymbol{\varphi}^{(i)}_{k_{0:t},0:t}, k_{0:t}^{(i)}) \in \boldsymbol{\Omega}_{k_{0:t}^{[l]},0:t} \times k_{0:t}^{[l]}\}$. Equations (7.8) and (7.9) follow.

7.2.2.2 Selection step

As discussed in the previous chapter, a selection scheme associates to each particle $(\widehat{\boldsymbol{\varphi}}^{(i)}_{\widehat{k}_{0:t},0:t}, \widehat{k}^{(i)}_{0:t})$ a number of "children", say $N_i \in \mathbb{N}$, such that $\sum_{i=1}^{N} N_i = N$. One can use multinomial, residual or systematic sampling to obtain the set of resampled particles: $((\widetilde{\boldsymbol{\varphi}}^{(i)}_{\widetilde{k}_{0:t},0:t}, \widetilde{k}^{(i)}_{0:t}); i = 1, \dots, N)$.

7.2.2.3 MCMC steps

After the selection scheme at time t, we obtain N particles distributed marginally approximately according to $p(k_{0:t}, \boldsymbol{\varphi}_{k_{0:t},0:t}|\mathbf{d}_{1:t})$. In practice, a skewed importance weights distribution implies that many particles have had no children, i.e. $N_i = 0$, whereas others have had a large number of children, the extreme case being $N_i = N$ for a particular value i. In this case, there is a severe depletion of samples. A strategy for improving the results consists of introducing MCMC steps of invariant distribution $p(k_{0:t}, \boldsymbol{\varphi}_{k_{0:t},0:t}|\mathbf{d}_{1:t})$ on each particle (Andrieu et al., 1999e; Carpenter et al., 1999; Doucet and Gordon, 1999; Gilks and Berzuini, 1998; MacEachern et al., 1999). The basic idea is that, by applying a Markov transition kernel, the total variation of the current distribution with respect to the invariant distribution can only decrease. Note, however, that we do not require this kernel to be ergodic. Convergence results for this type of algorithm are presented in (Gilks and Berzuini, 1998).

It is possible to generalise this idea and to introduce MCMC steps on the product space with an invariant distribution $\prod\limits_{i=1}^{N} p(\boldsymbol{\varphi}^{(i)}_{k_{0:t},0:t}, k^{(i)}_{0:t}|\mathbf{y}_{1:t})$, that is apply MCMC steps on the *whole population* of particles. In doing so, it allows us to introduce the algorithms developed in parallel MCMC computation. This chapter is limited to the use of simple MCMC transitions steps on each particle. In particular, reversible jump MCMC steps (Green, 1995) are proposed to allow the particles to move from one subspace to the others.

7.3 SMC Applied to RBF Networks

This section shows that it is possible to integrate the nuisance parameters $\boldsymbol{\alpha}_{1:m_t,t}$, leading to a more efficient algorithm. Subsequently, it introduces the reversible jump MCMC algorithm in the context of sequential sampling and describes the SMC algorithm for RBFs. Finally, it presents the implementation details for the sampling, selection and MCMC steps.

In the RBF signal model, $(\boldsymbol{\varphi}_{k_t,-t}, k_t) = (\boldsymbol{\theta}_{k_t,t}, k_t) = (\{\boldsymbol{\alpha}_{1:m_t,t}, \boldsymbol{\mu}_{1:k_t,t}, \boldsymbol{\sigma}^2_t\}, k_t)$. It is important to note that for the parameters $\boldsymbol{\alpha}_{1:m_t,t}$, conditionally upon $(\boldsymbol{\mu}_{1:k_t,t}, \boldsymbol{\sigma}^2_t\}, k_t)$, we have a linear Gaussian state space model. Consequently, it is possible to integrate out $\boldsymbol{\alpha}_{1:m_t,t}$, thereby reducing the variance of the estimates (Doucet et al., 1999, Propositions 1 and 2). The end result corresponds to a bank of Kalman filters for each network output[3] (Jazwinski, 1970; Kalman and Bucy, 1961):

$$K_t^{(i)}=(P^{(i)}_{k_t,t-1}+\delta^2_{\boldsymbol{\alpha}}\mathbf{I}_{m_t})\mathbf{D}'(\boldsymbol{\mu}_{1:k_t,t},\mathbf{x}_t)\left[\boldsymbol{\sigma}_t^2+\mathbf{D}(\boldsymbol{\mu}_{1:k_t,t},\mathbf{x}_t)(P^{(i)}_{k_t,t-1}+\delta^2_{\boldsymbol{\alpha}}\mathbf{I}_{m_t})\mathbf{D}'(\boldsymbol{\mu}_{1:k_t,t},\mathbf{x}_t)\right]^{-1} \tag{7.11}$$

$$\bar{\boldsymbol{\alpha}}^{(i)}_{1:m_t,t} = \bar{\boldsymbol{\alpha}}^{(i)}_{1:m_t,t-1} + K_t^{(i)}\left(\mathbf{y}_t - \mathbf{D}(\boldsymbol{\mu}_{1:k_t,t},\mathbf{x}_t)\bar{\boldsymbol{\alpha}}^{(i)}_{1:m_t,t-1}\right) \tag{7.12}$$

$$P^{(i)}_{k_t,t} = P^{(i)}_{k_t,t-1} + \delta^2_{\boldsymbol{\alpha}}\mathbf{I}_{m_t} - K_t^{(i)}\mathbf{D}(\boldsymbol{\mu}_{1:k_t,t},\mathbf{x}_t)(P^{(i)}_{k_t,t-1} + \delta^2_{\boldsymbol{\alpha}}\mathbf{I}_{m_t}) \tag{7.13}$$

where K_t denotes the Kalman gain and $\bar{\boldsymbol{\alpha}}_{1:m_t,t}$ and $P_{k_t,t}$ correspond to the conditional mean and covariance matrix of $\boldsymbol{\alpha}_{1:m_t,t}$. Note that the sizes of $\bar{\boldsymbol{\alpha}}_{1:m_t,t}$ and $P_{k_t,t}$ vary over time. If k_t decreases, then we should delete, at random, $k_{t-1} - k_t$ basis function coefficients and the corresponding row and column entries in the matrix $P_{k_{t-1},t-1}$. If k_t increases, say for example $k_t = k_{t-1} + 1$, we can extend $\bar{\boldsymbol{\alpha}}_{1:m_{t-1},t-1}$ and $P_{k_{t-1},t-1}$ as follows:

$$\bar{\boldsymbol{\alpha}}_{1:m_t,t-1} = \left[\bar{\boldsymbol{\alpha}}_{1:m_{t-1},t-1} \;\; \bar{\boldsymbol{\alpha}}_{1:m_0,0}\right]'$$

$$P_{k_t,t-1} = \begin{bmatrix} P_{k_{t-1},t-1} & 0 \\ 0 & P_{k_0,0} \end{bmatrix}$$

where $\bar{\boldsymbol{\alpha}}_{1:m_0,0}$ and $P_{k_0,0}$ depend on our prior knowledge about the data set being modelled. By integrating out $\boldsymbol{\alpha}_{1:m_t,t}$, we may select the following importance distribution:

$$q(\boldsymbol{\mu}_{1:k_t,t},\boldsymbol{\sigma}_t^2,k_t|\boldsymbol{\theta}_{k_{0:t-1},0:t-1},k_{0:t-1},\mathbf{d}_{1:t}) = p(\boldsymbol{\mu}_{1:k_t,t}|\boldsymbol{\mu}_{1:k_t,t-1},k_t)p(\boldsymbol{\sigma}_t^2|\boldsymbol{\sigma}_{t-1}^2)p(k_t|k_{t-1})$$

As pointed out in Chapter 5, standard MCMC methods are not able to "jump" between subspaces $\boldsymbol{\Omega}_{k_{0:t},0:t}$ of different dimension. However, recently, Green has introduced a flexible class of MCMC samplers, the so-called reversible jump MCMC, that are capable of jumping between subspaces of different dimensions (Green, 1995). Here, the chain must move across subspaces of different dimensions, and therefore the proposal distributions are more complex: see (Green, 1995; Richardson and Green, 1997) for details. For our problem, the following moves have been selected:

1. Birth of a new basis, *i.e.* proposing a new basis function.

2. Death of an existing basis, *i.e.* removing a basis function chosen randomly.

3. Update the RBF centres.

[3] Note that to simplify the notation, the Kalman filter equations are described for a single output. Thus $\boldsymbol{\sigma}_t^2$ is a scalar. In general, we would need to introduce a subindex $j = 1,\ldots,c$ (that is, write explicitly $\boldsymbol{\sigma}_{j,t}^2$, $\bar{\boldsymbol{\alpha}}_{j,1:m_t,t}$ and $P_{j,k_t,t}$) to indicate the j-th output of the neural network.

These moves are defined by heuristic considerations, the only condition to be fulfilled being to maintain the correct invariant distribution. A particular choice will only have influence on the convergence rate of the algorithm. The birth and death moves allow the network to grow from k_t to $k_t + 1$ and decrease from k_t to $k_t - 1$ respectively. By applying the bank of Kalman filters and incorporating reversible jump MCMC steps, the sequential model selection algorithm for RBF networks is as follows:

SMC Algorithm for RBFs Bayesian Model Selection

1. Sequential importance sampling step

   - For $i = 1, \ldots, N$, sample from the prior distributions:

$$\begin{array}{rcl} \widehat{k}_t^{(i)} | (\boldsymbol{\theta}_{k_{t-1},t-1}^{(i)}, k_{t-1}^{(i)}, \mathbf{d}_{1:t-1}) & \sim & p(k_t | k_{t-1}^{(i)}) \\ \widehat{\boldsymbol{\sigma}}_t^{2(i)} | (\widehat{k}_t^{(i)}, \boldsymbol{\theta}_{k_{t-1},t-1}^{(i)}, k_{t-1}^{(i)}, \mathbf{d}_{1:t-1}) & \sim & p(\boldsymbol{\sigma}_t^2 | \boldsymbol{\sigma}_{t-1}^{2(i)}) \\ \widehat{\boldsymbol{\mu}}_{1:k_t,t}^{(i)} | (\widehat{k}_t^{(i)}, \boldsymbol{\theta}_{k_{t-1},t-1}^{(i)}, k_{t-1}^{(i)}, \mathbf{d}_{1:t-1}) & \sim & p(\boldsymbol{\mu}_{1:k_t,t} | \boldsymbol{\mu}_{1:\widehat{k}_t,t-1}^{(i)}, \widehat{k}_t^{(i)}) \end{array}$$

   Change the dimension of $\bar{\boldsymbol{\alpha}}_{k_{t-1},t-1}$ and $P_{k_{t-1},t-1}$ to $\bar{\boldsymbol{\alpha}}_{\widehat{k}_t,t-1}$ and $P_{\widehat{k}_t,t-1}$ and set:

$$(\widehat{\boldsymbol{\theta}}_{\widehat{k}_{0:t},0:t}^{(i)}, \widehat{k}_{0:t}^{(i)}) \triangleq \left((\boldsymbol{\theta}_{k_{0:t-1},0:t-1}^{(i)}, k_{0:t-1}^{(i)}), (\widehat{\boldsymbol{\theta}}_{\widehat{k}_t,t}^{(i)}, \widehat{k}_t^{(i)}) \right)$$

   - For $i = 1, \ldots, N$, evaluate the importance weights up to a normalising constant:

$$w_t^{(i)} = p(\mathbf{y}_t | \widehat{\boldsymbol{\mu}}_{1:\widehat{k}_{0:t},0:t}^{(i)}, \widehat{\boldsymbol{\sigma}}_{0:t}^{2(i)}, \widehat{k}_{0:t}^{(i)}, \mathbf{x}_t, \mathbf{d}_{1:t-1})$$

   - For $i = 1, \ldots, N$, normalise the importance weights:

$$\widetilde{w}_t^{(i)} = w_t^{(i)} \left[\sum_{j=1}^{N} w_t^{(j)} \right]^{-1}$$

2. Selection step

   - Use residual sampling to multiply/suppress samples $(\widehat{\boldsymbol{\theta}}_{\widehat{k}_{0:t},0:t}^{(i)}, \widehat{k}_{0:t}^{(i)})$ with high/low importance weights $\widetilde{w}_t^{(i)}$, respectively, to obtain N random samples $(\widetilde{\boldsymbol{\theta}}_{\widetilde{k}_{0:t},0:t}^{(i)}, \widetilde{k}_{0:t}^{(i)})$ approximately distributed according to $p(\widetilde{\boldsymbol{\theta}}_{\widetilde{k}_{0:t},0:t}^{(i)}, \widetilde{k}_{0:t}^{(i)} | \mathbf{d}_{1:t})$.

3. MCMC step

   - Sample $u \sim \mathcal{U}_{[0,1]}$.

- If $(u \leq b_{k_t})$
  - then "birth" move (See Section 7.3.2.2).
  - else if $(u \leq b_{k_t} + d_{k_t})$ then "death" move (See Section 7.3.2.2).
  - else update $\widetilde{\boldsymbol{\mu}}_{1:\widetilde{k}_t,t}$ and $\widetilde{k}_t$ (See Section 7.3.2.1).

  End If.
- Update the parameters $\bar{\boldsymbol{\alpha}}^{(i)}_{1:m_t,t}$ and $P^{(i)}_{k_t,t}$ using a bank of Kalman filters (equations (7.11) to (7.13)).

■

7.3.1 Sampling and selection steps

In the prediction stage, samples from the importance function are obtained as follows:

$$\begin{aligned}
\widehat{k}_t|(\boldsymbol{\mu}_{1:k_{t-1},t-1}, \boldsymbol{\sigma}^2_{t-1}, k_{t-1}, \mathbf{d}_{1:t-1}) &\sim p(k_t|k_{t-1}) && (7.14)\\
\widehat{\boldsymbol{\sigma}}^2_t|(\widehat{k}_t, \boldsymbol{\mu}_{1:k_{t-1},t-1}, \boldsymbol{\sigma}^2_{t-1}, k_{t-1}, \mathbf{d}_{1:t-1}) &\sim p(\boldsymbol{\sigma}^2_t|\boldsymbol{\sigma}^2_{t-1}) && (7.15)\\
\widehat{\boldsymbol{\mu}}_{1:k_t,t}|(\widehat{k}_t, \boldsymbol{\mu}_{1:\widehat{k}_t,t-1}, \boldsymbol{\sigma}^2_{t-1}, k_{t-1}, \mathbf{d}_{1:t-1}) &\sim p(\boldsymbol{\mu}_{1:k_t,t}|\boldsymbol{\mu}_{1:\widehat{k}_t,t-1}, \widehat{k}_t) && (7.16)
\end{aligned}$$

Applying Bayes' rule, the posterior at time t is:

$$\begin{aligned}
&p(\boldsymbol{\mu}_{1:k_{0:t},0:t}, \boldsymbol{\sigma}^2_{0:t}, k_{0:t}|\mathbf{d}_{1:t}) \propto p(\mathbf{y}_t|\boldsymbol{\mu}_{1:k_{0:t},0:t}, \boldsymbol{\sigma}^2_{0:t}, k_{0:t}, \mathbf{x}_t, \mathbf{d}_{1:t-1})p(\boldsymbol{\mu}_{1:k_{0:t},0:t}, \boldsymbol{\sigma}^2_{0:t}, k_{0:t}|\mathbf{d}_{1:t-1})\\
&\quad = p(\mathbf{y}_t|\boldsymbol{\mu}_{1:k_{0:t},0:t}, \boldsymbol{\sigma}^2_t, k_{0:t}, \mathbf{x}_t, \mathbf{d}_{1:t-1})p(\boldsymbol{\mu}_{1:k_t,t}, \boldsymbol{\sigma}^2_t, k_t|\boldsymbol{\mu}_{1:k_{0:t-1},0:t-1}, \boldsymbol{\sigma}^2_{0:t-1}, k_{0:t-1}, \mathbf{d}_{1:t-1})\\
&\qquad \times p(\boldsymbol{\mu}_{1:k_{0:t-1},0:t-1}, \boldsymbol{\sigma}^2_{0:t-1}, k_{0:t-1}|\mathbf{d}_{1:t-1})\\
&\quad = p(\mathbf{y}_t|\boldsymbol{\mu}_{1:k_{0:t},0:t}, \boldsymbol{\sigma}^2_{0:t}, k_{0:t}, \mathbf{x}_t, \mathbf{d}_{1:t-1})p(\boldsymbol{\mu}_{1:k_t,t}|\boldsymbol{\mu}_{1:k_t,t-1}, k_t)p(\boldsymbol{\sigma}^2_t|\boldsymbol{\sigma}^2_{t-1})p(k_t|k_{t-1})\\
&\qquad \times p(\boldsymbol{\mu}_{1:k_{0:t-1},0:t-1}, \boldsymbol{\sigma}^2_{0:t-1}, k_{0:t-1}|\mathbf{d}_{1:t-1})
\end{aligned}$$

Hence, it follows that the unnormalised importance ratios w_t are given by:

$$\begin{aligned}
w_t &= \frac{p(\boldsymbol{\mu}_{1:k_{0:t},0:t}, \boldsymbol{\sigma}^2_{0:t}, k_{0:t}|\mathbf{d}_{1:t})}{q(\boldsymbol{\mu}_{1:k_t,t}, \boldsymbol{\sigma}^2_t, k_t|\boldsymbol{\theta}_{k_{0:t-1},0:t-1}, k_{0:t-1}, \mathbf{d}_{1:t})p(\boldsymbol{\mu}_{1:k_{0:t-1},0:t-1}, \boldsymbol{\sigma}^2_{0:t-1}, k_{0:t-1}|\mathbf{d}_{1:t-1})}\\
&\propto p(\mathbf{y}_t|\boldsymbol{\mu}_{1:k_{0:t},0:t}, \boldsymbol{\sigma}^2_{0:t}, k_{0:t}, \mathbf{x}_t, \mathbf{d}_{1:t-1}) \qquad (7.17)
\end{aligned}$$

That is, the importance ratios are the one-step-ahead densities (also known as the evidence or innovations) in the bank of Kalman filters. More precisely,

$$\begin{aligned}
&p(\mathbf{y}_t|\widehat{\boldsymbol{\mu}}^{(i)}_{1:\widehat{k}_{0:t},0:t}, \widehat{\boldsymbol{\sigma}}^{2(i)}_{0:t}, \widehat{k}^{(i)}_{0:t}, \mathbf{x}_t, \mathbf{d}_{1:t-1})\\
&= \mathcal{N}\left(\mathbf{D}(\widehat{\boldsymbol{\mu}}^{(i)}_{1:\widehat{k}_t,t}, \mathbf{x}_t)\bar{\boldsymbol{\alpha}}^{(i)}_{1:\widehat{m}_t,t-1}, \widehat{\boldsymbol{\sigma}}^{(i)2}_t + \mathbf{D}(\widehat{\boldsymbol{\mu}}^{(i)}_{1:\widehat{k}_t,t}, \mathbf{x}_t)(P^{(i)}_{\widehat{k}_t,t-1} + \delta^2_{\boldsymbol{\alpha}}\mathbf{I}_{m_t})\mathbf{D}'(\widehat{\boldsymbol{\mu}}^{(i)}_{1:\widehat{k}_t,t}, \mathbf{x}_t)\right)
\end{aligned}$$

(7.18)

The selection step is done according to the residual resampling scheme.

7.3.2 MCMC steps

For our problem, the target distribution is defined as:

$$p\left(k_t, \boldsymbol{\mu}_{1:k_t,t}\middle|\, \mathbf{x}_{1:t}, \mathbf{y}_{1:t}, k_{1:t-1}, \boldsymbol{\mu}_{1:k_{t-1},1:t-1}, \boldsymbol{\sigma}^2_{1:c,1:t}\right) \tag{7.19}$$

That is, the past trajectories are kept fixed to reduce the computational requirements. Note, however, that the methodology is more general and that it is possible to sample from the joint distribution of the trajectories $p\left(k_{1:t}, \boldsymbol{\mu}_{1:k_{1:t},1:t}, \boldsymbol{\sigma}^2_{1:c,1:t}\middle|\, \mathbf{x}_{1:t}, \mathbf{y}_{1:t}\right)$. The target distribution is defined on a union of disconnected spaces, $\left\{\widetilde{k}_{t-1}-1\right\} \times \boldsymbol{\Omega}_{\widetilde{k}_{t-1}-1} \cup \left\{\widetilde{k}_{t-1}\right\} \times \boldsymbol{\Omega}_{\widetilde{k}_{t-1}} \cup \left\{\widetilde{k}_{t-1}+1\right\} \times \boldsymbol{\Omega}_{\widetilde{k}_{t-1}+1}$, and admits the following expression:

$$\begin{aligned}
& p\left(k_t, \boldsymbol{\mu}_{1:k_t,t}\middle|\, \mathbf{x}_{1:t}, \mathbf{y}_{1:t}, \widetilde{k}_{1:t-1}, \boldsymbol{\mu}_{1:\widetilde{k}_{t-1},1:t-1}, \boldsymbol{\sigma}^2_{1:c,1:t}\right) \\
= \; & \mathbb{I}_{\left\{\widetilde{k}_{t-1}+1\right\}}(k_t)\, p_1\left(\widetilde{k}_{t-1}+1, \boldsymbol{\mu}_{1:k_{t-1}+1,t}\middle|\, \mathbf{x}_{1:t}, \mathbf{y}_{1:t}, \widetilde{k}_{1:t-1}, \boldsymbol{\mu}_{1:\widetilde{k}_{t-1},1:t-1}, \boldsymbol{\sigma}^2_{1:c,1:t}\right) \\
& + \mathbb{I}_{\left\{\widetilde{k}_{t-1}\right\}}(k_t)\, p_2\left(\widetilde{k}_{t-1}, \boldsymbol{\mu}_{1:\widetilde{k}_{t-1},t}\middle|\, \mathbf{x}_{1:t}, \mathbf{y}_{1:t}, \widetilde{k}_{1:t-1}, \boldsymbol{\mu}_{1:\widetilde{k}_{t-1},1:t-1}, \boldsymbol{\sigma}^2_{1:c,1:t}\right) \\
& + \mathbb{I}_{\left\{\widetilde{k}_{t-1}-1\right\}}(k_t)\, p_3\left(\widetilde{k}_{t-1}-1, \boldsymbol{\mu}_{1:\widetilde{k}_{t-1}-1,t}\middle|\, \mathbf{x}_{1:t}, \mathbf{y}_{1:t}, \widetilde{k}_{1:t-1}, \boldsymbol{\mu}_{1:\widetilde{k}_{t-1},1:t-1}, \boldsymbol{\sigma}^2_{1:c,1:t}\right)
\end{aligned} \tag{7.20}$$

To sample from this distribution over different models, we have to resort to reversible jump MCMC simulation. In particular, each particle is updated independently using reversible jump MCMC moves. The transition kernel is a mixture of different transition kernels. At each iteration, one of the candidate moves (birth, death or update) is randomly chosen[4]. The probabilities for choosing these moves are b_{k_t}, d_{k_t} and u_{k_t} respectively, such that $b_{k_t} + d_{k_t} + u_{k_t} = 1$ for all $1 \leq k_t \leq k_{\max}$. A move is performed if the algorithm accepts it. For $k_t = 1$ the death move is impossible, so that $d_0 \triangleq 0$. For $k = 0$ the death move is impossible, so that $d_0 \triangleq 0$. For $k = k_{\max}$, the birth is not allowed and therefore $b_{k_{\max}} \triangleq 0$. We set the transition model probabilities as follows $b_{k_t} = p(k_t+1|k_t)$, $d_{k_t} = p(k_t-1|k_t)$ and $b_{k_t} = p(k_t|k_t)$.

The update move simply updates the value of the parameters, while the birth and death moves allow the number of RBFs to increase or decrease. To ensure reversibility, we have designed a birth$_1$/death$_1$ move to communicate between the spaces $\left\{\widetilde{k}_{t-1}\right\} \times \boldsymbol{\Omega}_{\widetilde{k}_{t-1}}$ and $\left\{\widetilde{k}_{t-1}+1\right\} \times \boldsymbol{\Omega}_{\widetilde{k}_{t-1}+1}$ and a birth$_2$/death$_2$ move to communicate between $\left\{\widetilde{k}_{t-1}\right\} \times \boldsymbol{\Omega}_{\widetilde{k}_{t-1}}$ and $\left\{\widetilde{k}_{t-1}-1\right\} \times \boldsymbol{\Omega}_{\widetilde{k}_{t-1}-1}$. For all particles, the Markov chain is initialised at $\widetilde{k}_t$ and it obeys the following transition kernel:

$$\begin{aligned}
\mathcal{K} = \; & u_k K_{update} \mathbb{I}_{\{\widetilde{k}_{t-1}-1, \widetilde{k}_{t-1}, \widetilde{k}_{t-1}+1\}}(k_{t,j}) + b_k\{K_{birth_1} \mathbb{I}_{\{\widetilde{k}_{t-1}\}}(k_{t,j}) + K_{birth_2} \mathbb{I}_{\{\widetilde{k}_{t-1}-1\}}(k_{t,j})\} \\
& + d_k\{K_{death_2} \mathbb{I}_{\{\widetilde{k}_{t-1}\}}(k_{t,j}) + K_{death_1} \mathbb{I}_{\{\widetilde{k}_{t-1}+1\}}(k_{t,j})\}
\end{aligned} \tag{7.21}$$

[4] Note that although we are choosing MCMC moves that allow the number of RBFs to change only by 1, it is possible to design more complex and general moves within this sampling framework.

where the subscript j denotes the j-th iteration of the Markov chain. The expression tells us that if we are in the state $\{\widetilde{k}_{t-1}^{(i)}\}$, we can either execute the update, birth$_1$ or death$_2$ moves. If, for example, the algorithm selects a birth move in the first iteration, then it has to be a birth$_1$ move. If this move is then accepted, at the next iteration, one can only accept the update or death$_1$ move. That is, the death$_1$ move is the reversible move for the birth$_1$ move and the death$_2$ move is the reversible move for the birth$_2$. The possible trajectories of the Markov chain are summarised in Figure 7.2. In practice, we will only run one iteration of the MCMC sampler. We shall now proceed to describe the various moves in more detail.

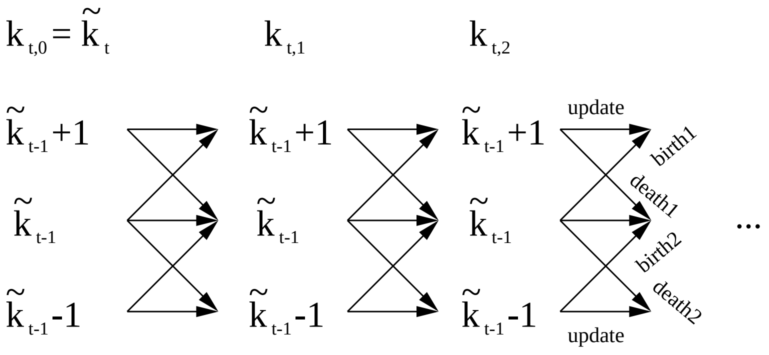

Figure 7.2 *Graph showing the possible transitions of the Markov chain.*

7.3.2.1 Update move

In this move, the dimension is kept fixed, *i.e.* $k_t^{(i)} = \widetilde{k}_t^{(i)}$. One merely samples the RBF centres; a difficult task because the distribution is nonlinear in these parameters. Here, they are sampled using the Metropolis-Hastings (MH) algorithm (Besag et al., 1995; Gilks et al., 1996; Tierney, 1994). The target distribution for each particle, at time t, is the posterior distribution:

$$p(\widetilde{\boldsymbol{\mu}}_{1:\widetilde{k}_{0:t},0:t}^{(i)}, \widetilde{\boldsymbol{\sigma}}_{0:t}^{2(i)}, \widetilde{k}_{0:t}^{(i)}|\mathbf{d}_{1:t})$$

while the proposal distribution corresponds to:

$$\boldsymbol{\mu}_{1:k_t,t}^{(i)\star}|\widetilde{\boldsymbol{\mu}}_{1:\widetilde{k}_t,t-1}^{(i)}, \widetilde{k}_t^{(i)}, \mathbf{d}_{t-1} \sim p(\boldsymbol{\mu}_{1:k_t,t}|\widetilde{\boldsymbol{\mu}}_{1:\widetilde{k}_t,t-1}^{(i)}, \widetilde{k}_t^{(i)})$$

Hence, the acceptance probability for the update move is given by:

$$\mathcal{A}(\widetilde{\boldsymbol{\mu}}_{\widetilde{k}_t,t}, \boldsymbol{\mu}_{\widetilde{k}_t,t}^{\star}) = \min\left\{1, \prod_{j=1}^{c} \frac{p(\mathbf{y}_{j,t}|\boldsymbol{\mu}_{1:\widetilde{k}_t,t}^{(i)\star}, \widetilde{\boldsymbol{\sigma}}_{j,t}^{2(i)}, \widetilde{k}_t^{(i)}, \mathbf{x}_t, \mathbf{d}_{1:t-1})}{p(\mathbf{y}_{j,t}|\widetilde{\boldsymbol{\mu}}_{1:\widetilde{k}_t,t}^{(i)}, \widetilde{\boldsymbol{\sigma}}_{j,t}^{2(i)}, \widetilde{k}_t^{(i)}, \mathbf{x}_t, \mathbf{d}_{1:t-1})}\right\} \tag{7.22}$$

7.3.2.2 Birth$_1$ and death$_1$ moves

As mentioned earlier[5], the birth$_1$ and death$_1$ moves allow us to jump between the spaces $\left\{\widetilde{k}_{t-1}\right\} \times \mathbf{\Omega}_{\widetilde{k}_{t-1}}$ and $\left\{\widetilde{k}_{t-1}+1\right\} \times \mathbf{\Omega}_{\widetilde{k}_{t-1}+1}$. Now, suppose that the current state of the Markov chain is in $\left\{\widetilde{k}_t = \widetilde{k}_{t-1}\right\} \times \mathbf{\Omega}_{\widetilde{k}_t}$, the birth$_1$ move will then involve the following steps:

Birth$_1$ move

- Propose a new RBF centre at random from $p\left(\boldsymbol{\mu}^{\star}\right)$.
- Evaluate $\mathcal{A}_{birth_1} = \min\left\{1, r_{birth_1}\right\}$, see equation (7.24), and sample $u \sim \mathcal{U}_{[0,1]}$.
- If $u \leq \mathcal{A}_{birth_1}$ then the state of the Markov chain becomes $\left(\widetilde{k}_{t-1}+1, \left(\boldsymbol{\mu}^{\star}, \boldsymbol{\mu}_{1:\widetilde{k}_{t-1}+1,t}\right)\right)$ else it remains equal to $\left(\widetilde{k}_{t-1}, \boldsymbol{\mu}_{1:\widetilde{k}_{t-1},t}\right)$.

■

The reverse move is described as follows. Suppose that the current state of the Markov chain is in $\left\{\widetilde{k}_t = \widetilde{k}_{t-1}+1\right\} \times \mathbf{\Omega}_{\widetilde{k}_t}$ the death$_1$ move will then involve the following steps:

Death$_1$ move

- Remove the basis that was added during the extension of the trajectory in the sampling stage.
- Evaluate $\mathcal{A}_{death_1} = \min\left\{1, r_{death_1}\right\}$, see equation (7.25), and sample $u \sim \mathcal{U}_{[0,1]}$.
- If $u \leq \mathcal{A}_{death_1}$ then the state of the Markov chain becomes $\left(\widetilde{k}_{t-1}, \boldsymbol{\mu}_{1:\widetilde{k}_{t-1},t}\right)$ else it remains equal to $\left(\widetilde{k}_{t-1}+1, \boldsymbol{\mu}_{1:\widetilde{k}_{t-1}+1,t}\right)$.

■

The acceptance ratios for the proposed moves are deduced from the following expression (Green, 1995):

$$r = (posterior\ distribution\ ratio) \times (proposal\ ratio) \times (Jacobian) \tag{7.23}$$

For the birth$_1$ move, we have:

$$\begin{aligned} r_{birth_1} \;&=\; \prod_{j=1}^{c} \frac{p\left(\mathbf{y}_{j,t}\middle|\, \boldsymbol{\mu}_{1:\widetilde{k}_{t-1}+1,t}, \widetilde{k}_{t-1}+1, \mathbf{x}_t, \mathbf{d}_{1:t-1}\right) p\left(\widetilde{k}_{t-1}+1\middle|\,\widetilde{k}_{t-1}\right)}{p\left(\mathbf{y}_{j,t}\middle|\, \boldsymbol{\mu}_{1:\widetilde{k}_{t-1},t}, \widetilde{k}_{t-1}, \mathbf{x}_t, \mathbf{d}_{1:t-1}\right) p\left(\widetilde{k}_{t-1}\middle|\,\widetilde{k}_{t-1}\right)} \\ &\quad \times \frac{p\left(\boldsymbol{\mu}^{\star}, \boldsymbol{\mu}_{1:\widetilde{k}_{t-1},t}\middle|\,\boldsymbol{\mu}_{1:\widetilde{k}_{t-1},t-1}\right)}{p\left(\boldsymbol{\mu}_{1:\widetilde{k}_{t-1},t}\middle|\,\boldsymbol{\mu}_{1:\widetilde{k}_{t-1},t-1}\right)} \frac{d_{\widetilde{k}_{t-1}+1}}{b_{\widetilde{k}_{t-1}}\, p\left(\boldsymbol{\mu}^{\star}\right)} \times J \end{aligned}$$

[5] To alleviate the notation, we shall drop the index $.^{(i)}$ when describing the birth and death moves.

where $p\left(\boldsymbol{\mu}^{\star}\right)$ is a Gaussian proposal distribution centered at the input data $\mathbf{x}_t$. The Jacobian J is clearly equal to 1. The above expression simplifies to:

$$r_{birth_1} = \prod_{j=1}^{c} \frac{p\left(\mathbf{y}_{j,t}\middle|\, \boldsymbol{\mu}_{1:\widetilde{k}_{t-1}+1,t}, \widetilde{k}_{t-1}+1, \mathbf{x}_t, \mathbf{d}_{1:t-1}\right) p\left(\widetilde{k}_{t-1}+1\middle| \widetilde{k}_{t-1}\right)}{p\left(\mathbf{y}_{j,t}\middle|\, \boldsymbol{\mu}_{1:\widetilde{k}_{t-1},t}, \widetilde{k}_{t-1}, \mathbf{x}_t, \mathbf{d}_{1:t-1}\right) p\left(\widetilde{k}_{t-1}\middle| \widetilde{k}_{t-1}\right)} \frac{d_{\widetilde{k}_{t-1}+1}}{b_{\widetilde{k}_{t-1}}} \tag{7.24}$$

The corresponding r_{death_1} is simply:

$$r_{death_1} = r_{birth_1}^{-1}$$

That is:

$$r_{death_1} = \prod_{j=1}^{c} \frac{p\left(\mathbf{y}_{j,t}\middle|\, \boldsymbol{\mu}_{1:\widetilde{k}_{t-1},t}, \widetilde{k}_{t-1}, \mathbf{x}_t, \mathbf{d}_{1:t-1}\right) p\left(\widetilde{k}_{t-1}\middle| \widetilde{k}_{t-1}\right)}{p\left(\mathbf{y}_{j,t}\middle|\, \boldsymbol{\mu}_{1:\widetilde{k}_{t-1}+1,t}, \widetilde{k}_{t-1}+1, \mathbf{x}_t, \mathbf{d}_{1:t-1}\right) p\left(\widetilde{k}_{t-1}+1\middle| \widetilde{k}_{t-1}\right)} \frac{b_{\widetilde{k}_{t-1}}}{d_{\widetilde{k}_{t-1}+1}} \tag{7.25}$$

7.3.2.3 Birth$_2$ and death$_2$ moves

The birth$_2$ and death$_2$ moves allows us to jump between the model spaces $\left\{\widetilde{k}_{t-1}\right\} \times \boldsymbol{\Omega}_{\widetilde{k}_{t-1}}$ and $\left\{\widetilde{k}_{t-1}-1\right\} \times \boldsymbol{\Omega}_{\widetilde{k}_{t-1}-1}$. Suppose that the current state of the Markov chain is in $\left\{\widetilde{k}_t = \widetilde{k}_{t-1}-1\right\} \times \boldsymbol{\Omega}_{\widetilde{k}_t}$ the birth$_2$ move will then involve the following steps:

Birth$_2$ move

- Propose a new RBF centre at random from $p\left(\boldsymbol{\mu}^{\star}\middle|\, \boldsymbol{\mu}_{deleted,t-1}\right)$, where $\boldsymbol{\mu}_{deleted,t-1}$ corresponds to the basis that was deleted during the extension of the trajectory.
- Evaluate $\mathcal{A}_{birth_2} = \min\left\{1, r_{birth_2}\right\}$, see equation (7.26), and sample $u \sim \mathcal{U}_{[0,1]}$.
- If $u \leq \mathcal{A}_{birth_2}$ then the state of the Markov chain becomes $\left(\widetilde{k}_{t-1}, \left(\boldsymbol{\mu}^{\star}, \boldsymbol{\mu}_{1:\widetilde{k}_{t-1},t}\right)\right)$ else it remains equal to $\left(\widetilde{k}_{t-1}-1, \mu_{1:\widetilde{k}_{t-1}-1,t}\right)$.

■

The reverse move is described as follows. Suppose that the current state of the Markov chain is in $\left\{\widetilde{k}_t = \widetilde{k}_{t-1}\right\} \times \boldsymbol{\Omega}_{\widetilde{k}_t}$, the death$_2$ move will then involve the following steps:

Death$_2$ move

- Choose the basis centre, to be deleted, at random among the $\widetilde{k}_{t-1}$ existing bases.
- Evaluate $\mathcal{A}_{death_2} = \min\left\{1, r_{death_2}\right\}$, see equation (7.27), and sample $u \sim \mathcal{U}_{[0,1]}$.

- If $u \leq \mathcal{A}_{death_2}$ then the state of the Markov chain becomes $\left(\widetilde{k}_{t-1}-1, \boldsymbol{\mu}_{1:\widetilde{k}_{t-1}-1,t}\right)$ else it remains equal to $\left(\widetilde{k}_{t-1}, \boldsymbol{\mu}_{1:\widetilde{k}_{t-1},t}\right)$.

■

Following the same steps as in the previous section, we obtain the following expression for r_{birth_2}:

$$
\begin{aligned}
r_{birth_2} \quad = \quad & \prod_{j=1}^{c} \frac{p\left(\mathbf{y}_{j,t}\middle|\, \boldsymbol{\mu}_{1:\widetilde{k}_{t-1},t}, \widetilde{k}_{t-1}, \mathbf{x}_t, \mathbf{d}_{1:t-1}\right) p\left(\widetilde{k}_{t-1}\middle|\, \widetilde{k}_{t-1}\right)}{p\left(\mathbf{y}_{j,t}\middle|\, \boldsymbol{\mu}_{1:\widetilde{k}_{t-1}-1,t}, \widetilde{k}_{t-1}-1, \mathbf{x}_t, \mathbf{d}_{1:t-1}\right) p\left(\widetilde{k}_{t-1}-1\middle|\, \widetilde{k}_{t-1}\right) \frac{1}{\widetilde{k}_{t-1}}} \\
& \times \frac{p\left(\boldsymbol{\mu}^{\bigstar}, \boldsymbol{\mu}_{1:\widetilde{k}_{t-1},t}\middle|\, \boldsymbol{\mu}_{1:\widetilde{k}_{t-1},t-1}\right)}{p\left(\boldsymbol{\mu}_{1:\widetilde{k}_{t-1}-1,t}\middle|\, \boldsymbol{\mu}_{1:\widetilde{k}_{t-1},t-1}\right)} \frac{d_{\widetilde{k}_{t-1}} \frac{1}{\widetilde{k}_{t-1}}}{b_{\widetilde{k}_{t-1}-1} p\left(\boldsymbol{\mu}^{\bigstar}\middle|\, \boldsymbol{\mu}_{deleted,t-1}\right)} \times J
\end{aligned}
$$

which simplifies to:

$$
r_{birth_2} = \prod_{j=1}^{c} \frac{p\left(\mathbf{y}_{j,t}\middle|\, \boldsymbol{\mu}_{1:\widetilde{k}_{t-1},t}, \widetilde{k}_{t-1}, \mathbf{x}_t, \mathbf{d}_{1:t-1}\right) p\left(\widetilde{k}_{t-1}\middle|\, \widetilde{k}_{t-1}\right)}{p\left(\mathbf{y}_{j,t}\middle|\, \boldsymbol{\mu}_{1:\widetilde{k}_{t-1}-1,t}, \widetilde{k}_{t-1}-1, \mathbf{x}_t, \mathbf{d}_{1:t-1}\right) p\left(\widetilde{k}_{t-1}-1\middle|\, \widetilde{k}_{t-1}\right)} \frac{d_{\widetilde{k}_{t-1}}}{b_{\widetilde{k}_{t-1}-1}} \qquad (7.26)
$$

The corresponding death ratio is:

$$
r_{death_2} = r_{birth_2}^{-1}
$$

That is:

$$
r_{death_2} = \prod_{j=1}^{c} \frac{p\left(\mathbf{y}_{j,t}\middle|\, \boldsymbol{\mu}_{1:\widetilde{k}_{t-1}-1,t}, \widetilde{k}_{t-1}-1, \mathbf{x}_t, \mathbf{d}_{1:t-1}\right) p\left(\widetilde{k}_{t-1}-1\middle|\, \widetilde{k}_{t-1}\right)}{p\left(\mathbf{y}_{j,t}\middle|\, \boldsymbol{\mu}_{1:\widetilde{k}_{t-1},t}, \widetilde{k}_{t-1}, \mathbf{x}_t, \mathbf{d}_{1:t-1}\right) p\left(\widetilde{k}_{t-1}\middle|\, \widetilde{k}_{t-1}\right)} \frac{b_{\widetilde{k}_{t-1}-1}}{d_{\widetilde{k}_{t-1}}} \qquad (7.27)
$$

7.4 Synthetic example

Data was generated from the following univariate, nonlinear and time-varying function using 500 covariate points uniformly distributed on $[0,1]$:

$$
y_t = \begin{cases} 4x_t - 2 + \left(2 + \frac{t}{150}\right) \exp(-15(x_t - 0.3)^2) + 2\exp(-15(x_t - 0.7)^2) + n_t & 1 \leq t \leq 250 \\ 4x_t - 2 + 2\exp(-15(x_t - 0.5)^2) + n_t & 250 < t \leq 500 \end{cases}
$$

where $n_t \sim \mathcal{N}(0, \sigma^2)$ with $\sigma^2 = 0.01$. Figure 7.3 shows the generated data. Note that the drift in the size of the Gaussian component at $x_t = 0.3$ is seriously affected by the noise corruption.

In the experiment, we selected Gaussian radial basis functions with the same variance as the Gaussian signal components. We chose $p(k_{t+1}|k_t)$ to have support $\{k_t - 1, k_t, k_t + 1\}$ and range $\{0.025, 0.95, 0.025\}$ and adopted a uniform prior for $\boldsymbol{\sigma}_0^2$ between 0 and 0.1. Since the domain of the data is the interval $[0,1]$, a multivariate Gaussian

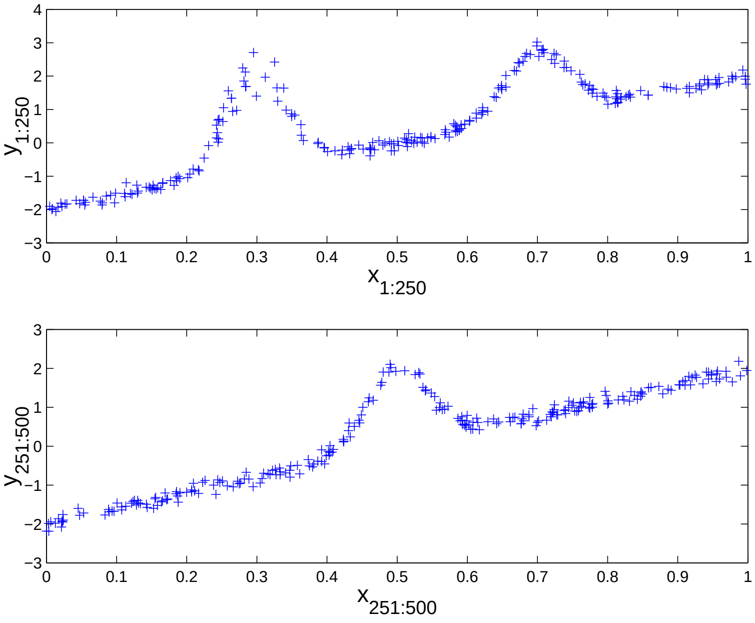

Figure 7.3 *Data generated for the synthetic experiment.*

prior for $\boldsymbol{\mu}_{1:k_0,0}$ with mean 0.5 and variance 0.1 was adopted. The diffusion parameters $\delta^2_{\boldsymbol{\mu}}$, $\delta^2_{\boldsymbol{\alpha}}$ and $\delta^2_{\boldsymbol{\sigma}}$ were all set to 0.001. The parameter $k_{\max}$ was set to 20. An informative multivariate Gaussian prior, with mean 2 and variance 2, was used for $\boldsymbol{\alpha}_{1:m_0,0}$. The extended means $\overline{\boldsymbol{\alpha}}_{1:m_0,0}$ of the bank of Kalman filters were drawn from the same prior. Finally, the Kalman filter covariance terms $P_{k_0,0}$ were set to 1.

The simulation results, using 1000 samples, are depicted in Figures 7.4 and 7.5. Figure 7.4 shows that the estimated model orders and noise variances (posterior means) converge to the right values. As a result, the output one-step-ahead predictions are very accurate. Figure 7.5 shows the estimates of the linear regression coefficients, basis centres and basis coefficients. These posterior mean estimates correspond to the centroids of the distributions computed using the main modes ($k = 2$ from $t = 1$ to $t = 500$ and $k = 1$ afterwards). Note how the integration of the coefficients leads to very fast and accurate estimates of the linear coefficients.

The algorithm performs extremely well when computing the linear regression parameters. However, it is clear that to obtain similar results for the other quantities, it requires more samples. It should also be mentioned that given the amount of noise, it is very difficult to track the drift in the basis function amplitude. Figures 7.6 and 7.7 show the histograms of the estimated noise variance and model order.

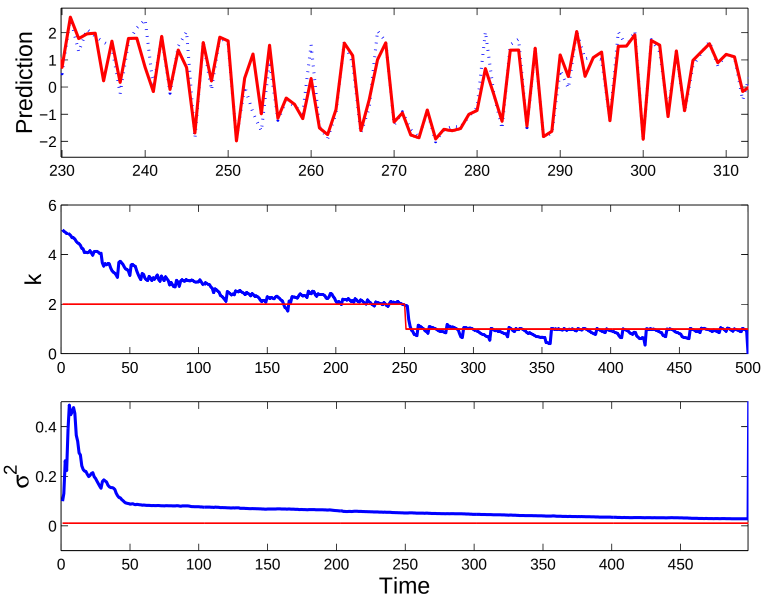

Figure 7.4 *The top plot shows the one-step-ahead output predictions [—] and the true outputs [···]. The middle and bottom plots show the true values and estimates of the model order and noise variance respectively.*

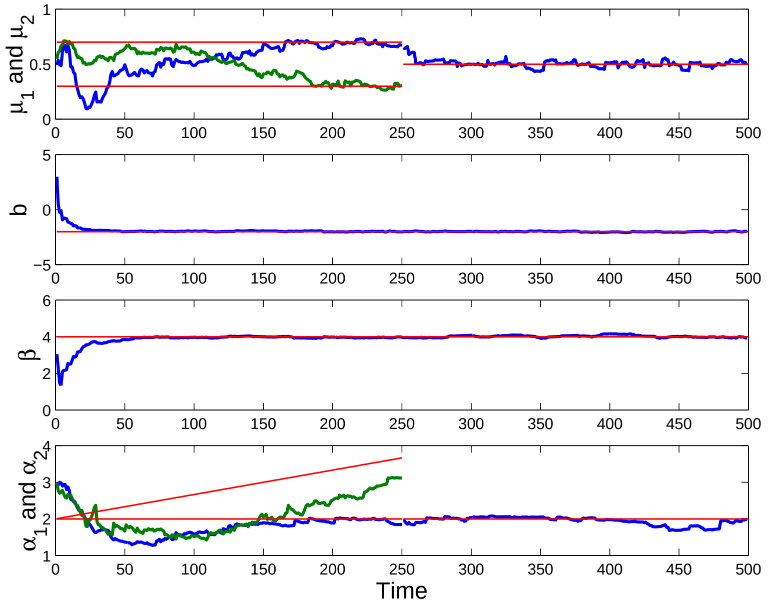

Figure 7.5 *Estimated and true values of the basis centres, linear regression parameters and basis coefficients.*

7.5 Summary

This chapter proposed a simulation-based approach to perform sequential Bayesian model selection. This strategy combines sequential Monte Carlo methods and reversible

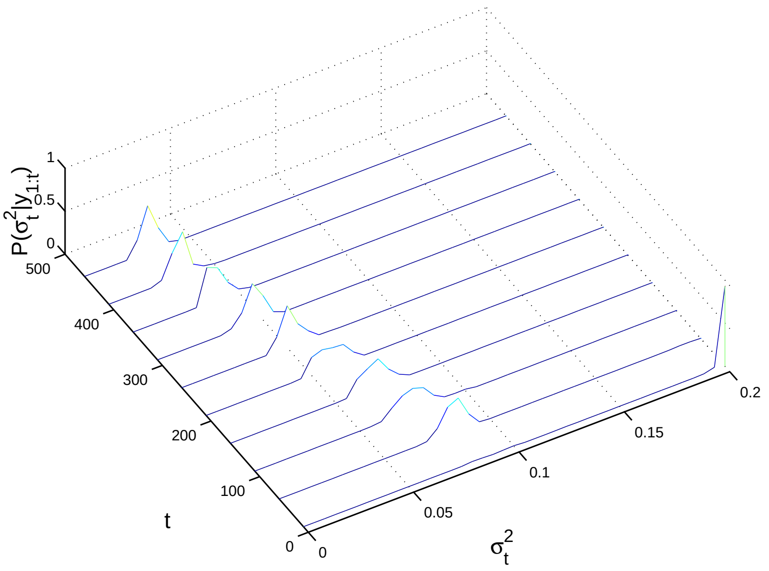

Figure 7.6 *Probabilities of the estimated noise variance.*

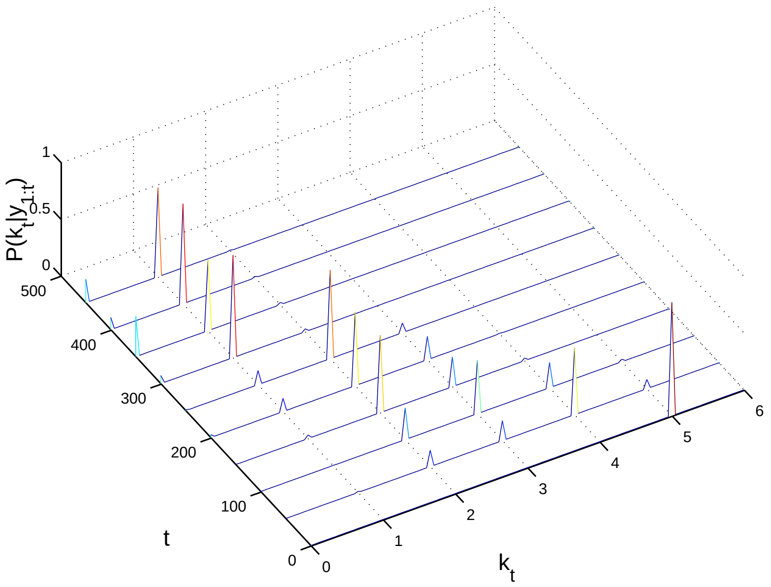

Figure 7.7 *Probabilities of the estimated model order.*

jump MCMC methods. It also makes use of variance reduction techniques. Its only drawback is that it is computationally expensive, yet it can be straightforwardly implemented on parallel computers. The effectiveness of the method was demonstrated on a complex joint estimation and model selection problem. There are many possibilities for future research. These include adopting MCMC schemes with forgetting factors to estimate the noise variance, thereby avoiding the need for the diffusion parameter $\delta_{\boldsymbol{\sigma}}^2$; allowing for models with different types of basis functions; performing input variable selection and experimenting with other noise distributions.

8

Applications

This chapter demonstrates the methods developed in Chapters 3 to 7 in a few application domains. Some comparative studies are used to evaluate the advantages and shortcomings of these methods.

Financial time series, by virtue of their nonlinearity, non-stationary and complex stochastic behaviour, constitute a difficult and challenging problem. It was decided, therefore, to test the sequential algorithms on two econometric problems, namely options pricing and foreign exchange forecasting. The batch learning methods are assessed on a robotics problem and a medical diagnosis application. The former is a standard benchmark for comparing the performance of learning algorithms for neural networks.

8.1 Experiment 1: Options Pricing

Derivatives are financial instruments whose value depends on some basic underlying cash product, such as interest rates, equity indices, commodities, foreign exchange, bonds, etc. An option is a particular type of derivative that gives the holder the right to do something. For example, a call option allows the holder to buy a cash product, at a specified date in the future, for a price determined in advance. The price at which the option is exercised is known as the strike price, while the date at which the option lapses is referred to as the maturity time. Put options, on the other hand, allow the holder to sell the underlying cash product.

In recent years, the mathematical modelling of financial derivatives has become increasingly important (Hull, 1997). Owing to the competitive nature of their environment, financial institutions have much interest in developing more sophisticated pricing models for options contracts. So far, the research results seem to provide clear evidence that there is a nonlinear and non-stationary relation between the options' price and the cash products' price, maturity time, strike price, interest rates and variance of the returns on the cash product (volatility). The standard model used to describe this relation is the Black-Scholes model (Black and Scholes, 1973). This basic model is, however, only valid under several conditions, namely no risk-less arbitrage opportunities, an instantaneous risk-less portfolio, continuous trading, no dividends, constant volatility and risk-free interest rate. In addition, the stock's price is assumed to be dictated by a geometric Brownian motion model.

To circumvent the limitations of the Black-Scholes model, Hutchinson et. al. (1994) and Niranjan (1996) have focused on the options pricing problem from a neural computing perspective. The former showed that good approximations to the widely used Black-Scholes formula may be obtained with neural networks, while the latter looked at the non-stationary aspects of the problem. Here, the work of Niranjan (1996) is extended, with the aim of showing that more accurate tracking of the options' prices can be achieved by employing the sequential methods proposed in this thesis. In particular, neural networks are employed to map the stock's price and time to maturity to the call and put options' prices. The stock's price and options' prices are normalised by the strike price. This approach, in conjunction with the use of volatility smiles and other strategic financial information, provides an indication of whether an option in the market is either overpriced or underpriced.

The experiment used five pairs of call and put option contracts on the FTSE100 index (daily close prices from February 1994 to December 1994) to evaluate the following pricing algorithms:

Trivial : This method simply involves using the current value of the option as the next prediction.

RBF-EKF : Represents a regularised radial basis function network with 4 hidden neurons, which was originally proposed in (Hutchinson et al., 1994). The output weights are estimated with a Kalman filter, while the means of the radial functions correspond to random subsets of the data, and their covariance is set to the identity matrix as in (Niranjan, 1996).

BS : Corresponds to a conventional Black-Scholes model with two outputs (normalised call and put prices) and two parameters (risk-free interest rate and volatility). The risk-free interest rate was set to 0.06, while the volatility was estimated over a moving window (of 50 time steps) as described in (Hull, 1997).

MLP-EKF : A multi-layer perceptron, with 6 sigmoidal hidden units and a linear output neuron, trained with the EKF algorithm (Chapter 3). The noise covariances R and Q and the states covariance P were set to diagonal matrices with entries equal to

10^{-6}, 10^{-7} and 10 respectively. The weights prior corresponded to a zero mean Gaussian density with variance equal to 1.

MLP-EKFQ : Represents an MLP, with 6 sigmoidal hidden units and a linear output neuron, trained with an hierarchical Bayesian model, whereby the weights are estimated with the EKF algorithm and the noise statistics are computed by maximising the evidence density function as suggested in Chapter 3 (Jazwinski's algorithm). The states covariance P was given by a diagonal matrix with entries equal to 10. The weights prior corresponded to a zero mean Gaussian density with variance equal to 1.

MLP-SIR : Corresponds to an MLP, with 6 sigmoidal hidden units and a linear output neuron, trained with the SIR algorithm. 1000 samples were employed in each simulation. The noise covariances R and Q were set to diagonal matrices with entries equal to 10^{-4} and 10^{-5} respectively. The prior samples for each layer were drawn from a zero mean Gaussian density with covariance equal to 1.

MLP-HySIR : Corresponds to an MLP, with 6 sigmoidal hidden units and a linear output neuron, trained with the HySIR algorithm (Chapter 6). Each simulation made use of 10 samples. The noise covariances R and Q were set to diagonal matrices with entries equal to 1 and 10^{-6} respectively. The prior samples for each layer were drawn from a zero mean Gaussian density with variance equal to 1. The Kalman filter parameters R^*, Q^* and P were set to diagonal matrices with entries equal to 10^{-4}, 10^{-5} and 1000 respectively.

MLP-SMC : Refers to an MLP, with 6 sigmoidal hidden units and a linear output neuron, trained with the SMC algorithm with linearised importance proposal and a mixture of Metropolis-Hastings correction steps, as discussed in Chapter 6. The simulations employed 500 particles. The noise covariances for the proposals R^* and Q^* were set to diagonal matrices with entries equal to 10^{-7} and 10^{-8} respectively. The prior samples for each layer were drawn from a zero mean Gaussian density with variance equal to 10. The diffusion parameter for the MLP weights was set to 10^{-8}. The initial measurement noise variance was drawn from a uniform prior between 0 and 10^{-3}. Its diffusion parameter was set to 10^{-8}.

SMC-RBF : This is the RBF model proposed in Chapter 7. Thin-plate spline basis functions were used in the approximation model. The number of samples was set to 500. In addition, a uniform prior for $\boldsymbol{\sigma}_0^2$, between 0 and 0.35, and a multivariate Gaussian prior for $\boldsymbol{\mu}_{1:k_0,0}$, with mean 0 and variance 0.05, were chosen. The diffusion parameters $\delta^2_{\boldsymbol{\mu}}$, $\delta^2_{\boldsymbol{\alpha}}$ and $\delta^2_{\boldsymbol{\sigma}}$ were set to 0.001, 0.00001 and 0.001 respectively. The parameters $k_{\max}$ and $\varsigma^\star$ were set to 20 and 0.05. A multivariate Gaussian prior for $\boldsymbol{\alpha}_{1:m_0,0}$, with mean 0 and variance 1, was adopted. The extended means $\overline{\boldsymbol{\alpha}}_{1:m_0,0}$ of the bank of Kalman filters were drawn from the same prior. Finally, the Kalman filter covariance terms $P_{k_0,0}$ were set to 10.

Strike price	2925	3025	3125	3225	3325
Trivial	0.0783	0.0611	0.0524	0.0339	0.0205
RBF-EKF	0.0538	0.0445	0.0546	0.0360	0.0206
BS	0.0761	0.0598	0.0534	0.0377	0.0262
MLP-EKF	0.0414	0.0384	0.0427	0.0285	0.0145
MLP-EKFQ	0.0404	0.0366	0.0394	0.0283	0.0150
MLP-SIR	0.0419	0.0405	0.0432	0.0314	0.0164
MLP-HySIR	0.0394	0.0362	0.0404	0.0273	0.0151
MLP-SMC	0.0414	0.0420	0.0421	0.0294	0.0143
SMC-RBF	0.0434	0.0369	0.0418	0.0289	0.0159

Table 8.1 *One-step-ahead prediction errors on call options.*

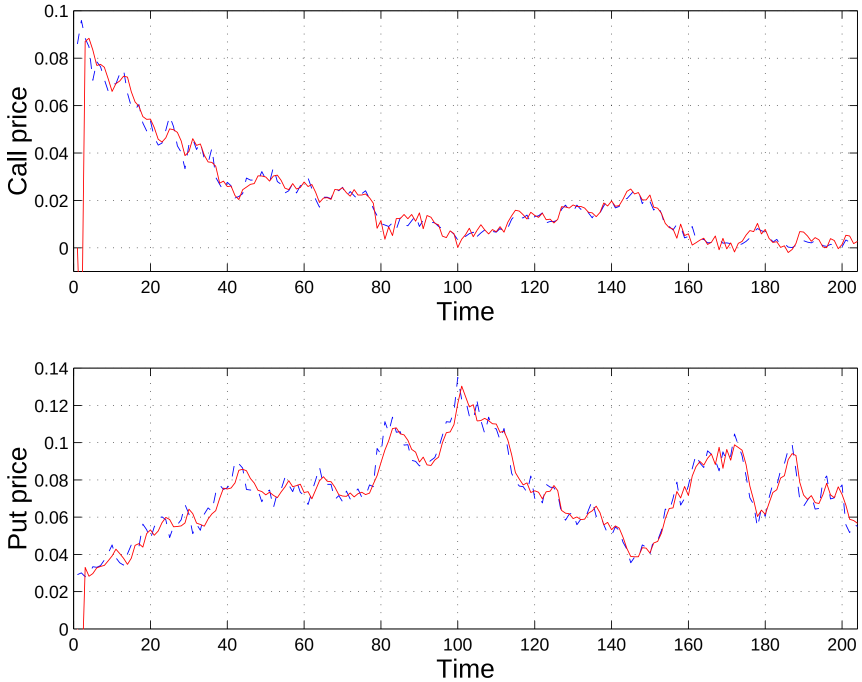

Figure 8.1 *Tracking call and put option prices with the MLP-HySIR method. Estimated values [—] and actual values [- -].*

Table 8.1 shows the mean squared error between the actual values of the options contracts and the one-step-ahead predictions for the various algorithms. Though the

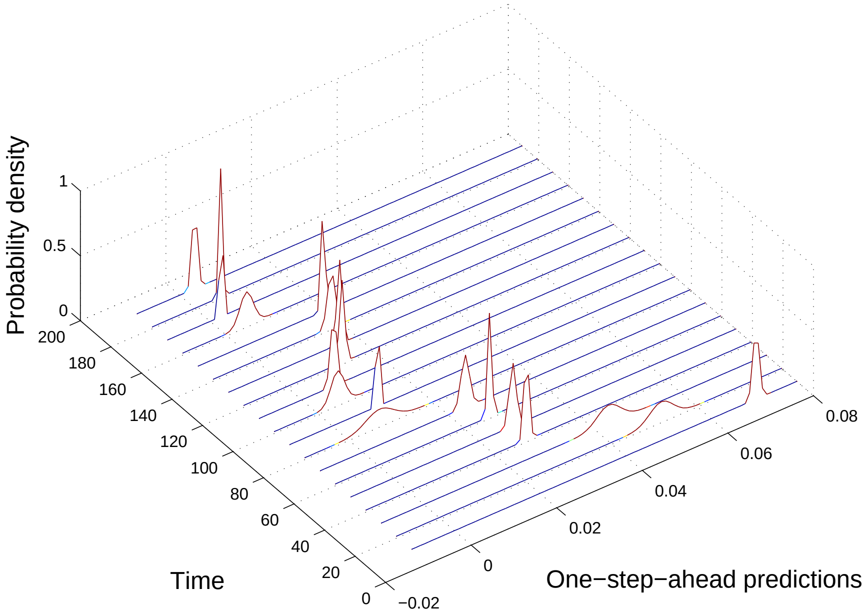

Figure 8.2 *Probability density function of the neural network's one-step-ahead prediction of the call price for the FTSE option with a strike price of* 3325.

differences are small, one sees that sequential algorithms produce lower prediction errors. The results for the MLP-EKFQ algorithm suggest that noise adaptation can lead to more accurate results. The HySIR algorithm seems to perform very well in comparison to the other algorithms. The one-step-ahead errors were computed on data corresponding to the last 100 days, to allow for convergence of the algorithms. Of course, this biases the results slightly in favour of the gradient descent methods.

Figure 8.1 shows the one-step-ahead predictions of the MLP-HySIR algorithm on the call and put option prices at the same strike price (3325). The algorithm exhibits fast convergence and accurate tracking. Figure 8.2 shows the probability density function of the network's one-step-ahead prediction, computed with the SIR algorithm. Typically, the one-step-ahead predictions for a group of options on the same cash product, but with different strike prices or duration to maturity, can be used to determine whether one of the options is being mispriced. Knowing the probability distribution of the network outputs allows us to design more interesting pricing tools.

8.2 Experiment 2: Foreign Exchange Forecasting

This section considers a foreign exchange spot rate forecasting problem, previously studied by (Nabney et al., 1996). The data set, shown in Figure 8.3, consists of the Deutsche Mark/French Franc daily closing prices from 30 September 1991 to June

1994. The foreign exchange market for this period was not entirely free as central banks often intervened to control the exchange rates between members of the European Exchange Rate Mechanism (ERM). Figure 8.3 shows one of these events, which took place in early August 1993 (time step 480), when the French and German central banks were forced to admit defeat against market guided speculation.

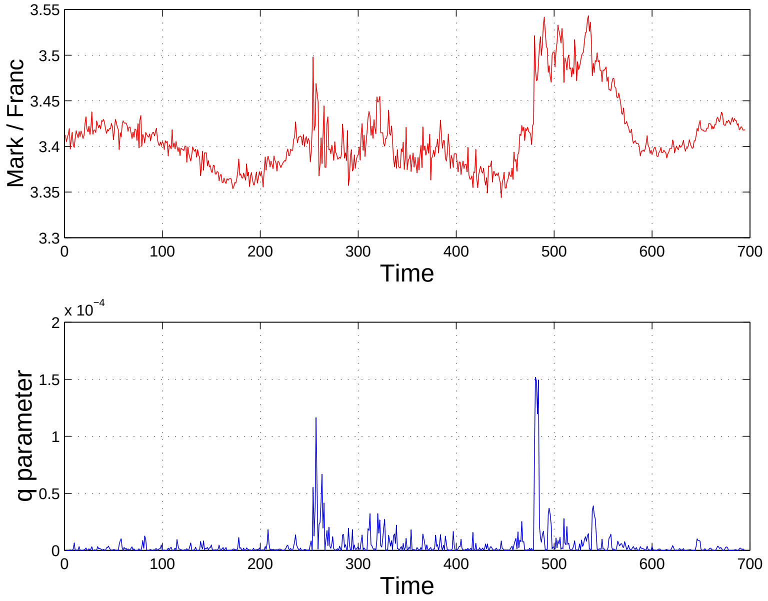

Figure 8.3 *The top plot shows the Deutsche Mark/French Franc daily closing prices from 30 September 1991 to June 1994. The bottom plots shows the regions of non-stationarity as detected by the MLP-EKFQ algorithm.*

To compare the algorithms proposed in this thesis with the ones of (Nabney et al., 1996), several on-line algorithms were implemented. The objective was to use a lag of five past observations to predict the next observation. With the exception of the trivial and SMC-Heavy methods, all the algorithms assumed the measurements to be Gaussian. The SMC-Heavy method uses a heavy tailed student-t distribution. A description of the algorithms follows:

Trivial : This method simply involves using the current value of the exchange rate as the next prediction.

MLP-EKF : A multi-layer perceptron, with 2 sigmoidal hidden units and a linear output neuron, trained with the EKF algorithm (Chapter 3). The noise covariances R and Q and the states covariance P were set to diagonal matrices with entries equal to 2×10^{-2}, 10^{-2} and 10 respectively. The weights prior corresponded to a zero mean Gaussian density with variance equal to 10.

MLP-EKFQ : Represents an MLP, with 2 sigmoidal hidden units and a linear output neuron, trained with an hierarchical Bayesian model, whereby the weights are estimated with the EKF algorithm and the process noise variance is computed by maximising the evidence density function as suggested in Section 3.3.3. The states covariance P was given by a diagonal matrix with entries equal to 10. The weights prior corresponded to a zero mean Gaussian density with variance equal to 10. The process noise covariance was set to a diagonal matrix with entries 2×10^{-7}.

MLP-SIR : Corresponds to an MLP, with 2 sigmoidal hidden units and a linear output neuron, trained with the SIR algorithm. 1000 samples were employed in each simulation. The noise covariances R and Q were both set to diagonal matrices with entries equal to 10^{-3}. The prior samples for each layer were drawn from a zero mean Gaussian density with variance equal to 10.

MLP-HySIR : Corresponds to an MLP, with 2 sigmoidal hidden units and a linear output neuron, trained with the HySIR algorithm (Chapter 6). Each simulation made use of 20 samples. The noise covariances R and Q were both set to diagonal matrices with entries equal to 10^{-3}. The prior samples for each layer were drawn from a zero mean Gaussian density with variance equal to 10. The Kalman filter parameters R^*, Q^* and P were set to diagonal matrices with entries equal to 10^{-1}, 10^{-7} and 100 respectively.

MLP-SMC : Refers to an MLP, with 2 sigmoidal hidden units and a linear output neuron, trained with the SMC algorithm with linearised importance proposal and a mixture of Metropolis-Hastings correction steps, as discussed in Chapter 6. The simulations employed 500 particles. The noise covariances for the proposals R^* and Q^* were both set to diagonal matrices with entries equal to 10^{-7}. The prior samples for each layer were drawn from a zero mean Gaussian density with variance equal to 10. The diffusion parameter for the MLP weights was set to 10^{-3}. The initial measurement noise variance was drawn from a uniform prior between 0 and 10^{-3}. Its diffusion parameter was set to 10^{-10}.

SMC-RBF : This is an RBF with thin plate spline basis functions trained with the model selection algorithm discussed in Chapter 7. The number of samples was set to 500. A uniform prior for $\boldsymbol{\sigma}_0^2$, between 0 and 0.5, and a multivariate Gaussian prior for $\boldsymbol{\mu}_{1:k_0,0}$, with mean 3.4 and variance 0.1, were chosen. The same prior was used for $\boldsymbol{\mu}^\star$. The diffusion parameters $\delta^2_{\boldsymbol{\mu}}$, $\delta^2_{\boldsymbol{\alpha}}$ and $\delta^2_{\boldsymbol{\sigma}}$ were set to 10^{-3}, 10^{-5} and 10^{-3} respectively. The parameters $k_{\max}$ and $\varsigma^\star$ were set to 20 and 0.005. A multivariate Gaussian prior with mean 0 and variance 2 was used for $\boldsymbol{\alpha}_{1:m_0,0}$.

The extended means $\overline{\boldsymbol{\alpha}}_{1:m_0,0}$ of the bank of Kalman filters were drawn from the same prior. Finally, the Kalman filter covariance terms $P_{k_0,0}$ were set to 100.

SMC-Heavy : Corresponds to an MLP, with 2 sigmoidal hidden units and a linear output neuron, trained with the SIR algorithm using a student-t likelihood with parameters $\alpha = 10^{-5}$ and $\lambda = 1$. 1000 samples were employed in each simulation. The noise covariance of the transition prior Q was set to a diagonal matrix with entries equal to 10^{-3}. The prior samples for each layer were drawn from a zero mean Gaussian density with variance equal to 10.

Method	Normalised RMSE
Trivial	0.2438
EKF with fixed size RBF (Nabney et al., 1996)	0.2351
RAN (Nabney et al., 1996)	0.2336
MLP-EKF	0.2349
MLP-EKFQ	0.2375
MLP-SIR	0.2373
MLP-HySIR	0.2346
MLP-SMC	0.2383
SMC-RBF	0.2371
SMC-Heavy	0.2335

Table 8.2 *Averaged normalised root mean square errors (fraction of unexplained variance) for the foreign exchange forecasting problem.*

Table 8.2 shows the errors incurred by each method. Once again they appear to be very similar, yet we need to keep in mind that small improvements in financial time series prediction often lead to large profits. Perhaps the most important conclusion one can draw is that when modelling financial time series one should use heavy tailed distributions instead of Gaussian approximations. The student-t distribution provides a better description of the high kurtosis typical of financial time series.

The SMC-RBF method did not perform very well in comparison to the other methods, yet it allowed us to obtain an estimate of the required number of basis functions. The one-step-ahead predictions, posterior mean model order estimates and prediction errors obtained with the SMC-RBF method are shown in Figure 8.4. The MLP-EKFQ was outperformed by the simple MLP-EKF algorithm. Nonetheless, it allowed us to determine the regions of non-stationarity as indicated in Figure 8.3.

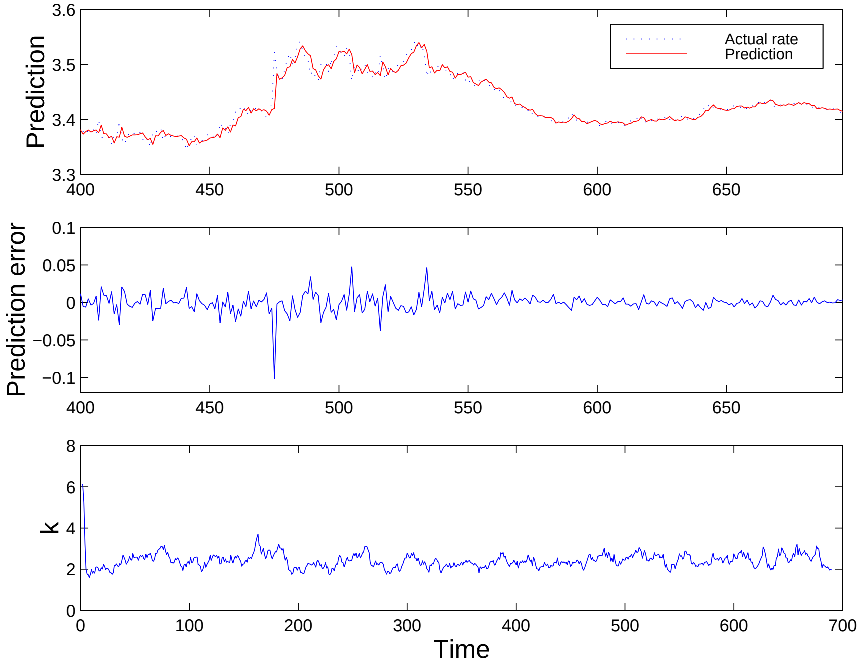

Figure 8.4 *One-step-ahead predictions, prediction errors and posterior mean model order estimate for the foreign exchange forecasting problem.*

8.3 Experiment 3: Robot Arm Mapping

This data set is often used as a benchmark to compare neural network algorithms[1]. It involves implementing a model to map the joint angle of a robot arm (x_1, x_2) to the position of the end of the arm (y_1, y_2). The data were generated from the following model:

$$\begin{aligned} y_1 &= 2.0\cos(x_1) + 1.3\cos(x_1 + x_2) + \epsilon_1 \\ y_2 &= 2.0\sin(x_1) + 1.3\sin(x_1 + x_2) + \epsilon_2 \end{aligned}$$

where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, $\sigma = 0.05$. The first 200 observations of the data set are used to train the models and the last 200 observations to test them.

Figure 8.5 shows the 3D plots of the training data and the contours of the training and test data. The contour plots also include the approximations that were obtained using the EM algorithm (Chapter 4) and an MLP with 2 linear output neurons and 20 sigmoidal hidden neurons. Figure 8.6 shows the convergence of the EM algorithm. In this particular run the training and test mean square errors were 0.00573 and 0.00810 (the minimum bound being $2\sigma^2 = 0.005$). Our mean square errors are of the same magnitude as the ones reported by other researchers, as shown in Table 8.3. Figure 8.6

[1] The data set can be found at `http://wol.ra.phy.cam.ac.uk/mackay/`

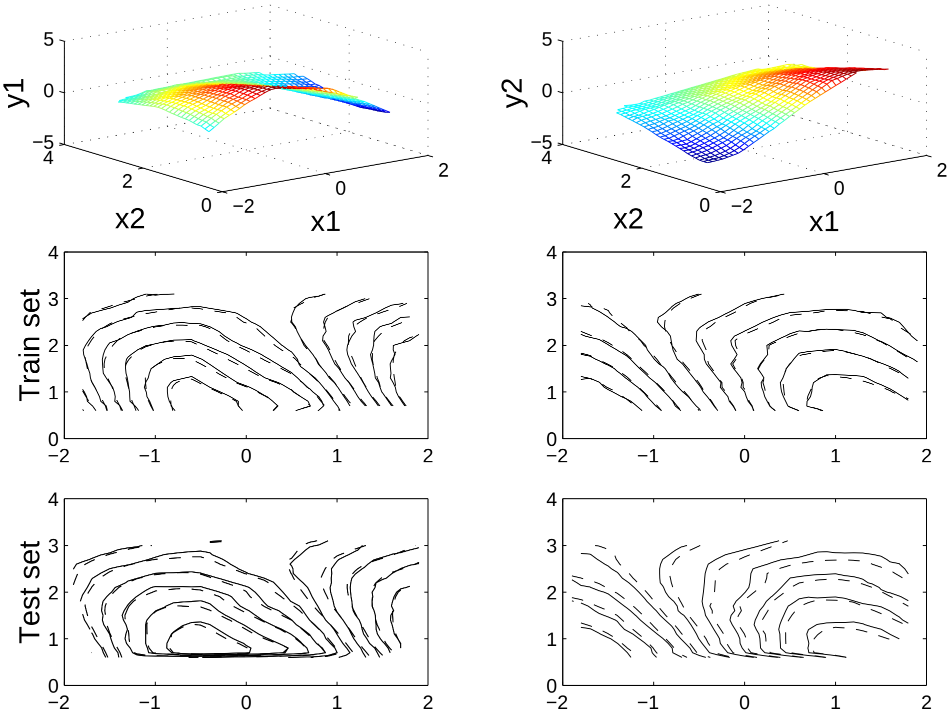

Figure 8.5 *The top plots show the training data surfaces corresponding to each coordinate of the robot arm's position. The middle and bottom plots show the training and validation data [- -] and the respective MLP mappings obtained with the EM algorithm [—].*

Method	MS error
Mackay's Gaussian approximation with highest evidence	0.00573
Mackay's Gaussian approximation with lowest test error	0.00557
Neal's hybrid MCMC	0.00554
Neal's hybrid MCMC with ARD	0.00549
Rios Insua's MLP with RJ-MCMC	0.00620
Holmes' RBF with RJ-MCMC	0.00535
EM method	0.00810
RJ-MCMC Bayesian method	0.00502
RJ-MCMC with MDL	0.00512
RJ-MCMC with AIC	0.00520

Table 8.3 *Mean square errors and number of basis functions for the robor arm data.*

also shows the two diagonal entries of the measurements noise covariance and the trace of the process noise covariance. They behave as expected. That is, the variance R converges to the true value, whereas the trace of Q goes to zero, indicating that the

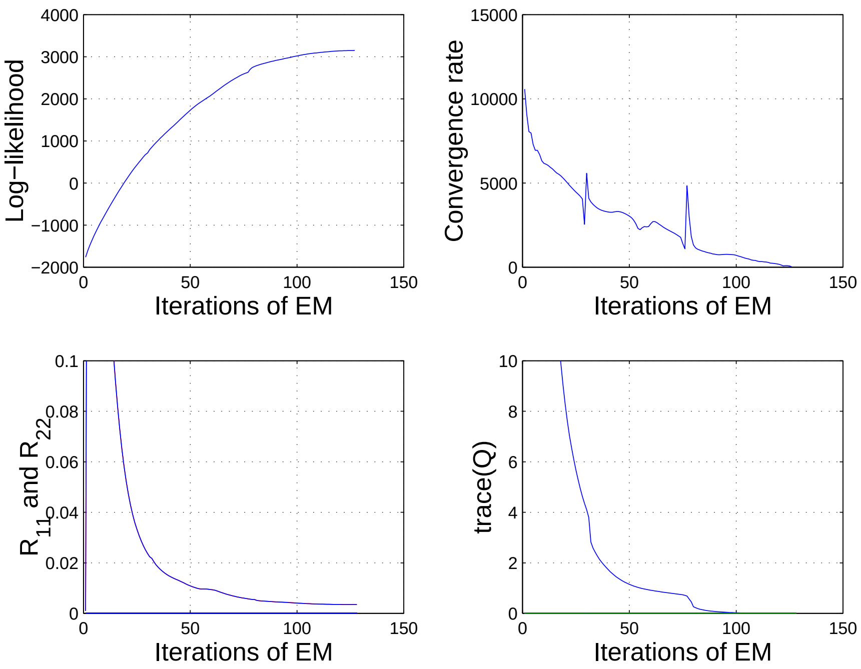

Figure 8.6 *The top plots show the log-likelihood function and the convergence rate (log-likelihood slope) for the robot arm problem. The bottom plots show the convergence of the diagonal entries of the measurements noise covariance R (almost identical) and the trace of the process noise covariance Q.*

model can approximate the stationary data perfectly well.

As indicated in Table 8.3, the performance of the reversible jump algorithm with the Bayesian model was also assessed in this data set. In all the simulations, cubic basis functions were adopted. Plots of the probabilities of each model order $\widehat{p}(k|\mathbf{x},\mathbf{y})$ in the chain (using equation (5.9)) for 50000 iterations were used to assess convergence, as shown in Figure 8.7. As the model orders begin to stabilise after 30000 iterations, the Markov chains were run for 50000 iterations with a burn in of 30000 iterations. It is possible to design more complex convergence diagnostic tools, however this topic is beyond the scope of this thesis.

Uninformative priors were chosen for all the parameters and hyper-parameters. In particular, the values shown in Table 8.4 were used. To demonstrate the robustness of the reversible jump Bayesian algorithm, different values for β_{δ^2} were chosen (the only critical hyper-parameter as it quantifies the mean of the spread $\boldsymbol{\delta}$ of $\boldsymbol{\alpha}_k$). The obtained mean square errors (Table 8.4) and probabilities for $\boldsymbol{\delta}_1$, $\boldsymbol{\delta}_2$, $\boldsymbol{\sigma}^2_{1,k}$, $\boldsymbol{\sigma}^2_{2,k}$ and k, shown in Figure 8.8, clearly indicate that the algorithm is robust with respect to prior specification.

The computed mean square errors are of the same magnitude as the ones reported

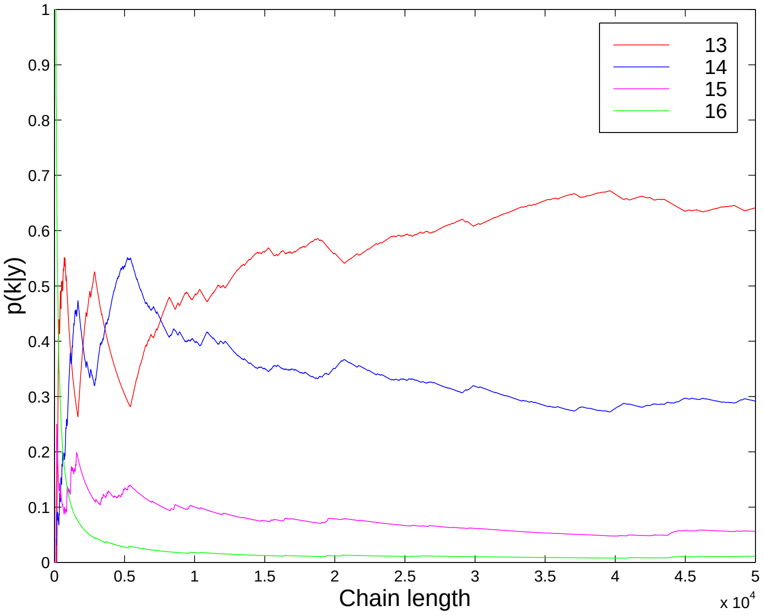

Figure 8.7 *Convergence of the reversible jump MCMC algorithm for RBF networks. The plot shows the probability of each model order given the data. The model orders begin to stabilise after 30000 iterations.*

α_{δ^2}	β_{δ^2}	v_0	γ_0	ε_1	ε_2	MS Error
2	0.1	0	0	0.0001	0.0001	0.00505
2	10	0	0	0.0001	0.0001	0.00503
2	100	0	0	0.0001	0.0001	0.00502

Table 8.4 *Simulation parameters and mean square errors for the robot arm data (test set) using the reversible jump MCMC algorithm and the Bayesian model.*

by other researchers (Holmes and Mallick, 1998; Mackay, 1992b; Neal, 1996; Rios Insua and Müller, 1998), indeed slightly better (not by more than 10%). Yet, the main point is that the algorithm exhibits the important quality of being robust to the prior specification. That is, it does not require extensive parameter tuning. Moreover, it leads to more parsimonious models than the ones previously reported.

The reversible jump simulated annealing algorithms with the AIC and MDL criteria were also tested on this problem. The results for the MDL criterion are depicted in Figure 8.9. We notice that the posterior increases stochastically with the number of iterations and, eventually, converges to a maximum. The figure also illustrates the convergence of the train and test set errors for each network in the Markov chain. The final network was the one that maximised the posterior. This network consisted of 12

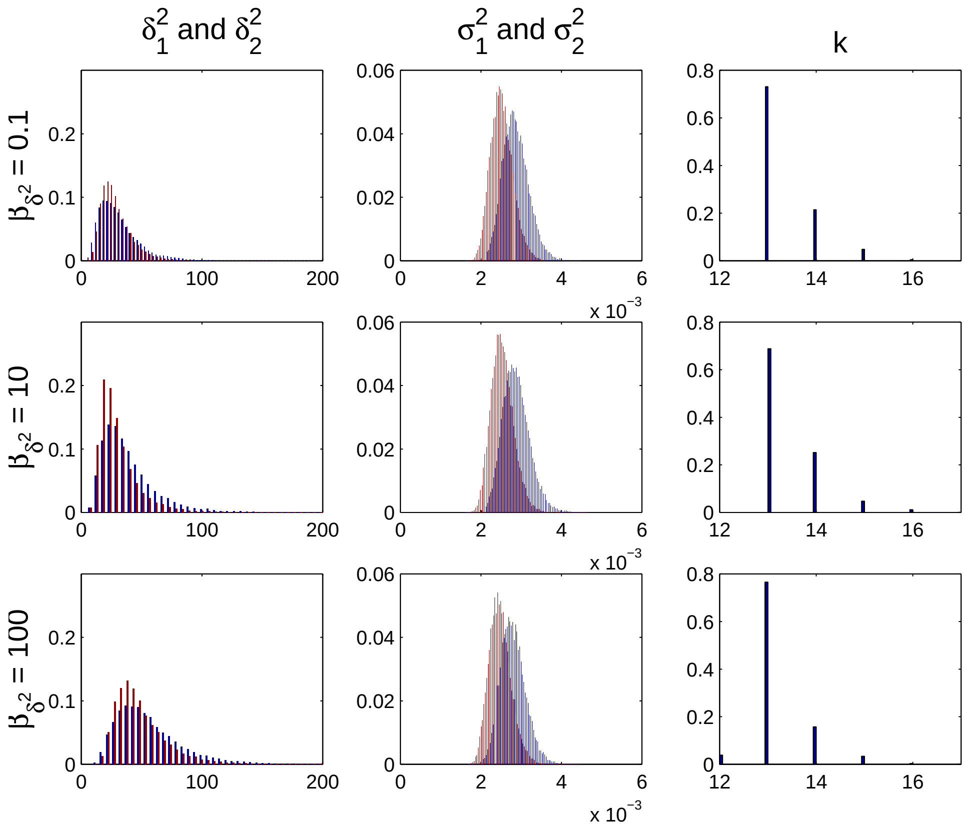

Figure 8.8 *Probabilities of smoothness constraints for each output ($\boldsymbol{\delta}_1$ and $\boldsymbol{\delta}_2$), noise variances ($\boldsymbol{\sigma}^2_{1,k}$ and $\boldsymbol{\sigma}^2_{2,k}$) and model order ($k$) for the robot arm data simulation using 3 different values for β_{δ^2}. The plots confirm that the algorithm is robust to the setting of β_{δ^2}.*

basis functions and incurred an error of 0.00512 in the test set. Following the same procedure, the AIC network consisted of 27 basis functions and incurred an error of 0.00520 in the test set. These results indicate that the full Bayesian model provides more accurate models. Moreover, it seems that the information criteria, in particular the AIC, can lead to over-fitting of the data.

These results confirm the well known fact that suboptimal techniques, that is the simulated annealing method with information criteria penalty terms and a rapid cooling schedule, can allow for faster computation at the expense of accuracy.

8.4 Experiment 4: Classification with Tremor Data

This section considers an interesting nonlinear classification data set[2] collected as part of a study to identify patients with muscle tremor (Roberts et al., 1996; Spyers-Ashby

[2] The data is available at `http://www.ee.ic.ac.uk/hp/staff/sroberts.html`

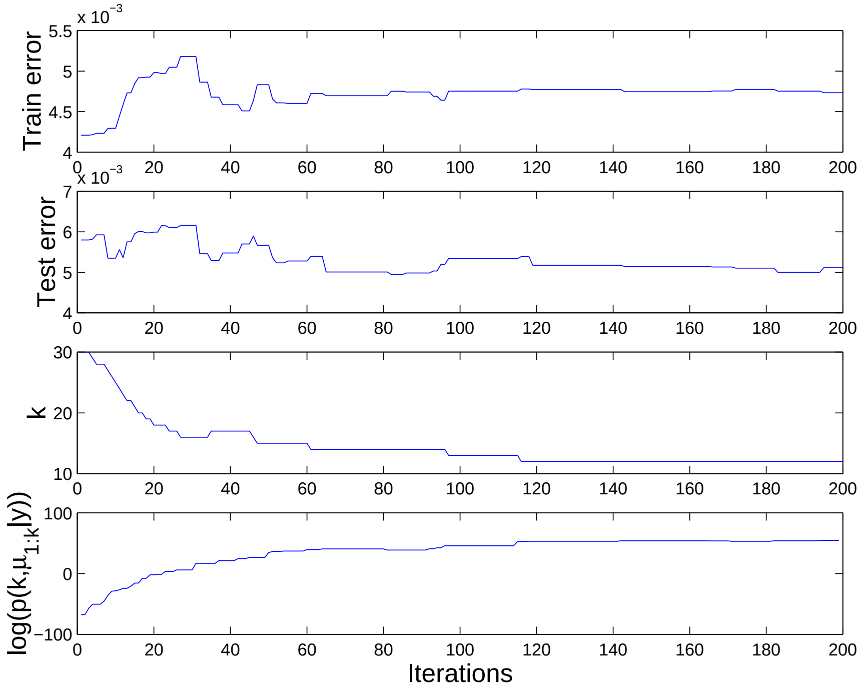

Figure 8.9 *Performance of the reversible jump simulated annealing algorithm for 200 iterations on the robot arm data, with the MDL criterion.*

et al., 1998). The data was gathered from a group of patients (9 with, primarily, Parkinson's disease or multiple sclerosis) and from a control group (not exhibiting the disease). Arm muscle tremor was measured with a 3-D mouse and a movement tracker in three linear and three angular directions. The time series of the measurements were parameterised using a set of autoregressive models. The number of features was then reduced to two (Roberts et al., 1996).

Figure 8.10 shows a plot of these features for patient (∘) and control groups (+). The figure also shows the decision boundaries (solid lines) and confidence intervals (dashed lines) obtained with an MLP, consisting of 10 sigmoidal hidden neurons and an output linear neuron. It needs to be pointed out, however, that having an output linear neuron leads to a classification framework based on discriminants. An alternative and more principled approach, which is not pursued in the batch learning case, is to use a logistic output neuron so that the classification scheme is based on probabilities of class membership. In the sequential learning case (see Chapter 6), an elegant Monte Carlo solution to this problem was presented. It is also possible to extend the batch methods to this probabilistic classification setting by adopting the generalised linear models framework with logistic, probit or softmax link functions (Gelman et al., 1995; Holmes, 1999; Nabney, 1999).

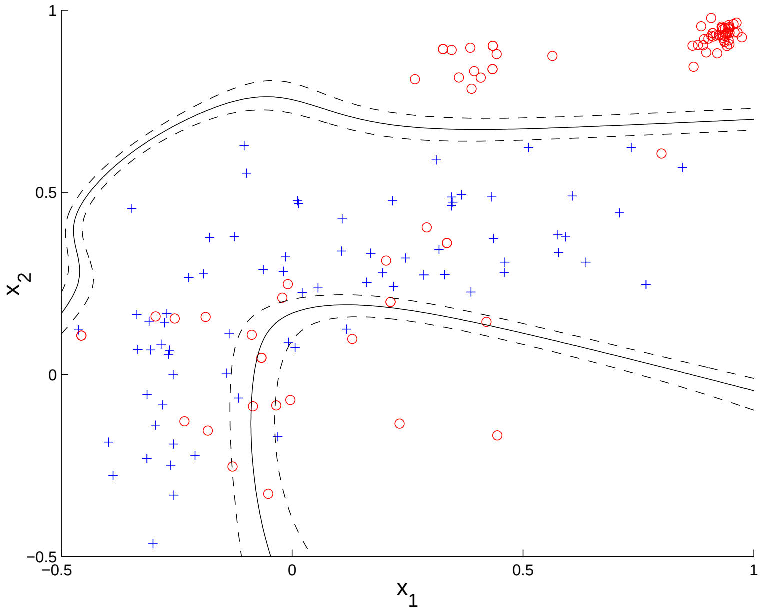

Figure 8.10 *Classification boundaries (–) and confidence intervals (- -) for the MLP classifier. The circles indicate patients, while the crosses represent the control group.*

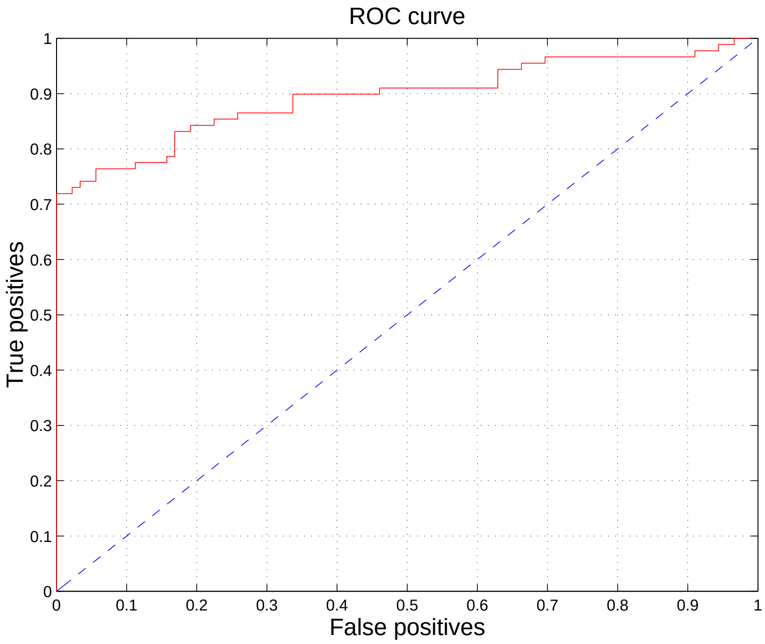

Figure 8.11 *Receiver operating characteristic (ROC) of the classifier for the tremor data.*

The size of the confidence intervals in Figure 8.10 is given by the innovations co-

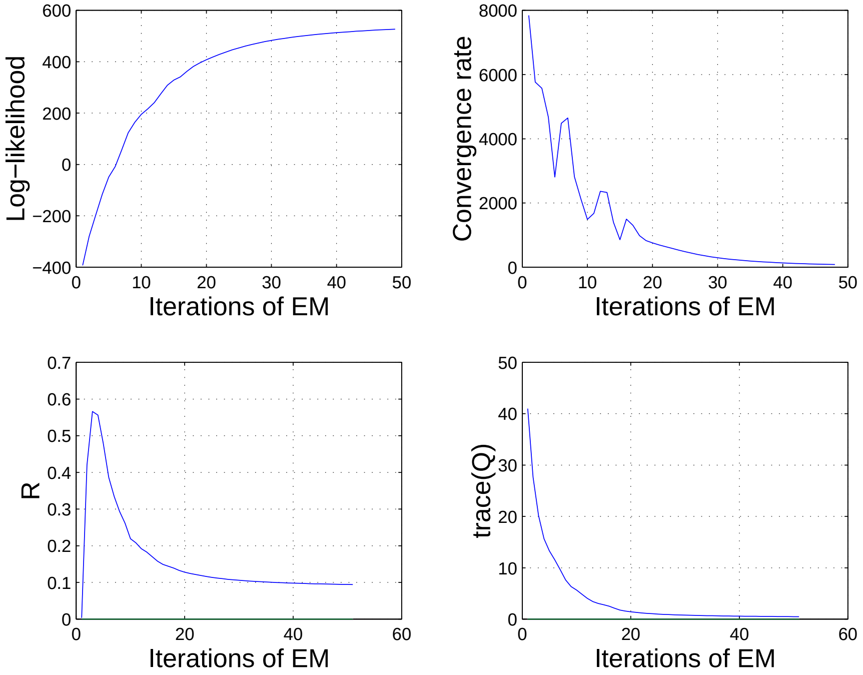

Figure 8.12 *The top plots show the log-likelihood function and the convergence rate (log-likelihood slope) for the tremor data classification problem. The bottom plots show the convergence of the measurements noise covariance R and the trace of the process noise covariance Q.*

variance. These intervals are a measure of uncertainty on the threshold that is applied to the linear output neuron. Our confidence of correctly classifying a sample occurring within these intervals should be very low. The receiver operating characteristic (ROC) curve, shown in Figure 8.11, indicates that we can expect to detect patients with a 70% confidence without making any mistakes. The percentage of classification errors in the test set was found to be 15.17. This error is of the same magnitude as previous results (Roberts and Penny, 1998). The convergence properties of the EM algorithm for this application are illustrated in Figure 8.12.

Figure 8.13 shows the decision boundaries (solid lines) and confidence intervals (dashed lines) obtained with the RJ-MCMC algorithm, using thin-plate spline hidden neurons and an output linear neuron. The size of the confidence intervals for the decision boundary is given by the noise variance (σ^2). The posterior mean ROC curve, shown in Figure 8.14, was obtained by averaging all the predictions for each classifier in the Markov chain. An alternative solution is to use the convex hull of the ROC curves for each classifier in the chain. This would yield the maximum realisable classifier according to the Neyman Pearson criterion (Andrieu et al., 1999c; Scott et al., 1998). The percentage of classification errors in the test set was found to be 14.60. Finally, the

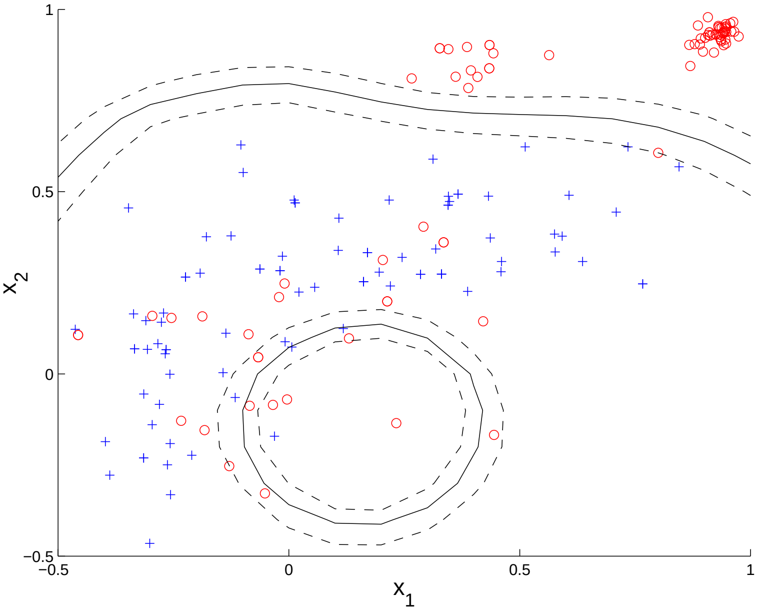

Figure 8.13 *Classification boundaries (–) and confidence intervals (- -) for the RBF classifier. The circles indicate patients, while the crosses represent the control group.*

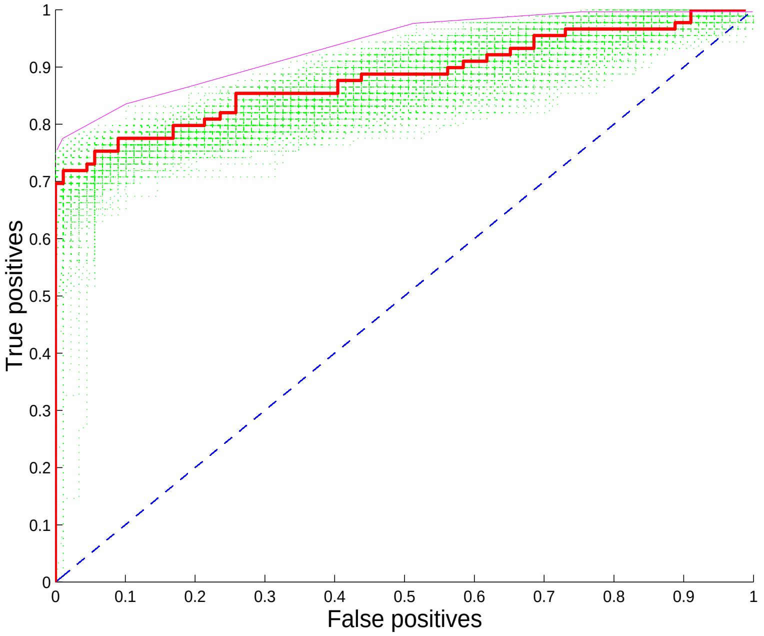

Figure 8.14 *Receiver operating characteristic (ROC) of the classifier for the tremor data. The solid line is the ROC curve for the posterior mean classifier, while the dotted lines correspond to the curves obtained for various classifiers in the Markov chain. The plot also shows the convex hull classifier.*

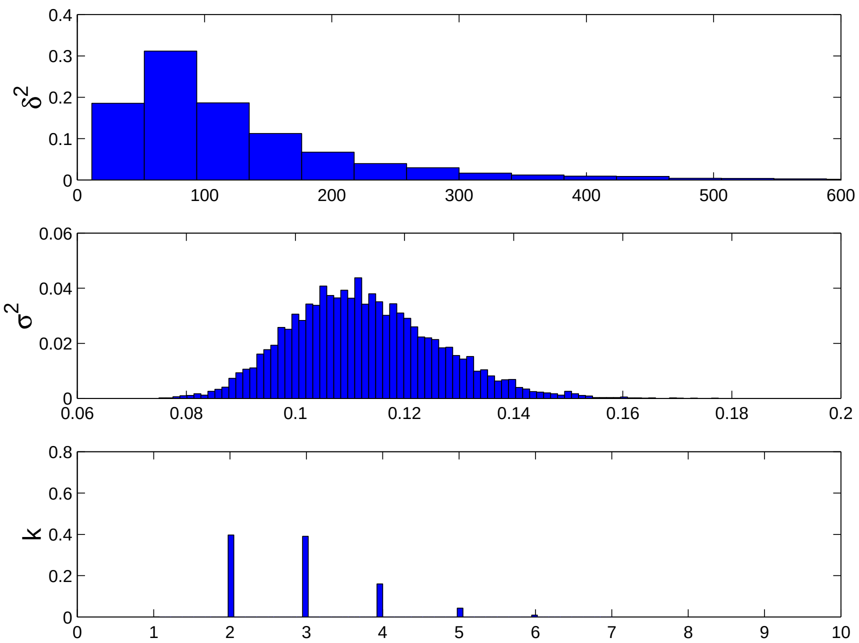

Figure 8.15 *Estimated probabilities of the signal to noise ratio (δ^2), noise variance (σ^2) and model order (k) for the classification example.*

estimated probabilities of the signal to noise ratio (δ^2), noise variance (σ^2) and model order (k) for this application are depicted in Figure 8.15.

8.5 Discussion

The experiments with the financial time series showed how to adopt a sequential learning framework to generate one-step-ahead predictions. Within this formulation, the algorithms seemed to yield similar results. It might, therefore, be argued that the EKF, by virtue of its simplicity and computational efficiency, provides the best solution. Yet, at a small increase in computational cost, the HySIR algorithm produces more accurate results. We begin noticing the power of SMC methods when we introduce non-Gaussian distributions and model selection schemes. SMC methods are still in their infancy, but considering the flexibility that they bring into the analysis, they should play a more significant role in future research.

The topic of on-line noise estimation, despite being the subject of a voluminous literature, still requires more research. Fixed-lag smoothing with SMC methods might provide a more satisfactory solution to this problem.

In the robot arm and tremor data sets, it was found that the EM algorithm performed reasonably well in terms of accuracy and computational requirements. From a convergence assessment point of view, this method proves to be very convenient. One simply has to monitor the rate of increase in the log-likelihood. In both data sets, however, the reversible jump MCMC method consistently yielded more accurate results. This shows that global search methods can be more precise than stochastic gradient descent methods.

Contrary to previously reported results, the experiments indicate that the Bayesian model with the reversible jump MCMC algorithm is robust with respect to the specification of the prior. In addition, it resulted in more parsimonious networks and better approximation errors than the ones previously reported in the literature. The resulting smaller networks are the result of the fact that the reversible jump MCMC algorithm only adds new neurons (new parameter dimensions) when they are required. It can then proceed to eliminate them if they no longer serve a purpose. On the other hand, fixed dimension methods, such as the EM algorithm proposed in Chapter 4, need to start with a large number of neurons to avoid getting trapped in local minima.

These experiments also indicate that the reversible jump simulated annealing algorithm with classical information criteria is an efficient and accurate variable-dimension, stochastic optimisation strategy.

The experiments presented in this chapter are by no means exhaustive. Yet, in conjunction with the the synthetic examples at the end of Chapters 3 to 7, they provide significant empirical support for the various proposed methods and algorithms. The software for most of the algorithms presented in this thesis has been made available in the internet at `http://www-svr.eng.cam.ac.uk/~jfgf`, so that they may be further tested on real applications. The EKFQ algorithm has already been adopted by researchers in the area of EEG segmentation (Penny and Roberts, 1998).

9

Conclusions

The Bayesian paradigm has extended the horizon of the field of learning for neural networks to fully embrace probability. In this new, mathematically rigorous context, it is possible to manage the uncertainty that arises in regression, prediction and classification. This paradigm, therefore, offers a general and coherent way of modelling many of the statistical aspects of real data sets, including persistence, outliers, variance, kurtosis and non-stationarity. It also enables us to perform parameter estimation and model selection jointly. Accordingly, one can automate the tasks of selecting the number of neurons, the type of basis functions and the number of inputs.

The Bayesian paradigm provides various mechanisms to incorporate *a priori* and domain specific knowledge into the learning process. These are of paramount importance. Without them, it is not possible to treat the ill-posed nature of the learning problem in a satisfactory manner.

The application of the Bayesian learning approach requires the evaluation of complex multi-dimensional integrals over the support of a probability distribution. The solution to this integration problem is far from trivial. Systematic numerical integration techniques prove to be prohibitively expensive. A more parsimonious strategy is to approximate the distribution in the integral and, subsequently, perform analytical integration. This approach may fail to embody all the richness of information available in the model and the data. A third alternative is to apply Monte Carlo methods to approximate the integrals by discrete sums over the regions of high probability.

In the past, various approaches based on Gaussian approximation and Monte Carlo methods have been proposed to solve the learning problem for neural networks (Buntine and Weigend, 1991; Holmes and Mallick, 1998; Mackay, 1992b; Marrs, 1998; Neal, 1996; Rios Insua and Müller, 1998). Despite these efforts, we are still far from reaching the summit of the trail. In this study, it is hoped that a significant distance has been covered.

We started the journey by exploring the potential of the Bayesian paradigm for sequential learning with neural networks. In doing so, a unifying framework was formulated for regularisation with smoothing priors, adaptive gradient descent methods and process noise estimation. A link between the evidence maximisation framework and many ideas developed much earlier in the control field was also established. This connection provided the basis for the formulation of an efficient EKF algorithm, where regularisation was achieved using noise estimation techniques.

The thesis went on to advance the field of Bayesian methods for sequential learning by introducing SMC techniques into the analysis of neural network models. Although these techniques tend to be more computationally demanding than Gaussian approximation schemes, there are four sound arguments for embracing them. Firstly, some distributions cannot be well approximated by Gaussian functions. Two examples of this were presented, namely sequential classification and robust time series analysis. In the former, the probability of class membership is either binomial or multinomial, while in the latter, it is advantageous to use heavy-tailed distributions to handle outliers. Secondly, the global search nature of SMC methods allows them to converge in fewer time steps than gradient descent methods. This was demonstrated by means of a simple example in Chapter 6. Thirdly, in many situations, the Taylor series approximations of the EKF may lead to poor representations of the measurements model. Examples of this limitation abound in the SMC literature. Finally, SMC methods tend to be more flexible. They can be easily modified to incorporate heuristic knowledge and perform noise estimation, model selection, smoothing and prediction.

The main drawback of SMC methods is their computational complexity. It is difficult to assess, *a priori*, the number of particles that a specific application will require. Yet it may be argued that the methods can be implemented in parallel. Moreover, if we have the right mathematical description of the distribution of the data, the computational requirements tend to be reasonably low.

In the batch learning scenario, the same trade-off between Gaussian approximation and Monte Carlo methods was encountered. An EM algorithm was derived using Gaussian approximations to estimate the weights of an MLP, measurement noise variance and process noise variance jointly. This method is applicable to non-stationary data sets. If, on the other hand, the data sets are stationary, the process noise variance provides an indication of how well the model fits the data. In addition to being computationally efficient, the method seems to be very stable.

MCMC simulation methods made it possible to devise more complex and powerful algorithms. A Bayesian model for RBFs, in which the estimation results seem to be robust with respect to the prior specification, was derived. This model, in conjunction with a reversible jump MCMC algorithm, was used to compute the number of neurons, parameters, regularisation terms and noise statistics to a high degree of accuracy. The same framework can be easily extended to automate the selection of input variables and type of basis functions. In the latter case, however, it is not clear what type of prior will lead to better results. In the same line of work, an efficient stochastic optimisation method, using the reversible jump and simulated annealing algorithms, was also proposed. This method can generate accurate results at a reasonable computational cost.

An important aspect of the work with MCMC methods is that it was possible to present convergence proofs for the integration and optimisation strategies. These proofs should play a significant role in the theory of MCMC learning for neural networks.

One of the greatest advantages of working with MCMC methods is their flexibility. It was shown how it is possible to incorporate domain specific knowledge using mixtures of Metropolis-Hastings steps. That is, instead of abandoning well known heuristics, it is possible to place these within a rigorous probabilistic setting. This applies to both batch learning and sequential learning problems.

A

Bayesian Derivation of the Kalman Filter

Consider the following linear Gauss-Markov process:

$$\begin{aligned} \boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t + \mathbf{u}_t & \text{(A.1)} \\ \mathbf{y}_t &= H_t \boldsymbol{\theta}_t + \mathbf{v}_t & \text{(A.2)} \end{aligned}$$

where $\mathbf{u}_t \sim \mathcal{N}(0, Q_t)$ denotes the process noise, $\mathbf{v}_t \sim \mathcal{N}(0, R_t)$ the measurement noise, $\mathbf{y}_{1:t} = \{\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \dots, \mathbf{y}_t\}$ the observations and $\boldsymbol{\theta}_t$ the latent states (the model parameters in our case). It is possible to apply Bayes' rule to estimate the posterior density function for the model parameters after the new data arrives:

$$p(\boldsymbol{\theta}_{t+1}|Y_{t+1}) = \frac{p(\mathbf{y}_{t+1}|\boldsymbol{\theta}_{t+1})}{p(\mathbf{y}_{t+1}|\mathbf{y}_{1:t})} p(\boldsymbol{\theta}_{t+1}|\mathbf{y}_{1:t}) \tag{A.3}$$

That is,

$$\text{Posterior} = \frac{\text{Likelihood}}{\text{Evidence}} \text{Prior}$$

Assuming Gaussian approximations to the probability density functions and that the noise processes are independent of each other and independent of the states, representations for the likelihood, evidence and prior can be derived in terms of the first and second order statistics (Anderson and Moore, 1979).

A.1 Prior Gaussian Density Function

The mean and covariance are given by:

$$\begin{aligned} \mathbb{E}(\boldsymbol{\theta}_{t+1}|\mathbf{y}_{1:t}) &= \mathbb{E}\big(\boldsymbol{\theta}_t + \mathbf{u}_t|\mathbf{y}_{1:t}\big) \\ &= \mathbb{E}\big(\boldsymbol{\theta}_t|\mathbf{y}_{1:t}\big) \\ &= \widehat{\boldsymbol{\theta}}_t \end{aligned}$$

and:

$$\begin{aligned}\text{Cov}(\boldsymbol{\theta}_{t+1}|\mathbf{y}_{1:t}) &= \mathbb{E}\big((\boldsymbol{\theta}_t + \mathbf{u}_t - \widehat{\boldsymbol{\theta}}_t)(\boldsymbol{\theta}_t + \mathbf{u}_t - \widehat{\boldsymbol{\theta}}_t)'|\mathbf{y}_{1:t}\big) \\ &= P_t + Q_t + 2\mathbb{E}\big((\boldsymbol{\theta}_t - \widehat{\boldsymbol{\theta}}_t)\mathbf{u}_t|\mathbf{y}_{1:t}\big) \\ &= P_t + Q_t\end{aligned}$$

Hence, the prior is:

$$\text{Prior} = p(\boldsymbol{\theta}_{t+1}|\mathbf{y}_{1:t}) = \mathcal{N}(\widehat{\boldsymbol{\theta}}_t, P_t + Q_t)$$

A.2 Evidence Gaussian Density Function

The mean is given by:

$$\begin{aligned}\mathbb{E}(\mathbf{y}_{t+1}|\mathbf{y}_{1:t}) &= \mathbb{E}\big(H_{t+1}\boldsymbol{\theta}_{t+1} + \mathbf{v}_{t+1}|\mathbf{y}_{1:t}\big) \\ &= H_{t+1}\mathbb{E}\big(\boldsymbol{\theta}_{t+1}|\mathbf{y}_{1:t}\big) \\ &= H_{t+1}\widehat{\boldsymbol{\theta}}_{t+1|t} \\ &= H_{t+1}\widehat{\boldsymbol{\theta}}_t\end{aligned}$$

and the covariance is given by:

$$\begin{aligned}\text{Cov}(\mathbf{y}_{t+1}|\mathbf{y}_{1:t}) &= \mathbb{E}\big((H_{t+1}\boldsymbol{\theta}_{t+1} + \mathbf{v}_{t+1} - H_{t+1}\widehat{\boldsymbol{\theta}}_t) \\ &\quad (H_{t+1}\boldsymbol{\theta}_{t+1} + \mathbf{v}_{t+1} - H_{t+1}\widehat{\boldsymbol{\theta}}_t)'|\mathbf{y}_{1:t}\big) \\ &= \mathbb{E}\big((H_{t+1}(\boldsymbol{\theta}_{t+1} - \widehat{\boldsymbol{\theta}}_t) + \mathbf{v}_{t+1})(H_{t+1}(\boldsymbol{\theta}_{t+1} - \widehat{\boldsymbol{\theta}}_t) + \mathbf{v}_{t+1})'|\mathbf{y}_{1:t}\big) \\ &= H_{t+1}(P_t + Q_t)H'_{t+1} + R_{t+1}\end{aligned}$$

Hence, the evidence is given by:

$$\text{Evidence: } p(\mathbf{y}_{t+1}|\mathbf{y}_{1:t}) = \mathcal{N}(H_{t+1}\widehat{\boldsymbol{\theta}}_{t+1}, H_{t+1}(P_t + Q_t)H'_{t+1} + R_{t+1})$$

A.3 Likelihood Gaussian Density Function

The mean is given by:

$$\begin{aligned}\mathbb{E}(\mathbf{y}_{t+1}|\boldsymbol{\theta}_{t+1}) &= \mathbb{E}\big(H_{t+1}\boldsymbol{\theta}_{t+1} + \mathbf{v}_{t+1}|\boldsymbol{\theta}_{t+1}\big) \\ &= H_{t+1}\mathbb{E}\big(\boldsymbol{\theta}_{t+1}|\boldsymbol{\theta}_{t+1}\big) \\ &= H_{t+1}\boldsymbol{\theta}_{t+1}\end{aligned}$$

while the covariance is given by:

$$\begin{aligned}\text{Cov}(\mathbf{y}_{t+1}|\boldsymbol{\theta}_{t+1}) &= \mathbb{E}\big((H_{t+1}\boldsymbol{\theta}_{t+1} + \mathbf{v}_{t+1} - H_{t+1}\boldsymbol{\theta}_{t+1}) \\ &\quad (H_{t+1}\boldsymbol{\theta}_{t+1} + \mathbf{v}_{t+1} - H_{t+1}\boldsymbol{\theta}_{t+1})'|\boldsymbol{\theta}_{t+1}\big) \\ &= R_{t+1}\end{aligned}$$

Hence, the expression for the likelihood is:

$$\text{Likelihood: } p(\mathbf{y}_{t+1}|\boldsymbol{\theta}_{t+1}, M_j, R_t, Q_t) = \mathcal{N}(H_{t+1}\boldsymbol{\theta}_{t+1}, R_{t+1})$$

A.4 Posterior Gaussian Density Function

Substituting the equations for the means and covariances of the evidence, prior and likelihood into equation (A.3) and completing squares in the exponent of the Gaussian exponential, yields the posterior density function:

$$p(\boldsymbol{\theta}_{t+1}|Y_{t+1}, M_j, R_t, Q_t) = A_{t+1} \exp\Big(-\frac{1}{2}(\boldsymbol{\theta}_{t+1} - \widehat{\boldsymbol{\theta}}_{t+1})P_{t+1}^{-1}(\boldsymbol{\theta}_{t+1} - \widehat{\boldsymbol{\theta}}_{t+1})\Big) \qquad \text{(A.4)}$$

where the coefficients A_{t+1} are represented by the following expression:

$$A_{t+1} = \frac{|H_{t+1}(P_t + Q_t)H'_{t+1} + R_{t+1}|^{1/2}}{(2\pi)^{m/2}|R_{t+1}|^{1/2}|P_t + Q_t|^{1/2}}$$

and $\widehat{\boldsymbol{\theta}}_{t+1}$ and P_{t+1} are given by:

$$\widehat{\boldsymbol{\theta}}_{t+1} = \widehat{\boldsymbol{\theta}}_t + K_{t+1}(\mathbf{y}_{t+1} - H_{t+1}\widehat{\boldsymbol{\theta}}_t) \qquad \text{(A.5)}$$

$$P_{t+1} = P_t + Q_t - K_{t+1}H_{t+1}(P_t + Q_t) \qquad \text{(A.6)}$$

where K_t is known as the Kalman gain:

$$K_{t+1} = (P_t + Q_t)H'_{t+1}\big(R_{t+1} + H_{t+1}(P_t + Q_t)H'_{t+1}\big)^{-1} \qquad \text{(A.7)}$$

B

Computing Derivatives for the Jacobian Matrix

For the network depicted in Figure B.1, the output layer mapping is given by:

$$y = \theta_1 + \theta_2 o_{11} + \theta_3 o_{12}$$

and consequently, the derivatives with respect to the weights are given by:

$$\begin{aligned}\frac{\partial y}{\partial \theta_1} &= 1\\ \frac{\partial y}{\partial \theta_2} &= o_{11}\\ \frac{\partial y}{\partial \theta_3} &= o_{12}\end{aligned}$$

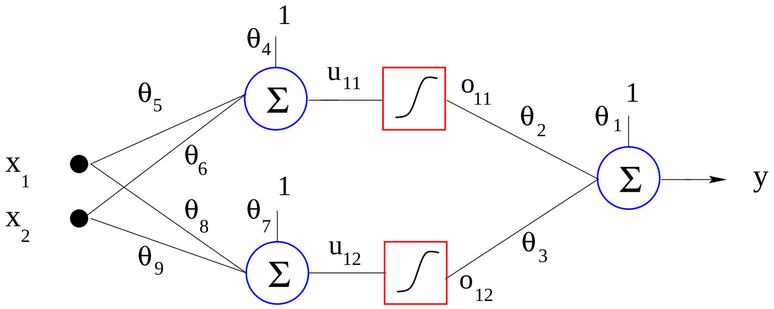

Figure B.1 *Simple MLP structure for regression problems.*

The hidden layer mapping for the top neuron is:

$$o_{11} = \frac{1}{1 + \exp(-u_{11})}, \qquad \text{where } u_{11} = \theta_4 + \theta_5 x_1 + \theta_6 x_2$$

The corresponding derivatives with respect to the weights are:

$$\begin{aligned}
\frac{\partial y}{\partial \theta_4} &= \frac{\partial y}{\partial o_{11}} \frac{\partial o_{11}}{\partial u_{11}} \frac{\partial u_{11}}{\partial \theta_4} \\
&= \theta_2 o_{11}(1 - o_{11}) \\
\frac{\partial y}{\partial \theta_5} &= \theta_2 o_{11}(1 - o_{11}) x_1 \\
\frac{\partial y}{\partial \theta_6} &= \theta_2 o_{11}(1 - o_{11}) x_2
\end{aligned} \tag{B.1}$$

The derivatives with respect to the weights of the other hidden layer neuron can be calculated following the same trivial procedure.

C

An Important Inequality

This appendix proves an important inequality that arises in the derivation of the EM algorithm (Baum et al., 1979):

$$\mathbb{E}\big(\ln p(\boldsymbol{\theta}|\mathbf{y},\boldsymbol{\varphi}^{\text{old}})\big) \geq \mathbb{E}\big(\ln p(\boldsymbol{\theta}|\mathbf{y},\boldsymbol{\varphi})\big)$$

where the expectations are taken as follows:

$$\int \big(\ln p(\boldsymbol{\theta}|\mathbf{y},\boldsymbol{\varphi}^{\text{old}})\big) p(\boldsymbol{\theta}|\mathbf{y},\boldsymbol{\varphi}^{\text{old}})\mathrm{d}\boldsymbol{\theta} \geq \int \big(\ln p(\boldsymbol{\theta}|\mathbf{y},\boldsymbol{\varphi})\big) p(\boldsymbol{\theta}|\mathbf{y},\boldsymbol{\varphi}^{\text{old}})\mathrm{d}\boldsymbol{\theta} \tag{C.1}$$

We begin by noticing that the function $x - 1$ is tangent to the function $\ln x$ at $x = 1$. In addition, the logarithmic function is concave, hence the following inequality holds:

$$\ln x \leq x - 1$$

As a result, it follows that:

$$\begin{aligned}
\mathbb{E}\big(\ln p(\boldsymbol{\theta}|\mathbf{y},\boldsymbol{\varphi})\big) - \mathbb{E}\big(\ln p(\boldsymbol{\theta}|\mathbf{y},\boldsymbol{\varphi}^{\text{old}})\big) &= \int \big(\ln p(\boldsymbol{\theta}|\mathbf{y},\boldsymbol{\varphi}) - \ln p(\boldsymbol{\theta}|\mathbf{y},\boldsymbol{\varphi}^{\text{old}})\big) p(\boldsymbol{\theta}|\mathbf{y},\boldsymbol{\varphi}^{\text{old}})\mathrm{d}\boldsymbol{\theta} \\
&= \int \ln \frac{p(\boldsymbol{\theta}|\mathbf{y},\boldsymbol{\varphi})}{p(\boldsymbol{\theta}|\mathbf{y},\boldsymbol{\varphi}^{\text{old}})} p(\boldsymbol{\theta}|\mathbf{y},\boldsymbol{\varphi}^{\text{old}})\mathrm{d}\boldsymbol{\theta} \\
&\leq \int \Big[\frac{p(\boldsymbol{\theta}|\mathbf{y},\boldsymbol{\varphi})}{p(\boldsymbol{\theta}|\mathbf{y},\boldsymbol{\varphi}^{\text{old}})} - 1 \Big] p(\boldsymbol{\theta}|\mathbf{y},\boldsymbol{\varphi}^{\text{old}})\mathrm{d}\boldsymbol{\theta} \\
&= 0
\end{aligned}$$

Hence:

$$\mathbb{E}\big(\ln p(\boldsymbol{\theta}|\mathbf{y},\boldsymbol{\varphi}^{\text{old}})\big) \geq \mathbb{E}\big(\ln p(\boldsymbol{\theta}|\mathbf{y},\boldsymbol{\varphi})\big)$$

D

An Introduction to Fixed and Variable Dimension MCMC

The objective of this appendix is to present a first-principles and "easy to understand" derivation of the Metropolis-Hastings and reversible jump Markov chain Monte Carlo (MCMC) algorithms (Metropolis et al., 1953; Green, 1995). These algorithms are of particular relevance to people working in fields where the estimation of parameters or latent variables is of paramount importance. These fields include machine learning, applied statistics, econometrics, forecasting, signal processing, communications, control and neural networks, among others.

The Metropolis-Hastings algorithm is useful to estimate, among other things, the posterior distribution of the model parameters in a Bayesian context. In doing so, it is possible to compute many quantities of interest such as the mean, modes and error bars on the estimates. The reversible jump algorithm allows one to cast the net much further. It estimates not only the unknown quantities but also the number of unknown quantities. Typical examples include estimating the number of neurons in a neural network (Andrieu et al., 1999d; Holmes and Mallick, 1998; Rios Insua and Müller, 1998), the number of sinusoids in a noisy signal (Andrieu, 1998), the number of lags in an autoregressive process (Troughton and Godsill, 1998), the number of components in a mixture (Richardson and Green, 1997) and the number of levels in a change-point process (Green, 1995).

In making the derivations of the algorithms more accessible, their usage in the engineering and computer science communities should increase. There are a few reasons for this premise. Firstly, people tend to feel more comfortable applying tools that they can perfectly understand. Secondly, the derivation sheds light on routes for modifying or tuning the algorithms to specific applications. Finally, a fair understanding of the pivotal assumptions, involved in the derivation, allows one to avoid incurring many common mistakes.

There is, nonetheless, a price to pay for this perspicuity in terms of generality and mathematical rigour. With this in mind, readers are encouraged to consult the excellent texts of (Gilks et al., 1996; Meyn and Tweedie, 1993; Robert and Casella, 1999) and papers by (Andrieu et al., 1999f; Besag et al., 1995; Brooks, 1998; Tierney, 1994). In addition, the appendix concentrate on the derivations of the algorithms so as to keep the discussion manageable. Topics such as applications, empirical convergence assessment techniques and algorithmic details are beyond the scope of this appendix. Chapters 5 to 7 and the afore-cited references provide valuable information on these topics.

The need for MCMC simulation is justified in Section D.1. Section D.2 introduces the concepts of measure, densities, distributions and σ-fields. Subsequently, Section D.3 defines Markov chains and describes some of the properties that play a fundamental role in MCMC simulation. Sections D.4 and D.5 present the Metropolis-Hastings and reversible jump MCMC algorithms respectively.

D.1 Why MCMC?

MCMC techniques are a set of powerful simulation methods that may be applied to solve integration and optimisation problems in large dimensional spaces. These two types of problems are the major stumbling blocks of Bayesian statistics and decision analysis. For instance, given some unknown variables $\boldsymbol{\theta} \in \mathbb{R}^m$ and data $\mathbf{y} \in \mathbb{R}^c$, the following three integration problems arise in Bayesian analysis:

Normalisation: To obtain the posterior distribution $p(\boldsymbol{\theta}|\mathbf{y})$ given the prior $p(\boldsymbol{\theta})$ and likelihood $p(\mathbf{y}|\boldsymbol{\theta})$, the normalising factor in Bayes' theorem needs to be computed:

$$p(\boldsymbol{\theta}|\mathbf{y}) = \frac{p(\mathbf{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int_{\mathbb{R}^m} p(\mathbf{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta}}$$

Marginalisation: Given the joint posterior of $(\boldsymbol{\theta}, \boldsymbol{\varphi}) \in \mathbb{R}^m \times \mathbb{R}^n$, we may often be interested in the marginal posterior:

$$p(\boldsymbol{\theta}|\mathbf{y}) = \int_{\mathbb{R}^n} p(\boldsymbol{\theta}, \boldsymbol{\varphi}|\mathbf{y})\mathrm{d}\boldsymbol{\varphi}$$

Expectation: The objective of the analysis is often to obtain summary statistics of the form:

$$\mathbb{E}(f(\boldsymbol{\theta})|\mathbf{y}) = \int_{\mathbb{R}^m} f(\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathbf{y})\mathrm{d}\boldsymbol{\theta}$$

It is usually impossible to evaluate the above integrals analytically. To overcome this difficulty, we have to resort to analytical approximation, deterministic numerical integration or Monte Carlo simulation. Approximations tend to disregard some of the salient statistical features of the processes under consideration, thereby often leading to poor results. In addition, it is often difficult to quantify the effect of the approximations on the estimates. Deterministic numerical integration in high dimensions, although accurate, is far too computationally expensive to be of any practical use. As we shall see soon, Monte Carlo methods provide the middle ground.

The idea of Monte Carlo integration is to draw an i.i.d. sample $\{\boldsymbol{\theta}^{(i)}; i = 1, 2, \ldots, N\}$ from the target distribution $p(\boldsymbol{\theta})$ (it could be the posterior in Bayesian analysis) so as to approximate the integrals by discrete sums:

$$\overline{\mathbb{E}(f(\boldsymbol{\theta}))} = \frac{1}{N}\sum_{i=1}^{N} f(\boldsymbol{\theta}^{(i)}) \xrightarrow[N\to\infty]{a.s.} \mathbb{E}(f(\boldsymbol{\theta})) = \int_{\mathbb{R}^m} f(\boldsymbol{\theta})p(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta}$$

The estimate $\overline{\mathbb{E}(f(\boldsymbol{\theta}))}$ is unbiased and by the strong law of large numbers, it will almost surely converge to $\mathbb{E}(f(\boldsymbol{\theta}))$. That is:

$$P\left(\lim_{N\to\infty} \overline{\mathbb{E}(f(\boldsymbol{\theta}))} = \mathbb{E}(f(\boldsymbol{\theta}))\right) = 1$$

If the variance σ_f^2 of $f(\boldsymbol{\theta})$ is finite, a central limit theorem yields convergence in distribution of the error:

$$\lim_{N\to\infty} \sqrt{N}\left(\overline{\mathbb{E}(f(\boldsymbol{\theta}))} - \mathbb{E}(f(\boldsymbol{\theta}))\right) \to \mathcal{N}(0, \sigma_f^2)$$

Note that the rate of convergence of the Monte Carlo estimate is of the order $\mathcal{O}(N^{-1/2})$, while the corresponding error for the deterministic integration method is typically $\mathcal{O}(N^{-m/2})$. The Monte Carlo estimate is independent of the dimension of $\boldsymbol{\theta}$. It should be mentioned, however, that the constant of proportionality will often depend on m. The advantage of Monte Carlo integration over deterministic integration arises from the fact that the former positions the integration grid (samples) in regions of high probability, as shown in Figure D.1.

The main disadvantage of simple Monte Carlo methods is that often it is not possible to draw samples from $p(\boldsymbol{\theta})$ directly. This problem can, however, be circumvented by the introduction of MCMC algorithms. Assuming that we can draw samples from a proposal distribution $q(\boldsymbol{\theta})$, the key idea of MCMC simulation is to design mechanisms that cause the proposal samples to migrate, so that their distribution approximates $p(\boldsymbol{\theta})$. This can be demonstrated by means of a simple example. Suppose we wish to draw samples from the target distribution:

$$p(\theta) = \frac{0.3}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2\sigma^2}\theta^2\right) + \frac{0.7}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2\sigma^2}(\theta - 10)^2\right)$$

This task can be accomplished by starting, for example, with a set of uniformly distributed samples $\{\theta_0^{(i)} \sim \mathcal{U}_{[0,20]}; i = 1, \ldots, 1000\}$ and sampling from the proposal

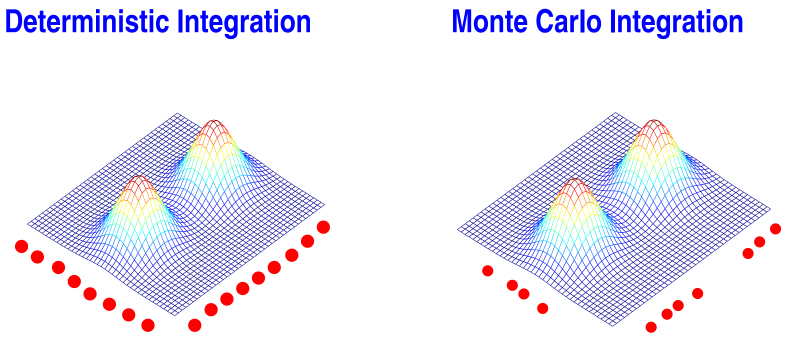

Figure D.1 *Deterministic versus Monte Carlo integration. The latter is more parsimonious in that it positions the integration grid (dots) in regions of high probability.*

$q(\theta) = \mathcal{N}(0, \sigma^{2\star})$ (see Figure D.2). An iterative mechanism that ensures that the samples become distributed according to $p(\theta)$ follows:

My first MCMC algorithm

```
1. Set $t = 0$ and for $i = 1, \ldots, 1000$ initialise $\theta_0^{(i)} \sim \mathcal{U}_{[0,20]}$
2. Iteration $t$, for $i = 1, \ldots, 1000$:
    • Sample $u \sim \mathcal{U}_{[0,1]}$.
    • Sample $\theta_t^{\star(i)}$ from $q(\theta_t) = \mathcal{N}(\theta_t^{(i)}, \sigma^{2\star})$.
    • If $u < \mathcal{A}(\theta_t^{(i)}, \theta_t^{\star(i)}) = \min\left\{1, \frac{\frac{0.3}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(\theta_t^{\star(i)})^2\right) + \frac{0.7}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(\theta_t^{\star(i)} - 10)^2\right)}{\frac{0.3}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(\theta_t^{(i)})^2\right) + \frac{0.7}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(\theta_t^{(i)} - 10)^2\right)}\right\}$
            $\theta_{t+1}^{(i)} = \theta_t^{\star(i)}$
      else
            $\theta_{t+1}^{(i)} = \theta_t^{(i)}$
3. $t \leftarrow t + 1$ and go to 2.
```

Figure D.3 shows that the distribution of the samples, for 9 iterations, $\sigma = 2$ and $\sigma^\star = 15$, converges to the the target distribution. Three important characteristics of the previous algorithm need to be emphasised. Firstly, the normalising constants of the target distribution are not required. We only need to know the target distribution up to a constant of proportionality. Secondly, although the algorithm makes use of 1000 chains, one single long chain would suffice to approximate the target distribution. Finally, the success or failure of the algorithm hinges on the choice of proposal

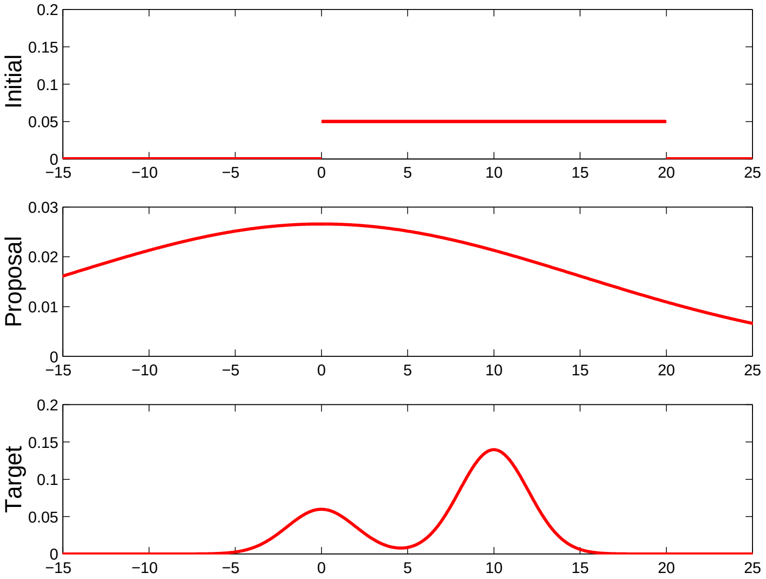

Figure D.2 *Initial, proposal and target distributions for the MCMC example.*

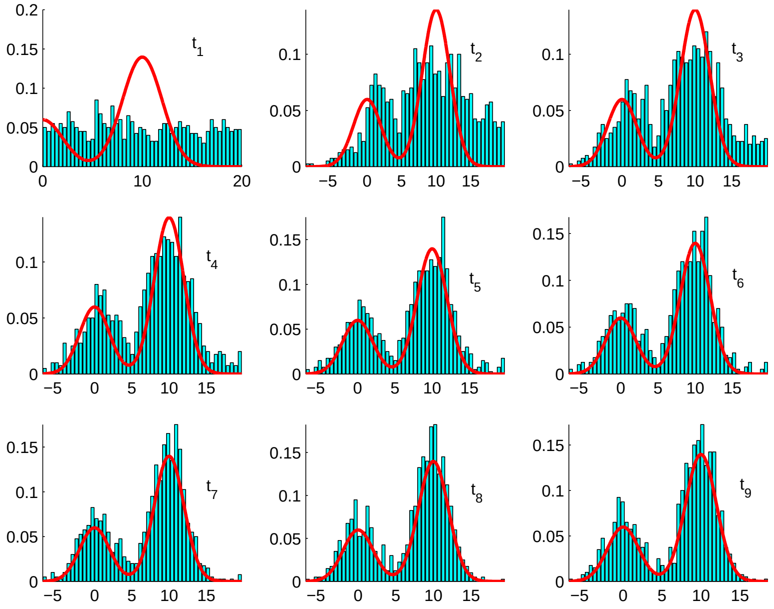

Figure D.3 *Approximating the target distribution in the MCMC example.*

distribution. As shown in Figure D.4, if the proposal is too narrow, only one mode of $p(\theta)$ might be visited. On the other hand, if it is too wide, the rejection rate can be very high. If all the modes are visited while the acceptance probability is high, the chain is said to “mix” well.

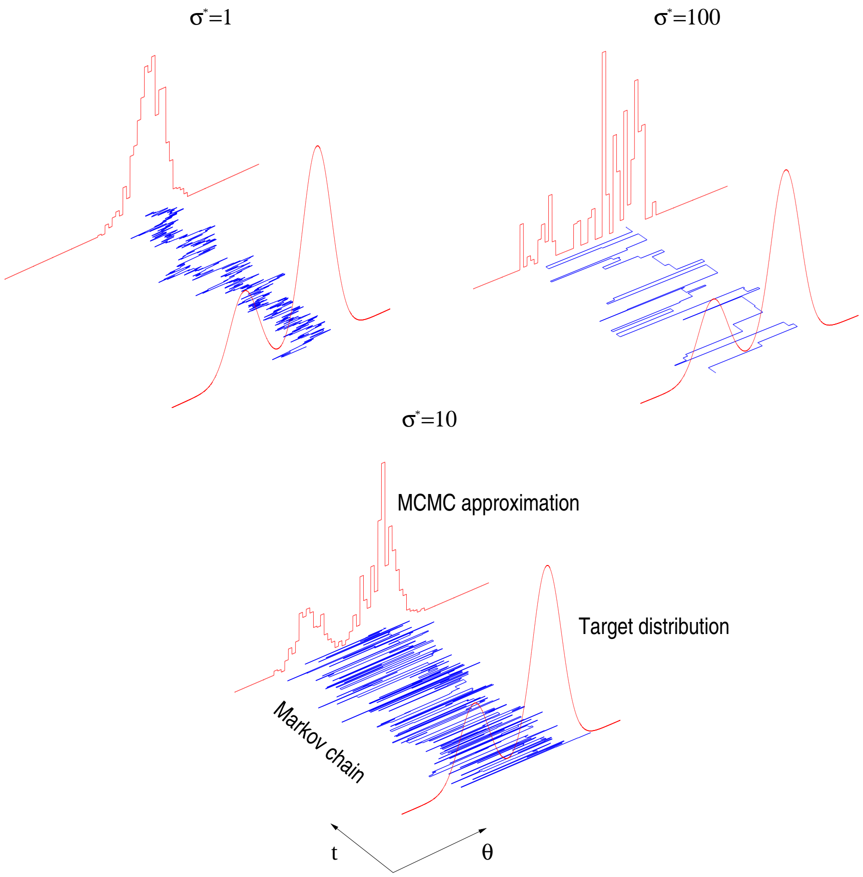

Figure D.4 *Approximations obtained for three proposals with different variances.*

A very powerful property of MCMC algorithms is that it is possible to combine several proposal distributions. For instance, if we suspect that the target distribution has two modes separated by a distance of ten units, we can adopt the following mixture of MCMC steps to improve the mixing:

Mixture of MCMC steps

```
1. Set $t = 0$ and for $i = 1, \dots, 1000$ initialise $\theta_0^{(i)} \sim \mathcal{U}_{[0,20]}$

2. Iteration $t$, for $i = 1, \dots, 1000$:

    • Sample $u \sim \mathcal{U}_{[0,1]}$ and $v \sim \mathcal{U}_{[0,1]}$.

    • if $v > 0.5$

        – Sample $\theta_t^{\star(i)}$ from $q(\theta_t) = \mathcal{N}(\theta_t^{(i)}, \sigma^{2\star})$.
        – If $u < \mathcal{A}(\theta_t^{(i)}, \theta_t^{\star(i)}) = \min\left\{1, \frac{p(\theta_t^{\star(i)}) \exp\left(-\frac{1}{2\sigma^{2\star}}(\theta_t^{(i)} - \theta_t^{\star(i)})^2\right)}{p(\theta_t^{(i)}) \exp\left(-\frac{1}{2\sigma^{2\star}}(\theta_t^{\star(i)} - \theta_t^{(i)})^2\right)}\right\}$
```

```
            $\theta_{t+1}^{(i)} = \theta_t^{\star(i)}$
        else
            $\theta_{t+1}^{(i)} = \theta_t^{(i)}$
  • Else if $v > 0.25$
    – Sample $\theta_t^{\star(i)}$ from $q(\theta_t) = \mathcal{N}(10 + \theta_t^{(i)}, \sigma^{2\star})$.
    – If $u < \mathcal{A}(\theta_t^{(i)}, \theta_t^{\star(i)}) = \min\left\{1, \frac{p(\theta_t^{\star(i)}) \exp\left(-\frac{1}{2\sigma^{2\star}}(\theta_t^{(i)} - 10 - \theta_t^{\star(i)})^2\right)}{p(\theta_t^{(i)}) \exp\left(-\frac{1}{2\sigma^{2\star}}(\theta_t^{\star(i)} - 10 - \theta_t^{(i)})^2\right)}\right\}$
            $\theta_{t+1}^{(i)} = \theta_t^{\star(i)}$
        else
            $\theta_{t+1}^{(i)} = \theta_t^{(i)}$
  • Else
    – Sample $\theta_t^{\star(i)}$ from $q(\theta_t) = \mathcal{N}(-10 + \theta_t^{(i)}, \sigma^{2\star})$.
    – If $u < \mathcal{A}(\theta_t^{(i)}, \theta_t^{\star(i)}) = \min\left\{1, \frac{p(\theta_t^{\star(i)}) \exp\left(-\frac{1}{2\sigma^{2\star}}(\theta_t^{(i)} + 10 - \theta_t^{\star(i)})^2\right)}{p(\theta_t^{(i)}) \exp\left(-\frac{1}{2\sigma^{2\star}}(\theta_t^{\star(i)} + 10 - \theta_t^{(i)})^2\right)}\right\}$
            $\theta_{t+1}^{(i)} = \theta_t^{\star(i)}$
        else
            $\theta_{t+1}^{(i)} = \theta_t^{(i)}$
3. $t \leftarrow t + 1$ and go to 2.
```

This algorithm is a particular implementation of the Metropolis-Hastings (MH) algorithm, which shall be presented in its more general form in Section D.4. The results obtained applying this algorithm are shown in Figure D.5. The mixture of MCMC steps consists of a local mode exploration step and steps to jump from one mode to the other. As a result, if the local exploration proposals are too narrow, as illustrated in the top left plot of figure D.4, the jump steps will allow the other modes to be explored. Mixtures of MCMC steps, therefore, permit the introduction of more ingenious strategies for exploring the parameter space. Indeed, one of the most powerful attributes of MCMC algorithms is that heuristics and domain specific knowledge can be incorporated using mixtures of MCMC steps. This allows us to obtain very accurate results at a reasonable computational cost.

MCMC methods are also suitable for optimisation. Here, the objective is to find the best among several alternatives. That is, we seek a decision with minimal cost, where, for convenience, the target distribution is assumed to be the cost function. The optimal choice is given by the peak of the target distribution (the maximum *a posteriori* (MAP) estimate in the Bayesian paradigm):

$$\boldsymbol{\theta}_{MAP} = \arg \max_{\boldsymbol{\theta} \in \mathbb{R}^m} p(\boldsymbol{\theta})$$

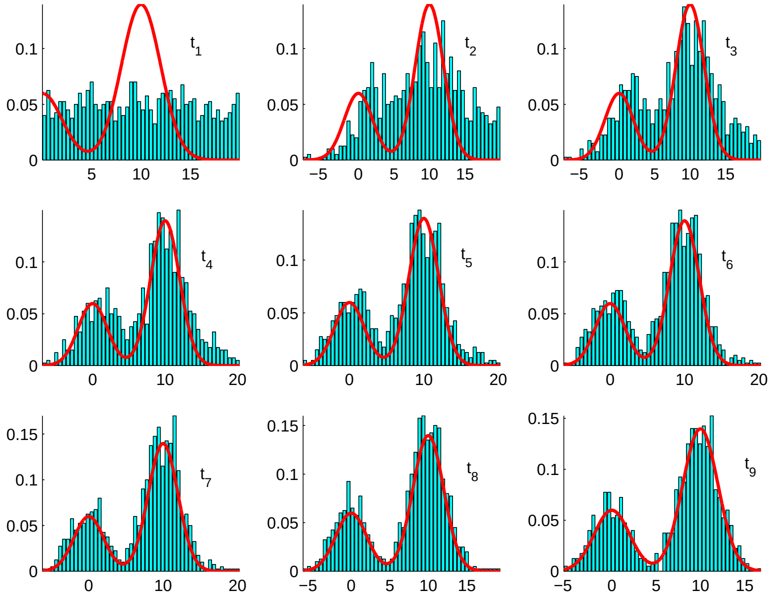

Figure D.5 *Approximating the target distribution with a mixture of MCMC steps.*

The following slight modification of our first algorithm achieves this goal:

MCMC optimisation algorithm

```
1. Set $t = 0$, $T_0 = 1$ and for $i = 1, \ldots, 1000$, initialise $\theta_0^{(i)} \sim \mathcal{U}_{[0,20]}$
2. Iteration $t$, for $i = 1, \ldots, 1000$:
   • Sample $u \sim \mathcal{U}_{[0,1]}$.
   • Sample $\theta_t^{\star(i)}$ from $q(\theta_t) = \mathcal{N}(\theta_t^{(i)}, \sigma^{2\star})$.
   • If $u < \mathcal{A}(\theta_t^{(i)}, \theta_t^{\star(i)}) = \min\left\{1, \frac{\frac{0.3}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2\sigma^2}(\theta_t^{\star(i)})^2\right)+\frac{0.7}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2\sigma^2}(\theta_t^{\star(i)}-10)^2\right)}{\frac{0.3}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2\sigma^2}(\theta_t^{(i)})^2\right)+\frac{0.7}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2\sigma^2}(\theta_t^{(i)}-10)^2\right)}\right\}^{\frac{1}{T_t}}$
         $\theta_{t+1}^{(i)} = \theta_t^{\star(i)}$
     else
         $\theta_{t+1}^{(i)} = \theta_t^{(i)}$
   • Set $T_{t+1} = 0.25 * T_t$.
3. $t \leftarrow t + 1$ and go to 2.
```

■

The results of applying this algorithm, known as simulated annealing (SA), are shown in Figure D.6. As can be seen, the samples are globally concentrated in the

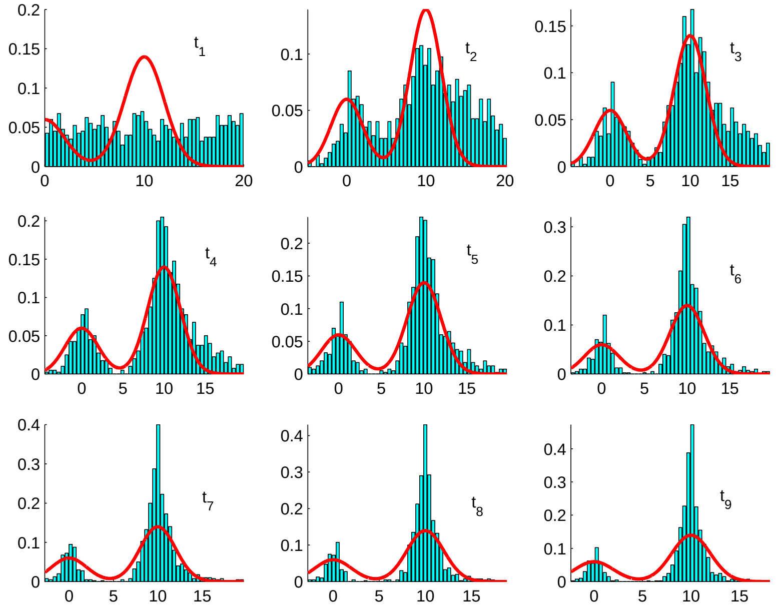

Figure D.6 *Discovering the modes of the target distribution with the simulated annealing algorithm.*

modes of the target distribution.

If the reader has been able to follow the presentation thus far, he or she should, without any major difficulty, be able to apply the MH and SA algorithms to a wide spectrum of problems. However, for complex applications, perhaps involving model selection, a reasonable grasp of some concepts of probability, measure theory and Markov chains is highly advisable. The following section reviews a few basic principles of probability and measure theory, which will make it possible to establish some important properties of Markov chains and, subsequently, present the MH and reversible jump MCMC algorithms in their general form.

D.2 Densities, Distributions, σ-Fields and Measures

The concepts of *density*, *distribution* and *measure* are essential to understand the derivations hereafter. A very simplified treatment is presented. Interested readers are advised to consult (Billingsley, 1985) for an in-depth presentation.

Let us consider a circular area on which we can obtain a quantity $m(A)$, say mass, for any area A within the circle. In particular, we need to focus on an infinitesimal area corresponding to the neighbourhood of a point $M(x, y)$. The density $D(x, y)$, that is the limit of the ratio of the mass of the neighbourhood to the area of the neighbourhood, depends on how we define the neighbourhood. In other words, on how we “measure” the area. **A density is thus a measure dependent quantity**.

This can be clarified by means of an example. In Cartesian coordinates, we can measure the circle's area by adding small rectangular areas $\triangle x \triangle y$ within the circle, as shown in Figure D.7. This type of measure is known as the *Lebesgue measure*. In polar coordinates, we can do the same thing using elements of area $\triangle\rho\triangle\theta$ along the radius ρ and angle θ, as illustrated in Figure D.7. As the size of the neighbourhoods become increasingly small, we find that:

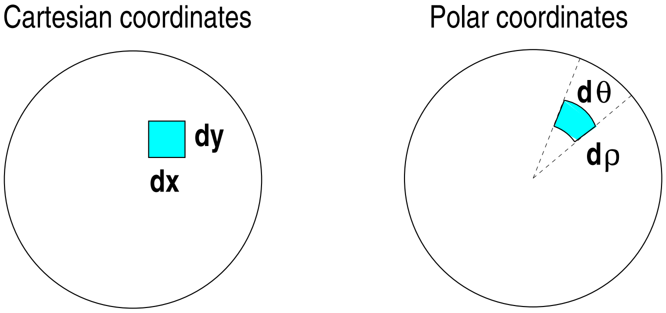

Figure D.7 *Two ways of measuring the area of a circle.*

$$D(x,y) = \lim_{\triangle x \triangle y \to 0} \frac{m(x,y)}{\triangle x \triangle y} \neq \lim_{\triangle \rho \triangle \theta \to 0} \frac{m(\rho,\theta)}{\triangle \rho \triangle \theta} = D(\rho,\theta)$$

That is, the densities in terms of the polar and Cartesian infinitesimal elements of area are different. This result follows from elementary calculus, where it is well known that:

$$\int_{A_\circ} D(x,y)\mathrm{d}x\mathrm{d}y = \int_{A_\circ} D(\rho,\theta)\mathrm{d}\rho\mathrm{d}\theta = \int_{A_\circ} D(\rho\cos(\theta), \rho\sin(\theta))\left|\frac{\partial(x,y)}{\partial(\rho,\theta)}\right|\mathrm{d}\rho\mathrm{d}\theta$$

where $A_\circ$ denotes the area of the circle and $\left|\frac{\partial(x,y)}{\partial(\rho,\theta)}\right| = \rho$ denotes the *Jacobian* of the transformation of coordinates. Hence, we have $D(\rho,\theta) = \rho D(x,y)$, thus proving that the value of the density depends on how we define the neighbourhood.

A further complication arises when we analyse spaces of different dimension. Here, it is possible to evaluate the ratio of the masses of two objects lying on separate dimensions, say a circle and a sphere. However, the ratio of the densities cannot be calculated as it requires a meaningless comparison between the area of the circle and the volume of the sphere. We need, therefore, to extend the dimension of the object in the lower dimensional space and define a *dominating measure* that is common to both objects. This dominating measure is, usually, the measure of the object in the higher dimension. This is illustrated in Figure D.8, where a univariate density has been expanded to a two-dimensional space by sampling a proposal from a uniform density. Needless to say, we could have used many other types of proposal densities. In this case, the Lebesgue measure $\triangle\theta_1\triangle\theta_2$ can be used as the dominating measure.

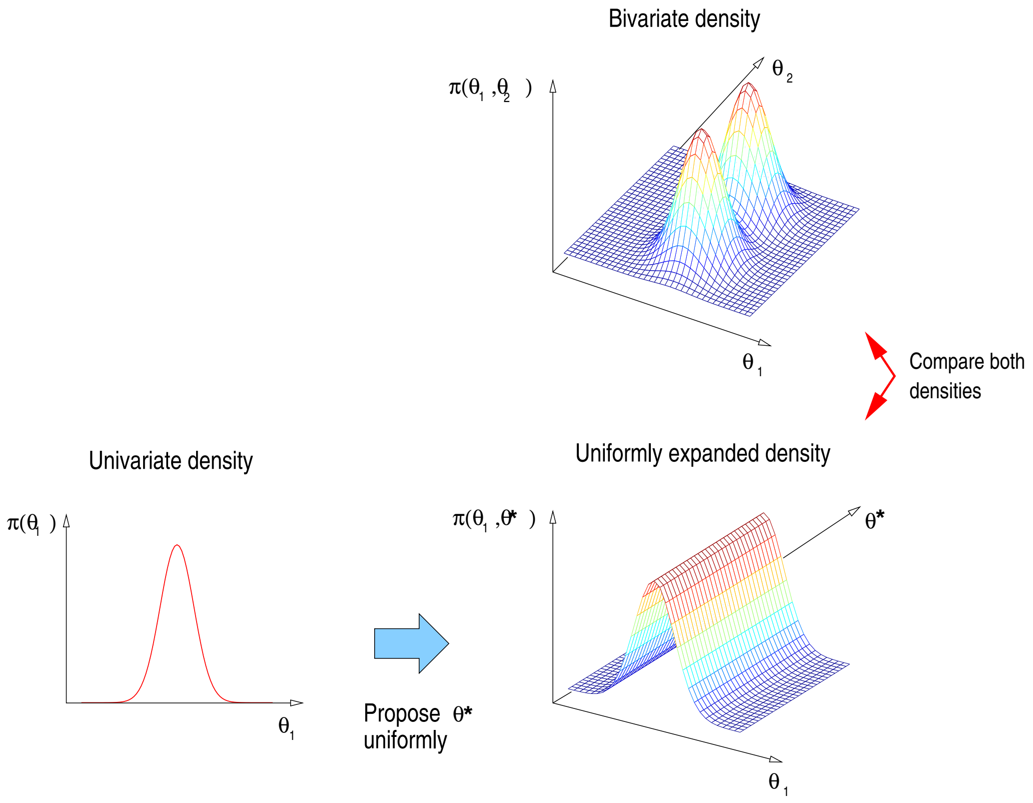

Figure D.8 *Comparing the densities* $\pi(\cdot)$ *between models with one and two dimensions.*

To understand the properties of Markov chains, we need to introduce the concepts of *probability space* and σ*-fields*. A probability space is a triple $(\boldsymbol{\Theta}, \mathcal{B}(\boldsymbol{\Theta}), P)$, where $\boldsymbol{\Theta}$ is the space containing all elements $\boldsymbol{\theta}_i$, $\mathcal{B}(\boldsymbol{\Theta})$ is a σ-field of subsets of $\boldsymbol{\Theta}$ and P is a probability law. This is perhaps best demonstrated by means of a simple example. Consider the tossing of a die and assume that we are only interested in the outcomes even and odd. In this experiment, the space $\boldsymbol{\Theta}$ contains six elements (the six faces of the die):

$$\boldsymbol{\Theta} = \{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6\}$$

Although this set has 2^6 possible subsets, we are only interested in measuring the class of events even and odd. The corresponding σ-field is:

$$\mathcal{B}(\boldsymbol{\Theta}) = \emptyset, \{\theta_1, \theta_3, \theta_5\}, \{\theta_2, \theta_4, \theta_6\}, \boldsymbol{\Theta}$$

That is, the σ-field contains the class of events that interest us, the impossible event $\emptyset$ and the certain event $\boldsymbol{\Theta}$ (odd or even outcome). This definition ensures that any set operations involving events is also an event. More precisely, $\mathcal{B}(\boldsymbol{\Theta})$ is a σ-field of subsets of $\boldsymbol{\Theta}$ if it satisfies the following conditions:

1. $\boldsymbol{\Theta} \in \mathcal{B}(\boldsymbol{\Theta})$.
2. if $A \in \mathcal{B}(\boldsymbol{\Theta})$ then $A^c \in \mathcal{B}(\boldsymbol{\Theta})$.
3. if $A_k \in \mathcal{B}(\boldsymbol{\Theta})$ for $k = 1, 2, \ldots$ then $\bigcup_{k=1}^{\infty} A_k \in \mathcal{B}(\boldsymbol{\Theta})$.

where A^c denotes the complement of the set A.

The pair $(\boldsymbol{\Theta}, \mathcal{B}(\boldsymbol{\Theta}))$ is known as the *measurable space* or *state space*. To complete the definition of a probability space, a number $P(A_i)$ is assigned to each event A_i in the σ-field. In the die example, since the outcomes even and odd are equally likely, we can assign the following probabilities to the events of the σ-field: $P(\emptyset) = 0$, $P(\{\theta_1, \theta_3, \theta_5\}) = 1/2$, $P(\{\theta_2, \theta_4, \theta_6\}) = 1/2$ and $P(\boldsymbol{\Theta}) = 1$. In general, the probabilities should satisfy the following conditions:

1. $P(\boldsymbol{\Theta}) = 1$.
2. $P(A_i) \geq 0$ for any $A_i \in \mathcal{B}(\boldsymbol{\Theta})$.
3. If $A_k \in \mathcal{B}(\boldsymbol{\Theta})$ for $k = 1, 2, \ldots$ and $A_i \cap A_j = \varnothing \;\; \forall i \neq j$, then $P\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} P(A_i)$.

On the real line $\mathbb{R} \triangleq (-\infty, \infty)$, events will consist of the sets that belong to the smallest σ-field containing all intervals $\theta < \theta_1$ for any θ_1. This σ-field, known as the *Borel* σ-field $\mathcal{B}(\mathbb{R})$, contains all open and closed intervals, all points and any countable union or intersection of intervals or points. The concept can be easily extended to higher dimensions. For example, in the plane the events will consist of the sets of points that can be expressed as countable unions or intersections of rectangles.

We can now be slightly more precise about the definition of measures. A measure μ on the space $(\boldsymbol{\Theta}, \mathcal{B}(\boldsymbol{\Theta}))$ is a function from $\mathcal{B}(\boldsymbol{\Theta})$ to $(-\infty, \infty]$ that is countably additive. That is, if $A_k \in \mathcal{B}(\boldsymbol{\Theta})$ for $k = 1, 2, \ldots$ and $A_i \cap A_j = \varnothing \;\; \forall i \neq j$, then

$$\mu\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} \mu(A_i)$$

For any interval $(a, b]$ on the real line, the Lebesgue measure μ^{Leb} is a positive measure ($\mu(A) \geq 0$ for any A) that satisfies $\mu^{\text{Leb}}(a, b] = b - a$. In higher dimensional Euclidean spaces $\mathbb{R}^m$, μ^{Leb} is constructed using the area of rectangles as the basic definition. A probability is, therefore, a positive measure that sums up to unity over the entire measurable space.

We can now define the *distribution* of a random variable $\boldsymbol{\theta}$ (for example, a vector of model parameters) as a probability measure μ defined for all events $A_i, \;\; i = 1, 2, \ldots,$ of $\mathcal{B}(\boldsymbol{\Theta})$ by:

$$\mu(A_i) = p(\boldsymbol{\theta} \in A_i)$$

If we assume that we have a function $p(\theta)$ on the real line that satisfies:

$$p(\theta) \geq 0 \qquad \text{and} \qquad \int_{-\infty}^{\infty} p(\theta)\mathrm{d}\theta = 1$$

that is, a *probability density*, then the *probability distribution* of the interval $(-\infty, \theta_1]$ is given by:

$$p(-\infty < \theta \leq \theta_1) = \int_{-\infty}^{\theta_1} p(\theta)\mathrm{d}\theta$$

For an infinitesimal interval $\mu(\mathrm{d}\theta) = \mathrm{d}\theta$, the distribution is given by:

$$p(\mathrm{d}\theta) = p(\theta)\mu(\mathrm{d}\theta) \tag{D.1}$$

Note that, as customary,p has been used to denote both the density and the distribution. This relation is illustrated in Figure D.9. The distribution of the infinitesimal element $\mathrm{d}\theta$ corresponds to the area of the rectangle. It is the probability given by the product of the measure (width of the rectangle) and the density (height of the rectangle).

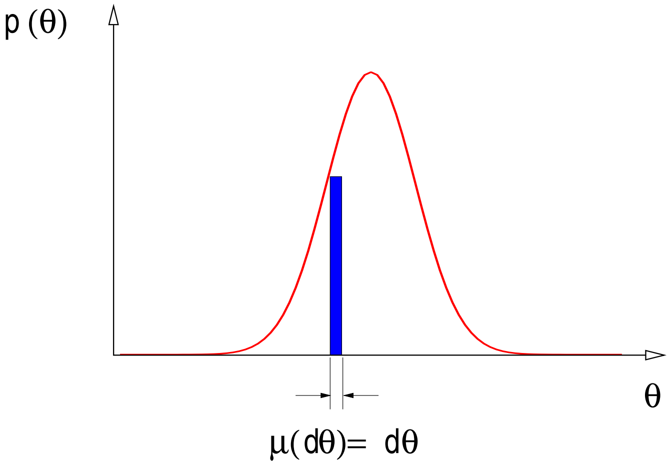

Figure D.9 *Relation between distributions, densities and measures.*

Most of the probability distributions used to represent continuous physical phenomena are meant to measure infinitesimal intervals or "spots" of the parameter space, and not single points (the delta Dirac mass is an exception). This is done, in simple terms, because it is possible to treat "spots" in $\mathbb{R}^m$ as events. It is important to appreciate that the theory underlying the algorithms we use in practice relies on the notion of probabilities that measure "spots" of the parameter space (distributions). However, the implementation of these algorithms makes use of points in the parameter space, not "spots". The latter is a result of the fact that computers are not able to deal with "spots", which consist of uncountable sets of points. Equation (D.1) bridges the gap between theory and implementation. It allows us to replace the distributions by products of functions of points (densities) and measures.

D.3 Markov Chains

This section reviews some of the fundamental properties of Markov chains. These permit the study of the convergence behaviour of many MCMC algorithms. In particular, it will be shown that if a Markov chain has an invariant distribution p, that is if $\theta_t \sim p$ then $\theta_{t+1} \sim p$, it follows that:

$$\lim_{t \to \infty} K^t(\boldsymbol{\theta}_0, \boldsymbol{\theta}_t \in A) = p(\boldsymbol{\theta}_t \in A)$$

where $\boldsymbol{\theta}_0$ is the initial state of the chain and $K(\boldsymbol{\theta}_t, \boldsymbol{\theta}_{t+1} \in A)$ is a mechanism by means of which the chain moves from one sample to the next. For example, for discrete state spaces, $K(\boldsymbol{\theta}_t, \boldsymbol{\theta}_{t+1} \in A)$ will simply be a transition matrix, while for more general state spaces, $K(\boldsymbol{\theta}_t, \boldsymbol{\theta}_{t+1} \in A)$ will correspond to various types of transition kernels. This convergence result states that if we iterate the transition mechanism many times, the samples produced by it will be approximately distributed according to the invariant distribution p. Typically, one devises transition mechanisms so that the target distribution corresponds to the invariant distribution.

The section will begin with a review of Markov chains and transition kernels. Subsequently, it will present a few concepts (irreducibility, recurrence, aperiodicity and reversibility) that allow us to construct mechanisms to generate samples from the target distribution p. It will also discuss some theoretical notions (drift, minorisation and small sets) that serve to determine how fast the convergence takes place.

D.3.1 Markov chains and transition kernels

It is intuitive to introduce Markov chains on finite (discrete) state spaces, where $\boldsymbol{\theta}$ can only take s possible values $\boldsymbol{\theta} \in \boldsymbol{\Theta} = \{\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_s\}$. The stochastic process $\boldsymbol{\theta}$ is called a Markov chain if:

$$p(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_t) = p(\boldsymbol{\theta}_0) \prod_{i=1}^{t} p(\boldsymbol{\theta}_i | \boldsymbol{\theta}_{i-1})$$

where $t \in \mathbb{N}$ is the time index. The chain is *homogeneous* if there is a transition matrix:

$$\mathcal{T} = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1s} \\ p_{21} & p_{22} & \cdots & p_{2s} \\ & & \ddots & \\ p_{s1} & p_{s2} & \cdots & p_{ss} \end{bmatrix}$$

such that the the transition probabilities $P(\boldsymbol{\theta}_t = \boldsymbol{\alpha}_j | \boldsymbol{\theta}_{t-1} = \boldsymbol{\alpha}_i) = p_{ij}$ for all t. That is, the evolution of the chain in a space $\boldsymbol{\Theta}$ depends solely on the current state of the chain and a fixed transition matrix. Note that the transition matrix is a mapping $\mathcal{T} : s \times s \rightarrow [0,1]$, with $\sum_i p_{ij} = 1$ for any i.

As an example, consider a Markov chain with three states ($s = 3$) and a transition graph as illustrated in Figure D.10. The transition matrix for this example is:

$$\mathcal{T} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0.1 & 0.9 \\ 0.6 & 0.4 & 0 \end{bmatrix}$$

If the probability vector for the initial state is $p(\boldsymbol{\theta}_0) = (0.5, 0.2, 0.3)$, it follows that $p(\boldsymbol{\theta}_1) = p(\boldsymbol{\theta}_0)\mathcal{T} = (0.18, 0.64, 0.18)$. After several iterations (multiplications by $\mathcal{T}$), the outer product $p(\boldsymbol{\theta}_t) = p(\boldsymbol{\theta}_0)\mathcal{T}^t$ converges to $(0.22, 0.41, 0.37)$. More importantly, no matter what initial distribution $p(\boldsymbol{\theta}_0)$ we use, the chain will stabilise at $(0.22, 0.41, 0.37)$. This result plays a fundamental role in MCMC simulation. We can start the chain anywhere in the space and be assured of convergence to the invariant distribution $p(\boldsymbol{\theta}) = (0.22, 0.41, 0.37)$. All we have to do to accomplish this is to design a suitable transition operator so that the target distribution, which is assumed to be known up to a normalising constant, corresponds to the invariant distribution. Section D.1 presented a few examples of such operators.

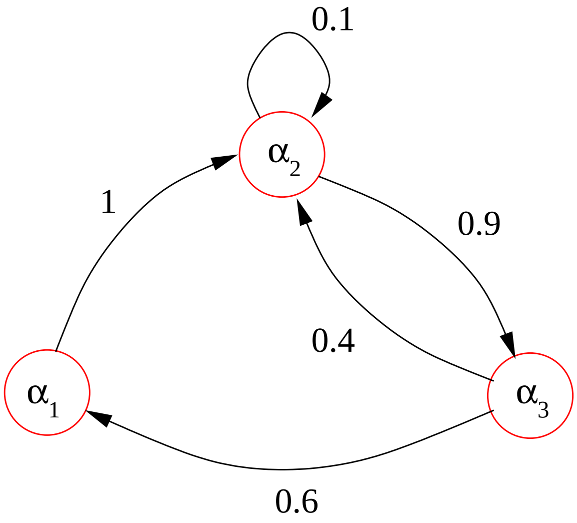

Figure D.10 *Transition graph for the Markov chain example with* $\boldsymbol{\Theta} = \{\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \boldsymbol{\alpha}_3\}$.

In general state spaces $(\boldsymbol{\Theta}, \mathcal{B}(\boldsymbol{\Theta}))$, Markov chains are defined as sequences of random variables $\{\boldsymbol{\theta}_t; t = 0, 1, 2, \ldots\}$, whose evolution in the space $\boldsymbol{\Theta}$ depends solely on the current state of the chain and a transition kernel. Here, the events correspond to the σ-algebra generated by a countable collection of subsets of $\mathbf{\Theta}$, for example intervals in $\mathbb{R}^m$. The transition kernel can be interpreted as a matrix with an infinite number of elements, where the elements in each row add up to one. It satisfies:

$$\begin{aligned} p(\boldsymbol{\theta}_{t+1} \in A | \boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_t) &= p(\boldsymbol{\theta}_{t+1} \in A | \boldsymbol{\theta}_t) \\ &= \int_A K(\boldsymbol{\theta}_t, \mathrm{d}\boldsymbol{\theta}_{t+1}) \end{aligned}$$

for all measurable sets $A \in \mathcal{B}(\mathbf{\Theta})$. That is, the kernel is a mapping $K : \mathbf{\Theta} \times \mathcal{B}(\mathbf{\Theta}) \to [0,1]$, with the following properties:

- For any fixed set $A \in \mathcal{B}(\mathbf{\Theta})$, $K(\cdot, A)$ is measurable.
- For any fixed $\boldsymbol{\theta} \in \mathbf{\Theta}$, $K(\theta, \cdot)$ is a probability measure.

The analysis is restricted to *time-homogeneous* Markov chains, that is the transition kernel is fixed over time. Time varying kernels are, however, necessary when dealing with MCMC optimisation algorithms such as *simulated annealing* (Andrieu et al., 1999d; Geman and Geman, 1984; Van Laarhoven and Arts, 1987).

As illustrated in the discrete case, once we know the initial distribution of the chain, say $p(\mathrm{d}\boldsymbol{\theta}_0)$, the transition kernel fully determines the behaviour of the chain. However, in general state spaces, the iterative multiplications by the transition matrix are replaced by the following recursion:

$$\begin{aligned} p(\boldsymbol{\theta}_0 \in A_0) &= \int_{A_0} p(\mathrm{d}\boldsymbol{\theta}_0) \\ p\big((\boldsymbol{\theta}_0, \boldsymbol{\theta}_1) \in A_0 \times A_1\big) &= \int_{A_0}\int_{A_1} p(\mathrm{d}\boldsymbol{\theta}_0) K(\boldsymbol{\theta}_0, \mathrm{d}\boldsymbol{\theta}_1) \\ p\big((\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2) \in A_0 \times A_1 \times A_2\big) &= \int_{A_0}\int_{A_1}\int_{A_2} p(\mathrm{d}\boldsymbol{\theta}_0) K(\boldsymbol{\theta}_0 | \mathrm{d}\boldsymbol{\theta}_1) K(\boldsymbol{\theta}_1, \mathrm{d}\boldsymbol{\theta}_2) \\ &\vdots \\ p\big((\boldsymbol{\theta}_0, \ldots, \boldsymbol{\theta}_T) \in A_0 \times \cdots \times A_T\big) &= \int_{A_0} \cdots \int_{A_T} \prod_{t=1}^{T} p(\mathrm{d}\boldsymbol{\theta}_0) K(\boldsymbol{\theta}_{t-1}, \mathrm{d}\boldsymbol{\theta}_t) \end{aligned}$$

By marginalising and choosing the initial condition $A_0 = \{\boldsymbol{\theta}_0\}$, if follows that the distribution $K^T(\boldsymbol{\theta}_0, \boldsymbol{\theta}_T \in A) \triangleq p(\boldsymbol{\theta}_T \in A | \boldsymbol{\theta}_0)$ is given by:

$$\begin{aligned}
K^1(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1 \in A) &= K(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1 \in A) \\
K^2(\boldsymbol{\theta}_0, \boldsymbol{\theta}_2 \in A) &= \int_{\boldsymbol{\Theta}} K(\boldsymbol{\theta}_0, \mathrm{d}\boldsymbol{\theta}_1) K(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in A) \\
K^3(\boldsymbol{\theta}_0, \boldsymbol{\theta}_3 \in A) &= \int_{\boldsymbol{\Theta}} \int_{\boldsymbol{\Theta}} K(\boldsymbol{\theta}_0, \mathrm{d}\boldsymbol{\theta}_1) K(\boldsymbol{\theta}_1, \mathrm{d}\boldsymbol{\theta}_2) K(\boldsymbol{\theta}_2, \boldsymbol{\theta}_3 \in A) \\
&= \int_{\boldsymbol{\Theta}} K(\boldsymbol{\theta}_0, \mathrm{d}\boldsymbol{\theta}_1) K^2(\boldsymbol{\theta}_1, \boldsymbol{\theta}_3 \in A) \\
&\vdots \\
K^T(\boldsymbol{\theta}_0, \boldsymbol{\theta}_T \in A) &= \int_{\boldsymbol{\Theta}} K(\boldsymbol{\theta}_0, \mathrm{d}\boldsymbol{\theta}_1) K^{T-1}(\boldsymbol{\theta}_1, \boldsymbol{\theta}_T \in A) \qquad \text{(D.2)}
\end{aligned}$$

In general, we obtain the *Chapman-Kolmogorov equation*:

$$K^{M+T}(\boldsymbol{\theta}, A) = \int_{\boldsymbol{\Theta}} K^M(\boldsymbol{\theta}, \mathrm{d}\boldsymbol{\alpha}) K^T(\boldsymbol{\alpha}, A) \qquad \text{(D.3)}$$

When considering discrete state spaces, it was shown, by means of an example, that the iterated application of the transition kernel (multiplication by the transition matrix) converges to an invariant distribution. In general state spaces, it is also possible to demonstrate that the iterated application of the transition kernel (equation (D.2)) converges to an invariant distribution. Moreover, we can choose the target distribution to be the invariant distribution, as demonstrated with the examples of Section D.1. The following subsections will discuss some conditions that need to be satisfied to achieve this result in general.

D.3.2 φ-Irreducibility

A Markov chain is irreducible if the transition kernel allows for moves over the entire state space. The chain is *φ-irreducible* if it has positive probability of entering any set to which the distribution φ assigns positive probability. That is, for every $A \in \mathcal{B}(\boldsymbol{\Theta})$ with $\varphi(A) > 0$, there exists a T such that $K^T(\boldsymbol{\theta}, A) > 0$ for all $\boldsymbol{\theta} \in \boldsymbol{\Theta}$. In a Bayesian context, irreducibility guarantees that we can visit all the sets of parameter values in the posterior's support. For example, the first two algorithms described in Section D.1 are p-irreducible because they allow for a complete exploration of the support of the target distribution p. As a result of this, we can draw samples from the target distribution p wherever it is bigger that zero, as illustrated in Figures D.3 and D.5.

D.3.3 Aperiodicity

A p-irreducible transition kernel is periodic if there exist an integer $T \geq 1$ and a sequence of non-empty disjoint sets $\{A_0, A_1, \ldots, A_T\}$ in $\mathcal{B}(\boldsymbol{\Theta})$ such that for all $t =$

$1, 2, \ldots, T-1$:

$$\begin{aligned} K(\boldsymbol{\theta}, A_{t+1}) &= 1 \qquad \text{for } \boldsymbol{\theta} \in A_t \\ K(\boldsymbol{\theta}, A_1) &= 1 \qquad \text{for } \boldsymbol{\theta} \in A_T \end{aligned}$$

Otherwise, the kernel is aperiodic. Aperiodicity holds if the transition kernel is such that there is a non-null probability of staying at a particular state $\boldsymbol{\theta}$ or if the kernel has a positive density in the neighbourhood of $\boldsymbol{\theta}$ (Robert and Casella, 1999). Section D.3.5 (Theorem 3) will show that aperiodicity allows us to avoid the use of Cesaro averages, leading to stronger convergence results.

D.3.4 Recurrence

To explain the concept of recurrence, the following definitions need to be introduced:

- The stopping time $\tau_A \triangleq \min\{t \geq 1; \boldsymbol{\theta}_t \in A\}$ is the first t for which the chain reaches the set A.
- The number of visits to the set A is $\eta_A = \sum_{t=1}^{\infty} \mathbb{I}_A(\boldsymbol{\theta}_t)$.
- The average number of passages in A is denoted $\mathbb{E}_{\boldsymbol{\theta}}(\eta_A)$.
- The probability of return to A in a finite number of steps is $p_{\boldsymbol{\theta}}(\tau_A < \infty)$.

Using the above definitions, a set A is called uniformly transient if $\mathbb{E}_{\boldsymbol{\theta}}(\eta_A) < M < \infty$ for all $\boldsymbol{\theta} \in A$ and recurrent if $\mathbb{E}_{\boldsymbol{\theta}}(\eta_A) = \infty$ for all $\boldsymbol{\theta} \in A$. Furthermore, the set is Harris recurrent if $p_{\boldsymbol{\theta}}(\eta_A = \infty) = 1$ for all $\boldsymbol{\theta} \in A$. Harris recurrence ensures that the Markov chain will exhibit the same limiting behaviour regardless of its starting value. This form of stability is, therefore, stronger than p-irreducibility.

D.3.5 Invariance and convergence

Most Markov chains constructed for MCMC simulation algorithms exhibit an important stability condition: if $\boldsymbol{\theta}_t \sim p$ then $\boldsymbol{\theta}_{t+1} \sim p$. When this happens, the distribution p is said to be invariant. More formally, a measure p is invariant if:

$$p(A) = \int_{\boldsymbol{\Theta}} p(\mathrm{d}\boldsymbol{\theta}) K(\boldsymbol{\theta}, A) \qquad \text{for all } A \in \mathcal{B}(\boldsymbol{\Theta}) \tag{D.4}$$

When dealing with Bayesian inference problems, p typically corresponds to the posterior distribution.

The appendix shall now proceed to present one the most fundamental theorems in the theory of MCMC simulation. Before doing so, the following definition is required: a chain is *positive recurrent* if the total mass of p is finite; otherwise it is *null recurrent*. The theorem can now be stated:

Theorem 3 (Meyn and Tweedie, 1993; Tierney, 1994) *Suppose we have a φ-irreducible Markov chain with invariant distribution p. Then the finite measure φ on $\mathcal{B}(\Theta)$ is absolutely continuous with respect to p, the chain is positive recurrent and p-irreducible and p is the unique invariant distribution. Moreover, for almost all $\boldsymbol{\theta}$:*

$$\frac{1}{T}\sum_{t=1}^{T} K^t(\boldsymbol{\theta}, .) \longrightarrow p$$

as t tends to infinity. If the chain is also aperiodic, then for almost all $\boldsymbol{\theta}$:

$$\|K^t(\boldsymbol{\theta}, .) - p\|_{\scriptscriptstyle TV} \to 0 \qquad\qquad \textit{as } t \to \infty$$

If the chain is Harris recurrent, then the convergence occurs for all $\boldsymbol{\theta}$.

Here, $\|\mu\|_{\text{TV}}$ denotes the *total variation norm*. For a measure μ on $(\Theta, \mathcal{B}(\Theta))$, this norm is defined as:

$$\|\mu\|_{\text{TV}} = \sup_{A\in\mathcal{B}(\Theta)} \mu(A) - \inf_{A\in\mathcal{B}(\Theta)} \mu(A)$$

This theorem is very powerful. Among other things, it tells us that if we construct an MCMC algorithm with invariant distribution p, we only need to prove φ-irreducibility and not the more demanding task of proving p-irreducibility. The theorem also says that there is no need for aperiodicity to obtain convergence of averages:

$$P\left(\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} f(\boldsymbol{\theta}_t) = \mathbb{E}(f(\boldsymbol{\theta}))\right) = 1$$

D.3.6 Reversibility and detailed balance

A Markov chain with invariant (stationary) distribution p is said to be reversible if it exhibits detailed balance:

$$p(\mathrm{d}\boldsymbol{\theta}_t)K(\boldsymbol{\theta}_t, \mathrm{d}\boldsymbol{\theta}_{t+1}) = p(\mathrm{d}\boldsymbol{\theta}_{t+1})K(\boldsymbol{\theta}_{t+1}, \mathrm{d}\boldsymbol{\theta}_t)$$

By integrating both sides with respect to $\boldsymbol{\theta}_t$, it follows that reversibility implies invariance:

$$\int_{\Theta} p(\mathrm{d}\boldsymbol{\theta}_t)K(\boldsymbol{\theta}_t, \mathrm{d}\boldsymbol{\theta}_{t+1}) = p(\mathrm{d}\boldsymbol{\theta}_{t+1})$$

Hence, if we know the target distribution p, we can ensure that this distribution will be the invariant distribution of a Markov chain by designing kernels that exhibit detailed balance.

D.3.7 Ergodicity and rates of convergence

A Markov chain is called ergodic if it is positive Harris recurrent and aperiodic. There are several stronger forms of ergodicity that describe how fast the chain is converging, namely *ergodicity of degree 2, geometric ergodicity* and *uniform ergodicity*:

Ergodicity of degree 2 : A Markov chain with invariant distribution p is ergodic of degree 2 if:

$$t\|K^t(\boldsymbol{\theta},.) - p\|_{\text{TV}} \to 0 \qquad\qquad \text{as } t \to \infty$$

for almost all $\boldsymbol{\theta} \in \boldsymbol{\Theta}$.

Geometric ergodicity : The chain is geometrically ergodic if there exists a nonnegative real valued function $M(\boldsymbol{\theta})$, with $\mathbb{E}_p\left(|M(\boldsymbol{\theta})|\right) < \infty$, and a positive constant $\rho < 1$ such that:

$$\|K^t(\boldsymbol{\theta},.) - p\|_{\text{TV}} \leq M(\boldsymbol{\theta})\rho^t$$

for all $\boldsymbol{\theta} \in \boldsymbol{\Theta}$.

Uniform (geometric) ergodicity : The chain is uniformly ergodic if there exists a positive constant $M < \infty$ and a positive constant $\rho < 1$ such that:

$$\|K^t(\boldsymbol{\theta},.) - p\|_{\text{TV}} \leq M\rho^t$$

for all $\boldsymbol{\theta} \in \boldsymbol{\Theta}$.

Uniform ergodicity implies geometric ergodicity, which in turn implies ergodicity of degree 2.

D.3.8 Minorisation and small sets

The minorisation condition and small sets are two important notions that will assist us greatly if we have to verify sufficient conditions for geometric and uniform convergence. A transition kernel satisfies the minorisation condition for an integer $T \geq 0$, a constant $\lambda > 0$, a set $C \in \mathcal{B}(\boldsymbol{\Theta})$ and a probability measure φ if:

$$K^T(\boldsymbol{\theta}, A) \geq \lambda\varphi(A) \qquad\qquad \text{for all } \boldsymbol{\theta} \in C \text{ and } A \in \mathcal{B}(\boldsymbol{\Theta}) \qquad\qquad (\text{D.5})$$

A set C satisfying this condition is called a small set. Under certain topological continuity conditions, Meyn and Tweedie (1993) show that every compact set (*e.g.* a bounded set in $\mathbb{R}^n$ such as $[-2, 2]$) is small. By using this result and small sets, we can easily prove that many MCMC algorithms are φ-irreducible (Chan, 1993; Roberts and Polson, 1994).

D.3.9 Drift conditions

Drift conditions, also known as Foster-Lyapunov conditions, are useful to derive convergence rates for MCMC algorithms (Meyn and Tweedie, 1993). The drift operator is given by:

$$\triangle V(\boldsymbol{\theta}_t) \triangleq \int_{\boldsymbol{\Theta}} K(\boldsymbol{\theta}_t, \mathrm{d}\boldsymbol{\theta}_{t+1}) V(\boldsymbol{\theta}_{t+1}) - V(\boldsymbol{\theta}_t) = \mathbb{E}_{K(\boldsymbol{\theta}_t|\mathrm{d}\cdot)}\big(V(\boldsymbol{\theta}_{t+1})|\boldsymbol{\theta}_t\big) - V(\boldsymbol{\theta}_t)$$

If a Markov chain is p-irreducible, then it is recurrent if for a small set $C \in \mathcal{B}(\boldsymbol{\Theta})$ and a positive function V unbounded off the small set, the following inequality holds:

$$\triangle V(\boldsymbol{\theta}_t) \leq 0 \qquad\qquad \text{for } \boldsymbol{\theta}_t \notin C$$

This inequality is equivalent to saying that the chain is a super-martingale. That is:

$$\mathbb{E}_{K(\boldsymbol{\theta}_t|\mathrm{d}\cdot)}\big(V(\boldsymbol{\theta}_{t+1})|\boldsymbol{\theta}_t\big) \leq V(\boldsymbol{\theta}_t)$$

As shown in Figure D.11, the super-martingale tells us that the expected value of the new state will be closer to the small set. That is, the chain drifts towards the small set.

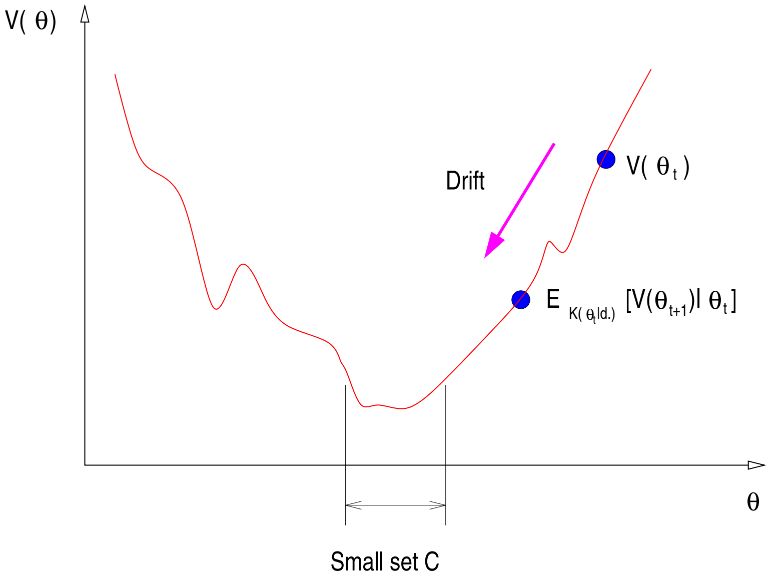

Figure D.11 *Graphical interpretation of Foster-Lyapunov drift stability conditions.*

A stronger drift condition is to ensure that for a small set C, $\lambda < 1$, $b < \infty$ and a drift function $V : \boldsymbol{\Theta} \to [1, \infty)$, we have:

$$\int_{\boldsymbol{\Theta}} K(\boldsymbol{\theta}_t, \mathrm{d}\boldsymbol{\theta}_{t+1}) V(\boldsymbol{\theta}_{t+1}) \leq \lambda V(\boldsymbol{\theta}_t) + b\mathbb{I}_C(\boldsymbol{\theta}_t)$$

This condition guarantees geometric convergence (Meyn and Tweedie, 1993). Thus, to prove geometric convergence for a particular MCMC algorithm the trick is to find a suitable small set and a tailored “potential” function $V(\cdot)$ (Roberts and Tweedie, 1996).

D.3.10 Important theorems on ergodicity

The following theorems are very useful when proving geometric or uniform ergodicity:

Theorem 4 [Theorem 15.0.1 of (Meyn and Tweedie, 1993)] *Suppose that a Markov chain $\{\boldsymbol{\theta}_t; t = 0, 1, 2, \dots\}$ is φ-irreducible and aperiodic. Then the following conditions are equivalent:*

1. *The Markov chain $\{\boldsymbol{\theta}_t; t = 0, 1, 2, \dots\}$ is geometrically ergodic.*

2. *The chain is positive recurrent with invariant measure p and there exist a function $M(\boldsymbol{\theta})$ and a positive constant $\rho < 1$ such that:*

$$\|K^t(\boldsymbol{\theta}, .) - p\|_{TV} \leq M(\boldsymbol{\theta})\rho^t$$

for all $\boldsymbol{\theta} \in \boldsymbol{\Theta}$.

3. *There exists a small set C, $\lambda < 1$, $b < \infty$ and a function $V(\boldsymbol{\theta})$ finite almost everywhere such that:*

$$\int_{\boldsymbol{\Theta}} K(\boldsymbol{\theta}_t, d\boldsymbol{\theta}_{t+1})V(\boldsymbol{\theta}_{t+1}) \leq \lambda V(\boldsymbol{\theta}_t) + b\mathbb{I}_C(\boldsymbol{\theta}_t)$$

In this case the geometric convergence can be strengthened to:

$$\|K^t(\boldsymbol{\theta}, .) - p\|_{TV} \leq MV(\boldsymbol{\theta})\rho^t$$

for some $M < \infty$. That is, the convergence occurs for any $\boldsymbol{\theta}$ for which $V(\boldsymbol{\theta})$ is finite.

Theorem 5 [Theorem 16.0.2 of (Meyn and Tweedie, 1993)] *For any Markov chain $\{\boldsymbol{\theta}_t; t = 0, 1, 2, \dots\}$, the following statements are equivalent:*

1. *The Markov chain $\{\boldsymbol{\theta}_t; t = 0, 1, 2, \dots\}$ is uniformly ergodic.*

2. *There exists a positive constant $M < \infty$ and a positive constant $\rho < 1$ such that:*

$$\|K^t(\boldsymbol{\theta}, .) - p\|_{TV} \leq M\rho^t$$

for all $\boldsymbol{\theta} \in \boldsymbol{\Theta}$.

3. *The space $\boldsymbol{\Theta}$ is small. For an integer $T \geq 0$, a constant $0 < \lambda \leq 1$ and a probability measure φ we have:*

$$K^T(\boldsymbol{\theta}, \cdot) \geq \lambda\varphi(\cdot) \qquad\qquad \textit{for all } \boldsymbol{\theta} \in \boldsymbol{\Theta}.$$

In particular, if $0 < \lambda < 1$, we have:

$$\|K^t(\boldsymbol{\theta}, .) - p\|_{TV} \leq 2(1 - \lambda)^{t/T}$$

4. *The chain is aperiodic and for $\lambda < 1$ and $b < \infty$, there exists a bounded solution $V(\boldsymbol{\theta}) \geq 1$ to:*

$$\|K^t(\boldsymbol{\theta}, .) - p\|_{TV} \leq MV(\boldsymbol{\theta})\rho^t$$

D.3.11 Limit behaviour

When we simulate long MCMC chains, one of our main interests is to estimate expectations of the type $\mathbb{E}_p(f)$ using sample averages:

$$\widehat{f} = \frac{1}{T}\sum_{t=1}^{T} f(\boldsymbol{\theta}_t) \tag{D.6}$$

The estimate is validated by the law of large numbers and central limit theorems. The former states that:

Proposition 6 [Corollary to Theorem 3.6 in Chapter 4 of (Revuz, 1975)] *Suppose we have an ergodic Markov chain with invariant distribution p and a real valued function f such that $\mathbb{E}_p(|f|) < \infty$. Then for any initial distribution, $\widehat{f} \to \mathbb{E}_p(f)$ almost surely.*

We have the following central limit theorems in terms of the rates of convergence:

Proposition 7 [Corollary 7.3 of (Cogburn, 1972)] *Suppose we have a Markov chain, which is ergodic of degree 2 and has invariant distribution p, and a bounded real valued function f. Then there exists a real number σ_f such that the distribution of:*

$$\sqrt{t}\big(\widehat{f} - \mathbb{E}_p(f)\big)$$

converges weakly to a normal distribution with mean 0 and variance σ_f^2 for any initial distribution.

Proposition 8 [Corollary 4.2(ii) of (Nummelin, 1984)] *Suppose we have a Markov chain, which is uniformly ergodic and has invariant distribution p, and a real valued function f such that $\mathbb{E}_p(f^2) < \infty$. Then there exists a real number σ_f such that the distribution of:*

$$\sqrt{t}\big(\widehat{f} - \mathbb{E}_p(f)\big)$$

converges weakly to a normal distribution with mean 0 and variance σ_f^2 for any initial distribution.

The theoretical concepts covered in this section can be used to present and validate the Metropolis-Hastings and reversible jump MCMC algorithms in their general form. This is done in the following sections.

D.4 The Metropolis-Hastings Algorithm

The Metropolis algorithm is the most ubiquitous class of MCMC algorithm. It was introduced in 1953 by (Metropolis et al., 1953) and subsequently generalised by (Hastings, 1970). Since then, many variants have been proposed (Gilks et al., 1996; Robert and Casella, 1999). Here, the discussion focuses, solely, on the Metropolis-Hastings (MH) algorithm as the derivation of the other variants is similar. An MH step of invariant distribution, say $p(\boldsymbol{\theta})$, and proposal distribution, say $q(\boldsymbol{\theta}^{\star}|\,\boldsymbol{\theta})$, involves sampling a candidate value $\boldsymbol{\theta}^{\star}$ given the current value $\boldsymbol{\theta}$ according to $q(\boldsymbol{\theta}^{\star}|\,\boldsymbol{\theta})$. The Markov chain then moves towards $\boldsymbol{\theta}^{\star}$ with probability $\mathcal{A}(\boldsymbol{\theta},\boldsymbol{\theta}^{\star}) = \min\{1,(p(\boldsymbol{\theta})q(\boldsymbol{\theta}^{\star}|\boldsymbol{\theta}))^{-1}p(\boldsymbol{\theta}^{\star})\; q(\boldsymbol{\theta}|\boldsymbol{\theta}^{\star})\}$, otherwise it remains equal to $\boldsymbol{\theta}$. The pseudo-code follows:

Metropolis-Hastings algorithm

```
1. Initialise $\boldsymbol{\theta}_0$ and set $t = 0$.
2. Iteration $t$
     • Sample $u \sim \mathcal{U}_{[0,1]}$.
     • Sample $\boldsymbol{\theta}_t^{\star}$ from $q(\boldsymbol{\theta}_t^{\star}|\boldsymbol{\theta}_t)$.
     • If $u < \mathcal{A}(\boldsymbol{\theta}_t,\boldsymbol{\theta}_t^{\star}) = \min\left\{1, \frac{p(\boldsymbol{\theta}_t{}^{\star})q(\boldsymbol{\theta}_t|\boldsymbol{\theta}_t{}^{\star})}{p(\boldsymbol{\theta}_t)q(\boldsymbol{\theta}_t{}^{\star}|\boldsymbol{\theta}_t)}\right\}$
               $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t^{\star}$
       else
               $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t$
3. $t \leftarrow t+1$ and go to 2.
```

This algorithm is very general, but to perform well in practice it is necessary to use "clever" proposal distributions to avoid excessive rejection of candidates. The transition kernel associated with the algorithm is:

$$K(\boldsymbol{\theta}_t, A) = \int_A \mathcal{K}(\boldsymbol{\theta}_t, \mathrm{d}\boldsymbol{\theta}_t^{\star}) + r(\boldsymbol{\theta}_t)\mathbb{I}_A(\boldsymbol{\theta}_t) \tag{D.7}$$

where

$$\mathcal{K}(\boldsymbol{\theta}_t, \mathrm{d}\boldsymbol{\theta}_t^{\star}) = q(\mathrm{d}\boldsymbol{\theta}_t^{\star}|\boldsymbol{\theta}_t)\mathcal{A}(\boldsymbol{\theta}_t,\boldsymbol{\theta}_t^{\star})$$

and

$$\mathcal{A}(\boldsymbol{\theta}_t,\boldsymbol{\theta}_t^{\star}) = \min\left\{1, \frac{p(\mathrm{d}\boldsymbol{\theta}_t{}^{\star})q(\mathrm{d}\boldsymbol{\theta}_t|\boldsymbol{\theta}_t{}^{\star})}{p(\mathrm{d}\boldsymbol{\theta}_t)q(\mathrm{d}\boldsymbol{\theta}_t{}^{\star}|\boldsymbol{\theta}_t)}\right\}$$

is the probability associated with a candidate being accepted and

$$r(\boldsymbol{\theta}_t) = 1 - \int_{\boldsymbol{\Theta}} q(\mathrm{d}\boldsymbol{\theta}_t^\star|\boldsymbol{\theta}_t)\mathcal{A}(\boldsymbol{\theta}_t, \boldsymbol{\theta}_t^\star)$$

is the probability of remaining at the same point.

If the target and proposal distributions admit densities with respect to the measures $\mathrm{d}\boldsymbol{\theta}_t$ and $\mathrm{d}\boldsymbol{\theta}_t^\star$, we can replace the distributions by products of densities and measures (equation (D.1)) to obtain:

$$K(\boldsymbol{\theta}_t, A) = \int_A q(\boldsymbol{\theta}_t^\star|\boldsymbol{\theta}_t)\mathcal{A}(\boldsymbol{\theta}_t, \boldsymbol{\theta}_t^\star)\mathrm{d}\boldsymbol{\theta}_t^\star + r(\boldsymbol{\theta}_t)\mathbb{I}_A(\boldsymbol{\theta}_t)$$

with

$$\begin{aligned}
\mathcal{A}(\boldsymbol{\theta}_t, \boldsymbol{\theta}_t^\star) &= \min\left\{1, \frac{p(\boldsymbol{\theta}_t{}^\star)\mathrm{d}\boldsymbol{\theta}_t{}^\star q(\boldsymbol{\theta}_t|\boldsymbol{\theta}_t{}^\star)\mathrm{d}\boldsymbol{\theta}_t}{p(\boldsymbol{\theta}_t)\mathrm{d}\boldsymbol{\theta}_t q(\boldsymbol{\theta}_t{}^\star|\boldsymbol{\theta}_t)\mathrm{d}\boldsymbol{\theta}_t{}^\star}\right\} \\
&= \min\left\{1, \frac{p(\boldsymbol{\theta}_t{}^\star)q(\boldsymbol{\theta}_t|\boldsymbol{\theta}_t{}^\star)}{p(\boldsymbol{\theta}_t)q(\boldsymbol{\theta}_t{}^\star|\boldsymbol{\theta}_t)}\right\}
\end{aligned}$$

A very interesting type of Metropolis-Hastings algorithm can be obtained when we adopt the full conditional distributions $p(\boldsymbol{\theta}_i|\boldsymbol{\theta}_{-i}) = p(\boldsymbol{\theta}_i|\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{i-1}, \boldsymbol{\theta}_{i+1}, \ldots, \boldsymbol{\theta}_m)$ as proposal distributions. This algorithm, known as the Gibbs sampler, has been very popular since its development (Geman and Geman, 1984). The following section describes it in more detail.

D.4.1 The Gibbs sampler

Suppose we have n unknown random variables $\boldsymbol{\theta}_t = \{\boldsymbol{\theta}_{t,1}, \boldsymbol{\theta}_{t,2}, \ldots, \boldsymbol{\theta}_{t,n}\}$. If, in addition, we know the full conditionals $p(\boldsymbol{\theta}_{t,i}|\boldsymbol{\theta}_{t,1}, \ldots, \boldsymbol{\theta}_{t,i-1}, \boldsymbol{\theta}_{t,i+1}, \ldots, \boldsymbol{\theta}_{t,n})$, it is advantageous to use the following proposal distribution for $i = 1, \ldots, n$:

$$q(\boldsymbol{\theta}_t^\star|\boldsymbol{\theta}_t) = \begin{cases} p(\boldsymbol{\theta}_{t,i}^\star|\boldsymbol{\theta}_{t,-i}) & \text{If } \boldsymbol{\theta}_{t,-i}^\star = \boldsymbol{\theta}_{t,-i} \\ 0 & \text{Otherwise} \end{cases}$$

The corresponding acceptance probability is:

$$\begin{aligned}
\mathcal{A}(\boldsymbol{\theta}_t, \boldsymbol{\theta}_t^\star) &= \min\left\{1, \frac{p(\boldsymbol{\theta}_t^\star)p(\boldsymbol{\theta}_{t,i}|\boldsymbol{\theta}_{t,-i}^\star)}{p(\boldsymbol{\theta}_t)p(\boldsymbol{\theta}_{t,i}^\star|\boldsymbol{\theta}_{t,-i})}\right\} \\
&= \min\left\{1, \frac{p(\boldsymbol{\theta}_t^\star)p(\boldsymbol{\theta}_{t,i}|\boldsymbol{\theta}_{t,-i})}{p(\boldsymbol{\theta}_t)p(\boldsymbol{\theta}_{t,i}^\star|\boldsymbol{\theta}_{t,-i}^\star)}\right\} \\
&= \min\left\{1, \frac{p(\boldsymbol{\theta}_{t,-i}^\star)}{p(\boldsymbol{\theta}_{t,-i})}\right\} \\
&= 1
\end{aligned}$$

Since the acceptance probability for each proposal is one, the Gibbs sampler algorithm is often presented as follows:

Gibbs sampler algorithm

```
1. Initialise $\boldsymbol{\theta}_{0,1:n}$ and set $t = 0$.
2. Iteration $t$
    • Sample $\boldsymbol{\theta}_{t+1,1} \sim p(\boldsymbol{\theta}_1|\boldsymbol{\theta}_{t,2}, \boldsymbol{\theta}_{t,3}, \dots, \boldsymbol{\theta}_{t,n})$.
    • Sample $\boldsymbol{\theta}_{t+1,2} \sim p(\boldsymbol{\theta}_2|\boldsymbol{\theta}_{t+1,1}, \boldsymbol{\theta}_{t,3}, \dots, \boldsymbol{\theta}_{t,n})$.
                                  $\vdots$
    • Sample $\boldsymbol{\theta}_{t+1,i} \sim p(\boldsymbol{\theta}_i|\boldsymbol{\theta}_{t+1,1}, \dots, \boldsymbol{\theta}_{t+1,i-1}, \boldsymbol{\theta}_{t,i+1}, \dots, \boldsymbol{\theta}_{t,n})$.
                                  $\vdots$
    • Sample $\boldsymbol{\theta}_{t+1,n} \sim p(\boldsymbol{\theta}_n|\boldsymbol{\theta}_{t+1,1}, \boldsymbol{\theta}_{t+1,2}, \dots \boldsymbol{\theta}_{t+1,n-1})$.
3. $t \leftarrow t + 1$ and go to 2.
```

Since the Gibbs sampler can be viewed as a special case of the MH algorithm, it is possible to introduce MH steps into the Gibbs sampler. That is, when the full conditionals are available and belong to the family of standard distributions (Gamma, Gaussian, etc.), we will draw the new samples directly. Otherwise, we can draw samples with MH steps embedded within the Gibbs algorithm. For $n = 2$, the Gibbs sampler is also known as the data augmentation algorithm, which is closely related to the EM algorithm (Dempster et al., 1977; Tanner and Wong, 1987).

D.4.2 On the convergence of the MH algorithm

Since $\mathcal{K}(\boldsymbol{\theta}_t, \boldsymbol{\theta}_t^\star) = q(\boldsymbol{\theta}_t^\star|\boldsymbol{\theta}_t)\mathcal{A}(\boldsymbol{\theta}_t, \boldsymbol{\theta}_t^\star)$ satisfies the reversibility condition:

$$p(\boldsymbol{\theta}_t)\mathcal{K}(\boldsymbol{\theta}_t, \boldsymbol{\theta}_t^\star) = p(\boldsymbol{\theta}_t{}^\star)\mathcal{K}(\boldsymbol{\theta}_t^\star, \boldsymbol{\theta}_t)$$

it follows that for any measurable set A:

$$\begin{aligned} \int_{\boldsymbol{\Theta}} \mathcal{K}(\boldsymbol{\theta}_t, A)p(\mathrm{d}\boldsymbol{\theta}_t) &= \int_{\boldsymbol{\Theta}}\int_A \mathcal{K}(\boldsymbol{\theta}_t, \boldsymbol{\theta}_t^\star)p(\boldsymbol{\theta}_t)\mathrm{d}\boldsymbol{\theta}_t^\star\mathrm{d}\boldsymbol{\theta}_t \\ &= \int_{\boldsymbol{\Theta}}\int_A \mathcal{K}(\boldsymbol{\theta}_t^\star, \boldsymbol{\theta}_t)p(\boldsymbol{\theta}_t^\star)\mathrm{d}\boldsymbol{\theta}_t^\star\mathrm{d}\boldsymbol{\theta}_t \\ &= \int_A p(\boldsymbol{\theta}_t^\star)\mathrm{d}\boldsymbol{\theta}_t^\star = p(A) \end{aligned} \tag{D.8}$$

since $\int_{\mathbf{\Theta}} \mathcal{K}(\boldsymbol{\theta}_t^{\star}, \boldsymbol{\theta}_t)\mathrm{d}\boldsymbol{\theta}_t = 1$. Thus, by construction, the MH algorithm admits p as invariant distribution.

If we can show the φ-irreducibility and aperiodicity of the MH kernel, then Theorem 3 guarantees the convergence of the algorithm. If the space $\mathbf{\Theta}$ is small (for example, bounded in $\mathbb{R}^m$), then it is possible to use the minorisation condition to prove uniform ergodicity. It is also possible to prove geometric ergodicity using drift conditions with test functions such as $V(\boldsymbol{\theta}) = \left(p(\boldsymbol{\theta})\right)^{-\beta}$ for some choice of β (Roberts and Tweedie, 1996).

D.5 The Reversible Jump MCMC Algorithm

This section introduces the reversible jump MCMC algorithm. For a more detailed introduction, readers are strongly encouraged to consult (Andrieu et al., 1999f). The reversible jump MCMC algorithm is a generalisation of the MH algorithm to spaces of different dimension (Green, 1995; Richardson and Green, 1997). It allows one to perform model selection and parameter estimation simultaneously. Figure D.12 shows a typical reversible jump MCMC move. A higher dimension is proposed by sampling $\boldsymbol{\theta}^*$ from a proposal distribution. If the move is accepted, the 2D parameter space expands to 3D. Otherwise, it remains the same. As shown in this section, the decision to accept or reject a move depends on a ratio involving the densities and measures for each dimension.

In the joint parameter estimation and model selection problem, we consider a family of N models $\{\mathcal{M}_n; n = 1, \ldots, N\}$, defined on a union of subspaces $\mathbf{\Theta} \triangleq \cup_{n=1}^{N}\{n\} \times \mathbf{\Theta}_n$, where N may be infinite. The full target distribution defined in this space is given by:

$$p(k, \mathrm{d}\boldsymbol{\theta}) = \sum_{n=1}^{N} p_n(n, \mathrm{d}\boldsymbol{\theta}_n)\mathbb{I}_{\{n\}\times\mathbf{\Theta}_n}(k, \boldsymbol{\theta})$$

That is, the probability of k being equal to n and $\boldsymbol{\theta}$ belonging to an infinitesimal set centred around θ_n is $p(n, \mathrm{d}\boldsymbol{\theta}_n)$. By marginalisation, the probability of being in subspace $\mathbf{\Theta}_n$ is:

$$p_n(n) = \int_{\mathbf{\Theta}} p_n(n, \mathrm{d}\boldsymbol{\theta}_n)$$

For simplicity, we shall consider only two subspaces: the more general case follows trivially (Andrieu et al., 1999f). The target distribution is reduced to:

$$p(k, \mathrm{d}\boldsymbol{\theta}) = p_1(1, \mathrm{d}\boldsymbol{\theta}_1)\mathbb{I}_{\{1\}\times\mathbf{\Theta}_1}(k, \boldsymbol{\theta}) + p_2(2, \mathrm{d}\boldsymbol{\theta}_2)\mathbb{I}_{\{2\}\times\mathbf{\Theta}_2}(k, \boldsymbol{\theta})$$

defined on the space $\mathbf{\Theta} = \{1\} \times \mathbf{\Theta}_1 \cup \{2\} \times \mathbf{\Theta}_2$.

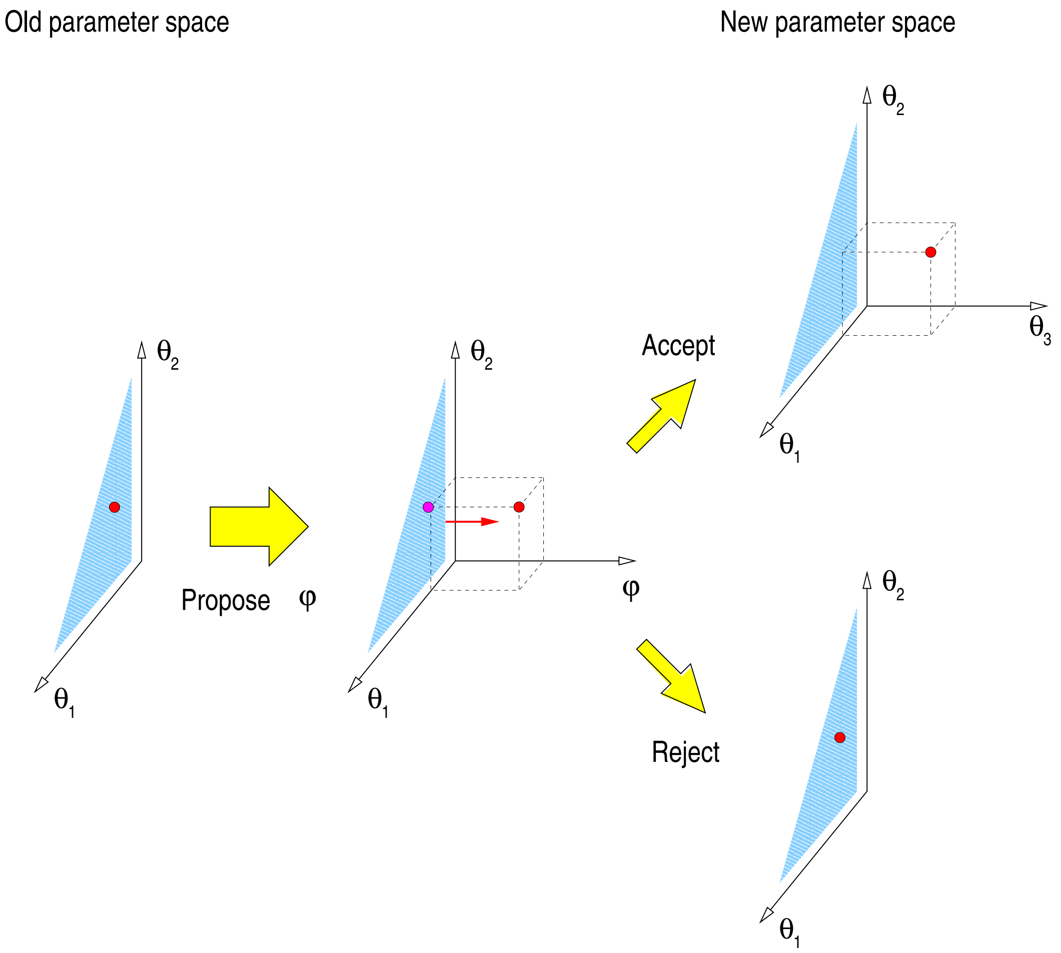

Figure D.12 *Proposing a birth move (dimension increase) with the reversible jump MCMC algorithm.*

We can extend the MH algorithm by adopting a transition kernel $K(k, \boldsymbol{\theta}; k^\star, \mathrm{d}\boldsymbol{\theta}^\star)$ that allows us to move from $(k, \boldsymbol{\theta})$ to $(k^\star, \boldsymbol{\theta}^\star)$. If the subspaces $\boldsymbol{\Theta}_1$ and $\boldsymbol{\Theta}_2$ are disjoint and, for clarity, we only consider moves from one subspace to the other (moves within a particular subspace can be done with the MH algorithm), the proposal distribution can be written as:

$$q(\mathrm{d}\boldsymbol{\theta}^\star|\boldsymbol{\theta}) = q_{1,2}(\mathrm{d}\boldsymbol{\theta}_2^\star|\boldsymbol{\theta}_1)\mathbb{I}_{\boldsymbol{\Theta}_1\times\boldsymbol{\Theta}_2}(\boldsymbol{\theta}^\star, \boldsymbol{\theta}) + q_{2,1}(\mathrm{d}\boldsymbol{\theta}_1^\star|\boldsymbol{\theta}_2)\mathbb{I}_{\boldsymbol{\Theta}_1\times\boldsymbol{\Theta}_2}(\boldsymbol{\theta}^\star, \boldsymbol{\theta})$$

That is, if we are in subspace $\{1\} \times \boldsymbol{\Theta}_1$, we move to $\{2\} \times \boldsymbol{\Theta}_2$ using the proposal $q_{1,2}$ and vice-versa. The corresponding acceptance ratio for a move from $(1, \boldsymbol{\theta}_1)$ to $(2, \boldsymbol{\theta}_2^\star)$ is given by:

$$\mathcal{A}(1, \boldsymbol{\theta}_1; 2, \boldsymbol{\theta}_2^\star) = \min\left\{1, \frac{p_2(2, \mathrm{d}\boldsymbol{\theta}_2^\star)q_{2,1}(\mathrm{d}\boldsymbol{\theta}_1|\boldsymbol{\theta}_2^\star)}{p_1(1, \mathrm{d}\boldsymbol{\theta}_1)q_{1,2}(\mathrm{d}\boldsymbol{\theta}_2^\star|\boldsymbol{\theta}_1)}\right\}$$

If the target and proposal distributions admit densities with respect to the measures $\mu_1(\mathrm{d}\boldsymbol{\theta}_1)$ and $\mu_2(\mathrm{d}\boldsymbol{\theta}_2^\star)$, we have (see equation (D.1)):

$$\begin{aligned}\mathcal{A}(1,\boldsymbol{\theta}_1;2,\boldsymbol{\theta}_2^\star) &= \min\left\{1, \frac{p_2(2,\boldsymbol{\theta}_2^\star)\mu_2(\mathrm{d}\boldsymbol{\theta}_2^\star)q_{2,1}(\boldsymbol{\theta}_1|\boldsymbol{\theta}_2^\star)\mu_1(\mathrm{d}\boldsymbol{\theta}_1)}{p_1(1,\boldsymbol{\theta}_1)\mu_1(\mathrm{d}\boldsymbol{\theta}_1)q_{1,2}(\boldsymbol{\theta}_2^\star|\boldsymbol{\theta}_1)\mu_2(\mathrm{d}\boldsymbol{\theta}_2^\star)}\right\}\\ &= \min\left\{1, \frac{p_2(2,\boldsymbol{\theta}_2^\star)q_{2,1}(\boldsymbol{\theta}_1|\boldsymbol{\theta}_2^\star)}{p_1(1,\boldsymbol{\theta}_1)q_{1,2}(\boldsymbol{\theta}_2^\star|\boldsymbol{\theta}_1)}\right\}\end{aligned}$$

It is often necessary to consider the more general case where the two sets are not disjoint: for example $\boldsymbol{\Theta}_2 = \boldsymbol{\Theta}_1 \times \boldsymbol{\Psi}_{1,2}$ and one wants to jump between $\boldsymbol{\Theta}_1$ and $\boldsymbol{\Theta}_2$. In many cases, it is natural to link the current state $\boldsymbol{\theta}_1$ and the candidate state $\boldsymbol{\theta}_2^\star$ deterministically. For instance, one could add an extra component $\boldsymbol{\varphi}_{1,2} \in \boldsymbol{\Psi}_{1,2}$ to $\boldsymbol{\theta}_1$ to obtain $\boldsymbol{\theta}_2^\star$ as follows:

$$\boldsymbol{\theta}_2^\star = (\boldsymbol{\theta}_1, \boldsymbol{\varphi}_{1,2}). \tag{D.9}$$

This is a reasonable proposal in the case of two nested models, where one wants to keep as much information as possible when moving from parameter space $\boldsymbol{\Theta}_1$ to $\boldsymbol{\Theta}_2$ and vice-versa. Examples of this type of model include neural networks, models of signal components (sinusoids, impulses, etc.) in noise, auto-regressive processes and many types of mixture models. For instance, one might wish to jump from a model consisting of k fixed-variance Gaussian components to one comprising $k+1$ components by proposing a new mean $\boldsymbol{\varphi}_{1,2}$, while keeping the current values of the other means. The reverse move, to jump from $\boldsymbol{\Theta}_2$ to $\boldsymbol{\Theta}_1$, is automatically defined and consists of removing an appropriate component from $\boldsymbol{\theta}_2$.

The acceptance ratio of the move from $\boldsymbol{\Theta}_1$ to $\boldsymbol{\Theta}_2$ is still equal to:

$$\mathcal{A}(1,\boldsymbol{\theta}_1;2,\boldsymbol{\theta}_2^\star) = \min\left\{1, \frac{p_2(2,\mathrm{d}\boldsymbol{\theta}_2^\star)q_{2,1}(\mathrm{d}\boldsymbol{\theta}_1|\boldsymbol{\theta}_2^\star)}{p_1(1,\mathrm{d}\boldsymbol{\theta}_1)q_{1,2}(\mathrm{d}\boldsymbol{\theta}_2^\star|\boldsymbol{\theta}_1)}\right\}$$

However, reparametrising in terms of $(\boldsymbol{\theta}_1, \boldsymbol{\varphi}_{1,2})$, results in the following acceptance ratio:

$$\begin{aligned}\mathcal{A}(1,\boldsymbol{\theta}_1;2,\boldsymbol{\theta}_2^\star) &= \min\left\{1, \frac{p_2\big(2,\mathrm{d}(\boldsymbol{\theta}_1,\boldsymbol{\varphi}_{1,2})\big)q_{2,1}\big(\mathrm{d}\boldsymbol{\theta}_1|(\boldsymbol{\theta}_1,\boldsymbol{\varphi}_{1,2})\big)}{p_1(1,\mathrm{d}\boldsymbol{\theta}_1)q_{1,2}\big(\mathrm{d}(\boldsymbol{\theta}_1,\boldsymbol{\varphi}_{1,2})|\boldsymbol{\theta}_1\big)}\right\}\\ &= \min\left\{1, \frac{p_2(2,\mathrm{d}\big(\boldsymbol{\theta}_1,\boldsymbol{\varphi}_{1,2})\big)\delta_{\boldsymbol{\theta}_1}(\mathrm{d}\boldsymbol{\theta}_1)}{p_1(1,\mathrm{d}\boldsymbol{\theta}_1)q_{1,2}(\mathrm{d}\boldsymbol{\varphi}_{1,2}|\boldsymbol{\theta}_1)q_{1,2}(\mathrm{d}\boldsymbol{\theta}_1|\boldsymbol{\theta}_1)}\right\}\\ &= \min\left\{1, \frac{p_2(2,\mathrm{d}\big(\boldsymbol{\theta}_1,\boldsymbol{\varphi}_{1,2})\big)\delta_{\boldsymbol{\theta}_1}(\mathrm{d}\boldsymbol{\theta}_1)}{p_1(1,\mathrm{d}\boldsymbol{\theta}_1)q_{1,2}(\mathrm{d}\boldsymbol{\varphi}_{1,2}|\boldsymbol{\theta}_1)\delta_{\boldsymbol{\theta}_1}(\mathrm{d}\boldsymbol{\theta}_1)}\right\}\end{aligned}$$

as only $\boldsymbol{\varphi}_{1,2}$ is sampled, whereas $\boldsymbol{\theta}_1$ is kept fixed. If we now convert the distributions to products of densities and measures (equation (D.1)), the acceptance ratio becomes:

$$\mathcal{A}(1,\boldsymbol{\theta}_1;2,\boldsymbol{\theta}_2^\star) = \min\left\{1, \frac{p_2\big(2,(\boldsymbol{\theta}_1,\boldsymbol{\varphi}_{1,2})\big)\mu_2\big(\mathrm{d}(\boldsymbol{\theta}_1,\boldsymbol{\varphi}_{1,2})\big)}{p_1(1,\boldsymbol{\theta}_1)q_{1,2}(\boldsymbol{\varphi}_{1,2}|\boldsymbol{\theta}_1)\mu_1(\mathrm{d}\boldsymbol{\theta}_1)\overline{\mu}_1(\mathrm{d}\boldsymbol{\varphi}_{1,2})}\right\}$$

which requires the existence and the evaluation of the ratio of measures:

$$\mathcal{J} = \frac{\mu_2(\mathrm{d}(\boldsymbol{\theta}_1, \boldsymbol{\varphi}_{1,2}))}{\mu_1(\mathrm{d}\boldsymbol{\theta}_1)\overline{\mu}_1(\mathrm{d}\boldsymbol{\varphi}_{1,2})}$$

In numerous cases, μ_2, μ_1 and $\overline{\mu}_1$ are the Lebesgue measures on the sets $\boldsymbol{\Theta}_2$, $\boldsymbol{\Theta}_1$ and $\boldsymbol{\Psi}_{1,2}$, and the ratio, known as the Jacobian, is equal to 1.

Finally, one can extend the derivation to a more general case where there exists a deterministic invertible relationship $f(\cdot)$ between $\boldsymbol{\Theta}_2 \times \boldsymbol{\Psi}_{2,1}$ and $\boldsymbol{\Theta}_1 \times \boldsymbol{\Psi}_{1,2}$ of the form:

$$\begin{pmatrix} \boldsymbol{\theta}_2^\star \\ \boldsymbol{\varphi}_{2,1}^\star \end{pmatrix} = \begin{pmatrix} f_{\boldsymbol{\theta}}(\boldsymbol{\theta}_1, \varphi_{1,2}) \\ f_{\boldsymbol{\varphi}}(\boldsymbol{\theta}_1, \varphi_{1,2}) \end{pmatrix} \tag{D.10}$$

The proposals $\boldsymbol{\varphi}_{1,2}$ and $\boldsymbol{\varphi}_{2,1}$ allow us to expand the dimensions of the parameter spaces ($\boldsymbol{\Theta}_1$ and $\boldsymbol{\Theta}_2$) to spaces ($\boldsymbol{\Theta}_1 \times \boldsymbol{\Psi}_{1,2}$ and $\boldsymbol{\Theta}_2 \times \boldsymbol{\Psi}_{2,1}$) that can "communicate". In other words, to spaces that can be compared. The acceptance ratio for a move from $\boldsymbol{\Theta}_1$ to $\boldsymbol{\Theta}_2$ is given by:

$$\begin{aligned}
\mathcal{A}(1, \boldsymbol{\theta}_1; 2, \boldsymbol{\theta}_2^\star) &= \min\left\{1, \frac{p_2(2, \mathrm{d}\boldsymbol{\theta}_2^\star)q_{2,1}(\mathrm{d}\boldsymbol{\theta}_1|\boldsymbol{\theta}_2^\star)}{p_1(1, \mathrm{d}\boldsymbol{\theta}_1)q_{1,2}(\mathrm{d}\boldsymbol{\theta}_2^\star|\boldsymbol{\theta}_1)}\right\} \\
&= \min\left\{1, \frac{p_2\big(2, \mathrm{d}f_{\boldsymbol{\theta}}(\boldsymbol{\theta}_1, \boldsymbol{\varphi}_{1,2})\big)q_{2,1}\big(\mathrm{d}f_{\boldsymbol{\varphi}}(\boldsymbol{\theta}_1, \boldsymbol{\varphi}_{1,2})|f_{\boldsymbol{\theta}}(\boldsymbol{\theta}_1, \boldsymbol{\varphi}_{1,2})\big)}{p_1(1, \mathrm{d}\boldsymbol{\theta}_1)q_{1,2}(\mathrm{d}\boldsymbol{\varphi}_{1,2}|\boldsymbol{\theta}_1)}\right\} \\
&= \min\left\{1, \frac{p_2(2, f_{\boldsymbol{\theta}}(\boldsymbol{\theta}_1, \boldsymbol{\varphi}_{1,2}))q_{2,1}(f_{\boldsymbol{\varphi}}(\boldsymbol{\theta}_1, \boldsymbol{\varphi}_{1,2})\big| f_{\boldsymbol{\theta}}(\boldsymbol{\theta}_1, \boldsymbol{\varphi}_{1,2}))}{p_1(1, \boldsymbol{\theta}_1)q_{1,2}(\boldsymbol{\varphi}_{1,2}\big|\boldsymbol{\theta}_1)}\right. \\
&\qquad \left. \times \frac{\mu_2(\mathrm{d}f_{\boldsymbol{\theta}}(\boldsymbol{\theta}_1, \boldsymbol{\varphi}_{1,2}))\overline{\mu}_2(\mathrm{d}f_{\boldsymbol{\varphi}}(\boldsymbol{\theta}_1, \varphi_{1,2}))}{\mu_1(\mathrm{d}\boldsymbol{\theta}_1)\overline{\mu}_1(\mathrm{d}\boldsymbol{\varphi}_{1,2})}\right\}
\end{aligned} \tag{D.11}$$

Again, in numerous cases, μ_2, μ_1, $\overline{\mu}_1$ and $\overline{\mu}_2$ are Lebesgue measures and therefore this ratio satisfies:

$$\frac{\mu_2(\mathrm{d}f_{\boldsymbol{\theta}}(\boldsymbol{\theta}_1, \boldsymbol{\varphi}_{1,2}))\overline{\mu}_2(\mathrm{d}f_{\boldsymbol{\varphi}}(\boldsymbol{\theta}_1, \boldsymbol{\varphi}_{1,2}))}{\mu_1(\mathrm{d}\boldsymbol{\theta}_1)\overline{\mu}_1(\mathrm{d}\boldsymbol{\varphi}_{1,2})} = \mathcal{J}_f \triangleq \left|\det \frac{\partial f(\boldsymbol{\theta}_1, \varphi_{1,2})}{\partial(\boldsymbol{\theta}_1, \varphi_{1,2})}\right| \tag{D.12}$$

where $\mathcal{J}_f$ is the Jacobian of the transformation $f(\cdot)$.

D.6 Summary

This appendix presented an introduction to Markov chains and Monte Carlo simulation. Starting with a brief discussion of measure theory and probability, it developed some important properties of Markov chains which were useful to describe the Metropolis-Hastings, Gibbs sampling and reversible jump MCMC algorithms. The introduction is by no means exhaustive. Readers are encouraged to use it as a starting point and proceed to study more advanced topics in MCMC simulation, such as empirical convergence diagnostics, simulated tempering, coupling, perfect sampling, sequential Monte

Carlo, hit and run algorithms, auxiliary variables and continuous time sampling methods (Besag et al., 1995; Brooks, 1998; Gilks et al., 1996; Meyn and Tweedie, 1993; Robert and Casella, 1999; Tierney, 1994).

E

Proof of Theorem 1

The proof[1] of Theorem 1 follows from similar proofs in the field of signal processing (Andrieu and Doucet, 1999; Andrieu et al., 1999a). It relies on the following theorem, which is a result of Theorems 14.0.1 and 15.0.1 in (Meyn and Tweedie, 1993):

Theorem 6 *Suppose that a Markovian transition kernel P on a space $\mathbf{Z}$*

1. *is a $\phi-$irreducible (for some measure ϕ) aperiodic Markov transition kernel with invariant distribution π.*
2. *has geometric drift towards a small set C with drift function $V : \mathbf{Z} \to [1, +\infty)$ i.e. there exists $0 < \lambda < 1$, $b > 0$, k_0 and an integrable measure ν such that:*

$$PV(\mathbf{z}) \leq \lambda V(\mathbf{z}) + b\mathbb{I}_C(\mathbf{z}) \tag{E.1}$$

$$P^{k_0}(\mathbf{z}, d\mathbf{z}') \geq \mathbb{I}_C(\mathbf{z})\nu(d\mathbf{z}') \tag{E.2}$$

then for π-almost all $\mathbf{z}_0$, some constants $\rho < 1$ and $R < +\infty$, we have:

$$\|P^n(\mathbf{z}_0, \cdot) - \pi(\cdot)\|_{TV} \leq RV(\mathbf{z}_0)\rho^n \tag{E.3}$$

That is, P is geometrically ergodic.

We need to prove five lemmas that will allow us to prove the different conditions required to apply Theorem 6. These lemmas will enable us to prove Proposition 9 which will establish the minorisation condition (E.2) for some k_0 and measure ϕ (to be described). The ϕ-irreducibility and aperiodicity of the Markov chain are then proved in Corollary 3, thereby ensuring the simple convergence of the Markov chain. To complete the proof, Proposition 10 will establish the drift condition (E.1). To simplify the

[1] Readers not familiar with the concepts of irreducibility, small sets and drift conditions are strongly encouraged to consult Appendix D.

presentation, only one network output is considered. The proof for multiple outputs follows trivially.

Before presenting the various lemmas and their respective proofs, some notation needs to be introduced. Let $\mathcal{K}(\Lambda_{k_1}, \delta^2_{k_1}, k_1, \boldsymbol{\mu}_{1:k_1}; d\Lambda_{k_2}, d\delta^2_{k_2}, k_2, d\boldsymbol{\mu}_{1:k_2})$ denote[2] the transition kernel of the Markov chain. Thus, for fixed $(\Lambda_{k_1}, \delta^2_{k_1}, k_1, \boldsymbol{\mu}_{1:k_1}) \in \mathbb{R}^{+2} \times \boldsymbol{\Omega}$, we have:

$$\begin{aligned}
&p\big((\Lambda_{k_2}, \delta^2_{k_2}, k_2, \boldsymbol{\mu}_{1:k_2}) \in A_{k_2} | (\Lambda_{k_1}, \delta^2_{k_1}, k_1, \boldsymbol{\mu}_{1:k_1})\big) \\
&\qquad\qquad = \int_{A_{k_2}} \mathcal{K}(\Lambda_{k_1}, \delta^2_{k_1}, k_1, \boldsymbol{\mu}_{1:k_1}; d\Lambda_{k_2}, d\delta^2_{k_2}, k_2, d\boldsymbol{\mu}_{1:k_2}) \qquad \text{(E.4)}
\end{aligned}$$

where $A_{k_2} \in \mathcal{B}(\mathbb{R}^{+2} \times \{k_2\} \times \boldsymbol{\Omega}_{k_2})$. This transition kernel is by construction (Section 5.3.2) a mixture of transition kernels. Hence:

$$\begin{aligned}
&\mathcal{K}(\Lambda_{k_1}, \delta^2_{k_1}, k_1, \boldsymbol{\mu}_{1:k_1}; d\Lambda_{k_2}, d\delta^2_{k_2}, k_2, d\boldsymbol{\mu}_{1:k_2}) \\
&\quad = \Big(b_{k_1} \mathcal{K}_{birth}(\Lambda_{k_1}, \delta^2_{k_1}, k_1, \boldsymbol{\mu}_{1:k_1}; d\Lambda_{k_1}, d\delta^2_{k_1}, k_1+1, d\boldsymbol{\mu}_{1:k_1+1}) \\
&\quad + d_{k_1} \mathcal{K}_{death}(\Lambda_{k_1}, \delta^2_{k_1}, k_1, \boldsymbol{\mu}_{1:k_1}; d\Lambda_{k_1}, d\delta^2_{k_1}, k_1-1, d\boldsymbol{\mu}_{1:k_1-1}) \\
&\quad + s_{k_1} \mathcal{K}_{split}(\Lambda_{k_1}, \delta^2_{k_1}, k_1, \boldsymbol{\mu}_{1:k_1}; d\Lambda_{k_1}, d\delta^2_{k_1}, k_1+1, d\boldsymbol{\mu}_{1:k_1+1}) \\
&\quad + m_{k_1} \mathcal{K}_{merge}(\Lambda_{k_1}, \delta^2_{k_1}, k_1, \boldsymbol{\mu}_{1:k_1}; d\Lambda_{k_1}, d\delta^2_{k_1}, k_1-1, d\boldsymbol{\mu}_{1:k_1-1}) \\
&\quad + (1 - b_{k_1} - d_{k_1} - s_{k_1} - m_{k_1}) \mathcal{K}_{update}(\Lambda_{k_1}, \delta^2_{k_1}, k_1, \boldsymbol{\mu}_{1:k_1}; d\Lambda_{k_1}, d\delta^2_{k_1}, k_1, d\boldsymbol{\mu}^*_{1:k_1}) \Big) \\
&\qquad \times p(\delta^2_{k_2} | \delta^2_{k_1}, k_2, \boldsymbol{\mu}_{1:k_2}, \mathbf{x}, \mathbf{y}) p(\Lambda_{k_2} | k_2) d\Lambda_{k_2} d\delta^2_{k_2} \qquad \text{(E.5)}
\end{aligned}$$

where $\mathcal{K}_{birth}$ and $\mathcal{K}_{death}$ correspond to the reversible jumps described in Section 5.3.2.1, $\mathcal{K}_{split}$ and $\mathcal{K}_{merge}$ to the reversible jumps described in Section 5.3.2.2 and $\mathcal{K}_{update}$ is described in Section 5.3.2.3. The different steps for sampling the parameters $\delta^2_{k_2}$ and Λ_{k_2} are described in Section 5.3.1.3.

Lemma 1 *Let* $\mathbf{P}^*_k$ *denote the matrix* $\mathbf{P}_k$ *for which* $\delta^2 \to +\infty$ *and let* $\mathbf{v} \in \mathbb{R}^N$*, then* $\mathbf{v}'\mathbf{P}^*_k\mathbf{v} = 0$ *if and only if* $\mathbf{v}$ *belongs to the space spanned by the columns of* $\mathbf{D}\left(\boldsymbol{\mu}_{1:k}, \mathbf{x}\right)$*, with* $\boldsymbol{\mu}_{1:k} \in \boldsymbol{\Omega}_k$*.*

Proof. Let us first assume that $\mathbf{v}$ belongs to the space spanned by the columns of $\mathbf{D}\left(\boldsymbol{\mu}_{1:k}, \mathbf{x}\right)$, then it can be written as a linear combination as follows:

$$\mathbf{v} = \mathbf{D}(\boldsymbol{\mu}_{1:k}, \mathbf{x})Q$$

Hence:

$$\begin{aligned}
\mathbf{v}'\mathbf{P}^*_k\mathbf{v} &= Q'\mathbf{D}'(\boldsymbol{\mu}_{1:k}, \mathbf{x}) \Big(\mathbf{I}_N - \mathbf{D}(\boldsymbol{\mu}_{1:k}, \mathbf{x}) \big(\mathbf{D}'(\boldsymbol{\mu}_{1:k}, \mathbf{x}) \mathbf{D}(\boldsymbol{\mu}_{1:k}, \mathbf{x})\big)^{-1} \mathbf{D}'(\boldsymbol{\mu}_{1:k}, \mathbf{x}) \Big) \mathbf{D}(\boldsymbol{\mu}_{1:k}, \mathbf{x})Q \\
&= 0
\end{aligned}$$

[2] In what follows, the notation Λ_k, δ^2_k is adopted for ease of presentation. This does not mean that these variables depend on the dimension k.

which proves that if $\mathbf{v}$ belongs to the space spanned by the columns of $\mathbf{D}\left(\boldsymbol{\mu}_{1:k},\mathbf{x}\right)$ then $\mathbf{v}'\mathbf{P}_k^*\mathbf{v}=0$. To complete the proof, let us assume that $\mathbf{v}'\mathbf{P}_k^*\mathbf{v}=0$ then:

$$\mathbf{v}'\left(\mathbf{v}-\mathbf{D}(\boldsymbol{\mu}_{1:k},\mathbf{x})\big(\mathbf{D}'(\boldsymbol{\mu}_{1:k},\mathbf{x})\mathbf{D}(\boldsymbol{\mu}_{1:k},\mathbf{x})\big)^{-1}\mathbf{D}'(\boldsymbol{\mu}_{1:k},\mathbf{x})\mathbf{v}\right)=0$$

Thus, taking into account that $\mathbf{D}(\boldsymbol{\mu}_{1:k},\mathbf{x})\big(\mathbf{D}'(\boldsymbol{\mu}_{1:k},\mathbf{x})\mathbf{D}(\boldsymbol{\mu}_{1:k},\mathbf{x})\big)^{-1}\mathbf{D}'(\boldsymbol{\mu}_{1:k},\mathbf{x})\mathbf{v}$ is the projection of $\mathbf{v}$ onto the space spanned by the columns of $\mathbf{D}(\boldsymbol{\mu}_{1:k},\mathbf{x})$, it follows that $\mathbf{v}'\mathbf{P}_k^*\mathbf{v}=0$ only if $\mathbf{v}$ belongs to the space spanned by the columns of $\mathbf{D}\left(\boldsymbol{\mu}_{1:k},\mathbf{x}\right)$ ■

Then, noting that $\mathbf{y}'\mathbf{P}_k\mathbf{y}=\frac{1}{1+\delta^2}\mathbf{y}'\mathbf{y}+\frac{\delta^2}{1+\delta^2}\mathbf{y}'\mathbf{P}_k^*\mathbf{y}$, we obtain the following corollary:

Corollary 2 *If the observed data $\mathbf{y}$ is really noisy, i.e. it cannot be described as the sum of k basis functions and a linear mapping, then there exists a number $\varepsilon>0$ such that for all $k\leq k_{\max}$, $\delta^2\in\mathbb{R}^+$ and $\boldsymbol{\mu}_{1:k}\in\boldsymbol{\Omega}_k$*

$$\mathbf{y}'\mathbf{P}_k\mathbf{y}\geq\varepsilon>0 \tag{E.6}$$

■

Lemma 2 *For all $k\leq k_{\max}$, $\delta^2\in\mathbb{R}^+$ and $\boldsymbol{\mu}_{1:k}\in\boldsymbol{\Omega}_k$*

$$\mathbf{y}'\mathbf{P}_k\mathbf{y}\leq\mathbf{y}'\mathbf{y} \tag{E.7}$$

■

Lemma 3 *Let K_1 be the transition kernel corresponding to $\mathcal{K}$ such that Λ and δ^2 are kept fixed. Then there exists $M_1>0$ such that for any M_2 sufficiently large, any $\delta^2\in\mathbb{R}^+$ and $k_1=1,\ldots,k_{\max}$:*

$$K_1(\Lambda,\delta^2,k_1,\boldsymbol{\mu}_{1:k_1};\Lambda,\delta^2,k_1-1,d\boldsymbol{\mu}_{1:k_1-1})\geq\frac{c^\star\mathbb{I}_{\{\Lambda;\Lambda<M_2\}}(\Lambda)}{M_1M_2k_1}\delta_{S_{\boldsymbol{\mu}_{1:k_1}}}(d\boldsymbol{\mu}_{1:k_1-1}) \tag{E.8}$$

with $c^\star>0$ as defined in equation (5.20).

Proof. According to the definition of the transition kernel, for all pairs $((k_1,\boldsymbol{\mu}_{1:k_1}),(k_2,\boldsymbol{\mu}_{1:k_2}))\in\boldsymbol{\Omega}^2$, one has the following inequality:

$$K_1(\Lambda,\delta^2,k_1,\boldsymbol{\mu}_{1:k_1};\Lambda,\delta^2,k_2,d\boldsymbol{\mu}_{1:k_2})\geq\min\{1,r_{death}\}d_{k_1}\frac{\delta_{S_{\boldsymbol{\mu}_{1:k_1}}}(d\boldsymbol{\mu}_{1:k_2})}{k_1} \tag{E.9}$$

where $1/k_1$ is the probability of choosing one of the basis functions for the purpose of removing it and $S_{\boldsymbol{\mu}_{1:k_1}}\triangleq\{\boldsymbol{\mu}'\in\boldsymbol{\Omega}_{k_1-1}/\exists l\in\{1,\ldots,k_1\}$ such that $\boldsymbol{\mu}'=\boldsymbol{\mu}_{-l}\}$. Then from equation (5.23) and for all $k_1=1,\ldots,k_{\max}$, we have:

$$r_{death}^{-1}=\left(\frac{\gamma_0+\mathbf{y}_{1:N}'\mathbf{P}_{k_1-1}\mathbf{y}_{1:N}}{\gamma_0+\mathbf{y}_{1:N}'\mathbf{P}_{k_1}\mathbf{y}_{1:N}}\right)^{\left(\frac{N+v_0}{2}\right)}\frac{1}{k_1(1+\delta^2)^{1/2}}$$

As a result, we can use Lemmas 1 and 2 to obtain ε and M_1 such that:

$$r_{death}^{-1} \le \left(\frac{\gamma_0 + \mathbf{y}'_{1:N}\mathbf{y}_{1:N}}{\varepsilon}\right)^{(\frac{N+v_0}{2})} \frac{1}{k_1(1+\delta^2)^{1/2}} < M_1 < +\infty \tag{E.10}$$

Thus there exists M_1 sufficiently large such that for any M_2 sufficiently large (from equation (5.20)), $\delta^2 \in \mathbb{R}^+$, $1 \le k_1 \le k_{\max}$ and $\boldsymbol{\mu}_{1:k_1} \in \mathbf{\Omega}_{k_1}$

$$K_1(\Lambda, \delta^2, k_1, \boldsymbol{\mu}_{1:k_1}; \Lambda, \delta^2, k_1 - 1, d\boldsymbol{\mu}_{1:k_1-1}) \ge \mathbb{I}_{\{\Lambda;\Lambda<M_2\}}(\Lambda)\frac{c^\star}{M_2}\frac{1}{M_1 k_1}\delta_{S_{\boldsymbol{\mu}_{1:k_1}}}(d\boldsymbol{\mu}_{1:k_1-1}) \tag{E.11}$$

■

Lemma 4 *The transition kernel $\mathcal{K}$ satisfies the following inequality for $k = 0$:*

$$\mathcal{K}(\Lambda_0, \delta_0^2, 0, \boldsymbol{\mu}_0; d\Lambda_0^*, d\delta_0^{*2}, 0, d\boldsymbol{\mu}_0) \ge \zeta\varphi(\delta_0^{*2}|0)p(\Lambda_0|0)d\delta_0^{*2}d\Lambda_0 \tag{E.12}$$

with $\zeta > 0$ and φ a probability density.

Proof. From the definition of the transition kernel $\mathcal{K}$, it follows that[3]:

$$\begin{aligned}\mathcal{K}(\Lambda_0, \delta_0^2, 0, \boldsymbol{\mu}_0; d\Lambda_0^*, d\delta_0^{*2}, 0, d\boldsymbol{\mu}_0) &\ge u_0 p(\delta_0^{*2}|\delta_0^2, 0, d\boldsymbol{\mu}_0, \mathbf{x}, \mathbf{y})p(\Lambda_0|0)d\delta_0^{*2}d\Lambda_0 \\ &\ge (1-c^\star)p(\delta_0^{*2}|\delta_0^2, 0, d\boldsymbol{\mu}_0, \mathbf{x}, \mathbf{y})p(\Lambda_0|0)d\delta_0^{*2}d\Lambda_0\end{aligned} \tag{E.13}$$

as $0 < 1 - c^\star \le u_0 \le 1$ and the notation $\varphi(\delta^{*2}|0) \triangleq p(\delta^{*2}|\delta^2, 0, \boldsymbol{\mu}_0, \mathbf{x}, \mathbf{y})$ is adopted
■

Lemma 5 *There exists a constant $\xi > 0$ and a probability density φ such that for all $\delta^2 \in \mathbb{R}^+$, $0 \le k \le k_{\max}$ and $\boldsymbol{\mu}_{1:k} \in \mathbf{\Omega}_k$ one obtains:*

$$p(\delta^{*2}|\delta^2, k, \boldsymbol{\mu}_{1:k}, \mathbf{x}, \mathbf{y}) \ge \xi\varphi(\delta^{*2}|k) \tag{E.14}$$

Proof. From Section 4.1.2, to update δ^2 at each iteration one draws from the distribution $p(\boldsymbol{\alpha}_{1:m}, \sigma^2|\delta^2, k, \boldsymbol{\mu}_{1:k}, \mathbf{x}, \mathbf{y})$, that is, one draws σ^2 from:

$$p(\sigma^2|\delta^2, k, \boldsymbol{\mu}_{1:k}, \mathbf{x}, \mathbf{y}) = \frac{\left(\frac{\gamma_0 + \mathbf{y}'_{1:N}\mathbf{P}_k\mathbf{y}_{1:N}}{2}\right)^{\frac{N+v_0}{2}}}{\Gamma\left(\frac{N+v_0}{2}\right)(\sigma^2)^{\frac{N+v_0}{2}+1}} \exp\left(\frac{-1}{2\sigma^2}(\gamma_0 + \mathbf{y}'_{1:N}\mathbf{P}_k\mathbf{y}_{1:N})\right) \tag{E.15}$$

then $\boldsymbol{\alpha}_{1:m}$ from:

$$p(\boldsymbol{\alpha}_{1:m}|\delta^2, k, \boldsymbol{\mu}_{1:k}, \sigma^2, \mathbf{x}, \mathbf{y}) = \frac{1}{|2\pi\sigma^2\mathbf{M}_k|^{1/2}} \exp\left(\frac{-1}{2\sigma^2}(\boldsymbol{\alpha}_{1:m} - \mathbf{h}_k)'\mathbf{M}_k^{-1}(\boldsymbol{\alpha}_{1:m} - \mathbf{h}_k)\right) \tag{E.16}$$

[3] When $k = 0$, the same notation for the transition kernel is used for convenience, even if $\boldsymbol{\mu}_0$ does not exist.

and finally one draws δ^{*2} according to:

$$p(\delta^{*2}|\delta^2, k, \boldsymbol{\theta}, \mathbf{x}, \mathbf{y}) = \frac{\left(\frac{\boldsymbol{\alpha}'_{1:m}\mathbf{D}'(\boldsymbol{\mu}_{1:k},\mathbf{x})\mathbf{D}(\boldsymbol{\mu}_{1:k},\mathbf{x})\boldsymbol{\alpha}_{1:m}}{2\sigma^2} + \beta_{\delta^2}\right)^{m/2+\alpha_{\delta^2}}}{\Gamma(m/2+\alpha_{\delta^2})(\delta^{*2})^{m/2+\alpha_{\delta^2}+1}} \times \exp\left(\frac{-1}{\delta^{*2}}\left(\frac{\boldsymbol{\alpha}'_{1:m}\mathbf{D}'(\boldsymbol{\mu}_{1:k},\mathbf{x})\mathbf{D}(\boldsymbol{\mu}_{1:k},\mathbf{x})\boldsymbol{\alpha}_{1:m}}{2\sigma^2} + \beta_{\delta^2}\right)\right) \quad \text{(E.17)}$$

Consequently:

$$\begin{aligned} & p(\delta^{*2}|\delta^2, k, \boldsymbol{\theta}, \mathbf{x}, \mathbf{y})p(\boldsymbol{\alpha}_{1:m}, \sigma^2|\delta^2, k, \boldsymbol{\mu}_{1:k}, \mathbf{x}, \mathbf{y}) \\ &= \frac{\left(\frac{\gamma_0+\mathbf{y}'_{1:N}\mathbf{P}_k\mathbf{y}_{1:N}}{2}\right)^{\frac{N+v_0}{2}}\left(\frac{\boldsymbol{\alpha}'_{1:m}\mathbf{D}'(\boldsymbol{\mu}_{1:k},\mathbf{x})\mathbf{D}(\boldsymbol{\mu}_{1:k},\mathbf{x})\boldsymbol{\alpha}_{1:m}}{2\sigma^2} + \beta_{\delta^2}\right)^{m/2+\alpha_{\delta^2}}}{\Gamma(\frac{N+v_0}{2})\Gamma(m/2+\alpha_{\delta^2})(2\pi)^{m/2}(\sigma^2)^{(N+v_0+m)/2+1}(\delta^{*2})^{m/2+\alpha_{\delta^2}+1}|\mathbf{M}_k|^{1/2}} \\ & \times \exp\left(\frac{-1}{2\sigma^2}\left[(\boldsymbol{\alpha}_{1:m}-\mathbf{h}_k)'\mathbf{M}_k^{-1}(\boldsymbol{\alpha}_{1:m}-\mathbf{h}_k) + \gamma_0 + \mathbf{y}'_{1:N}\mathbf{P}_k\mathbf{y}_{1:N}\right.\right. \\ & \qquad \left.\left. + \frac{\boldsymbol{\alpha}'_{1:m}\mathbf{D}'(\boldsymbol{\mu}_{1:k},\mathbf{x})\mathbf{D}(\boldsymbol{\mu}_{1:k},\mathbf{x})\boldsymbol{\alpha}_{1:m}}{\delta^{*2}}\right] - \frac{\beta_{\delta^2}}{\delta^{*2}}\right) \qquad \text{(E.18)} \end{aligned}$$

The minorisation condition, given by equation (E.14), can be obtained by integrating with respect to the nuisance parameters $\boldsymbol{\alpha}_{1:m}$ and σ^2. To accomplish this, some algebraic manipulations are carried out to obtain the following relation:

$$\begin{aligned} & (\boldsymbol{\alpha}_{1:m}-\mathbf{h}_k)'\mathbf{M}_k^{-1}(\boldsymbol{\alpha}_{1:m}-\mathbf{h}_k) + (\gamma_0 + \mathbf{y}_{1:N}\mathbf{P}_k\mathbf{y}_{1:N}) + \frac{\boldsymbol{\alpha}'_{1:m}\mathbf{D}'(\boldsymbol{\mu}_{1:k},\mathbf{x})\mathbf{D}(\boldsymbol{\mu}_{1:k},\mathbf{x})\boldsymbol{\alpha}_{1:m}}{\delta^{*2}} \\ = \ & (\boldsymbol{\alpha}_{1:m}-\mathbf{h}_k^\bullet)'\mathbf{M}_k^{\bullet-1}(\boldsymbol{\alpha}_{1:m}-\mathbf{h}_k^\bullet) + \gamma_0 + \mathbf{y}_{1:N}\mathbf{P}_k^\bullet\mathbf{y}_{1:N} \qquad \text{(E.19)} \end{aligned}$$

with:

$$\begin{aligned} \mathbf{M}_k^{\bullet-1} &= \left(1+\frac{1}{\delta^2}+\frac{1}{\delta^{*2}}\right)\mathbf{D}'(\boldsymbol{\mu}_{1:k},\mathbf{x})\mathbf{D}(\boldsymbol{\mu}_{1:k},\mathbf{x}) \\ \mathbf{h}_k^\bullet &= \mathbf{M}_k^\bullet\mathbf{D}'(\boldsymbol{\mu}_{1:k},\mathbf{x})\mathbf{y}_{1:N} \\ \mathbf{P}_k^\bullet &= \mathbf{I}_N - \mathbf{D}(\boldsymbol{\mu}_{1:k},\mathbf{x})\mathbf{M}_k^\bullet\mathbf{D}'(\boldsymbol{\mu}_{1:k},\mathbf{x}) \end{aligned}$$

We can now integrate with respect to $\boldsymbol{\alpha}_{1:m}$ (Gaussian distribution) and with respect to $\boldsymbol{\sigma}^2$ (inverse Gamma distribution) to obtain the minorisation condition for δ^{*2}:

$$\begin{aligned}
&p(\delta^{*2}|\delta^2, k, \boldsymbol{\mu}_{1:k}, \mathbf{x}, \mathbf{y}) \\
\geq\ & \int_{\mathbb{R}^m \times \mathbb{R}^+} \frac{\left(\frac{\gamma_0 + \mathbf{y}'_{1:N}\mathbf{P}_k \mathbf{y}_{1:N}}{2}\right)^{\frac{N+v_0}{2}} \left(\beta_{\delta^2}\right)^{m/2+\alpha_{\delta^2}}}{\Gamma(\frac{N+v_0}{2})\Gamma(m/2+\alpha_{\delta^2})(2\pi)^{m/2}(\sigma^2)^{(N+v_0+m)/2+1}(\delta^{*2})^{m/2+\alpha_{\delta^2}+1}|\mathbf{M}_k|^{1/2}} \\
&\times \exp\left(\frac{-1}{2\sigma^2}\left[(\boldsymbol{\alpha}_{1:m} - \mathbf{h}_k^\bullet)'\mathbf{M}_k^{\bullet -1}(\boldsymbol{\alpha}_{1:m} - \mathbf{h}_k^\bullet) + \gamma_0 + \mathbf{y}'_{1:N}\mathbf{P}_k^\bullet \mathbf{y}_{1:N}\right] - \frac{\beta_{\delta^2}}{\delta^{*2}}\right) d\boldsymbol{\alpha}_{1:m} d\sigma^2 \\
=\ & \frac{|\mathbf{M}_k^\bullet|^{1/2}}{|\mathbf{M}_k|^{1/2}} \frac{\left(\frac{\gamma_0 + \mathbf{y}'_{1:N}\mathbf{P}_k \mathbf{y}_{1:N}}{2}\right)^{\frac{N+v_0}{2}} \left(\beta_{\delta^2}\right)^{m/2+\alpha_{\delta^2}}}{\Gamma(m/2+\alpha_{\delta^2})\left(\frac{\gamma_0 + \mathbf{y}'_{1:N}\mathbf{P}_k^\bullet \mathbf{y}_{1:N}}{2}\right)^{\frac{N+v_0}{2}} (\delta^{*2})^{m/2+\alpha_{\delta^2}+1}} \exp\left(-\frac{\beta_{\delta^2}}{\delta^{*2}}\right) \\
\geq\ & \left(\frac{1+\frac{1}{\delta^2}}{1+\frac{1}{\delta^2}+\frac{1}{\delta^{*2}}}\right)^{m/2} \frac{\varepsilon^{\frac{N+v_0}{2}} \beta_{\delta^2}^{m/2+\alpha_{\delta^2}}}{(\gamma_0 + \mathbf{y}'_{1:N}\mathbf{y}_{1:N})^{\frac{N+v_0}{2}}\Gamma(m/2+\alpha_{\delta^2})} \frac{1}{(\delta^{*2})^{m/2+\alpha_{\delta^2}+1}} \\
&\times \exp\left(-\frac{\beta_{\delta^2}}{\delta^{*2}}\right) \\
\geq\ & \left(\frac{1}{1+\delta^{*2}}\right)^{k_{\max}/2} \frac{\varepsilon^{\frac{N+v_0}{2}} \min_{k\in\{0,\ldots,k_{\max}\}} \beta_{\delta^2}^{m/2+\alpha_{\delta^2}}}{(\gamma_0 + \mathbf{y}'_{1:N}\mathbf{y}_{1:N})^{\frac{N+v_0}{2}}\Gamma\left((k_{\max}+d+1)/2+\alpha_{\delta^2}\right)} \\
&\times \frac{1}{(\delta^{*2})^{\frac{k_{\max}+d+1}{2}+\alpha_{\delta^2}+1}} \exp\left(-\frac{\beta_{\delta^2}}{\delta^{*2}}\right) \qquad \text{(E.20)}
\end{aligned}$$

where Lemma 1, its corollary and Lemma 2 have been used ■

Proposition 9 *For any M_2 large enough, there exists $\eta_{M_2} > 0$ such that for all pairs $((\Lambda_{k_1}, \delta_{k_1}^2, k_1, \boldsymbol{\mu}_{1:k_1}), (\Lambda_{k_2}, \delta_{k_2}^2, k_2, \boldsymbol{\mu}_{1:k_2})) \in (\mathbb{R}^{+2} \times \boldsymbol{\Omega})^2$*

$$\begin{aligned}
&\mathcal{K}^{(k_{\max})}(\Lambda_{k_1}, \delta_1^2, k_1, \boldsymbol{\mu}_{1:k_1}; d\Lambda_{k_2}, d\delta_{k_2}^2, k_2, d\boldsymbol{\mu}_{1:k_2}) \\
&\qquad \geq \mathbb{I}_{\{\Lambda_{k_1};\Lambda_{k_1}<M_2\}}(\Lambda_{k_1})\eta_{M_2}\phi(d\Lambda_{k_2}, d\delta_{k_2}^2, k_2, d\boldsymbol{\mu}_{1:k_2}) \qquad \text{(E.21)}
\end{aligned}$$

where $\phi(d\Lambda, d\delta^2, k, d\boldsymbol{\mu}_{1:k}) \triangleq p(\Lambda|k)d\Lambda\varphi(\delta^2|k)d\delta^2\mathbb{I}_{\{0\}}(k)\delta_{\{\boldsymbol{\mu}_0\}}(d\boldsymbol{\mu}_{1:k})$.

Proof. From Lemmas 3 and 5, one obtains for $k_1 = 1, \ldots, k_{\max}$:

$$\begin{aligned}
\mathcal{K}(\Lambda_{k_1}, \delta_{k_1}^2, k_1, \boldsymbol{\mu}_{1:k_1}; d\Lambda_{k_1-1}, d\delta_{k_1-1}^2, k_1-1, d\boldsymbol{\mu}_{k_1-1}) \ &\geq\ \mathbb{I}_{\{\Lambda_{k_1};\Lambda_{k_1}<M_2\}}(\Lambda_{k_1})\frac{c^\star}{M_2}\frac{1}{M_1 k_1} \\
&\times \xi p(\Lambda_{k_1-1}|k_1-1)d\Lambda_{k_1-1}\varphi(\delta_{k_1-1}^2|k_1-1)d\delta_{k_1-1}^2\delta_{S_{\boldsymbol{\mu}_{1:k_1}}}(d\boldsymbol{\mu}_{k_1-1}) \\
&\qquad \text{(E.22)}
\end{aligned}$$

Consequently for $k_1 = 1, \ldots, k_{\max}$, when one iterates the kernel $\mathcal{K}$ $k_{\max}$ times, the resulting transition kernel denoted $\mathcal{K}^{(k_{\max})}$ satisfies:

$$\begin{aligned}
&\mathcal{K}^{(k_{\max})}(\Lambda_{k_1}, \delta^2_{k_1}, k_1, \boldsymbol{\mu}_{1:k_1}; d\Lambda^*_0, d\delta^{*2}_0, 0, d\boldsymbol{\mu}^*_0) \\
&\quad = \int_{\mathbb{R}^+ \times \mathbb{R}^+ \times \boldsymbol{\Omega}} \mathcal{K}^{(k_1)}(\Lambda_{k_1}, \delta^2_{k_1}, k_1, \boldsymbol{\mu}_{1:k_1}; d\Lambda_l, d\delta^2_l, l, d\boldsymbol{\mu}_{1:l}) \\
&\qquad\qquad \times \mathcal{K}^{(k_{\max}-k_1)}(\Lambda_l, \delta^2_l, l, \boldsymbol{\mu}_{1:l}; d\Lambda^*_0, d\delta^{*2}_0, 0, d\boldsymbol{\mu}^*_0) \\
&\quad \geq \int_{\mathbb{R}^+ \times \mathbb{R}^+} \int_{\{0\} \times \boldsymbol{\Omega}_0} \mathcal{K}^{(k_1)}(\Lambda_{k_1}, \delta^2_{k_1}, k_1, \boldsymbol{\mu}_{1:k_1}; d\Lambda_l, d\delta^2_l, l, d\boldsymbol{\mu}_{1:l}) \\
&\qquad\qquad\qquad \times \mathcal{K}^{(k_{\max}-k_1)}(\Lambda_l, \delta^2_l, l, \boldsymbol{\mu}_{1:l}; d\Lambda^*_0, d\delta^{*2}_0, 0, d\boldsymbol{\mu}^*_0) \\
&\quad = \mathcal{K}^{(k_1)}(\Lambda_{k_1}, \delta^2_{k_1}, k_1, \boldsymbol{\mu}_{1:k_1}; d\Lambda_0, d\delta^2_0, 0, d\boldsymbol{\mu}_0)\mathcal{K}^{(k_{\max}-k_1)}(\Lambda_0, \delta^2_0, 0, \boldsymbol{\mu}_0; d\Lambda^*_0, d\delta^{*2}_0, 0, d\boldsymbol{\mu}^*_0) \\
&\quad \geq \mathbb{I}_{\{\Lambda_{k_1}; \Lambda_{k_1} < M_2\}}(\Lambda_{k_1}) M_3^{k_1 - 1}(\frac{\xi c^\star}{M_1 M_2})^{k_1} \varsigma^{k_{\max}-k_1} \phi(d\Lambda^*_0, d\delta^{*2}_0, 0, d\boldsymbol{\mu}^*_0) \qquad \text{(E.23)}
\end{aligned}$$

where Lemma 4 has been used and $M_3 = \min_{k=1,\ldots,k_{\max}} \int_{\{\Lambda;\Lambda<M_2\}} p(\Lambda|k) d\Lambda > 0$. The conclusion follows with $\eta_{M_2} \triangleq \min\{\varsigma^{k_{\max}}, \min_{k \in \{1,\ldots,k_{\max}\}} M_3^{k-1}(\frac{\xi c^\star}{M_1 M_2})^k \varsigma^{k_{\max}-k}\} > 0$
■

Corollary 3 *The transition kernel $\mathcal{K}$ is ϕ-irreducible. As $p(d\Lambda, d\delta^2, k, d\boldsymbol{\mu}_{1:k}|\mathbf{x}, \mathbf{y})$ is an invariant distribution of $\mathcal{K}$ and the Markov chain is ϕ-irreducible, then from (Tierney, 1994, Theorem 1*, pp. 1758) the Markov chain is $p(d\Lambda, d\delta^2, k, d\boldsymbol{\mu}_{1:k}|\mathbf{x}, \mathbf{y})$-irreducible. Aperiodicity is straightforward. Indeed, there is a non-zero probability of choosing the update move in the empty configuration from equation (E.12) and to move anywhere in $\mathbb{R}^2 \times \{0\} \times \{\boldsymbol{\mu}_0\}$. Therefore the Markov chain admits $p(d\Lambda, d\delta^2, k, d\boldsymbol{\mu}_{1:k}|\mathbf{x}, \mathbf{y})$ as unique equilibrium distribution (Tierney, 1994, Theorem 1*, pp. 1758).*

The drift condition will be proved subsequently.

Proposition 10 *Let $V(\Lambda, \delta^2, k, \boldsymbol{\mu}_{1:k}) \triangleq \max\{1, \Lambda^v\}$ for $v > 0$, then:*

$$\lim_{\Lambda \to +\infty} \mathcal{K}V(\Lambda, \delta^2, k, \boldsymbol{\mu}_{1:k}) / V(\Lambda, \delta^2, k, \boldsymbol{\mu}_{1:k}) = 0 \qquad \text{(E.24)}$$

where by definition:

$$\mathcal{K}V(\Lambda, \delta^2, k, \boldsymbol{\mu}_{1:k}) \triangleq \int_{\mathbb{R}^+ \times \mathbb{R}^+ \times \boldsymbol{\Omega}} \mathcal{K}(\Lambda, \delta^2, k, \boldsymbol{\mu}_{1:k}; d\Lambda^*, d\delta^{*2}, k^*, d\boldsymbol{\mu}^*_{1:k}) V(\Lambda^*, \delta^{*2}, k^*, \boldsymbol{\mu}^*_{1:k}) \qquad \text{(E.25)}$$

Proof. The transition kernel of the Markov chain is of the form (some arguments are removed for convenience):

$$\begin{aligned}
\mathcal{K} = &\left(b_{k_1}\mathcal{K}_{birth} + d_{k_1}\mathcal{K}_{death} + m_{k_1}\mathcal{K}_{merge} + s_{k_1}\mathcal{K}_{split} + (1 - b_{k_1} - d_{k_1} - s_{k_1} - m_{k_1})\mathcal{K}_{update}\right) \\
&\times p(\delta^2_{k_2}|\delta^2_{k_1}, k_2, \boldsymbol{\mu}_{1:k_2}, \mathbf{x}, \mathbf{y}) p(\Lambda_{k_2}|k_2) \qquad \text{(E.26)}
\end{aligned}$$

Now, study the following expression:

$$\begin{aligned}
& \mathcal{K}V(\Lambda_{k_1},\delta_1^2,k_1,\boldsymbol{\mu}_{1:k_1}) \\
= \; & b_{k_1}\sum_{k_2\in\{k_1,k_1+1\}}\int_{\Phi_{k_2}}\mathcal{K}_{birth}\int_{\mathbb{R}+}p(\delta_{k_2}^2|\delta_{k_1}^2,k_2,\boldsymbol{\mu}_{1:k_2},\mathbf{x},\mathbf{y})d\delta_{k_2}^2\int_{\mathbb{R}+}p(\Lambda_{k_2}|k_2)\Lambda_{k_2}^{v}d\Lambda_{k_2} \\
& +d_{k_1}\sum_{k_2\in\{k_1,k_1-1\}}\int_{\Phi_{k_2}}\mathcal{K}_{death}\int_{\mathbb{R}+}p(\delta_{k_2}^2|\delta_{k_1}^2,k_2,\boldsymbol{\mu}_{1:k_2},\mathbf{x},\mathbf{y})d\delta_{k_2}^2\int_{\mathbb{R}+}p(\Lambda_{k_2}|k_2)\Lambda_{k_2}^{v}d\Lambda_{k_2} \\
& +s_{k_1}\sum_{k_2\in\{k_1,k_1+1\}}\int_{\Phi_{k_2}}\mathcal{K}_{split}\int_{\mathbb{R}+}p(\delta_{k_2}^2|\delta_{k_1}^2,k_2,\boldsymbol{\mu}_{1:k_2},\mathbf{x},\mathbf{y})d\delta_{k_2}^2\int_{\mathbb{R}+}p(\Lambda_{k_2}|k_2)\Lambda_{k_2}^{v}d\Lambda_{k_2} \\
& +m_{k_1}\sum_{k_2\in\{k_1,k_1-1\}}\int_{\Phi_{k_2}}\mathcal{K}_{merge}\int_{\mathbb{R}+}p(\delta_{k_2}^2|\delta_{k_1}^2,k_2,\boldsymbol{\mu}_{1:k_2},\mathbf{x},\mathbf{y})d\delta_{k_2}^2\int_{\mathbb{R}+}p(\Lambda_{k_2}|k_2)\Lambda_{k_2}^{v}d\Lambda_{k_2} \\
& +(1-b_{k_1}-d_{k_1}-s_{k_1}-m_{k_1})\int_{\boldsymbol{\Omega}_{k_1}}\mathcal{K}_{update}\int_{\mathbb{R}+}p(\delta_{k_1}^{*2}|\delta_{k_1}^2,k_1,\boldsymbol{\mu}_{1:k_1}^{*},\mathbf{x},\mathbf{y})d\delta_{k_1}^2\int_{\mathbb{R}+}p(\Lambda_{k_1}^{*}|k_1)\Lambda_{k_1}^{*v}d\Lambda_{k_1}^{*} \\
\leq \; & b_{k_1}\sum_{k_2\in\{k_1,k_1+1\}}\int_{\mathbb{R}+}p(\Lambda_{k_2}|k_2)\Lambda_{k_2}^{v}d\Lambda_{k_2}+d_{k_1}\sum_{k_2\in\{k_1,k_1-1\}}\int_{\mathbb{R}+}p(\Lambda_{k_2}|k_2)\Lambda_{k_2}^{v}d\Lambda_{k_2} \\
& +s_{k_1}\sum_{k_2\in\{k_1,k_1+1\}}\int_{\mathbb{R}+}p(\Lambda_{k_2}|k_2)\Lambda_{k_2}^{v}d\Lambda_{k_2}+m_{k_1}\sum_{k_2\in\{k_1,k_1-1\}}\int_{\mathbb{R}+}p(\Lambda_{k_2}|k_2)\Lambda_{k_2}^{v}d\Lambda_{k_2} \\
& +(1-b_{k_1}-d_{k_1}-s_{k_1}-m_{k_1})\int_{\mathbb{R}+}p(\Lambda_{k_1}^{*}|k_1)\Lambda_{k_1}^{*v}d\Lambda_{k_1}^{*}
\end{aligned}$$

Since $p(\Lambda|k)$ is a Gamma distribution, for any $0\leq k\leq k_{\max}$, one obtains the inequality $\int_{\mathbb{R}+}p(\Lambda|k)\Lambda^{v}d\Lambda<+\infty$ and the result follows immediately ■

Proof of Theorem 6

Proof. By construction, the transition kernel $\mathcal{K}(\Lambda_{k_1},\delta_{k_1}^2,k_1,\boldsymbol{\mu}_{1:k_1};d\Lambda_{k_2},d\delta_{k_2}^2,k_2,d\boldsymbol{\mu}_{1:k_2})$ admits $p(d\Lambda,d\delta^2,k,d\boldsymbol{\mu}_{1:k}|\mathbf{x},\mathbf{y})$ as invariant distribution. Proposition 9 proved the ϕ-irreducibility and the minorisation condition with $k_0=k_{\max}$ and Proposition 10 proved the drift condition, thus Theorem 6 applies ■

Bibliography

Akaike, H. (1974). A new look at statistical model identification. *IEEE Transactions on Automatic Control*, AC-19:716–723.

Akashi, H. and Kumamoto, H. (1977). Random sampling approach to state estimation in switching environments. *Automatica*, 13:429–434.

Anderson, B. D. and Moore, J. B. (1979). *Optimal Filtering*. Prentice-Hall, New Jersey.

Andrieu, C. (1998). *MCMC Methods for the Bayesian Analysis of Nonlinear Parametric Regression Models*. PhD thesis, University Cergy-Pontoise, Paris XV, France. In French.

Andrieu, C., Barat, E., and Doucet, A. (1999a). Bayesian deconvolution of noisy filtered point processes. Technical Report CUED/F-INFENG/TR 352, Cambridge University Engineering Department.

Andrieu, C., Breyer, L. A., and Doucet, A. (1999b). Convergence of simulated annealing using Foster-Lyapunov criteria. Technical Report CUED/F-INFENG/TR 346, Cambridge University Engineering Department.

Andrieu, C., de Freitas, J. F. G., and Doucet, A. (1999c). A brief note on maximum realisable MCMC classifiers. Technical Report CUED/F-INFENG/TR 358, Cambridge University Engineering Department.

Andrieu, C., de Freitas, J. F. G., and Doucet, A. (1999d). Robust full Bayesian learning for neural networks. Technical Report CUED/F-INFENG/TR 343, Cambridge University Engineering Department.

Andrieu, C., de Freitas, J. F. G., and Doucet, A. (1999e). Sequential MCMC for Bayesian model selection. In *IEEE Higher Order Statistics Workshop*, pages 130–134, Ceasarea, Israel.

Andrieu, C., Djurić, P. M., and Doucet, A. (1999f). Model selection by MCMC computation. To appear in *Signal Processing*.

Andrieu, C. and Doucet, A. (1998a). Efficient simulated annealing algorithms for Bayesian parameter estimation. In Djafari, A., editor, *Proceedings of the SPIE Conference on Bayesian Inference for Inverse Problems,* volume 3459, pages 283–294, San Diego, California.

Andrieu, C. and Doucet, A. (1998b). Efficient stochastic maximum a posteriori estimation for harmonic signals. In *EUSIPCO*, volume 3, pages 1821–1824, Island of Rhodes.

Andrieu, C. and Doucet, A. (1999). Joint Bayesian detection and estimation of noisy sinusoids via reversible jump MCMC. *IEEE Transactions on Signal Processing*, 47(10). To appear.

Avitzour, D. (1995). A stochastic simulation Bayesian approach to multitarget tracking. *IEE Proceedings on Radar, Sonar and Navigation*, 142(2):41–44.

Bakshi, B. R. and Stephanopoulos, G. (1993). Wave-net: A multiresolution, hierarchical neural network with localized learning. *AIChE Journal*, 39(1):57–80.

Bar-Shalom, Y. and Li, X. R. (1993). *Estimation and Tracking: Principles, Techniques and Software*. Artech House, Boston.

Barron, A. R. (1993). Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945.

Baum, L. E., Petrie, T., Soules, G., and Weiss, N. (1979). A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Annals of Mathematical Statistics*, 41:164–171.

Baxt, W. G. (1990). Use of an artificial neural network for data analysis in clinical decision-making: The diagnosis of acute coronary occlusion. *Neural Computation*, 2:480–489.

Beadle, E. R. and Djurić, P. M. (1997). A fast weighted Bayesian bootstrap filter for nonlinear model state estimation. *IEEE Transactions on Aerospace and Electronic Systems*, 33(1):338–343.

Bergman, N. (1999). *Recursive Bayesian Estimation: Navigation and Tracking Applications*. PhD thesis, Department of Electrical Engineering, Linköping University, Sweeden.

Bernardo, J. M. and Smith, A. F. M. (1994). *Bayesian Theory*. Wiley Series in Applied Probability and Statistics.

Berzuini, C., Best, N. G., Gilks, W. R., and Larizza, C. (1997). Dynamic conditional independence models and Markov Chain Monte Carlo methods. *Journal of the American Statistical Association*, 92(440):1403–1412.

Besag, J., Green, P. J., Hidgon, D., and Mengersen, K. (1995). Bayesian computation and stochastic systems. *Statistical Science*, 10(1):3–66.

Billings, S. A. (1980). Identification of nonlinear systems – a survey. *IEE Proceedings Part D*, 127(6):272–285.

Billingsley, P. (1985). *Probability and Measure*. Wiley.

Bishop, C. (1995a). Training with noise is equivalent to Tikhonov regularization. *Neural Computation*, 7(1):108–116.

Bishop, C. M. (1995b). *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford.

Black, F. and Scholes, M. (1973). The pricing of options and corporate liabilities. *Journal of Political Economy*, 81:637–659.

Blom, H. A. P. and Bar-Shalom, Y. (1988). The interacting multiple model algorithm for systems with Markovian switching coefficients. *IEEE Transactions on Automatic Control*, 33(8):780–783.

Bolviken, E. and Storvic, G. (To appear in 2000). Stochastic and deterministic particle filters. In Doucet, A., de Freitas, J. F. G., and Gordon, N. J., editors, *Sequential Monte Carlo Methods in Practice*. Springer-Verlag.

Brass, A., Pendleton, B. J., Chen, Y., and Robson, B. (1993). Hybrid Monte Carlo simulations theory and initial comparison with molecular dynamics. *Biopolymers*, 33(8):1307–1315.

Breiman, L. (1993). Hinging hyperplanes for regression, classification, and function approximation. *IEEE Transactions on Information Theory*, 39(3):999–1013.

Brooks, S. P. (1998). Markov chain Monte Carlo method and its application. *The Statistician*, 47(1):69–100.

Buntine, W. L. and Weigend, A. S. (1991). Bayesian back-propagation. *Complex Systems*, 5:603–643.

Candy, J. V. (1986). *Signal Processing: The Model Based Approach*. McGraw-Hill, New York.

Carpenter, J., Clifford, P., and Fearnhead, P. (1999). Building robust simulation-based filters for evolving data sets. Unpublished. Department of Statistics, Oxford University.

Chan, K. S. (1993). Asymptotic behaviour of the Gibbs sampler. *Journal of the American Statistical Association*, 88:320–326.

Chen, C. F. (1981). The EM algorithm to the multiple indicators and multiple causes model via the estimation of the latent variable. *Journal of the American Statistical Association*, 76(375):704–708.

Clapp, T. C. and Godsill, S. J. (1999). Fixed-lag blind equalization and sequence estimation in digital communications systems using sequential importance sampling. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 2495–2498, Arizona.

Cogburn, R. (1972). The central limit theorem for Markov processes. In *Sixth Berkeley Symposium on Mathematics, Statistics and Probability*, volume 2, pages 485–512, Berkeley. University of California Press.

Cornford, D., Ramage, G., and Nabney, I. T. (1998). A neural network sensor model with input noise. Technical Report NCRG/98/021, Neural Computing Research Group, Aston University, Birmingham, UK.

Crisan, D., Del Moral, P., and Lyons, T. (1999). Discrete filtering using branching and interacting particle systems. To appear in *Markov Processes and Related Fields*, vol. 3.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314.

de Freitas, J. F. G. (1997). Neural network based nonparametric regression for nonlinear system identification and fault detection. Master's thesis, University of the Witwatersrand, Johannesburg.

de Freitas, J. F. G., Gaylard, A., Stevens, A. L., Ridley, J. N., and Landy, C. F. (1996). Identification of vibrating structures and fault detection using neural networks. In *IEEE International Conference on Neural Networks*, volume 4, pages 2044–2048, Washington.

de Freitas, J. F. G., Niranjan, M., and Gee, A. H. (1997). Hierarchical Bayesian-Kalman models for regularisation and ARD in sequential learning. Technical Report CUED/F-INFENG/TR 307, Cambridge University Engineering Department.

Del Moral, P. and Guionnet, A. (1999). On the stability of measure valued processes. Applications to non linear filtering and interacting particle systems. Technical report, Publications du Laboratoire de Statistiques et Probabilités, No 03-98.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society Series B*, 39:1–38.

Denison, D. (1998). Bayesian MARS. *Statistics and Computing*, 8:337–346.

Digalakis, V., Rohlicek, J. R., and Ostendorf, M. (1993). ML estimation of a stochastic linear system with the EM algorithm and its application to speech recognition. *IEEE Transactions on Speech and Audio Processing*, 1(4):431–442.

Djurić, P. (To appear in 2000). Short term forecasting of electricity load. In Doucet, A., de Freitas, J. F. G., and Gordon, N. J., editors, *Sequential Monte Carlo Methods in Practice*. Springer-Verlag.

Djurić, P. M. (1996). A model selection rule for sinusoids in white Gaussian noise. *IEEE Transactions on Signal Processing*, 44(7):1744–1751.

Djurić, P. M. (1998). Asymptotic MAP criteria for model selection. *IEEE Transactions on Signal Processing*, 46(10):2726–2735.

Djurić, P. M. (1999). Monitoring and selection of dynamic models by Monte Carlo sampling. In *IEEE Higher Order Statistics Workshop*, pages 191–194, Ceasarea, Israel.

Doucet, A. (1997). *Monte Carlo Methods for Bayesian Estimation of Hidden Markov Models. Application to Radiation Signals*. PhD thesis, University Paris-Sud, Orsay, France. Chapters 4 and 5 in English.

Doucet, A. (1998). On sequential simulation-based methods for Bayesian filtering. Technical Report CUED/F-INFENG/TR 310, Department of Engineering, Cambridge University.

Doucet, A. and Gordon, N. J. (1999). Simulation-based optimal filter for manoeuvring target tracking. In *SPIE Signal and Data Processing of Small Targets*, volume SPIE 3809. To appear.

Doucet, A., Gordon, N. J., and Krishnamurthy, V. (1999). Particle filters for state estimation of jump Markov linear systems. Technical Report CUED/F-INFENG/TR 359, Cambridge University Engineering Department.

Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. (1987). Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222.

Efron, B. (1982). *The Bootstrap, Jacknife and other Resampling Plans*. SIAM, Philadelphia.

Everson, R. and Roberts, S. J. (1999). Non-stationary independent component analysis. Technical Report TR-99-1, Department of Electrical Engineering, Imperial College of London. To appear in *International Conference on Artificial Neural Networks*.

Fahlman, S. E. and Lebiere, C. (1988). The cascade-correlation learning architecture. In Touretzky, D. S., editor, *Proceedings of the Connectionist Models Summer School*, volume 2, pages 524–532, San Mateo, CA.

Fearnhead, P. (1998). *Sequential Monte Carlo Methods in Filter Theory*. PhD thesis, Department of Statistics, Oxford University, England.

Fisher, R. A. (1929). Tests of significance in harmonic analysis. *Proceedings of the Royal Society A*, 125:54–59.

Frean, M. (1990). The upstart algorithm: A method for constructing and training feedforward neural networks. *Neural Computation*, 2(2):198–209.

Friedman, J. H. (1991). Multivariate adaptive regression splines. *The Annals of Statistics*, 19(1):1–141.

Friedman, J. H. and Stuetzle, W. (1981). Projection pursuit regression. *Journal of the American Statistical Association*, 76(376):817–823.

Gelb, A., editor (1974). *Applied Optimal Estimation*. MIT Press.

Gelfand, A. E. and Dey, D. K. (1997). Bayesian model choice: Asymptotics and exact calculations. *Journal of the Royal Statistical Society B*, 56(3):501–514.

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.

Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741.

Genest, C. and Zidek, J. V. (1986). Combining probability distributions: A critique and annotated bibliography. *Statistical Science*, 1(1):114–148.

George, E. I. and Foster, D. P. (1997). Calibration and empirical Bayes variable selection. Unpublished. Department of Management Science and Information Systems, University of Texas.

Geweke, J. (1989). Bayesian inference in econometric models using Monte Carlo integration. *Econometrica*, 24:1317–1399.

Ghahramani, Z. (1997). Learning dynamic Bayesian networks. In Giles, C. L. and Gori, M., editors, *Adaptive Processing of Temporal Information*, volume 1387 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag.

Gilks, W. R. and Berzuini, C. (1998). Monte Carlo inference for dynamic Bayesian models. Unpublished. Medical Research Council, Cambridge, UK.

Gilks, W. R., Richardson, S., and Spiegelhalter, D. J., editors (1996). *Markov Chain Monte Carlo in Practice*. Chapman and Hall, Suffolk.

Girosi, F. and Anzellotti, G. (1995). Convergence rates of approximation by translates. Technical Report AIM-1288, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, MA.

Girosi, F., Jones, M., and Poggio, T. (1995). Regularization theory and neural networks architectures. *Neural Computation*, 7(2):219–269.

Gordon, N. and Whitby, A. (1995). A Bayesian approach to target tracking in the presence of a glint. In Drummond, O. E., editor, *Signal and Data Processing of Small Targets*, volume SPIE 2561, pages 472–483.

Gordon, N. J. (1994). *Bayesian Methods for Tracking*. PhD thesis, Imperial College, University of London.

Gordon, N. J., Salmond, D. J., and Smith, A. F. M. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings-F*, 140(2):107–113.

Graham, A. (1981). *Kronecker Products and Matrix Calculus with Applications*. Ellis Horwood Limited.

Green, P. J. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82:711–732.

Hammersley, J. H. and Handscomb, D. C. (1968). *Monte Carlo Methods*. Methuen, London.

Handschin, J. E. (1970). Monte Carlo techniques for prediction and filtering of non-linear stochastic processes. *Automatica*, 6:555–563.

Handschin, J. E. and Mayne, D. Q. (1969). Monte Carlo techniques to estimate the conditional expectation in multi-stage non-linear filtering. *International Journal of Control*, 9(5):547–559.

Hannan, E. J. (1961). Testing for a jump in the spectral function. *Journal of the Royal Statistical Society B*, 23:394–404.

Harvey, A. C. (1989). *Forecasting, Structural Time Series Models, and the Kalman Filter*. Cambridge University Press.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their Applications. *Biometrika*, 57:97–109.

Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation*. Macmillan College Publishing Company.

Hecht-Nielsen, R. (1990). *Neurocomputing*. Addison-Wesley.

Higuchi, T. (1997). Monte Carlo filter using the genetic algorithm operators. *Journal of Statistical Computation and Simulation*, 59(1):1–23.

Hinton, G. (1987). Learning translation invariant recognition in massively parallel networks. In de Bakker, J. W., Nijman, A. J., and Treleaven, P. C., editors, *Proceedings of the Conference on Parallel Architectures and Languages*, pages 1–13, Berlin.

Holmes, C. (1999). A Bayesian approach to generalised nonlinear modelling with multivariate smoothing splines. Unpublished. Statistics Section, Department of Mathematics, Imperial College of London.

Holmes, C. C. and Mallick, B. K. (1998). Bayesian radial basis functions of variable dimension. *Neural Computation*, 10(5):1217–1233.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward neural networks are universal approximators. *Neural Networks*, 2(5):359–366.

Huber, P. J. (1985). Projection pursuit. *The Annals of Statistics*, 13(2):435–475.

Hull, J. C. (1997). *Options, Futures, and Other Derivative*. Prentice Hall, third edition.

Hürzeler, M. (1998). *Statistical Methods for General State-Space Models*. PhD thesis, Swiss Federal Institute of Technology, Zurich, Switzerland.

Hürzeler, M. and Künsch, H. R. (1998). Monte Carlo approximation for general state-space models. *Journal of Computational and Graphical Statistics*, 7(2):175–193.

Hutchinson, J. M., Lo, A. W., and Poggio, T. (1994). A nonparametric approach to pricing and hedging derivative securities via learning networks. *The Journal of Finance*, 49(3):851–889.

Isard, M. and Blake, A. (1996). Contour tracking by stochastic propagation of conditional density. In *European Conference on Computer Vision*, pages 343–356, Cambridge, UK.

Jacobs, R. A. (1988). Increased rates of convergence through learning rates adaptation. *Neural Networks*, 1:295–307.

Jacobs, R. A. (1995). Methods for combining experts probability assessments. *Neural Computation*, 7(5):867–888.

Jang, J. S. R. and Sun, C. T. (1993). Functional equivalence between radial basis function networks and fuzzy inference systems. *IEEE Transactions on Neural Networks*, 4(1):156–159.

Jazwinski, A. H. (1969). Adaptive filtering. *Automatica*, 5:475–485.

Jazwinski, A. H. (1970). *Stochastic Processes and Filtering Theory*. Academic Press.

Jazwinski, A. H. and Bailie, A. E. (1967). Adaptive filtering. Technical Report 67–6, Analytical Mechanics Associates, Maryland.

Jones, L. K. (1992). A simple lemma on greedy approximation in Hilbert space and convergence rates for projection pursuit regression and neural network training. *The Annals of Statistics*, 20(1):608–613.

Juditsky, A., Hjalmarsson, H., Benveniste, A., Delyon, B., Ljung, and Sjöberg, J. (1995). Non-linear black-box models in system identification: Mathematical foundations. *Automatica*, 31(12):1725–1750.

Kadirkamanathan, V. and Kadirkamanathan, M. (1995). Recursive estimation of dynamic modular RBF networks. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems 8*, pages 239–245.

Kadirkamanathan, V. and Niranjan, M. (1993). A function estimation approach to sequential learning with neural networks. *Neural Computation*, 5(6):954–975.

Kalman, R. E. and Bucy, R. S. (1961). New results in linear filtering and prediction theory. *Transactions of the ASME (Jounal of Basic Engineering)*, 83D:95–108.

Kitagawa, G. (1987). Non-Gaussian state-space modeling of nonstationary time series. *Journal of the American Statistical Association*, 82(400):1032–1063.

Kitagawa, G. (1996). Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5:1–25.

Kong, A., Liu, J. S., and Wong, W. H. (1994). Sequential imputations and Bayesian missing data problems. *Journal of the American Statistical Association*, 89(425):278–288.

Kramer, S. C. and Sorenson, H. W. (1988). Recursive Bayesian estimation using piece-wise constant approximations. *Automatica*, 24(6):789–801.

Lawrence, S., Tsoi, A., and Giles, C. L. (1996). Local minima and generalization. In *IEEE International Conference on Neural Networks*, volume 1, pages 371–376.

Le Cun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., and Hubbard, W. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551.

Le Cun, Y., Denker, J. S., and Solla, S. A. (1990). Optimal brain damage. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems*, volume 2, pages 598–605, San Mateo, CA.

Lee, H. (1999). *Model Selection and Model Averaging for Neural Networks*. PhD thesis, Department of Statistics, Carnegie Mellon University, Pittsburgh, USA.

Leen, T. (1995). From data distributions to regularization in invariant learning. *Neural Computation*, 7(5):974–981.

Li, X. R. and Bar-Shalom, Y. (1994). A recursive multiple model approach to noise identification. *IEEE Transactions on Aerospace and Electronic Systems*, 30(3):671–684.

Liu, J. and West, M. (To appear in 2000). Combined parameter and state estimation in simulation-based filtering. In Doucet, A., de Freitas, J. F. G., and Gordon, N. J., editors, *Sequential Monte Carlo Methods in Practice*. Springer-Verlag.

Liu, J., Wong, W. H., and Kong, A. (1994). Covariance structure of the Gibbs sampler with applications to the comparisons of estimators and augmentation schemes. *Biometrika*, 81(1):27–40.

Liu, J. S. and Chen, R. (1995). Blind deconvolution via sequential imputations. *Journal of the American Statistical Association*, 90(430):567–576.

Liu, J. S. and Chen, R. (1998). Sequential Monte Carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93:1032–1044.

Ljung, L. (1987). *System Identification: Theory for the User*. Prentice-Hall.

Lowe, D. (1989). Adaptive radial basis function nonlinearities and the problem of generalisation. In *Proceedings of the IEE Conference on Artificial Neural Networks*, pages 171–175, London.

MacEachern, S. N., Clyde, M., and Liu, J. S. (1999). Sequential importance sampling for nonparametric Bayes models: the next generation. *Canadian Journal of Statistics*, 27:251–267.

Mackay, D. J. C. (1992a). Bayesian interpolation. *Neural Computation*, 4(3):415–447.

Mackay, D. J. C. (1992b). A practical Bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472.

Mackay, D. J. C. (1994). Bayesian nonlinear modelling for the prediction competition. In *ASHRAE Transactions*, volume 100, pages 1053–1062, Atlanta, Georgia.

Mackay, D. J. C. (1995). Probable networks and plausible predictions – a review of practical Bayesian methods for supervised neural networks. *Network-Computation in Neural Systems*, 6(3):469–505.

Mackay, D. J. C. (1996). Hyperparameters: Optimise or integrate out? In Heidbreder, G. R., editor, *Fundamental Theories of Physics*, pages 43–59. Kluwer Academic Publishers.

Marquardt, D. W. and Snee, R. D. (1975). Ridge regression in practice. *American Statistician*, 29(1):3–20.

Marrs, A. (To appear in 2000). In-situ ellipsometry. In Doucet, A., de Freitas, J. F. G., and Gordon, N. J., editors, *Sequential Monte Carlo Methods in Practice*. Springer-Verlag.

Marrs, A. D. (1998). An application of reversible-jump MCMC to multivariate spherical Gaussian mixtures. In Jordan, M. I., Kearns, M. J., and Solla, S. A., editors, *Advances in Neural Information Processing Systems*, volume 10, pages 577–583.

Mathews, V. J. (1991). Adaptive polynomial filters. *IEEE Signal Processing Magazine*, pages 10–26.

Mehra, R. K. (1970). On the identification of variances and adaptive Kalman filtering. *IEEE Transactions on Automatic Control*, AC-15(2):175–184.

Mehra, R. K. (1971). On-line identification of linear dynamic systems with applications to Kalman filtering. *IEEE Transactions on Automatic Control*, AC-16(1):12–21.

Mehra, R. K. (1972). Approaches to adaptive filtering. *IEEE Transactions on Automatic Control*, AC-17:693–698.

Melvin, D. G. (1996). A comparison of statistical and connectionist techniques for liver transplant monitoring. Technical Report CUED/F-INFENG/TR 282, Cambridge University Engineering Department.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1091.

Meyn, S. P. and Tweedie, R. L. (1993). *Markov Chains and Stochastic Stability*. Springer-Verlag, New York.

Moody, J. and Darken, C. (1988). Learning with localized receptive fields. In Hinton, G., Sejnowski, T., and Touretzsky, D., editors, *Proceedings of the 1988 Connectionist Models Summer School*, pages 133–143, Palo Alto, CA.

Müller, P. (1991). Monte Carlo integration in general dynamic models. *Contemporary Mathematics*, 115:145–163.

Müller, P. (1992). Posterior integration in dynamic models. *Computer Science and Statistics*, 24:318–324.

Müller, P. and Rios Insua, D. (1998). Issues in Bayesian analysis of neural network models. *Neural Computation*, 10:571–592.

Musso, C., Oudjane, N., and Le Gland, F. (To appear in 2000). Improving regularized particle filters. In Doucet, A., de Freitas, J. F. G., and Gordon, N. J., editors, *Sequential Monte Carlo Methods in Practice*. Springer-Verlag.

Myers, K. A. and Tapley, B. D. (1976). Adaptive sequential estimation of unknown noise statistics. *IEEE Transactions on Automatic Control*, AC-21:520–523.

Nabney, I. T. (1999). Efficient training of RBF networks for classification. Technical Report NCRG/99/002, Neural Computing Research Group, Aston University, Birmingham, UK.

Nabney, I. T., McLachlan, A., and Lowe, D. (1996). Practical methods of tracking non-stationary time series applied to real world data. In Rogers, S. K. and Ruck, D. W., editors, *AeroSense '96 : Applications and Science of Artificial Neural Networks II. SPIE Proceedings No. 2760*, pages 152–163.

Narendra, K. S. and Parthasarathy, K. (1990). Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1:4–27.

Neal, R. M. (1996). *Bayesian Learning for Neural Networks*. Lecture Notes in Statistics No. 118, Springer-Verlag, New York.

Neal, R. M. (1998). Annealed importance sampling. Technical Report No 9805, University of Toronto.

Nelder, J. A. and Wedderburn, R. W. M. (1972). Generalized linear models. *Journal of the Royal Statistical Society, Series A*, 135:370–384.

Niranjan, M. (1996). Sequential tracking in pricing financial options using model based and neural network approaches. In Mozer, M. C., Jordan, M. I., and Petsche, T., editors, *Advances in Neural Information Processing Systems*, volume 8, pages 960–966.

Niyogi, P. and Girosi, F. (1994). On the relationship between generalization error, hypothesis complexity, and sample complexity for radial basis functions. Technical Report AIM-1467, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, MA.

North, B. and Blake, A. (1998). Learning dynamical models using expectation-maximisation. In *International Conference on Computer Vision*, pages 384–389, Mumbai, India.

Nummelin, E. (1984). *Irreducible Markov Chains and Non-Negative Operators*. Cambridge University Press, Cambridge.

Pao, Y. H. (1989). *Adaptive Pattern Recognition and Neural Networks*. Addisson-Wesley.

Penny, W. D. and Roberts, S. J. (1998). Dynamic models for nonstationary signal segmentation. To appear in *Computers and Biomedical Research*.

Penny, W. D., Roberts, S. J., Curran, E., and Stokes, M. (1999). EEG-based communication: a pattern recognition approach. To appear in *IEEE Transactions on Rehabilitation Engineering*.

Perrone, M. P. (1995). Averaging/modular techniques for neural networks. In Arbib, M. A., editor, *The Handbook of Brain Theory and Neural Networks*, pages 126–129. MIT Press.

Perrone, M. P. and Cooper, L. N. (1993). When networks disagree: Ensemble methods for hybrid neural networks. In Mammone, R. J., editor, *Artificial Neural Networks for Speech and Vision*, pages 126–142, London.

Pitt, M. K. and Shephard, N. (1999). Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, 94(446):590–599.

Platt, J. (1991). A resource allocating network for function interpolation. *Neural Computation*, 3:213–225.

Poggio, T. and Girosi, F. (1990). Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497.

Pole, A. and West, M. (1990). Efficient Bayesian learning in non-linear dynamic models. *Journal of Forecasting*, 9:119–136.

Puskorius, G. V. and Feldkamp, L. A. (1991). Decoupled extended Kalman filter training of feedforward layered networks. In *International Joint Conference on Neural Networks*, pages 307–312, Seattle.

Puskorius, G. V. and Feldkamp, L. A. (1994). Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent networks. *IEEE Transactions on Neural Networks*, 5(2):279–297.

Qin, S., Su, H., and McAvoy, T. J. (1992). Comparison of four neural network learning methods for dynamic system identification. *IEEE Transactions on Neural Networks*, 3(1):122–130.

Rao, R. P. N. and Ballard, D. H. (1997). Dynamic model of visual recognition predicts neural response properties in the visual cortex. *Neural Computation*, 9(4):721–763.

Rauch, H. E., Tung, F., and Striebel, C. T. (1965). Maximum likelihood estimates of linear dynamic systems. *AIAA Journal*, 3(8):1445–1450.

Refenes, A., editor (1995). *Neural Networks in the Capital Markets*. John Wiley and Sons.

Revuz, D. (1975). *Markov Chains*. North Holland, Amsterdam.

Richardson, S. and Green, P. J. (1997). On Bayesian analysis of mixtures with an unknown number of components. *Journal of the Royal Statistical Society B*, 59(4):731–792.

Rios Insua, D. and Müller, P. (1998). Feedforward neural networks for nonparametric regression. Technical Report 98–02, Institute of Statistics and Decision Sciences, Duke University.

Ripley, B. D. (1987). *Stochastic Simulation*. Wiley, New York.

Ripley, B. D. (1996). *Pattern Recognition and Neural Networks*. Cambridge University Press.

Rissanen, J. (1987). Stochastic complexity. *Journal of the Royal Statistical Society*, 49:223–239.

Robert, C. P. and Casella, G. (1999). *Monte Carlo Statistical Methods*. Springer-Verlag, New York.

Roberts, G. and Tweedie, R. (1996). Geometric convergence and central limit theorems for multidimensional Hastings and Metropolis algorithms. *Biometrika*, 83:95–110.

Roberts, G. O. and Polson, N. G. (1994). On the geometric convergence of the Gibbs sampler. *Journal of the Royal Statistical Society B*, 56(2):377–384.

Roberts, S. J. and Penny, W. D. (1998). Bayesian neural networks for classification: How useful is the evidence framework? To appear in *Neural Networks*.

Roberts, S. J., Penny, W. D., and Pillot, D. (1996). Novelty, confidence and errors in connectionist systems. In *IEE Colloquium on Intelligent Sensors and Fault Detection*, number 261, pages 10/1–10/6.

Robinson, T. (1994). The application of recurrent nets to phone probability estimation. *IEEE Transactions on Neural Networks*, 5(2):298–305.

Rosenblatt, A. (1959). *Principles of Neurodynamics*. Spartan, New York.

Roweis, S. and Ghahramani, Z. (1999). A unifying review of linear Gaussian models. *Neural Computation*, 11(2):305–345.

Rubin, D. B. (1988). Using the SIR algorithm to simulate posterior distributions. In Bernardo, J. M., DeGroot, M. H., Lindley, D. V., and Smith, A. F. M., editors, *Bayesian Statistics 3*, pages 395–402, Cambridge, MA. Oxford University Press.

Ruck, D. W., Rogers, S. K., Kabrisky, M., Maybeck, P. S., and Oxley, M. E. (1992). Comparative analysis of backpropagation and the extended Kalman filter for training multilayer perceptrons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(6):686–690.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In Rumelhart, D. E. and McClelland, J. L., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, pages 318–362, Cambridge, MA.

Saarinen, S., Bramley, R., and Cybenko, G. (1993). Ill-conditioning in neural network training problems. *SIAM Journal of Scientific Computing*, 14(3):693–714.

Schottky, B. and Saad, D. (1999). Statistical mechanics of EKF learning in neural networks. *Journal of Physics A*, 32(9):1605–1621.

Schwarz, G. (1985). Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464.

Scott, M. J., Niranjan, M., and Prager, R. W. (1998). Parcel: Feature subset selection in variable cost domains. Technical Report CUED/F-INFENG/TR 323, Cambridge University Engineering Department.

Shah, S., Palmieri, F., and Datum, M. (1992). Optimal filtering algorithms for fast learning in feedforward neural networks. *Neural Networks*, 5(5):779–787.

Shumway, R. H. and Stoffer, D. S. (1982). An approach to time series smoothing and forecasting using the EM algorithm. *Journal of Time Series Analysis*, 3(4):253–264.

Shumway, R. H. and Stoffer, D. S. (1991). Dynamic linear models with switching. *Journal of the American Statistical Association*, 86(415):763–769.

Sibisi, S. (1989). Regularization and inverse problems. In Skilling, J., editor, *Maximum Entropy and Bayesian Methods*, pages 389–396. Kluwer Academic Publishers.

Silverman, B. W. (1986). *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, UK.

Singhal, S. and Wu, L. (1988). Training multilayer perceptrons with the extended Kalman algorithm. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems*, volume 1, pages 133–140, San Mateo, CA.

Sjöberg, J. (1995). *Non-Linear System Identification with Neural Networks*. PhD thesis, Department of Electrical Engineering, Linköping University, Sweeden.

Sjöberg, J., Zhang, Q., Ljung, L., Benveniste, A., Deylon, B., Glorenec, P., Hjalmarsson, H., and Juditzky, A. (1995). Non-linear black-box modelling in system identification: A unified overview. *Automatica*, 31(12):1691–1724.

Smith, A. F. M. and Gelfand, A. E. (1992). Bayesian statistics without tears: a sampling-resampling perspective. *American Statistician*, 46(2):84–88.

Smith, M. and Kohn, R. (1996). Nonparametric regression using Bayesian variable selection. *Journal of Econometrics*, 75(2):317–343.

Sorenson, H. W. and Alspach, D. L. (1971). Recursive Bayesian estimation using Gaussian sums. *Automatica*, 7:465–479.

Spyers-Ashby, J. M., Bain, P., and Roberts, S. J. (1998). A comparison of fast Fourier transform (FFT) and autoregressive (AR) spectral estimation techniques for the analysis of tremor data. *Journal of Neuroscience Methods*, 83(1):35–43.

Stavropoulos, P. (1998). *Computational Methods for the Bayesian Analysis of Dynamic Models*. PhD thesis, Department of Statistics, University of Glasgow, Scotland.

Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society*, B 36(1):111–147.

Stone, M. (1978). Cross-validation: A review. *Math. Operationsforsch. Statist. Ser. Statistics*, 9(1):127–139.

Sutton, R. S. (1992a). Adapting bias by gradient descent: An incremental version of delta-bar-delta. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 171–176. MIT Press, San Jose, California.

Sutton, R. S. (1992b). Gain adaptation beats least squares? In *Proceedings of the Seventh Yale Workshop on Adaptive Learning Systems*, pages 161–166.

Tanner, M. A. and Wong, W. H. (1987). The calculation of posterior distributions by data augmentation. *Journal of the American Statistical Association*, 82(398):528–550.

Tenney, R. R., Hebbert, R. S., and Sandell, N. S. (1977). A tracking filter for manoeuvring sources. *IEEE Transactions on Automatic Control*, 22(2):246–251.

Tierney, L. (1994). Markov chains for exploring posterior distributions. *The Annals of Statistics*, 22(4):1701–1762.

Troughton, P. T. and Godsill, S. J. (1998). A reversible jump sampler for autoregressive time series. In *International Conference on Acoustics, Speech and Signal Processing*, volume IV, pages 2257–2260.

Van Laarhoven, P. J. and Arts, E. H. L. (1987). *Simulated Annealing: Theory and Applications*. Reidel Publishers, Amsterdam.

Vapnik, V. N. (1982). *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, Berlin.

Wahba, G. and Wold, S. (1969). A completely automatic French curve: Fitting spline functions by cross-validation. *Communications on Statistics, Series A*, 4(1):1–17.

Watson, M. W. and Engle, R. F. (1983). Alternative algorithms for the estimation of dynamic factor, MIMIC and varying coefficient regression models. *Journal of Econometrics*, 23(3):385–400.

West, M. and Harrison, J. (1996). *Bayesian Forecasting and Dynamic Linear Models*. Springer-Verlag.

Wetherill, G. B. (1986). *Regression Analysis with Applications – Monographs on Statistics and Applied Probability*. Chapman and Hall.

White, H. (1989). Learning in artificial neural networks: A statistical perspective. *Neural Computation*, 1:425–464.

Williams, P. M. (1995). Bayesian regularization and pruning using a Laplace prior. *Neural Computation*, 7(1):117–143.

Wolpert, D. H. (1993). On the use of evidence in neural networks. In Hanson, S. J., Cowan, J. D., and Giles, C. L., editors, *Advances in Neural Information Processing Systems*, volume 5, pages 539–546, San Mateo, CA.

Wright, W. A. (1998). Neural network regression with input uncertainty. In Constantinides, T., Kung, S. Y., Niranjan, M., and Wilson, E., editors, *IEEE International Workshop on Neural Networks in Signal Processing*, pages 284–293, Cambridge, England.

Wu, L. and Moody, J. (1996). A smoothing regularizer for feedforward and recurrent neural networks. *Neural Computation*, 8(3):461–489.

Yamada, T. and Yabuta, T. (1993). Dynamic system identification using neural networks. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(1):204–211.

Yingwei, L., Sundararajan, N., and Saratchandran, P. (1997). A sequential learning scheme for function approximation using minimal radial basis function neural networks. *Neural Computation*, 9:461–478.

Zaritskii, V. S., Svetnik, V. B., and Shimelevich, L. I. (1975). Monte-Carlo techniques in problems of optimal information processing. *Automation and Remote Control*, 36(3):2015–2022.

Zellner, A. (1986). On assessing prior distributions and Bayesian regression analysis with g-prior distributions. In Goel, P. and Zellner, A., editors, *Bayesian Inference and Decision Techniques*, pages 233–243. Elsevier.