Chapter 4
A New Bayesian Framework for ANNs

4.1 Introduction and Motivation

The primary motivation for developing a Bayesian framework applicable to ANNs in the field of water resources modelling is that Bayesian methods allow the uncertainty in inferences made from data to be explicitly quantified (*Gelman et al.*, 2004). As discussed in Section 2.3.3, there are two levels of Bayesian inference that can be performed in ANN modelling: inference of the network weights under the assumption that the selected ANN architecture is correct (Bayesian training and prediction); and inference of the appropriate model architecture given the estimated weight distributions (Bayesian model selection). It is considered that the adoption of a Bayesian framework may lead to the wider acceptance of ANNs in the field of water resources modelling, as these two levels of inference work hand in hand to help address the three most significant issues facing ANN modellers in this field; namely generalisability, interpretability and uncertainty, as discussed in Section 2.2.5.

Firstly, the generalisability of an ANN may be improved through Bayesian training, as a range of plausible weight vectors (in the form of the posterior weight distribution) is considered when making predictions, rather than allowing a single, possibly incorrect, weight vector to completely dominate. This, together with the incorporation of 'regularisation' type priors, helps to overcome some of the difficulties associated with training an ANN due to the existence of local minima on the error surface and the potential of being overtrained. As it was seen in the previous chapter (and summarised in Appendix A), the "optimal" weight vector obtained during training, and hence the predictions generated, can be sensitive to the initial weights, the size of the network and the training method used. However, since the weights do not have any physical interpretation, it is generally impossible to determine which set of weights results in the best representation of the data-generating relationship. Simply finding the weight vector that provides the best fit to the training data does not necessarily result in a correct model of the system, because, due to the stochastic nature of hydrological systems, each different set of training data would most likely yield a different set of weights. Bayesian training overcomes the need to define an “optimal” weight vector by accounting for all sets of weights that provide a good fit to the training data; hence improving the likelihood of correctly describing the underlying system. Additionally, Bayesian model selection, which automatically penalises complexity, may help to improve the generalisability of an ANN by enabling the selection of the smallest network architecture suitable for modelling the data; thus, reducing the potential of overfitting.

Secondly, the interpretability of an ANN may be improved under the Bayesian modelling paradigm, as smaller, simpler models, chosen through Bayesian model selection, are more transparent, or interpretable, than large complex models with many weights. Furthermore, as it was shown in Section 3.4, input importance measures used to describe the relationship modelled by an ANN can be sensitive to the weights obtained during training. If a single incorrect weight vector is used in the assessment of a trained ANN, misleading information about the underlying system may be extracted from the optimised weights. Therefore, ANN interpretability can also be improved through Bayesian training, as the posterior weight distribution may be used to express the input importance measures as probability distributions which quantify the uncertainty in these estimates.

Finally, Bayesian training allows uncertainty in the weight estimates to be explicitly accounted for, which can significantly enhance the usability of the resulting predictions. Relying on a single optimal weight vector overestimates the confidence of the model predictions, as the uncertainty associated with the weight estimates is not incorporated. Subsequently, this can lead to inappropriate design and management decisions being made if the predictions are taken as being 100% reliable. On the other hand, by accounting for the full range of plausible weights, probabilistic predictions can be generated, enabling prediction intervals to be assigned that quantify the level of confidence in the model predictions and allow decisions to be made based on a known level of reliability. Bayesian model selection also helps to address the uncertainty issue by increasing the identifiability of weights. The weights of larger models, and therefore, the resulting predictions of large ANNs, have a higher degree of associated uncertainty than those of smaller models, since the information contained in the training data is more sparse in relation to the number of weights as the size of the network increases. Therefore, if the smallest model capable of fitting the training data can be selected, the information in the data can be used more efficiently to estimate the values of the weights; thus, minimising the uncertainty in the ANN weight estimates and the associated predictions.

In this chapter, a new Bayesian framework for ANNs, incorporating a “Bayesian training and prediction” component and a “Bayesian model selection” component, is proposed and the methods comprising the framework are described. A review of some of the statistical methods commonly used in Bayesian analysis is also presented to aid the understanding of the methods proposed in this chapter and explain how the proposed methods were arrived at. Based on this review, the methods adopted in the proposed framework were selected for their ability to produce accurate results, while maintaining simplicity and ease of implementation and coding. The simplicity of the framework is particularly important for its adoption by practitioners in the field of water resources modelling, as it is likely that the difficulties associated with coding the more complex Bayesian training methods proposed primarily in the statistical and computer sciences literature (see Section 4.2.2) have hindered their use in this field, with practitioners opting to disregard prediction uncertainty and relying on deterministic predictions, rather than apply such methods. The finer details of the proposed methods are determined through investigation and application of the methods on the three synthetic data sets used in the previous chapter (see Section 3.4.1). Fine-tuning of these details was necessary to ensure the successful implementation of the overall Bayesian approach.

4.2 BAYESIAN TRAINING AND PREDICTION

As discussed in Section 2.3.4, training an ANN and making predictions using Bayesian methodology involves solving equations (2.14), (2.16) and (2.17). Possibly the most straightforward way of doing this involves the use of Gaussian approximations, where the posterior weight distribution is approximated by a Gaussian distribution centred on the mode found (i.e. the optimal weight vector found using deterministic training methods) with a variance that can be calculated from the Hessian. However, the hyperparameters and the multimodal nature of ANN weights present a problem for such an approximation, as the resulting posterior is by no means Gaussian (*Neal*, 1993). Therefore, in this research, only more sophisticated methods based on Markov chain Monte Carlo methods will be considered.

4.2.1 Markov Chain Monte Carlo Methods

Markov chain Monte Carlo (MCMC) methods provide a means for generating a sequence of samples $\theta_1, \theta_2, \ldots, \theta_n$ from virtually any continuous distribution, or *target density*, $p(\theta|\mathbf{y})$. This target density need only be known to within a multiplicative constant, as follows:

$$p(\theta|\mathbf{y}) = p^*(\theta|\mathbf{y})/Z \tag{4.1}$$

where $p^*(\theta|\mathbf{y})$ is the unnormalised density, which must be calculable for a given vector θ, and Z is the (possibly unknown) normalising constant (*Thyer*, 2001). Such methods are generally used when it is impossible (or computationally infeasible) to sample from $p(\theta|\mathbf{y})$ directly and, although there are other methods available for generating samples from $p(\theta|\mathbf{y})$ (e.g. uniform sampling, rejection sampling), when applied carefully, MCMC methods are the easiest way to obtain reliable results (*Gelman et al.*, 2004). When MCMC methods are applied to simulate a Bayesian posterior distribution, $p^*(\theta|\mathbf{y}) = p(\mathbf{y}|\theta)p(\theta)$ and Z is the normalising constant $p(\mathbf{y})$. The fact that MCMC methods do not require knowledge of Z is a major advantage, since the calculation of $p(\mathbf{y})$ may be complicated.

To generate samples from $p(\theta|\mathbf{y})$ using MCMC methods, the process is started at an arbitrary point θ_0 and samples are drawn sequentially, using a transition distribution $T(\theta_t|\theta_{t-1})$ to represent the probability of moving from point θ_{t-1} to point θ_t. Since the transition distribution depends only on the previous state θ_{t-1}, the generated sequence forms a *Markov chain*. After a large number of iterations, the samples generated from the simulation converge to a stationary distribution corresponding to the target density $p(\theta|\mathbf{y})$. However, in order for this to be true, the following two conditions must be satisfied:

1. The generated sequence must be *ergodic*, which means that the Markov chain must converge to a stationary distribution. This will be the case if the Markov chain is irreducible and aperiodic, which means that there must be a positive probability of eventually reaching any state from any other state and that the number of moves required to move from one state to another is not required to be a multiple of some integer (*Chib and Greenberg*, 1995).

2. The target distribution $p(\theta|\mathbf{y})$ must be an *invariant distribution* of the Markov chain constructed under the transition distribution $T(\cdot)$. It has been shown by *Chib and Greenberg* (1995) that this will be the case if $T(\cdot)$ satisfies the *detailed balance*, or *reversibility*, condition:

$$T(\theta_a|\theta_b)p(\theta_b|\mathbf{y}) = T(\theta_b|\theta_a)p(\theta_a|\mathbf{y}) \tag{4.2}$$

A variety of Markov chains may be constructed to meet these requirements. The three simplest and most commonly used MCMC algorithms, namely the Metropolis-Hastings algorithm, the Metropolis algorithm and the Gibbs sampler, are described in this section, as well as a variation of the standard Metropolis algorithm known as the adaptive Metropolis (AM) algorithm.

4.2.1.1 The Metropolis-Hastings Algorithm

The Metropolis-Hastings algorithm (*Hastings*, 1970) is the most general of the MCMC algorithms described in this section. The transition from θ_{t-1} to θ_t is determined, firstly, by employing a *proposal density* Q, conditional on θ_{t-1}, to generate a candidate state θ^*, where Q may be any fixed density from which samples can be drawn (*MacKay*, 2003). However, the use of $Q(\theta^*|\theta_{t-1})$ alone will most likely not satisfy the reversibility condition necessary for the Markov chain to converge to the target distribution. Therefore, an acceptance probability α is introduced to determine whether or not to accept the candidate state, which corrects this condition and enables the Markov chain to continually adapt to the target distribution (*Chib and Greenberg*, 1995). Thus, transitions from θ_{t-1} to θ_t are made according to:

$$T(\theta_t|\theta_{t-1}) = Q(\theta^*|\theta_{t-1})\alpha(\theta^*|\theta_{t-1}) \tag{4.3}$$

It is shown in *Chib and Greenberg* (1995) that, in order to satisfy reversibility, the acceptance probability must be set to:

$$\alpha(\theta^*|\theta_{t-1}) = \min\left[\frac{p^*(\theta^*|\mathbf{y})Q(\theta^*|\theta_{t-1})}{p^*(\theta_{t-1}|\mathbf{y})Q(\theta_{t-1}|\theta^*)}, 1\right] \tag{4.4}$$

Thus, the algorithm proceeds as follows (as adapted from *Gelman et al.* (2004)):

1. Initialise the algorithm with an arbitrary starting point θ_0 for which $p^*(\theta_0|\mathbf{y}) > 0$.

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

   (a) Sample a candidate θ^* from the proposal distribution $Q(\theta^*|\theta_{t-1})$.

   (b) Evaluate the ratios $p^*(\theta^*|\mathbf{y})/p^*(\theta_{t-1}|\mathbf{y})$ and $Q(\theta^*|\theta_{t-1})/Q(\theta_{t-1}|\mathbf{y})$.

   (c) Generate u from $U(0,1)$ and if:

   $$u \leq \alpha(\theta^*|\theta_{t-1}) \tag{4.5}$$

   set $\theta_t = \theta^*$ (i.e. accept θ^*), otherwise set $\theta_t = \theta_{t-1}$ (i.e. reject θ^*).

3. Return the samples $\{\theta_1, \theta_2, \ldots, \theta_n\}$.

The samples are only regarded as draws from the target distribution after the algorithm has converged to a stationary distribution and once the effects of the initial starting value θ_0 are small enough to be ignored.

The Metropolis-Hastings algorithm is specified by its proposal density $Q(\theta^*|\theta_{t-1})$, which, as mentioned, can be any fixed density from which samples can be drawn. However, the rate of convergence to the stationary distribution $p(\theta|\mathbf{y})$ can crucially depend on the form of the proposal distribution adopted. Some general families from which Q may be selected are discussed in *Chib and Greenberg* (1995). Typically, Q is chosen such that it can be easily sampled and evaluated and so that only few tuning parameters (e.g. location and scale parameters) require specification.

4.2.1.2 The Metropolis Algorithm

The Metropolis algorithm (*Metropolis et al.*, 1953) is a special case of the Metropolis-Hastings algorithm, or rather, the latter is a generalisation of the former, since the Metropolis algorithm was developed first. The Metropolis algorithm requires that the proposal distribution $Q(\theta^*|\theta_{t-1})$ is symmetric, satisfying the condition $Q(\theta^*|\theta_{t-1}) = Q(\theta_{t-1}|\theta^*)$. Therefore, the acceptance probability reduces to:

$$\alpha(\theta^*|\theta_{t-1}) = \min\left[\frac{p^*(\theta^*|\mathbf{y})}{p^*(\theta_{t-1}|\mathbf{y})}, 1\right] \tag{4.6}$$

Thus, if a jump from θ_{t-1} to θ^* goes "uphill", it is always accepted, whereas if the jump goes "downhill", it is only accepted with probability $p^*(\theta^*|\mathbf{y})/p^*(\theta_{t-1}|\mathbf{y})$. This is illustrated in Figure 4.1, where a jump from θ_0 to θ_1 would occur with certainty; however, a jump from θ_1 to θ_2 would occur with probability $p^*(\theta_2|\mathbf{y})/p^*(\theta_1|\mathbf{y})$.

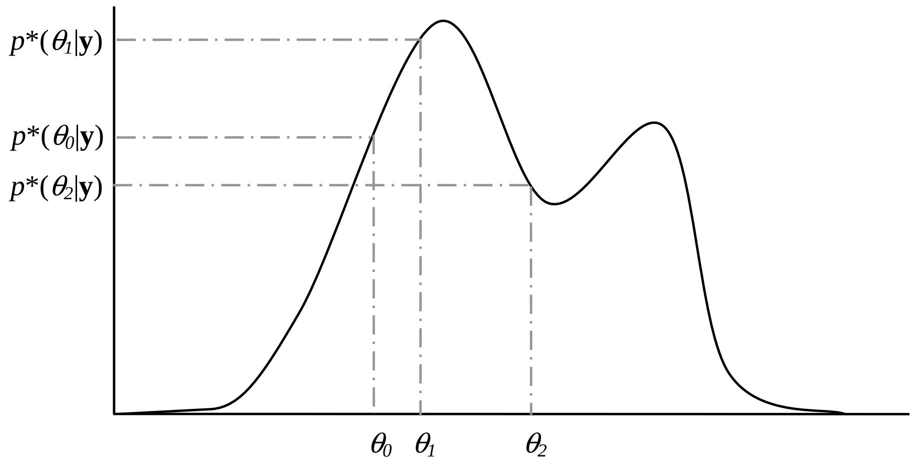
$p^*(\theta_1|\mathbf{y})$
$p^*(\theta_0|\mathbf{y})$
$p^*(\theta_2|\mathbf{y})$
θ_0
θ_1
θ_2

Figure 4.1 The probability of a jump using the Metropolis Algorithm

A special case of the Metropolis algorithm can be defined by the relation $Q(\theta^*|\theta_{t-1}) = Q(\theta_{t-1} - \theta^*)$. Here, the candidates are drawn according to:

$$\theta^* = \theta_{t-1} + \upsilon \tag{4.7}$$

where υ is called the increment random variable and follows the distribution Q (*Chib and Greenberg*, 1995). As the candidates are equal to the previous value plus noise, the sequence of samples generated forms a *random walk* chain. The proposal distribution employed by this very popular algorithm has a form that is indexed by a scale parameter (*Roberts*, 1996). A simple Gaussian distribution centred on the current state θ_{t-1} with fixed covariance Σ is a common choice for $Q(\theta^*|\theta_{t-1})$ since the Gaussian distribution is symmetric and is one of few high-dimensional densities from which it is easy to draw samples (*MacKay*, 2003). The scale parameter chosen for Q has important implications on the convergence properties and efficiency of the algorithm, particularly in the case of complex models with correlated parameters. Shown in Figure 4.2 is an example of a bivariate target density together with two example proposal distributions, Q_1 and Q_2. Using the larger proposal distribution Q_1, denoted by the dashed line, it can be seen that a jump made from the state θ_{t-1} in almost any direction will result in a decrease in $p^*(\theta|\mathbf{y})$, and as such, a large proportion of jumps will be rejected leading to slow convergence of the chain. On the other hand, if the smaller proposal distribution Q_2 is selected, the acceptance rate will increase; however, the algorithm will take longer to traverse $p^*(\theta|\mathbf{y})$, which will again lead to slow convergence (*Thyer et al.*, 2002). This problem is exacerbated in problems where parameters in the target distribution are highly correlated, since the scale

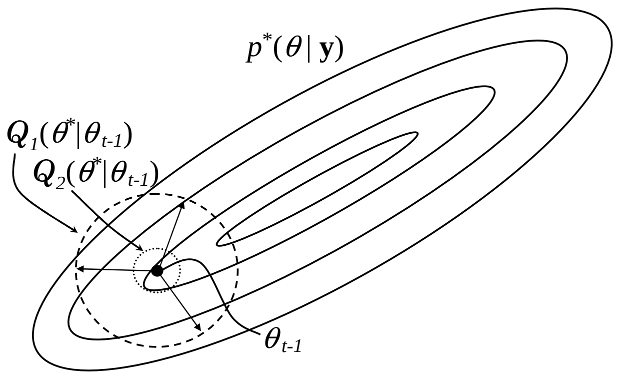

Figure 4.2 The effect of the scale of the proposal distribution used by the Metropolis algorithm. Q_1 and Q_2 represent two proposal distributions having larger and smaller scale parameters, respectively. Ellipses denote contours of $p^*(\theta|\mathbf{y})$.

of the proposal must be made very small to ensure sufficient acceptance of states. Many iterations will therefore be required to move between effectively independent states.

4.2.1.3 The Gibbs Sampler

The Gibbs sampler (*Geman and Geman*, 1984) is the simplest form of MCMC algorithm and is also a special case of the Metropolis-Hastings algorithm (*Gelman et al.*, 2004). The Gibbs sampler is defined in terms of subvectors of θ and the transition distributions are the conditional distributions of the joint distribution $p^*(\theta|\mathbf{y})$. If θ is divided into l subvectors $\theta = \left\{\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(l)}\right\}$, the transition distribution for the jth subvector is:

$$T(\theta_t^{(j)}|\theta_{t-1}^{(j)}) = p^*(\theta_t^{(j)}|\theta_{t-1}^{(-j)}, \mathbf{y}) \tag{4.8}$$

where

$$\theta_{t-1}^{(-j)} = \left\{\theta_t^{(1)}, \ldots, \theta_t^{(j-1)}, \theta_{t-1}^{(j+1)}, \ldots, \theta_{t-1}^{(l)}\right\} \tag{4.9}$$

The acceptance probability $\alpha(\theta_t|\theta_{t-1})$ is always equal to one due to the *product of kernels principle*, which states that if the transition distributions are conditional distributions of $p^*(\theta|\mathbf{y})$, then the product of the transition distributions has $p^*(\theta|\mathbf{y})$ as its invariant distribution (*Chib and Greenberg*, 1995). A single iteration of the Gibbs sampler involves l steps, where each subvector is drawn in turn conditional on the other subvectors, as follows:

$$\begin{aligned}
\theta_t^{(1)} &\sim p^*(\theta_t^{(1)}|\theta_{t-1}^{(2)}, \ldots, \theta_{t-1}^{(l)}, \mathbf{y}) \\
\theta_t^{(2)} &\sim p^*(\theta_t^{(2)}|\theta_t^{(1)}, \theta_{t-1}^{(3)}, \ldots, \theta_{t-1}^{(l)}, \mathbf{y}) \\
\theta_t^{(3)} &\sim p^*(\theta_t^{(3)}|\theta_t^{(1)}, \theta_t^{(2)}, \theta_{t-1}^{(4)}, \ldots, \theta_{t-1}^{(l)}, \mathbf{y}), \text{ etc.}
\end{aligned} \tag{4.10}$$

A requirement of the Gibbs sampler is that it is possible to generate samples directly from the conditional distributions. For many problems, this will not be possible since the conditional distributions may be complex with non-standard forms. However, for problems where the conditional distributions have standard forms for which sampling methods have been developed, the Gibbs sampler is attractive due to its simplicity in both implementation and coding and is often the first choice of MCMC method. Additionally, unlike most MCMC methods, the Gibbs sampler has no adjustable parameters, which is an advantage in that it minimises the amount of tuning required to obtain good performance of the algorithm. A disadvantage of the Gibbs sampler, however, is that it can be quite inefficient if the subvectors of θ are correlated, since many iterations are needed to move from one state to another that is largely independent of the first. Moreover, like the random walk Metropolis algorithm, the Gibbs sampler explores the search space by a random walk, which further slows the rate at which independent states are visited (*Neal*, 1993).

For problems where some conditional distributions in a model can be sampled from directly and some cannot, a combination of the Gibbs sampler and Metropolis algorithm may be used, with the Gibbs sampler used where possible and the Metropolis algorithm used otherwise.

4.2.1.4 The Adaptive Metropolis (AM) algorithm

The adaptive Metropolis (AM) algorithm, developed by *Haario et al.* (2001), is a variation of the random walk Metropolis algorithm that was designed to overcome the problems associated with selecting an appropriate scale parameter for the proposal distribution. This algorithm employs an adaptation strategy that forces the proposal distribution to approach an appropriately scaled Gaussian approximation of the target density, which increases the efficiency of the algorithm and improves the rate of convergence to the stationary distribution.

The proposal distribution used in the AM algorithm is a multivariate Gaussian distribution centred on the current state θ_{t-1} with a covariance calculated based on all of the previously sampled states $\Sigma_t = \Sigma_t(\theta_0, \ldots, \theta_{t-1})$. This ensures that information gained about the target distribution throughout the simulation is used to adapt the proposal towards this distribution, allowing the algorithm to overcome the random walk properties of the standard Metropolis algorithm. Apart from this, the definition of the algorithm is the same as the general Metropolis process described in Section 4.2.1.2. However, since $Q(\theta^*|\theta_0, \ldots, \theta_{t-1})$ depends on all of the previous states, it is no longer symmetric, nor is the process generated Markovian. Nevertheless, it was proven by *Haario et al.* (2001) that the AM process still has the correct ergodicity properties, which enable it to converge to a stationary distribution, under the assumptions that the target distribution has a compact support and is bounded from above. These assumptions correspond reasonably well to practical situations.

To initialise the algorithm, an arbitrary, positive definite covariance matrix, Σ_0, is selected. For an initial period $t_0 > 0$, the covariance of the proposal is fixed at the initial covariance, after which time the adaptation strategy begins, as follows:

$$\Sigma_t = \begin{cases} \Sigma_0 & t \le t_0 \\ c^2 \mathrm{cov}\left(\theta_0, \theta_1, \ldots, \theta_{t-1}\right) + c^2 e \mathbf{I}_d & t > t_0 \end{cases} \tag{4.11}$$

where c is an adaptive scaling parameter used to maintain an appropriate acceptance rate, e is a small constant used to ensure that Σ_t will not become singular, and $\mathbf{I}_d$ is the d-

dimensional identity matrix, with d being the dimension of θ. It is worth mentioning that the actual value of e is not important for the performance of the algorithm. For $t > t_0$, calculation of Σ_t satisfies the following recursion formula:

$$\Sigma_{t+1} = \frac{t-1}{t}\Sigma_t + \frac{c^2}{t}\left[t\,\bar{\theta}_{t-1}\bar{\theta}_{t-1}^T - (t+1)\,\bar{\theta}_t\bar{\theta}_t^T + \theta_t\theta_t^T + e\mathbf{I}_d\right] \tag{4.12}$$

where $\bar{\theta}_t = 1/(t+1)\sum_{i=0}^{t}\theta_i$. Therefore, the covariance may be updated at each iteration with little additional computational cost. The choice of the initial fixed period t_0 should reflect the confidence in the initial covariance Σ_0. The longer this period, the more slowly the adaptation is felt and the greater the effect of the initial covariance on the simulated draws. Therefore, if the initial fixed period is short, even a poor initial choice of Σ_0 should only have a minor impact on the overall convergence of the algorithm.

The AM algorithm has been found to have a number of advantages over other variants of the Metropolis-Hastings algorithm in terms of efficiency, both computational and statistical, and ease of use (*Marshall et al.*, 2004). Unlike the conventional Metropolis and Metropolis-Hastings algorithms, great care is not required in selecting an appropriate proposal distribution and, in fact, the only requirement is that the initial covariance Σ_0 be strictly positive definite and scaled such that the algorithm moves at least a little during the initial period t_0. Furthermore, by using the covariance matrix of the weights to determine jumps to candidate states, interactions between the weights are automatically accounted for, allowing the chain to move more efficiently through weight space. However, although the AM algorithm has been successfully tested on problems with up to 200 parameters (*Haario et al.*, 2001), it has been noted by *Haario et al.* (2005), who proposed a variation of the algorithm for high dimensional problems, that the AM algorithm becomes less robust when applied to problems with greater than 50 parameters. As the variation of the AM algorithm proposed by *Haario et al.* (2005) is based on updating a single parameter at a time, this algorithm will not be considered in this research, since the target function would need to be evaluated separately for each parameter, increasing complexity and computational cost.

4.2.2 MCMC Methods Previously Applied to ANNs

When the MCMC methods discussed in Section 4.2.1 are applied to ANN training, the generic parameter θ is substituted with the network weights $\mathbf{w}$ and/or the hyperparameters $\{\sigma_{\mathbf{w}}^2, \sigma_{\mathbf{y}}^2\}$. While the Gibbs sampler may be a convenient and simple way to sample the hyperparameters, there appears to be no reasonable way of applying Gibbs sampling to $\mathbf{w}$, since the use of this method depends on the ability to easily sample from the full conditional distribution of one parameter, given the values of all other parameters and the data, which is extremely messy and multimodal for ANN weights (*Neal*, 1996a). The Metropolis algorithm, on the other hand, may be used to sample $\mathbf{w}$; however, owing to its random walk nature, many authors warn against the use of the simple Metropolis algorithm for ANN training, as the generally high dimension of $\mathbf{w}$ and the interactions between the individual weights make this algorithm prohibitively slow (*MacKay*, 1995b; *Neal*, 1996a; *Rasmussen*, 1996). A number of MCMC methods have been proposed for training ANNs, of which the approach developed by *Neal* (1992, 1993, 1996a) is the most widely advocated. In this approach, the hyperparameters are sampled using the Gibbs sampler and the weights are sampled using the hybrid Monte Carlo (HMC) algorithm of *Duane et al.* (1987), as mentioned in Section 2.3.4. In this algorithm, the stationary distribution is defined by $p(\mathbf{w}|\mathbf{y}) \propto \exp\{-E(\mathbf{w})\}$, where $E(\mathbf{w})$ is referred to as the potential energy function of the "position vector" $\mathbf{w}$ and is defined by (2.13). Each weight is assigned a momentum variable, which collectively form the vector $\mathbf{v}$. Both $\mathbf{w}$ and $\mathbf{v}$ are updated together in a Metropolis step using *Hamiltonian dynamics*, where a jump to $\mathbf{w}^*$ is determined largely by the momentum $\mathbf{v}$, which is updated according to the gradient of $E(\mathbf{w})$. Therefore, successive jumps tend to be in the same direction, which overcomes the random walk nature of simple Metropolis. The acceptance probability is employed to stop movement when regions of low probability are reached, at which point the momentum is altered until jumping can continue. The HMC method of Neal has been adopted in a number of studies, including those of *Rasmussen* (1996), *Vivarelli and Williams* (1997, 2001), *Husmeier et al.* (1999), *Vehtari et al.* (2000) and *Lampinen and Vehtari* (2001).

Müller and Rios Insua (1998) and *Rios Insua and Müller* (1998) proposed a MCMC method for training ANNs based on the reversible jump method of *Green* (1995). Using this method, the model architecture is also treated as a variable parameter, which helps to overcome inefficiencies due to multiple modes. At most, J^* hidden nodes are considered and indicators d_j are associated with each node $j = 1, \ldots, J^*$ to determine whether the nodes are included or not. If a node j is included, $d_j = 1$; otherwise, $d_j = 0$. A prior distribution is included on the indicators, which enables any number of hidden nodes $J \leq J^*$ and favours parsimony. Additionally, the algorithm used to update the weights treats the input-hidden weights γ and the hidden-output weights β separately (i.e. $\mathbf{w} = \{\gamma, \beta\}$). This enables linearization of the model in terms of β and the matrix of hidden node outputs $\mathbf{Z}$, as $y = \beta\mathbf{Z} + \beta_0$, which in turn allows marginalisation over β to compute a marginal likelihood $p(\mathbf{y}|\gamma)$. It also enables β to be sampled using the Gibbs sampler and allows γ to be updated using the Metropolis algorithm, based only on the values of the hyperparameters. Combined with a variable model architecture, it is claimed by *Müller and Rios Insua* (1998), that this algorithm allows for “fast and efficient mixing over various local modes in the posterior distribution”.

de Freitas et al. (2000) proposed yet another sequential MCMC method to train ANNs. This method is based on extended Kalman filter (EKF) weight updates, which incorporate gradient information, and importance sampling, which is a precursor to the Metropolis algorithm (*Gelman et al.*, 2004). For each iteration of this algorithm, a sample of n candidate weight vectors $\mathbf{w}^*$ is selected using an EKF step. These candidates are then assigned corresponding importance ratios, calculated using the importance sampling method, and each candidate is accepted based on the normalised importance ratios of the sample. If a candidate $\mathbf{w}_i^*$ is not accepted, $\mathbf{w}_{t,i}$ is selected by resampling a weight vector from the set $\mathbf{w}_1^*, \ldots, \mathbf{w}_n^*$ based on its normalised importance ratio, such that vectors with larger importance ratios end up with a greater number of copies in the sample $\mathbf{w}_{t,1}, \ldots, \mathbf{w}_{t,n}$. It is claimed that this algorithm is an improvement over simple MCMC methods in terms of computational time and accuracy (*de Freitas et al.*, 2000).

Lee (2003) advocates the use of a simpler MCMC algorithm to those discussed above and proposed a method similar, yet less complex, to that proposed by *Müller and Rios Insua* (1998). While this algorithm is also based on the linearization of the model in the form of $y = \beta\mathbf{Z} + \beta_0$, which enables β to be sampled using the Gibbs sampler, it assumes a non-variable architecture. A noninformative prior distribution in the form of $p(\gamma, \beta, \sigma_{\mathbf{y}}^2) \propto 1/\sigma_{\mathbf{y}}^2$ on restricted weight space is also assumed, which significantly simplifies the Metropolis step used to sample γ. It was shown that this method was comparable to the more complex MCMC algorithms applied to ANNs in terms of accuracy; however, a noted disadvantage of the method is that weight regularisation is not incorporated and overfitting may occur.

Recently, *Liang* (2005) proposed an evolutionary Monte Carlo (EMC) algorithm for simultaneously training and selecting the complexity of an ANN by sampling from the joint posterior of the weights and the network structure. Each connection in the network is given an indicator function, which determines how effective the connection is. Collectively, the indicator functions then determine the model structure; however, this requires that a maximum number of hidden nodes be specified. The EMC algorithm involves using genetic mutation and crossover operators to generate a new population of samples, which are then accepted or rejected based on the Metropolis-Hastings rule. A disadvantage of this algorithm is that it involves a number of user-defined parameters that may affect the performance of the algorithm.

4.2.3 Proposed Bayesian Training Approach

In this section, details of the Bayesian training approach proposed in this research are presented, together with issues that require further investigation. The methods and details presented were selected based on a review of the available MCMC methods and the approaches previously applied for ANN training, while keeping in mind that a major objective of the proposed approach is simplicity.

4.2.3.1 Proposed Likelihood Function

Ideally, for a model to correctly describe a system, the model residuals should be independently and identically distributed (i.i.d.) with zero mean and constant variance (i.e. white noise) and should be independent of the model inputs. The likelihood function proposed in this research is that given by (2.11), which makes the assumption that the model residuals ϵ correspond to additive, uncorrelated Gaussian noise in the observed response data. This is a common choice for $L(\mathbf{w})$, since its easy evaluation and few parameters (the distribution is controlled by a single variance hyperparameter $\sigma_{\mathbf{y}}^2$) results in a simple analysis. However, apart from simplicity, there are other good reasons to assume a Gaussian error model. Firstly, when the errors in the data are the sum of a large number of small independent component errors, the *central limit theorem* states that the distribution of the errors should be approximately Gaussian (*Bishop*, 1995). Therefore, deviations from normality tend to indicate systematic errors which have been unaccounted for. Secondly, it is often the case that data are roughly normally distributed, either in their original form or in some simple transformation such as the logarithm (*Box and Tiao*, 1973); hence, if the functional form of the model is appropriate, the residuals should also be roughly normally distributed.

In many practical problems, the Gaussian residual model with fixed variance may be inappropriate. As well as being subject to random disturbances in the response data, the modelled outputs are affected by measurement and sampling errors in the input variables and the lack of a correctly specified model. This may result in non-Gaussian residuals and dependence between the residual variance and the inputs. Therefore, a more robust choice for $L(\mathbf{w})$ might be Student's t-distribution with unknown degrees of freedom, as suggested by *Lampinen and Vehtari* (2001), which has longer tails than the Gaussian density, allowing it to accommodate occasional unusual observations. However, for simplicity and consistency with standard Bayesian regression methods and the deterministic error model used in this research (minimising the SSE is equivalent to maximising the likelihood under the assumption of Gaussian residuals), the Gaussian likelihood function will be adopted. Transformations to approximate normality will be applied to the data if necessary to help meet the assumptions of this error model (i.e. to decrease the effects of outliers in the data - see Section 3.2.3.2).

4.2.3.2 Proposed Prior Distributions

In this research, three different forms of $p(\mathbf{w})$ will be investigated to determine the most suitable prior distribution for $\mathbf{w}$ to use in the proposed Bayesian training approach. These priors were selected by taking into account their simplicity, their effect on generalisability, their provision of insight into the model, and how realistically they represent actual prior knowledge of the weights. The different priors considered are described as follows:

Noninformative uniform prior: As ANN weights have no physical interpretation, little can be known about the values of these parameters before observing the data. In such cases, it is often recommended to choose a *noninformative* prior distribution that plays a minimal role in the posterior distribution (*Lee*, 1989). Such a prior is said to 'let the data speak for themselves' as it does not restrict the estimation of the posterior (*Gelman et al.*, 2004). The use of a wide uniform prior distribution $U(-a, a)$ specifies an equal likelihood of positive and negative values, but an otherwise lack of prior knowledge about the values of the weights. The value of a is fixed and is selected to incorporate a wide range of weight values for which the likelihood may be appreciable. Therefore, this prior involves no variable hyperparameters, which results in the simplest MCMC training procedure. However, a disadvantage of this prior is that, since the weights are effectively unrestricted, no form of regularisation is incorporated into the training approach and overfitting may occur. Nevertheless, as simplicity was of major importance in developing the Bayesian training approach proposed in this research, this prior was considered regardless of the fact that it does not incorporate weight regularisation.

Hierarchical prior: As an alternative to the use of a noninformative prior to define the lack of prior knowledge about the weights, a hierarchical prior distribution, governed by unknown hyperparameters, may be used. The hyperparameters are then given rather noninformative hyperprior distributions which allows their values, and hence the prior distribution, to be determined from the data, as discussed in Section 2.3.4.1. To reiterate, a suitable form of hierarchical prior for $\mathbf{w}$ is given by (2.15), which is the product of G independent normal distributions, defined by zero means and different variance hyperparameters $\sigma^2_{\mathbf{w}_1}, \ldots, \sigma^2_{\mathbf{w}_G}$, where G is the number of weight groups considered. This type of prior is consistent with regularisation theory, allowing unnecessary connections to have smaller weights. It is also consistent with the fact that the values of the weights have an equal likelihood of being positive or negative and have a finite variance. Four weight groups were considered, corresponding to the input-hidden layer weights, the hidden layer biases, the hidden-output layer weights and the output biases, as this is generally the minimum requirement for suitable regularisation (*Sarle*, 2002).

ARD prior: Using the automatic relevance determination (ARD) method of *MacKay* (1995a) and *Neal* (1996a), the hierarchical prior discussed above is employed; however, the input-hidden layer weights are further divided into groups associated with each input variable, such that the weights on connections exiting an input have a prior distribution controlled by a hyperparameter associated with that input. Thus, the number of weight groups and, hence, the number of hyperparameters considered is $K + 3$, where K is equal to the number of inputs. Since the hyperparameters determine how closely the weights are distributed about the zero mean, the use of this form of prior enables the relevance of each input to be determined automatically from the data as the values of the hyperparameters are adapted (*Neal*, 1996a). In terms of realistically describing prior knowledge of the weights, the ARD prior can be considered correct, given the knowledge that some inputs may be less relevant than others. However, of the prior distributions investigated in this research, this prior results in the most complex MCMC training method, as it involves the greatest number of hyperparameters, for which hyperprior distributions must also be specified.

Neal (1996a) applied the following three levels of ARD prior to a number of test cases to demonstrate that ARD was indeed able to suppress the values of weights associated with irrelevant inputs:

- No ARD - corresponding to the hierarchical prior described above.
- 1-level ARD - all hyperparameters were independently given rather vague hyperpriors.
- 2-level ARD - an additional level of hierarchy was used, where a *top-level* hyperparameter, common to all inputs, was used to control the lower level input-hidden weight hyperparameters. The top-level hyperparameter was given a very vague hyperprior, while lower level hyperpriors were less vague.

It was also shown that, in some cases, the use of ARD could lead to an improvement in predictive performance. *Husmeier et al.* (1999) extended this investigation by applying the three levels of ARD priors to five different benchmark data sets in order to compare the resulting predictive performances of the models developed and evaluate the efficiency of the ARD scheme. However, the study was inconclusive, since each level of ARD prior resulted in at least one best and one worst solution for the different benchmark data sets, and no prior ever obviously resulted in better performance than the others. However, it was confirmed that the use of either 1-level or 2-level ARD resulted in the suppression of irrelevant input-hidden weights.

Since the 2-level ARD prior results in an increase in complexity, yet, provided no obvious advantages in the comparisons, only 1-level ARD will be considered in this research.

4.2.3.3 Proposed Hyperprior Distributions

For the likelihood and prior distributions considered in this research, the hyperparameters (e.g. $\sigma^2_{\mathbf{y}}$ and $\sigma^2_{\mathbf{w}} = \{\sigma^2_{\mathbf{w}_1}, \ldots, \sigma^2_{\mathbf{w}_G}\}$) are all Gaussian variances. A suitable prior for the Gaussian variance is the scaled inverse chi-squared distribution $\chi^{-2}(\nu^0, S^0)$, where ν^0 and S^0 are degrees of freedom and scale parameters, respectively, chosen to express the level of prior knowledge (*Lee*, 1989). This is a natural *conjugate prior* to the Gaussian likelihood, which means that, for this likelihood, the posterior distribution will have the same parametric form as the prior. Recall that the full conditional distributions of $\sigma^2_{\mathbf{w}}$ and $\sigma^2_{\mathbf{y}}$ are given by (2.19) and (2.20), respectively. Generating samples from these distributions is relatively easy if conjugate priors are chosen, since it is then known that $p(\sigma^2_{\mathbf{y}}|\mathbf{w}, \mathbf{y})$ and $p(\sigma^2_{\mathbf{w}_g}|\mathbf{w}_g)$ are also scaled inverse chi-squared distributions, given by:

$$p(\sigma^2_{\mathbf{y}}|\mathbf{w}, \mathbf{y}) \sim \chi^{-2}\left(\nu^*_{\mathbf{y}} = \nu^0_{\mathbf{y}} + N, S^*_{\mathbf{y}} = \frac{S^0_{\mathbf{y}}\nu^0_{\mathbf{y}} + S_{\mathbf{y}}N}{\nu^0_{\mathbf{y}} + N}\right) \tag{4.13}$$

$$p(\sigma^2_{\mathbf{w}_g}|\mathbf{w}_g) \sim \chi^{-2}\left(\nu^*_{\mathbf{w}_g} = \nu^0_{\mathbf{w}_g} + d_g, S^*_{\mathbf{w}_g} = \frac{S^0_{\mathbf{w}_g}\nu^0_{\mathbf{w}_g} + S_{\mathbf{w}_g}d_g}{\nu^0_{\mathbf{w}_g} + d_g}\right) \tag{4.14}$$

respectively, where $S_{\mathbf{y}}$ is equal to $\sum_{i=1}^{N}(y_i - f(\mathbf{x}_i, \mathbf{w}))^2/N$, $S_{\mathbf{w}_g}$ is equal to $\sum_{i=1}^{d_g} w_i^2/d_g$, N is the number of training samples and d_g is the dimension of the gth weight group. Draws from these distributions may be easily obtained by sampling X from the $\chi^2_{\nu^*}$ distribution and letting $\sigma^2 = \nu^* S^*/X$.

To enable $\sigma^2_{\mathbf{y}}$ and $\sigma^2_{\mathbf{w}_g}$ to be determined from the data, it is necessary that the respective hyperpriors $p(\sigma^2_{\mathbf{y}})$ and $p(\sigma^2_{\mathbf{w}_g})$ are rather noninformative. In general, the smaller ν^0 is relative to N and d_g, the less informative is the scaled inverse chi-squared prior distribution (*Gelman et al.*, 2004). This can be seen in Figure 4.3, which shows that S^0 affects the

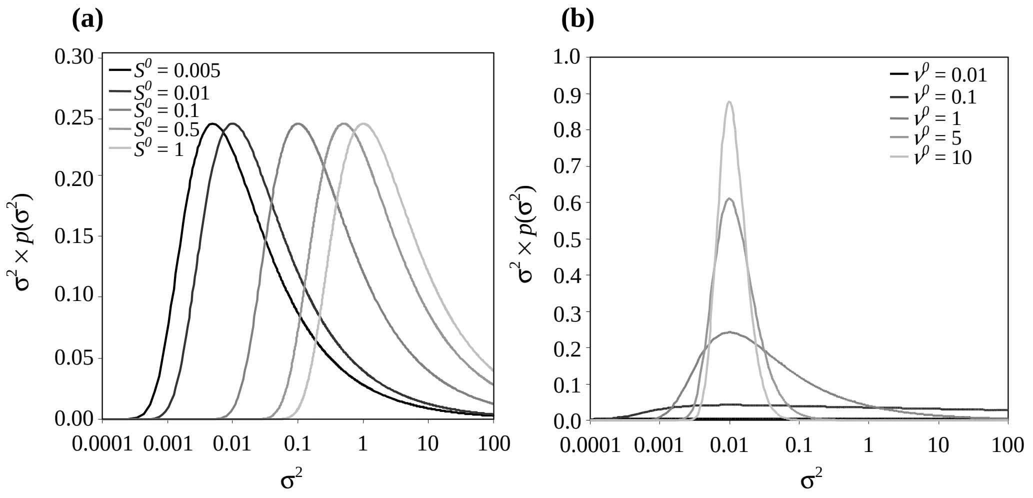

Figure 4.3 Effect of (a) scale S^0 and (b) degrees of freedom ν^0 on the scaled inverse chi-squared distribution $\chi^{-2}(\nu^0, S^0)$.

location of the scaled inverse chi-squared distribution, while ν^0 affects the scale. Details of the hyperpriors are typically uncritical unless extreme or unrealistic values are chosen for their modes (*Husmeier et al.*, 1999); however, both *Neal* (1996a) and *Husmeier et al.* (1999) caution that the use of very vague priors may slow convergence of the MCMC sampling procedure, which may lead to poor predictive performance if the posterior has not been adequately sampled.

4.2.3.4 Proposed MCMC Sampling and Prediction Approach

Given the advantages of the AM algorithm over the more straightforward MCMC sampling procedures (e.g. the random walk Metropolis algorithm), and its simplicity in comparison to some of the more complex methods discussed in Section 4.2.2, the AM algorithm will be employed to sample the weight vectors in the proposed MCMC sampling approach. Furthermore, since the use of conjugate hyperprior distributions enables straightforward sampling from the full conditional distributions of the variance hyperparameters, the Gibbs sampler will be employed to sample the hyperparameters. Therefore, the proposed MCMC Bayesian training approach follows a two-step iterative procedure, where, in the first step, the hyperparameters $\left\{\sigma^2_{\mathbf{y}}, \sigma^2_{\mathbf{w}}\right\}$ are held constant while the weight vector $\mathbf{w}$ is sampled from $p(\mathbf{w}|\sigma^2_{\mathbf{w}}, \sigma^2_{\mathbf{y}}, \mathbf{y})$ using the AM algorithm, while, in the second step, $\mathbf{w}$ is held constant while $\sigma^2_{\mathbf{y}}$ and $\sigma^2_{\mathbf{w}}$ are sampled their respective full conditional distributions (given by (4.13) and (4.14), respectively) using the Gibbs sampler. This is consistent with the MCMC approach used by *Neal* (1996a) and discussed in Section 2.3.4.2, except that the complicated HMC algorithm has been substituted with the much simpler AM algorithm. Due to the simplicity of both the AM algorithm and the Gibbs sampler, programming errors are less likely in the implementation of this algorithm than if a more complex approach was used.

To initialise the training procedure, arbitrary values of $\mathbf{w}_0$, $\sigma^2_{\mathbf{y},0}$, $\sigma^2_{\mathbf{w},0}$ and Σ_0 are required. Appropriate values of $\sigma^2_{\mathbf{y},0}$ and $\sigma^2_{\mathbf{w},0}$ are $S_\mathbf{y}$ and $S_\mathbf{w}$, respectively, which define the location of the prior distributions for these hyperparameters. *Haario et al.* (2001) suggest using maximum likelihood estimates of $\mathbf{w}$, which give a rough estimate of the location of the posterior distribution, to avoid the AM algorithm starting slowly. Not only does this increase convergence speed, but these values provide a useful check of the accuracy of the Bayesian training algorithm. Therefore, in this research, $\mathbf{w}_0$ was set equal to the deterministic estimate $\hat{\mathbf{w}}$ obtained by minimising the SSE (see Chapter 3). *Haario et al.* (2001) also suggest that if Σ_0 is chosen to approximate the covariance of the target distribution, the AM algorithm will be more efficient during the initial stages of the simulation. An approximation of the posterior covariance is described by *MacKay* (1995a), which involves evaluating the Hessian matrix of the regularised error function, given by (2.13). However, as discussed in Section 2.2.5.3, this Hessian matrix is often ill-conditioned, resulting in a (nearly) singular covariance matrix. This, in turn, can cause instability of the AM algorithm, as use of the recursion formula (4.12) means that the covariance is updated based on an initial ill-conditioned matrix. Nevertheless, for the AM algorithm, the only actual requirements for the choice of Σ_0 are that it is positive definite and allows the algorithm to move at least a little in the initial fixed covariance stage. Therefore, in the proposed implementation, Σ_0 is defined by:

$$\Sigma_0 = c^2 \times \begin{bmatrix} \sigma^2_{w_1,0} & 0 & \cdots & 0 \\ 0 & \sigma^2_{w_2,0} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma^2_{w_d,0} \end{bmatrix} \tag{4.15}$$

Gelman et al. (2004) give a number of recommendations for achieving optimal efficiency using a multinormal proposal distribution centred on the current weight state with the same shape as the target distribution, such as that used by the AM algorithm. One of these is that the optimal jumping rule has an acceptance rate of approximately 23% (for dimension $d > 5$), with the most efficient scaling parameter being $c \approx 2.4/\sqrt{d}$. *Gelman et al.* (2004) also recommend that this scaling parameter should be tuned up or down if the acceptance rate is too high or low, respectively. However, since the AM algorithm already incorporates an adaptation strategy, in this research, c will only be tuned in the initial part of the simulation, which is particularly important to ensure that sufficient states are accepted when Σ_0 is fixed.

The algorithm is run for sufficient iterations, t_F, firstly, to achieve convergence to the stationary distribution and secondly, to sample enough weight states following convergence to provide an adequate representation of the posterior distribution. However, weight states simulated prior to when convergence is reached at $t = t_b << t_F$ will still be influenced by the initial distribution, rather than the posterior distribution; therefore, these simulations are discarded. The practice of discarding early iterations is commonly referred to as *burn-in*. A conservative choice for the burn-in period is half of the total iterations (i.e. $t_F/2$) (*Gelman et al.*, 2004). The remainder of the simulations (or a smaller representative subset) are used as the basis for making Monte Carlo estimates from the predictive distribution of a data set. In this research, the predictive distribution will be summarised by 95% prediction limits, which are useful for the visualisation of prediction uncertainty, and the mean predictions, which enable a direct comparison of predictive performance with that of a deterministic ANN (i.e. an ANN based on single valued weight estimates).

Given the above considerations, the full Bayesian training procedure used in this study was carried out as follows:

```
1. Set $\mathbf{w}_0 = \hat{\mathbf{w}}$; $\sigma^2_{\mathbf{y},0} = S^0_{\mathbf{y}}$; $\sigma^2_{\mathbf{w},0} = S^0_{\mathbf{w}}$ and evaluate
   $\log p^*_0(\mathbf{w}_0|\mathbf{y}) = \left[\log L(\mathbf{w}_0|\sigma^2_{\mathbf{y},0}) + \log p(\mathbf{w}_0|\sigma^2_{\mathbf{w},0})\right]$. Initialise $\Sigma_0$ according to (4.15)

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

   (a) If $t \leq t_0$,
           let $\Sigma_t = \Sigma_0$
   (b) Sample a candidate $\mathbf{w}^*$ from $Q(\mathbf{w}^*|\mathbf{w}_{t-1}) = N(\mathbf{w}_{t-1}, \Sigma_t)$
   (c) Evaluate
           $\log p^*_t(\mathbf{w}^*|\mathbf{y}) = \left[\log L(\mathbf{w}^*|\sigma^2_{\mathbf{y},t-1}) + \log p(\mathbf{w}^*|\sigma^2_{\mathbf{w},t-1})\right]$,
       and calculate
           $\alpha(\mathbf{w}^*|\mathbf{w}_{t-1}) = \min\left\{\exp\left[\log p^*_t(\mathbf{w}^*|\mathbf{y}) - \log p^*_{t-1}(\mathbf{w}_{t-1}|\mathbf{y})\right], 1\right\}$
   (d) Generate $u$ from $U(0,1)$, and if $u \leq \alpha(\mathbf{w}^*|\mathbf{w}_{t-1})$,
           set $\mathbf{w}_t = \mathbf{w}^*$,
       otherwise,
           set $\mathbf{w}_t = \mathbf{w}_{t-1}$
   (e) Sample $\sigma^2_{\mathbf{y},t}$ from $p(\sigma^2_{\mathbf{y},t}|\mathbf{w}_t, \mathbf{y}) = \chi^{-2}\left(\nu^*_{\mathbf{y}}, S^*_{\mathbf{y}}\right)$ and $\sigma^2_{\mathbf{w},t}$ from
       $p(\sigma^2_{\mathbf{w},t}|\mathbf{w}_t) = \chi^{-2}\left(\nu^*_{\mathbf{w}}, S^*_{\mathbf{w}}\right)$
```

   (f) If $t = t_0$,
      calculate $\Sigma_{t+1} = c^2\text{cov}(\mathbf{w}_0, \mathbf{w}_1, \ldots, \mathbf{w}_{t_0}) + c^2 e I_d$, where
      $\text{cov}(\mathbf{w}_0, \mathbf{w}_1, \ldots, \mathbf{w}_{t_0}) = \frac{1}{t_0}\left(\sum_{i=0}^{t_0} \mathbf{w}_i\mathbf{w}_i^T - (t_0+1)\overline{\mathbf{w}}_{t_0}\overline{\mathbf{w}}_{t_0}^T\right)$,
      reset $c = 2.4/\sqrt{d}$
   else if $t > t_0$,
      calculate Σ_{t+1} according to (4.12), substituting $\mathbf{w}$ for θ

3. Discard initial samples $(\mathbf{w}_0, \sigma^2_{\mathbf{y},0}, \sigma^2_{\mathbf{w},0}), \ldots, (\mathbf{w}_{t_b}, \sigma^2_{\mathbf{y},t_b}, \sigma^2_{\mathbf{w},t_b})$ to diminish the effects of the initial distribution. Use samples $\left\{\mathbf{w}_{t_b+1}, \sigma^2_{\mathbf{y},t_b+1}, \sigma^2_{\mathbf{w},t_b+1}\right\}, \ldots,$ $\left\{\mathbf{w}_{t_F}, \sigma^2_{\mathbf{y},t_F}, \sigma^2_{\mathbf{w},t_F}\right\}$ for analysis.

4. For $i = 1, 2, \ldots, N_{testset}$

   (a) calculate the network predictions $y_{i,t_b+1}, \ldots, y_{i,t_F}$ based on $\mathbf{w}_{t_b+1}, \ldots, \mathbf{w}_{t_F}$ and input vector $\mathbf{x}_i$.

   (b) Rank predictions $y_{i,t_b+1}, \ldots, y_{i,t_F}$ in ascending order and determine 95% simulation limits.

   (c) Calculate mean prediction
   $\bar{y}_i = 1/(t_F - t_b)\sum_{t=t_b+1}^{t_F} y_t$.

4.2.3.5 Diagnosing and Improving Convergence

The most critical issue associated with MCMC simulations is determining when convergence to the posterior distribution has been achieved. Due to the generally high dimension of $\mathbf{w}$ and the correlations between individual weights, convergence to the posterior weight distribution is usually relatively slow, as mentioned in Section 4.2.2. Multiple chains can be used to more widely explore the weight space and help to speed convergence to the posterior distribution. Furthermore, multiple chains can be used to reveal problems with convergence that cannot be seen by looking at a single chain (*Kass et al.*, 1998). Therefore, in this research, a number of parallel chains are simulated simultaneously using the MCMC approach discussed above.

Gelman et al. (2004) recommend that the multiple chains should be initialised with starting points drawn from a distribution believed to be overdispersed with respect to the posterior distribution. However, to verify, or feel confident, that the points are in fact overdispersed requires knowledge of the posterior distribution. Approximating the posterior would generally involve evaluation of the Hessian matrix at the posterior mode, which, as discussed, may be near singular for an ANN. Therefore, it is proposed that the multiple chains are each initialised at the maximum likelihood estimate $\hat{\mathbf{w}}$. As the posterior distribution of ANN weights is often multimodal, it is acknowledged that this initialisation may bias the resulting posterior distribution if the algorithm becomes trapped in the vicinity of the local mode. However, it is hypothesised that the bias that may be caused by this initialisation may be decreased if, for an initial period $t_{\sigma_0^2}$, $\sigma_{\mathbf{y}}^2$ and $\sigma_{\mathbf{w}}^2$ are fixed in such a way that would allow the simulated chains to move more freely about the weight space during this period. For example, if $\sigma_{\mathbf{y}}^2$ was to quickly adapt to some value close to $\hat{\sigma}_{\mathbf{y}}^2$, where $\hat{\sigma}_{\mathbf{y}}^2$ results in the (locally) maximum likelihood value, given $\hat{\mathbf{w}}$, only small jumps around $\hat{\mathbf{w}}$ would be accepted, making it difficult to move away from the initial location. However, by fixing $\sigma_{\mathbf{y}}^2 > \hat{\sigma}_{\mathbf{y}}^2$ such that the magnitude of the initial likelihood is somewhat reduced, but the scale is increased, the acceptance rate of weight vectors further from $\hat{\mathbf{w}}$ would be increased. To ensure that the weights are not restricted during this period, it would also be important to fix $\sigma_{\mathbf{w}}^2 = 1$. The effects that fixing $\sigma_{\mathbf{y}}^2$ and $\sigma_{\mathbf{w}}^2$ have on the ability of the MCMC training algorithm to escape a local minimum will be investigated in this research.

Müller and Rios Insua (1998) and *Neal* (1993, 1996b) discuss the multimodality of ANN posterior distributions at length, as well as methods for efficiently sampling from such distributions. While many of these "mode hunting" schemes are relatively complicated, a rather straightforward simulated annealing method is discussed by *Neal* (1992, 1993). Using simulated annealing, the Metropolis acceptance probability is modified as follows:

$$\alpha(\mathbf{w}^*|\mathbf{w}_{t-1}) = \min\left\{\frac{\exp\left[\log p_t^*(\mathbf{w}^*|\mathbf{y}) - \log p_{t-1}^*(\mathbf{w}_{t-1}|\mathbf{y})\right]}{T}, 1\right\} \tag{4.16}$$

where T is referred to as the *temperature*. Therefore, the sampled weight vectors asymptotically represent the distribution $p^*(\mathbf{w}|\mathbf{y})/T$ (*Bishop*, 1995). The idea is that for $T >> 1$, the MCMC sampling procedure may explore the weight space much more freely, allowing it to converge to its stationary distribution more quickly and to escape local minima. The process is started at a high initial temperature T, which is gradually cooled (reduced) throughout the simulation until it reaches $T = 1$, where $p^*(\mathbf{w}|\mathbf{y})/T$ represents the desired target distribution. It is hoped that, as the temperature is reduced, the algorithm will settle into a region of high posterior probability. This process is illustrated in Figure 4.4, which shows a multimodal distribution $p^*(\mathbf{w}|\mathbf{y})/T$ that may be sampled from relatively freely, given that T is large, but is represented by two disjointed modes when $T = 1$. The success of simulated annealing, however, depends on whether or not the distribution at higher temperatures is a good guide to the distribution at lower temperatures; therefore, good re-

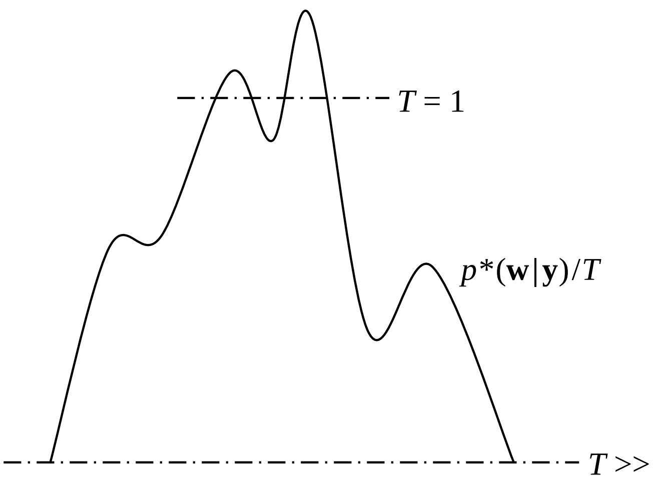

Figure 4.4 The effect of different temperatures T on the distribution $p^*(\mathbf{w}|\mathbf{y})/T$ using simulated annealing.

sults cannot be guaranteed in all cases (*Neal*, 1993). The effect of simulated annealing on convergence will also be investigated in this research.

There are numerous diagnostic tools available to help determine whether or not an MCMC sampling procedure has converged, many of which are reviewed by *Cowles and Carlin* (1996) and *Brooks and Roberts* (1998). However, caution is needed when using any of these tools because, in general, although the diagnostics may often succeed at detecting convergence failure, they may also fail at doing this (*Cowles and Carlin*, 1996). The most commonly used diagnostics of convergence are trace plots of sample MCMC values versus iteration (*Kass et al.*, 1998). Traces of various important quantities, such as log posterior probability, log likelihood, log prior probability, and important hyperparameters, can be visually inspected to determine when or whether approximate convergence has been reached. For example, as stated in *Kass et al.* (1998), if the log posterior probability is increasing, the main mode has yet to be reached, whereas if it is decreasing, the algorithm was initialised near a tall, narrow mode and is moving towards a more representative part of the distribution. Therefore, it can be assumed that convergence has been reached when this plot flattens out. By overlaying traces obtained by different chains on a common graph, it is generally easier to detect convergence or failure thereof. However, a limitation of MCMC posterior simulation, particularly when applied to multimodal problems like ANNs, is that it can never be guaranteed that convergence to the true posterior has been obtained, as there may still be modes that have been undiscovered.

4.3 Bayesian Model Selection

As discussed in Section 2.3.5, Bayesian model selection (BMS) involves the comparison of a set of H competing models $\{\mathcal{H}_i; i = 1, \ldots, H\}$ based on the posterior probability that each model $\mathcal{H}_i$ is the "true" model of the system, given the observed data, which is calculated by (2.21). When the prior probabilities assigned to the different models are approximately equal, as is generally the case, this simplifies to (2.22), which states that the relative probabilities of the competing models can be compared based on their evidence $p(\mathbf{y}|\mathcal{H})$. The evidence of a model can be evaluated by the integral given by (2.6); however, except for the simplest of models, this integral is analytically intractable. Therefore, alternative methods are needed to estimate $p(\mathbf{y}|\mathcal{H})$. A number of methods have been proposed for this (*Gelfand and Dey*, 1994; *Kass and Raftery*, 1995); however, given that MCMC posterior simulation is used for the proposed Bayesian training approach, only methods for estimating $p(\mathbf{y}|\mathcal{H})$ based on sampled draws from the posterior will be considered in this research.

4.3.1 Computation of Evidence via Posterior Simulation

While this section by no means provides an exhaustive discussion of methods available for approximating the evidence based on sampled draws from the posterior distribution (*DiCiccio et al.* (1997) gives a review of such methods), several methods with general applicability, which use samples from the posterior directly, are described. These methods include the Newton-Raftery estimator (*Newton and Raftery*, 1994), the Gelfand-Dey estimator (*Gelfand and Dey*, 1994) and the Chib-Jeliazkov estimator (*Chib and Jeliazkov*, 2001).

4.3.1.1 Newton-Raftery Estimator

The Newton-Raftery estimator for $p(\mathbf{y}|\mathcal{H}_i)$ is based on sampled draws from the posterior distribution and importance sampling. Importance sampling is a useful method for computing the expectation of a function using samples generated from a density Q^*, known as the *importance sampling function* (*Geweke*, 1989; *Newton and Raftery*, 1994). In terms of a model's evidence, it can be used to provide an estimate of the integral in (2.6) in the form of:

$$\hat{p}(\mathbf{y}|\mathcal{H}) = \frac{\sum_{i=1}^{ns} q_i p(\mathbf{y}|\theta_i, \mathcal{H})}{\sum_{i=1}^{ns} q_i} \tag{4.17}$$

where ns is the number of samples generated from Q^* and $q_i = p(\theta_i)/Q^*(\theta_i)$.

When the samples are drawn from the posterior distribution of θ, the importance sampling function is $Q^* = p(\mathbf{y}|\theta, \mathcal{H})p(\theta|\mathcal{H})/p(\mathbf{y}|\mathcal{H})$, which, when substituted into (4.17), gives the estimator:

$$\hat{p}(\mathbf{y}|\mathcal{H}) = \left\{ \frac{1}{ns} \sum_{i=1}^{ns} p(\mathbf{y}|\theta_i, \mathcal{H})^{-1} \right\}^{-1} \tag{4.18}$$

which is the harmonic mean of the likelihood values. As $ns \rightarrow \infty$, this estimate converges to the correct value. However, although (4.18) is very easy to calculate, it is unstable, with occasional values of small likelihood having a large effect on the final result (*Newton and Raftery*, 1994). Therefore, *Newton and Raftery* (1994) proposed that the importance sampling function be a mixture of the prior and posterior densities $Q^* = \delta p(\theta_i|\mathcal{H}_i) + (1-\delta)p(\theta_i|\mathbf{y}, \mathcal{H}_i)$, where $0 < \delta < 0.5$ such that the estimator is based mostly on high likelihood values of θ. This estimator overcomes the instability of that given by (4.18) and, as $ns \rightarrow \infty$, also converges to the correct value. However, this method has the disadvantage that the prior must be sampled from, as well as the posterior.

4.3.1.2 Gelfand-Dey Estimator

Gelfand and Dey (1994) also proposed a modification of the harmonic mean estimator given by (4.18). The Gelfand-Dey estimator is given by:

$$\hat{p}(\mathbf{y}|\mathcal{H}) = \left\{ \frac{1}{ns} \sum_{i=1}^{ns} \frac{\tau(\theta_i)}{p(\mathbf{y}|\theta_i, \mathcal{H})p(\theta_i|\mathcal{H})} \right\}^{-1} \tag{4.19}$$

where $\tau(\theta)$ is any proper density, which is sometimes called a *tuning function* (*Chib*, 1995) and plays the role of the importance sampling function. This is an unbiased and consistent estimator of $p(\mathbf{y}|\mathcal{H})$, which is stable, provided the tails of $\tau(\theta)$ are sufficiently thin to prevent occasional small likelihood values from significantly influencing the final estimate. Furthermore, for $\hat{p}(\mathbf{y}|\mathcal{H})$ to have a small variance, $\tau(\theta)$ must be similar to $p^*(\theta|\mathbf{y}, \mathcal{H}) = p(\mathbf{y}|\theta, \mathcal{H})p(\theta|\mathcal{H})$ (*DiCiccio et al.*, 1997). A natural choice for $\tau(\theta)$ would therefore be a multivariate normal with mean and covariance computed from the sampled draws from the posterior distribution $\{\theta_1, \ldots, \theta_{ns}\}$ (*Gelfand and Dey*, 1994). In a comparison of the Gelfand-Dey and Newton-Raftery estimators for selecting an appropriate annual rainfall model, *Frost* (2004) found the Gelfand-Dey estimator to be more accurate and consistent between simulations than the Newton-Raftery estimator.

4.3.1.3 Chib-Jeliazkov Estimator

By rearranging the general Bayes' theorem expression given by (2.5) and taking the logarithm at some fixed point $\hat{\theta}$, the following expression for $\log p(\mathbf{y}|\mathcal{H})$ is obtained:

$$\begin{aligned} \log p(\mathbf{y}|\mathcal{H}) &= \log p(\mathbf{y}|\acute{\theta}, \mathcal{H}) + \log \hat{p}(\acute{\theta}|\mathcal{H}) \\ &\quad - \log p(\acute{\theta}|\mathbf{y}, \mathcal{H}) \end{aligned} \tag{4.20}$$

Thus, if $\hat{\theta}$ is a sampled draw from the posterior, the only unknown in this equation is $\log p(\hat{\theta}|\mathbf{y}, \mathcal{H})$. The estimator proposed by *Chib and Jeliazkov* (2001) is based on estimating the posterior density at a single point $\acute{\theta}$ sampled from the posterior using the Metropolis-Hastings algorithm (see Section 4.2.1.1), in order to solve (4.20). This is done using the following equation:

$$p(\acute{\theta}|\mathbf{y}, \mathcal{H}) = \frac{ns_1^{-1} \sum_{i=1}^{ns_1} \alpha(\acute{\theta}|\theta_i) Q(\acute{\theta}|\theta_i)}{ns_2^{-1} \sum_{j=1}^{ns_2} \alpha(\theta_j|\acute{\theta})} \tag{4.21}$$

where θ_i are sampled draws from the posterior distribution, θ_j are sampled draws from the proposal distribution $Q(\theta_i|\acute{\theta})$ used by the Metropolis-Hastings algorithm and $\alpha(\cdot)$ is given by (4.4). *Chib and Jeliazkov* (2001) note that while the choice of $\acute{\theta}$ is arbitrary, for estimation efficiency it is appropriate to choose a point that has high posterior density. Furthermore, they state that although ns_1 and ns_2 may be different, in practice they are set to be equal (i.e. $ns = ns_1 = ns_2$). In a study conducted by *Marshall et al.* (2005), the Chib-Jeliazkov estimator was used for selecting the appropriate level of complexity for a conceptual rainfall-runoff model, given MCMC samples from the posterior parameter distributions of various competing models. Using synthetically generated data, it was shown that this estimator was able to successfully identify the model by which the data were generated as having the highest evidence.

4.3.2 Bayes Factors

The evidence ratio of a pair of competing models is known as the *Bayes' factor* (BF), defined by:

$$BF_{j,k} = \frac{p(\mathbf{y}|\mathcal{H}_j)}{p(\mathbf{y}|\mathcal{H}_k)}, \tag{4.22}$$

which is interpreted as the evidence of model $\mathcal{H}_j$ in favour of model $\mathcal{H}_k$. Regardless of the value of the prior odds (i.e. the prior probability of $\mathcal{H}_j$ in favour of $\mathcal{H}_k$), the Bayes factor is the ratio of the posterior odds of $\mathcal{H}_j$ to its prior odds, given by:

$$BF_{j,k} = \frac{p(\mathcal{H}_j|\mathbf{y})}{p(\mathcal{H}_k|\mathbf{y})} \div \frac{p(\mathcal{H}_j)}{p(\mathcal{H}_k)} \tag{4.23}$$

Thus, when assuming equal prior probabilities, the Bayes factor is equal to the posterior odds in favour of $\mathcal{H}_j$. BMS therefore involves estimating the evidence of each competing model, performing pairwise comparisons between the models and ranking them according to their Bayes factors.

4.3.2.1 Interpreting Bayes Factors

It may be useful to interpret the Bayes factor in terms of its logarithm, as the log Bayes factor is the difference in predictive scores between two models, or the relative success of $\mathcal{H}_j$ and $\mathcal{H}_k$ at predicting the data (*Kass and Raftery*, 1995). In order to interpret the strength of evidence in favour of model $\mathcal{H}_j$ over $\mathcal{H}_k$ provided by $BF_{j,k}$, *Kass and Raftery* (1995) gave an interpretive scale based on twice the natural logarithm of the Bayes factor, which is on the same scale as the more familiar likelihood ratio test. This scale is reproduced in Table 4.1; however, it is given in terms of $\log_e BF_{j,k}$ rather than $2\log_e BF_{j,k}$, since the predictive score interpretation is preferred in this research.

Table 4.1 Interpretive scale for Bayes factors (Source: adapted from *Kass and Raftery* (1995)).

4.3.2.2 Sensitivity to Prior Distributions

In general, when using Bayes factors for model comparison, care needs to be taken in choosing an appropriate prior for θ. In order to estimate θ using Bayesian methods, the prior distribution is often chosen for convenience, since it is known that if the data sample is fairly large, the influence of the prior will be small. However, the prior distribution has a more significant influence on the evidence of a model than it does on the posterior, as shown by *Kass* (1993). There has been some criticism regarding the use of Bayes factors when applied to models where there is little *a priori* knowledge, as the use of improper (i.e. does not integrate to 1) noninformative priors can be problematic (*Kass and Raftery*, 1995). Therefore, the prior distributions must be proper and not have too big a scale. *Raftery* (1996) argues that subjective priors that are relatively flat in the region where the likelihood is large should only have a small impact on Bayes factors. An advantage of using hierarchical priors is that a subjective choice of prior distribution is not required, as it is determined by the data.

4.3.3 BMS Previously Applied to ANNs

Although not in relation to ANNs, *Berger and Rios Insua* (1998) stated that the Bayesian approach to model selection is less widely used than the Bayesian approach to parameter estimation in hydrological applications. This is also true in the case of ANNs, where, apart from the evidence approach of *MacKay* (1992a, 1995a) and the automatic model selection methods proposed by *Müller and Rios Insua* (1998) and *Liang* (2005), there have been very few applications of BMS, particularly using methods relying on sampled draws from the posterior weight distribution. *Neal* (1993) discusses a number of methods for estimating the Bayes factor based on posterior simulation in terms of “free energy estimation”, as proposed in the statistical physics literature. All of these methods, however, require analytical effort to tailor them to a particular application (*Kass and Raftery*, 1995). Later, *Neal* (1994, 1996a) advocated the use of ANN models with unrestricted complexity and suitably chosen hierarchical prior distributions to prevent overfitting, thereby ignoring the problem of model selection.

Lee (2001) proposed a BMS approach using the Metropolis-Hastings algorithm to sample from the model space (where the free parameters included inputs and hidden nodes) in order to find models with high posterior probability. To do this, minus half the BIC, given by (3.8), was used to approximate $\log p(\mathbf{y}|\mathcal{H}_i)$ for each of the models visited by the Metropolis-Hastings algorithm, as it has been shown that (*Schwarz*, 1978):

$$p(\mathbf{y}|\mathcal{H}) = \exp\left(-\frac{1}{2}\text{BIC}\right) + O_P(1) \tag{4.24}$$

While this may not seem to be a great approximation (since the error $O_P(1)$ does not decrease for a large sample size N), the BIC has been shown to be asymptotically consistent for model selection and has been found to work well in practice (*Lee*, 2001). The acceptance probability used for the Metropolis-Hastings algorithm was, therefore, $\alpha(\mathcal{H}^*|\mathcal{H}_{t-1}) = \min\{1, \exp(-1/2\text{BIC}^* + 1/2\text{BIC}_{t-1})\}$. However, while MCMC was used to sample from the model space, this method did not use samples from the posterior weight distribution. Instead, a standard algorithm was used to train each of the sampled models and MCMC was only used at the end of the process to estimate the posterior weight distribution for the best selected model. However, *Lee* (2002) then compared a number of methods for approximating $p(\mathbf{y}|\mathcal{H})$ based on sampled draws from the posterior distribution, together with the BIC approximation. The methods were applied to two data sets: one real and one simulated. Overall, all of the methods had difficulty in approximating $p(\mathbf{y}|\mathcal{H})$, particularly on the real data, where the methods did not agree at all. It was found that the sample-based methods (i.e. all methods except the BIC) were very sensitive to the MCMC sample, showing highly variable results for a given model size, based on different MCMC samples. It was concluded that the BIC, which was able to correctly indicate the best model structure for each data set, was reasonably stable and reliable and was apparently a more useful approximation for standard data set sizes than methods based on MCMC samples.

Vehtari and Lampinen (2002) proposed a framework for using the "expected utilities" of two models for model comparison. If the utility of a model is the posterior predictive distribution of y given input $\mathbf{x}_i$, in other words the utility of a model is a measure of how good its predictions are, *Vehtari and Lampinen* (2002) suggested that the expected utility can then be used to assess how good the model is. Importance sampling was employed to estimate the leave-one-out predictive densities for a given model, which were then used to calculate the expected utility of the model and the expected utility distribution. The difference between the expected utilities of two models $\mathcal{H}_j$ and $\mathcal{H}_k$ can be used for model selection, where the expected utility distributions of the models are used to determine the probability that the expected utility of $\mathcal{H}_j$ is greater than that of $\mathcal{H}_k$.

4.3.4 Proposed BMS Framework

4.3.4.1 Evaluating the Evidence

In this research, three methods will be investigated for their ability to estimate $p(\mathbf{y}|\mathcal{H})$. These include the Gelfand-Dey (G-D) and Chib-Jeliazkov (C-J) estimators described in Sections 4.3.1.2 and 4.3.1.3, respectively, since these estimators have been used successfully for hydrological model selection (*Marshall et al.*, 2005; *Frost*, 2004), as well as a BIC approximation, since *Lee* (2002) found the BIC to be a more stable approximation than sample-based methods. The G-D, C-J and BIC estimators are easy to program and only require a relatively simple step after training with MCMC. The Newton-Raftery estimator, on the other hand, is more difficult to apply, since sampled draws from the prior distribution are required, as well as sampled draws from the posterior. Therefore, this method is not considered further in this research.

Each of the estimators investigated has relative merits over the others. The G-D estimator uses sampled draws directly and requires no further samples to be generated from any other distribution. It has also been shown to be very accurate, provided that the tuning function τ is chosen appropriately (*Kass and Raftery,* 1995). However, this method may also produce very poor results given a bad choice of τ. Nevertheless, this estimator seems well suited for application in conjunction with AM training. For good results with the G-D estimator, τ should be chosen so that it approximates the posterior distribution and, since the proposal distribution is adapted towards the posterior throughout the AM simulation, a good choice for τ is determined automatically. Therefore, in this research, $\tau(\mathbf{w}_i, \sigma^2_{\mathbf{y}}, \sigma^2_{\mathbf{w}}) = Q(\mathbf{w}_i|\tilde{\mathbf{w}}) = N(\tilde{\mathbf{w}}, \Sigma_f)$, where $\tilde{\mathbf{w}}$ is the median of the posterior distribution and Σ_f is the value covariance of the proposal (estimated by (4.12)) at the final iteration of the AM algorithm. If values of $p(\mathbf{y}|\mathbf{w}_i, \sigma^2_{\mathbf{y}}, \mathcal{H})p(\mathbf{w}_i|\sigma^2_{\mathbf{w}}, \mathcal{H})$ are recorded throughout the AM simulation, only $\tau(\mathbf{w}_i)$ requires evaluation for $i = 1, \ldots, ns$ in order to calculate $\hat{p}(\mathbf{y}|\mathcal{H})$.

The C-J estimator has an advantage over the G-D method in that a tuning function is not required. However, samples are required from the proposal distribution, as well as from the posterior. This step is simple, since sampling from the proposal is carried out throughout the AM simulation; however, it increases the computational cost of the overall algorithm. Furthermore, evaluation of $\alpha(\acute{\mathbf{w}}|\mathbf{w}_i)$, $Q(\acute{\mathbf{w}}|\mathbf{w}_i)$ and $\alpha(\mathbf{w}_j|\acute{\mathbf{w}})$ are required for $i, j = 1, \ldots, ns$ in order to calculate $\hat{p}(\mathbf{y}|\mathcal{H})$. In this research, $\acute{\mathbf{w}}$ was set equal to the median of the posterior distribution $\tilde{\mathbf{w}}$.

For the proposed BIC estimator, $-1/2\text{BIC}$ values are evaluated for sampled weight vectors $\mathbf{w}_1, \ldots, \mathbf{w}_{ns}$, resulting in a distribution of $-1/2\text{BIC}$ values, which roughly approximates a distribution of $\hat{p}(\mathbf{y}|\mathcal{H})$ values. This is different to the BIC approximation used by *Lee* (2002), where only a single $-1/2\text{BIC}$ value was evaluated for each network. Although it was shown in Chapter 3 that the in-sample BIC was a promising model selection tool, it was also concluded that, when applied deterministically, its value could be sensitive to the weights obtained; thus, affecting the results. By evaluating the $-1/2\text{BIC}$ distribution, this variability is accounted for. The $-1/2\text{BIC}$ is given by:

$$-\frac{1}{2}\text{BIC} = \log p(\mathbf{y}|\mathbf{w}_i, \sigma^2_{\mathbf{y}}, \mathcal{H}) - \frac{d}{2}\log N \tag{4.25}$$

where d is the dimension of $\mathbf{w}$ and N is the size of the training data set. If the likelihood values $p(\mathbf{y}|\mathbf{w}_i, \sigma^2_{\mathbf{y}}, \mathcal{H})$ are recorded throughout the AM simulation, subtraction of the constant $\frac{d}{2}\log N$ from each log likelihood value is all that is required to evaluate the distribution for $\hat{p}(\mathbf{y}|\mathcal{H})$. This distribution can then be used to evaluate a mean $\hat{p}(\mathbf{y}|\mathcal{H})$ value for use in Bayes factor calculations. This is much simpler than either the G-D or the C-J estimators; however, it is not expected to be as accurate due to the error in (4.24).

4.3.4.2 Checking Bayes Factors with Posterior Weight Distributions

As discussed in *DiCiccio et al.* (1997) and *Lee* (2002), it can be difficult to obtain accurate estimates of $p(\mathbf{y}|\mathcal{H})$ based on posterior simulations, particularly in the case of ANNs. Therefore, in this framework, it is proposed that the Bayes factors calculated using the approximated evidence values be used as a guide for model selection, but a final check of the model rankings be carried out using the posterior weight distributions. If the marginal posterior distribution of a hidden-output layer weight includes the value zero within the 95% highest density region, this suggests that the associated hidden node may be pruned from the network, with a 95% level of confidence that model performance will not be affected. If there are more than one hidden-output layer weights with marginal posterior distributions that include zero within the 95% highest density region, scatter plots of pairs of these weights should be inspected to determine whether the joint distribution of the weights passes through the origin (0,0), which would indicate that both weights in the pair, and the corresponding hidden nodes, may be pruned from the network. Otherwise, if the joint distribution does not pass through (0,0), only one of the hidden nodes may be pruned. This is illustrated in Figure 4.5, which shows two different examples of a pair of hidden-output layer weights for which the marginal posterior distributions include zero within the 95% highest density regions. In the example shown in Figure 4.5 (a), the

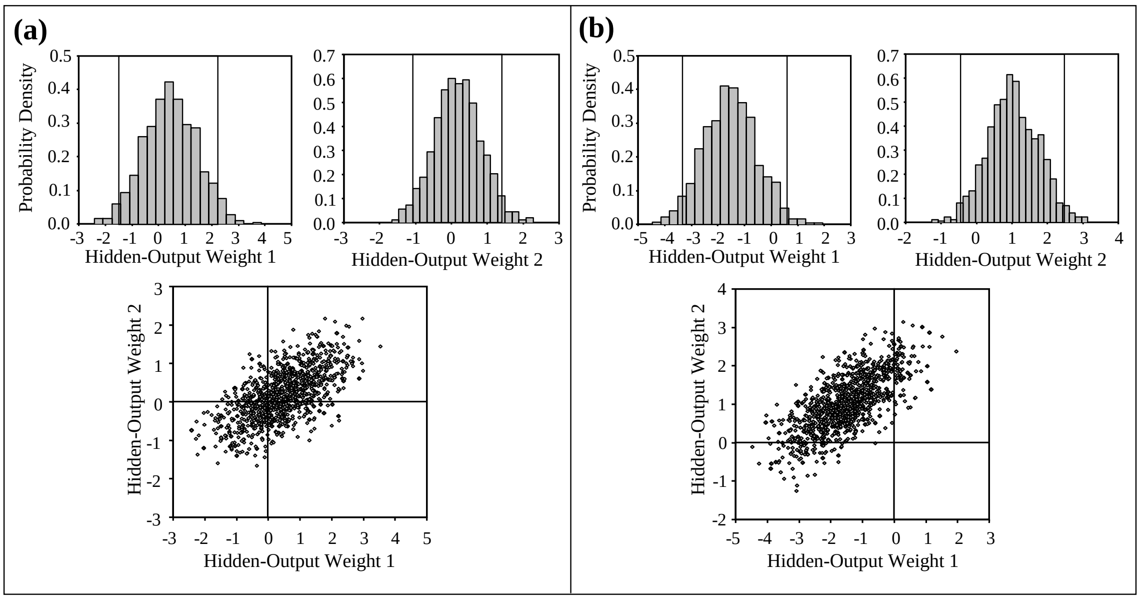

Figure 4.5 Examples of a pair of hidden-output layer weights with marginal posterior distributions that include zero within the 95% highest density regions, where (a) the joint distribution passes through (0,0) and (b) where it does not.

scatter plot of hidden-output weight 1 versus hidden-output weight 2 passes though (0,0), indicating that both hidden nodes are unnecessary in the model. However, in the example illustrated in Figure 4.5 (b), the scatter plot does not pass through the origin, indicating that at least one of the hidden nodes in the pair is necessary. In this research, histograms of the hidden-output weight probability density functions were constructed, from which the 95% highest density regions were estimated.

4.4 Fine-Tuning and Assessment of Bayesian Techniques with Synthetic Data

4.4.1 Bayesian Training and Prediction

Fine-tuning of the proposed Bayesian training and prediction method involved selection of the user-defined parameters for the AM/Gibbs sampler algorithm, determination of the most appropriate form of prior distribution and investigation and recommendation of the best way to improve convergence of the algorithm. Investigations were carried out on synthetic data sets I, II, and III, described in Section 3.4.1.

4.4.1.1 User-Defined Parameters

The user-defined parameters required for the basic AM/Gibbs sampler algorithm (excluding additional parameters required for improving convergence) include the hyperprior parameters $S_{\mathbf{w}}$, $\nu_{\mathbf{w}}$, $S_{\mathbf{y}}$, and $\nu_{\mathbf{y}}$; the number of chains simulated; the adaptive scaling parameter c; the total number of iterations t_F; the number of burn-in iterations t_b; and the initial fixed period t_0. Values of these parameters were selected primarily based on recommendations or typical values used in the Bayesian ANN literature and then were adjusted, if necessary, to suit the application of the proposed Bayesian training framework on the synthetic data sets. Although the number of iterations required for convergence is highly problem dependent, the number of iterations necessary to achieve approximate convergence was investigated when the proposed MCMC algorithm, using the hierarchical prior distribution (since it results in intermediate complexity), was used to train the 2-, 6- and 10-hidden node networks applied to data set II (since this was considered to have intermediate nonlinearity and noise properties of the data sets considered), in order to provide a rough guide for the other data sets and network sizes.

The hyperprior parameters were selected based on those used by *Neal* (1996a). For the input-hidden layer weights and the hidden layer biases, $S_{\mathbf{w}} = 0.01$ and $\nu_{\mathbf{w}} = 0.1$. For the hidden-output layer weights, $S_{\mathbf{w}} = (\frac{1}{100J})^2$, where J is the number of hidden layer nodes, and $\nu_{\mathbf{w}} = 0.1$. A fixed Gaussian prior with $\sigma^2_{\mathbf{w}} = 1$ was used for the output bias. For the noninformative uniform prior distribution, the lower and upper bounds of the distribution were set to -100 and 100, as it was considered that this range would easily incorporate all of the possible weight values (although it does not account for cases where the weights can take any real value due to correlations between the weights). It was decided to simulate four parallel chains based on the number of chains generally used by *Cowles and Carlin* (1996), which is between 3-5. The maximum number of iterations used by *Haario et al.* (2001) and *Marshall et al.* (2004) for the AM algorithm was 80,000 and 100,000, respectively. However, in each of these studies, only fairly low dimensional problems were considered (no greater than a dimension of 8). For the higher dimensional problems considered in this research, it was expected that the AM algorithm would take longer to converge; therefore, t_F was initially set to 400,000. Traces of the log prior density ($\log p(\mathbf{w}|\sigma^2_{\mathbf{w}})$), the log likelihood ($\log L(\mathbf{w}|\sigma^2_{\mathbf{y}})$) and the log unnormalised posterior density ($p^*(\mathbf{w}|\mathbf{y}) = \log L(\mathbf{w}|\sigma^2_{\mathbf{y}}) + \log p(\mathbf{w}|\sigma^2_{\mathbf{w}})$) were then inspected to determine when or whether convergence had been reached for the 2-, 6- and 10-hidden node ANNs applied to data set II. The number of burn-in iterations t_b was then determined from these plots based on the point when the traces became approximately flat. The adaptive scaling parameter c was initially set equal to $2.4/\sqrt{d}$, as recommended by *Gelman et al.* (2004), and was scaled up or down during the initial fixed period t_0 to ensure that the chains moved away from the initial position during this time. The initial period t_0 was selected so that it was short relative to the total number of iterations t_F, yet long enough to allow c to be appropriately scaled such that a sufficient number of candidate states were accepted within this period. The value of t_0 was therefore set equal to 1000.

4.4.1.2 Escaping Local Modes

It is considered that the prior distribution adopted may not only influence the potential of the MCMC algorithm to overfit the data, but also the ability of the simulated chains to escape local modes in the posterior distribution. It is expected that the ability to escape local modes could be increased using the hierarchical and ARD priors described in Section 4.2.3.2, since the simulated chains are encouraged to move to different regions of the posterior not only based on the likelihood value, as is the case for the noninformative uniform prior. Therefore, if the weight vector was initialised at a locally maximum likelihood value, there would be a greater chance of moving away from this mode, as candidate states with lower likelihood values would more easily be accepted if they have a higher prior density. However, both the hierarchical and ARD priors result in an increase in complexity of the MCMC algorithm than when the uniform prior distribution is used. Therefore, the ability of the MCMC algorithm to move away from a poor initialisation of $\mathbf{w}_0$ was investigated to determine whether the increase in complexity is warranted, given the advantages gained by using either a hierarchical or ARD prior.

This investigation has been carried out using data sets II and III, as examples of poor local mode solutions were evident when ANNs were developed to model these data sets in Chapter 3. In contrast, no poor local mode solutions were obtained when ANNs were developed to model data set I, making it unsuitable for this investigation. For data set II, the most obvious example of a poor local mode was found using a 3 hidden node ANN, where each of the deterministic training algorithms obtained a MSE value that was at least 20% worse than the best MSE value obtained (see Tables A.4-A.6 in Appendix A). For data set III, the most obvious example of a poor local mode was found when a 4 hidden node ANN was used to model this data set. In this case, each of the deterministic training algorithms obtained a MSE value that was at least 60% worse than the best MSE value obtained (see Tables A.7-A.9 in Appendix A). These examples are illustrated in Figures 4.6 and 4.7, which show trace plots of the training and test set SSE values when the SCE-UA algorithm, initialised with different sets of random weights, was used to train the 3 hidden node ANN applied to data set II and the 4 hidden node ANN applied to data set III, respectively. In each of these figures, it can be seen that both the training and test set errors obtained in plot (a) are significantly greater than those obtained in plot (b), indicating that the algorithm became trapped in a local minimum when starting from the weight initialisations shown in (a). In this investigation, the MCMC algorithm was initialised with the weights obtained when the SCE-UA algorithm became trapped in these

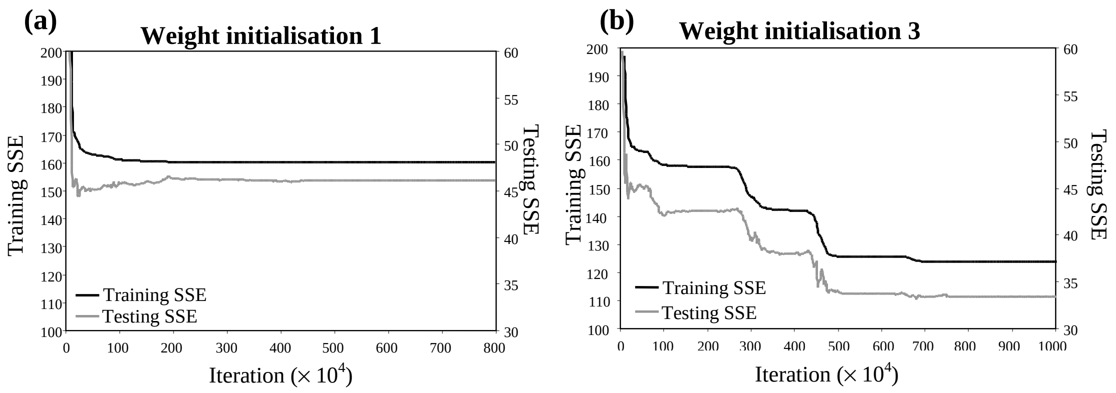

Figure 4.6 Training and test set SSE traces obtained when (a) local minimum and (b) global minimum solutions were converged to when a 3 hidden node ANN was developed to model data set II, using different weight initialisations.

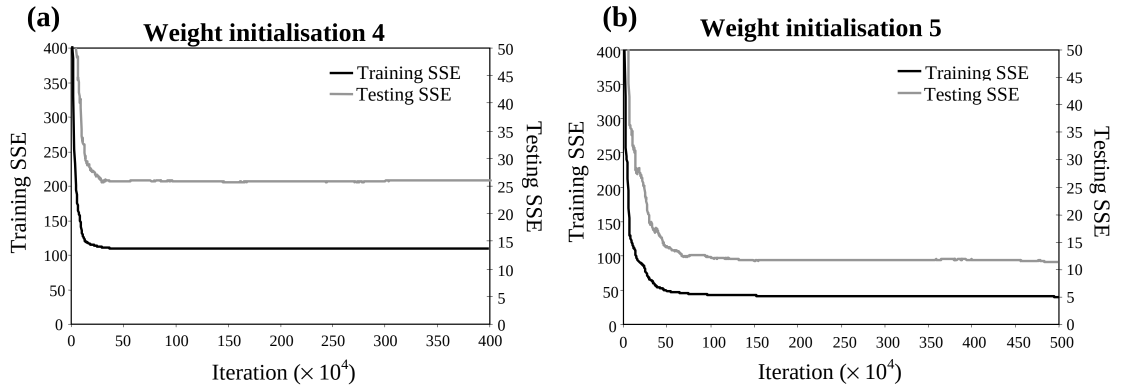

Figure 4.7 Training and test set SSE traces obtained when (a) local minimum and (b) global minimum solutions were converged to when a 4 hidden node ANN was developed to model data set III, using different weight initialisations.

local minimum solutions, in order to investigate the ability of the MCMC algorithm to escape poor local modes, under the assumptions of noninformative uniform, hierarchical and ARD prior distributions.

Simulated annealing and the effect of fixing $\sigma^2_{\mathbf{y}} > \hat{\sigma}^2_{\mathbf{y}}$ and $\sigma^2_{\mathbf{w}} = 1$ for a period $t_{\sigma^2_0}$ were also investigated using these weight initialisations. For data set II, the residual variance $\hat{\sigma}^2_{\mathbf{y}}$ calculated at the local maximum likelihood (based on scaled input and output data) was approximately equal to 0.45, corresponding to a log likelihood value of approximately -727.5. For data set III, $\hat{\sigma}^2_{\mathbf{y}}$ was approximately equal to 0.18 when calculated at the locally maximum log likelihood value of -680.6. Values of 0.5, 1.0, 2.0 and 3.0 were therefore considered for $\sigma^2_{\mathbf{y},0}$, as these values all produce flatter, wider likelihood functions in each case. An additional value of $\sigma^2_{\mathbf{y},0} = 0.3$ was considered for data set III, since the value of $\hat{\sigma}^2_{\mathbf{y}}$ obtained was somewhat smaller than that obtained for data set II. To prevent the covariance of the AM proposal distribution, calculated at iteration $t_0 + 1$, from becoming overly large as a result of the altered posterior distribution, it was considered that $t_{\sigma^2_0}$ should be less than t_0. However, $t_{\sigma^2_0}$ also needs to be long enough such that the chains have a chance to move away from $\mathbf{w}_0$. Therefore, $t_{\sigma^2_0}$ was set equal to half t_0 (i.e. 500). The results obtained by fixing the hyperparameters were compared to the results obtained when the hyperparameters were updated immediately after the first weight updates, rather than being fixed for any period of time. In this case, the values of $\sigma^2_{\mathbf{y},0}$ and $\sigma^2_{\mathbf{w},0}$ were set equal to the hyperprior parameters $S_{\mathbf{y}}$ and $S_{\mathbf{w}}$, respectively. However, these values have little consequence, since they are updated straight away.

The success of simulated annealing is vitally dependent upon the schedule of temperatures used. A commonly used and simple schedule is the geometric annealing schedule given by $T_{t+1} = (1 - \varphi)T_t$, where the initial temperature T_0 is set high enough so that the initial rejection rate is very low and φ is chosen such that the temperature is reduced slightly after each iteration (*Neal*, 1993). In this research, appropriate values for φ and T_0 were found by trial-and-error using the values of $\varphi = 5.0 \times 10^{-5}$ and $T_0 = 15$ adopted by *Neal* (1992) to guide the range of values considered, which included $T_0 = 5, 15, 30, 50$ and $\varphi = 1.0 \times 10^{-3}, 1.0 \times 10^{-4}, 5.0 \times 10^{-5}$. By reducing φ by a factor of 10, the number of iterations required for the temperature to cool to one is increased by a factor of 10, whereas a reduction in φ by a factor of 2, means that the number of iterations required is approximately doubled. Values of φ less than 5.0×10^{-5} were not considered, as the number of iterations required for the temperature to cool to one was too great (e.g. $> 390,000$ for $T_0 = 50$, $\varphi = 1.0 \times 10^{-5}$). To maintain an approximately constant acceptance rate, *Neal* (1993) recommends scaling the width of the proposal distribution by $T_t^{1/2}$; however, this recommendation is based on the use of a constant proposal distribution, unlike the adaptive proposal used by the AM algorithm. At high temperatures, the width of the adaptive proposal will automatically become wider due to the acceptance of a wider range of weight vectors. Therefore, in this research, the proposal distribution was scaled by $(T_t/T_0)^{1/2}$, in order to maintain adequate acceptance rates as the temperature cooled. *Neal* (1993) also suggests that the prior should be exempt from the effect of temperature, as the prior beliefs should guide the search for the weight values even in the initial stages of the MCMC algorithm and should not be scaled down. This was achieved by modifying (4.16) as follows:

$$\alpha(\mathbf{w}^*|\mathbf{w}_{t-1}) = \min\Big\{ \quad \exp\left[\log L(\mathbf{w}^*) - \log L(\mathbf{w}_{t-1})\right]/T \\ + \exp\left[\log p(\mathbf{w}^*) - \log p(\mathbf{w}_{t-1})\right], 1\Big\} \tag{4.26}$$

4.4.1.3 Prevention of Overfitting

The primary advantage of the hierarchical and ARD priors over the noninformative uniform prior described in Section 4.2.3.2 is the automatic incorporation of weight regularisation and the prevention of overfitting. As the ARD prior was proposed as a method for automatically determining the relevance of ANN inputs, it was originally considered that the ARD prior had this additional advantage over the other forms of prior distribution considered in this research. However, it was shown by *Lampinen and Vehtari* (2001) that the nonlinearity of an input has the largest effect on the relevance score of the ARD and that inputs with a large, but linear, effect on the output are given a low relevance measure. This is because when an ANN models a linear relationship, the input-hidden layer weights are small, such that they lie on the linear part of the hidden layer sigmoidal activations, and the strength of the relation is determined by scaling the hidden-output weights. Therefore, the input-hidden weights do not measure a linear input-output relationship regardless of its importance. This means that ARD is not appropriate for selecting, removing or ranking ANN inputs according to the ARD relevance measures. Nevertheless, the ARD prior is the most correct form of prior of those considered, as it does not assume that the input weight groups all have the same variance, which was shown by *Lampinen and Vehtari* (2001) to correspond to nonlinearity.

As the hierarchical prior distribution results in a more complex MCMC training algorithm than when the uniform prior is assumed, and as the ARD prior results in a further increase in complexity above that, the different forms of prior were assessed to determine whether or not the additional complexity is warranted given their ability to prevent overfitting. To do this, the MCMC algorithm was initialised with the optimised weights obtained from networks identified as overtrained in the investigations carried out in Chapter 3. Similar to the local modes case discussed in Section 4.4.1.2 above, obvious examples of overtraining were only observed for two of the three synthetic data sets. It was found that overtraining did not occur when ANNs were trained to model data set III, which is likely due to the fact that this data set had the greatest signal-to-noise ratio and was also the longest of the three data sets considered (see Section 3.4.1). For data sets I and II, the most obvious examples of overfitting were observed when 10 hidden node ANNs were trained to fit these data sets. This can be seen in Figures 4.8 and 4.9, respectively, which show trace plots of the training and test set SSE values obtained when data sets I and II were modelled with (a) the corresponding optimal ANN structures and (b) 10 hidden node ANNs. As can be seen, the final training set SSE values obtained using the 10 hidden node ANNs were somewhat lower than those obtained using the optimal network

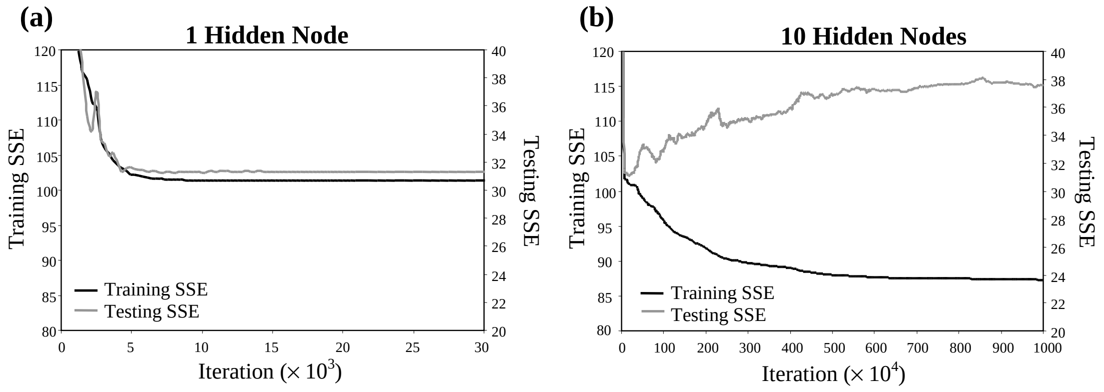

Figure 4.8 Training and test set SSE traces obtained when (a) a 1 hidden node ANN (optimal structure) and (b) a 10 hidden node ANN (overfitted) were trained to fit data set I.

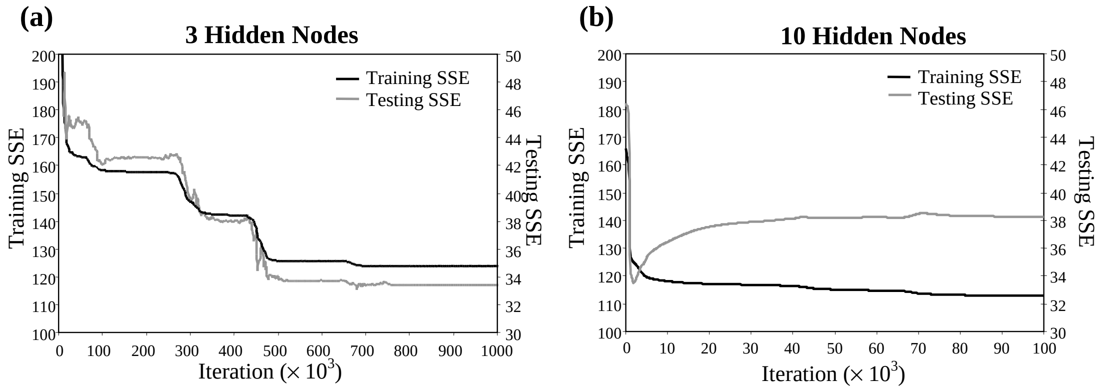

Figure 4.9 Training and test set SSE traces obtained when (a) a 3 hidden node ANN (optimal structure) and (b) a 10 hidden node ANN (overfitted) were trained to fit data set II.

sizes (i.e. a 1 hidden node ANN for data set I and a 3 hidden node ANN for data set II); however, the corresponding test set SSE values were significantly higher than those obtained using the optimal network sizes. In this investigation, the MCMC algorithm was initialised with two different sets of weights obtained when 10 hidden node ANNs were trained to fit data sets I and II: (1) the weights obtained when training was stopped early before overfitting of the data had begun; and (2) the weights obtained when the training algorithm was allowed to run to convergence. The first set of weights was used to assess the abilities of the prior distributions to prevent overfitting, given a set of weights where overfitting could, but has not yet, occurred, while the second set of weights was used to assess the ability of the MCMC algorithm to find a more generalised fit to the data, given a set of weights that result in an overfitted model, under the assumptions of the different prior distributions.

To determine whether the 10 hidden node models trained using the MCMC algorithm were overfitting the data or not, the likelihood values were compared to those obtained for the optimal network sizes, trained with the MCMC algorithm using a uniform prior distribution (since overtraining was not expected for these networks). If the likelihood values for the 10 hidden node models were less than those for the optimal ANN structures, it was considered that overfitting was occurring to some extent.

4.4.2 Bayesian Model Selection

The MCMC algorithm was used to train 10 networks containing between 1 and 10 hidden nodes, applied to data sets I, II and III, given the optimal configuration of the algorithm determined by the investigations described in Section 4.4.1. For each network size, the MCMC algorithm was initialised with the optimised weights obtained when the SCE-UA algorithm was used for training and early stopping was employed to prevent overfitting. Each of the evidence estimators discussed in Section 4.3.4 was used to evaluate the evidence of each model, from which Bayes factors were then calculated and used to rank the models in order of posterior probability. The models' rankings were then compared to the knowledge that a 1 hidden node network is best for data set I, a 3 hidden node network is most suitable for data set II and a 5-6 hidden node network is optimal for modelling data set III (see Section 3.4.5.2), in order to determine which estimator, if any, is most appropriate for BMS within the proposed framework. Inspection of marginal posterior distributions was carried out, as described in Section 4.3.4.2, to check or confirm the results of the Bayes factor comparisons.

4.4.3 Results

4.4.3.1 Fine-Tuning of Bayesian Training Algorithm

Detecting convergence

The trace plots in Figure 4.10 show the mean values of the $\log p^*(\mathbf{w}|\mathbf{y})$, the $\log L(\mathbf{w})$ and the $\log p(\mathbf{w})$ densities, calculated by taking the average of the four parallel chains, over the 400,000 iterations of the MCMC simulation for (a) the 2 hidden node ANN, (b) the 6 hidden node ANN and (c) the 10 hidden node ANN applied to data set II. As can be seen, these traces all become approximately flat roughly within 100,000 iterations. However, while it may appear as though the algorithm has converged, it is difficult to trust a single trace (even if it is the average of a number of chains) for each of these MCMC quantities. In Figure 4.11, traces of the $\log p^*(\mathbf{w}|\mathbf{y})$ values for the individual chains are

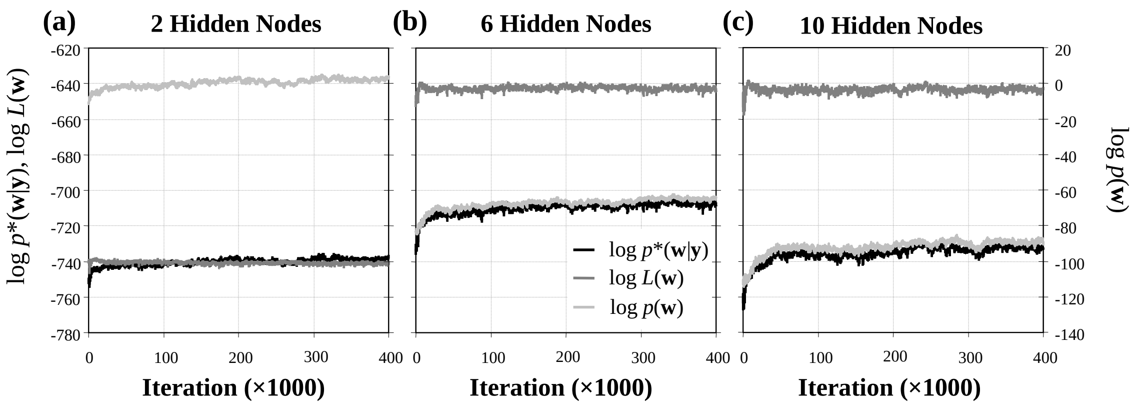

Figure 4.10 Mean $\log p^*(\mathbf{w}|\mathbf{y})$, $\log L(\mathbf{w})$ and $\log p(\mathbf{w})$ traces for the $2, 6, \ldots, 10$ hidden node ANNs applied to data set II.

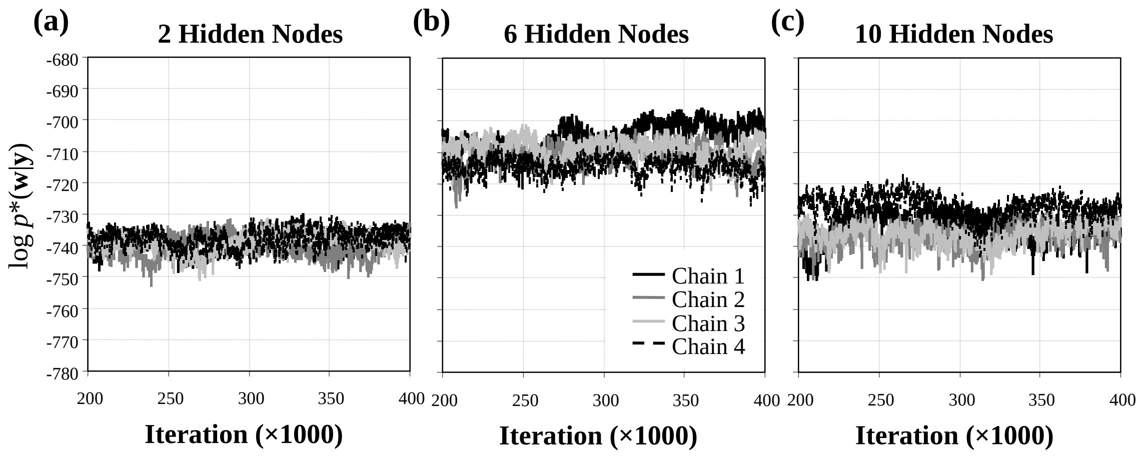

Figure 4.11 Log $p^*(\mathbf{w}|\mathbf{y})$ traces obtained from the 4 parallel MCMC chains for the $2, 6, \ldots, 10$ hidden node ANNs applied to data set II.

shown over the final 200,000 iterations of the MCMC algorithm. It can be seen that these plots give a better picture of whether or not the chains are sampling from the same distribution. For the 2 hidden node network, it is evident that the chains were mixing well, indicating that the algorithm had converged (at least to a stationary distribution, if not to the true posterior) within the first 200,000 iterations of the simulation. The $\log p^*(\mathbf{w}|\mathbf{y})$ traces for the 6 and 10 hidden node ANNs, on the other hand, are a little more spread out. For the 6 hidden node ANN, it appears as though one of the chains is still increasing, indicating that the algorithm had not properly converged within 400,000 iterations, while it is apparent that the chains simulated for the 10 hidden node ANN were not mixing properly until around 300,000 iterations. However, as the chains were not obviously different from one another, nor were they obviously increasing or decreasing for any of the 2-, 6- or 10 hidden node ANNs, it was considered that 400,000 iterations is a good rough guide for the synthetic test cases considered in this research as to the number of iterations required, firstly, to achieve convergence and, secondly, to sample an appropriate number of draws from the stationary distribution. Therefore, 400,000 iterations were used for the remainder of the investigations presented in this section, as these investigations were exploratory of the performance of the MCMC algorithm, but the results did not depend on convergence. However, for the BMS investigations, of which the results are presented in Section 4.4.3.2, more care was taken to achieve convergence, as these results were dependent on convergence of the MCMC algorithm.

As well as being useful for detecting convergence, or failure thereof, the plots shown in Figures 4.10 and 4.11 also provide interesting information about the models. It can be seen that the log prior probability of the weights of the 2 hidden node ANN is significantly higher than the corresponding log prior densities for the 6 and 10 hidden node ANNs. This is due to the use of the hierarchical prior distribution, which incorporates weight regularisation and favours fewer weights. It can also be seen that the log likelihood values for the 2 hidden node ANN are significantly lower than those of the 6 and 10 hidden node ANNs, which have approximately equal log likelihood values. This indicates that the 2 hidden node network provides a significantly worse fit to the data than either the 6 or 10 hidden node models. Overall, the weights of the 10 hidden node ANN, with high likelihood, but low prior, have approximately the same log unnormalised posterior density as the 2 hidden node ANN weights, which have a high prior probability but low likelihood. The weights of the 6 hidden node ANN have the highest overall unnormalised posterior density, which provides useful information about the level of complexity required for modelling data set II, since the evidence of a model is also proportional to the unnormalised posterior density.

Escaping local modes

Shown in Figures 4.12 and 4.13 are the mean $\log p^*(\mathbf{w}|\mathbf{y})$, $\log L(\mathbf{w})$ and $\log p(\mathbf{w})$ traces resulting from each form of prior distribution investigated when the MCMC algorithm was applied to train the 3 and 4 hidden node ANNs used for modelling data sets II and III, respectively, with poor (local minimum) weight initialisations, and when the hyperparameters were updated immediately (i.e. were not held constant for any period). For the uniform prior, the $\log p^*(\mathbf{w}|\mathbf{y})$ values correspond to the $\log L(\mathbf{w})$ values, as the value

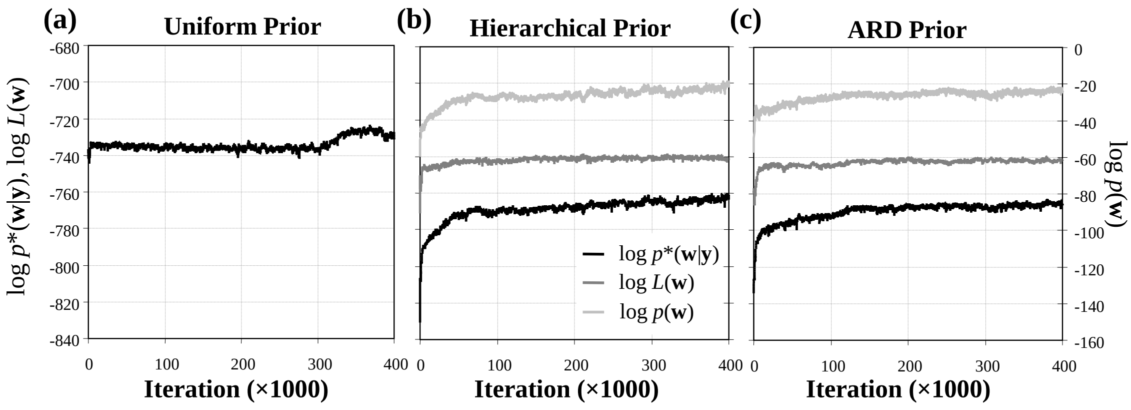

Figure 4.12 Mean $\log p^*(\mathbf{w}|\mathbf{y})$, $\log L(\mathbf{w})$ and $\log p(\mathbf{w})$ traces resulting from the (a) uniform, (b) hierarchical and (c) ARD prior distributions when the MCMC algorithm with unfixed hyperparameters was used to train the poorly initialised data set II model.

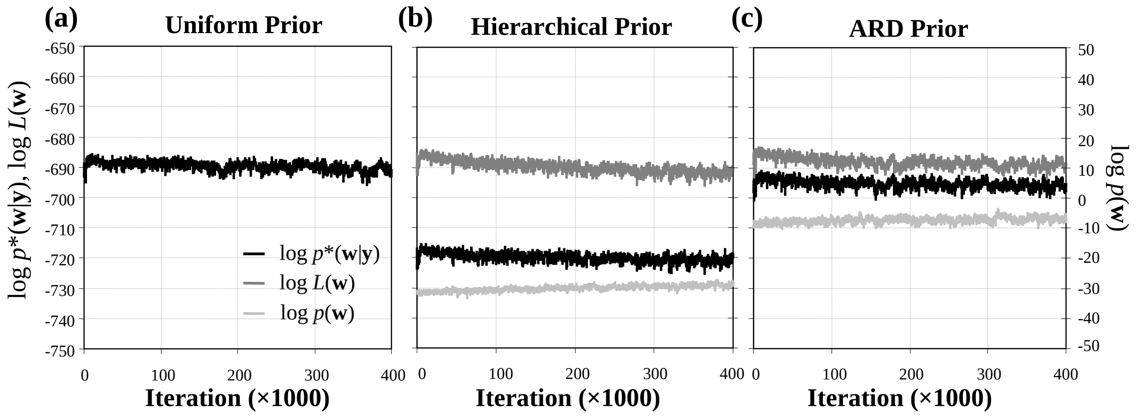

Figure 4.13 Mean $\log p^*(\mathbf{w}|\mathbf{y})$, $\log L(\mathbf{w})$ and $\log p(\mathbf{w})$ traces resulting from the (a) uniform, (b) hierarchical and (c) ARD prior distributions when the MCMC algorithm with unfixed hyperparameters was used to train the poorly initialised data set III model.

of the prior was disregarded since it was constant across the range of weights considered. It can be seen in these figures that the uniform prior resulted in the highest $\log p^*(\mathbf{w}|\mathbf{y})$ values in each case, primarily because it does not account for a negative log prior probability as the other $\log p^*(\mathbf{w}|\mathbf{y})$ values do. However, for the data set II model, the uniform prior was the only form of prior which enabled the likelihood value to obviously increase above its initial value, indicating that at least one of the simulated chains was able to move away from the initial weights. On the other hand, the log likelihood traces resulting from the hierarchical and ARD priors were flat throughout the simulation, indicating that the chains were unable to escape the local mode. The $\log p^*(\mathbf{w}|\mathbf{y})$ traces for these models did increase throughout the simulation, but as a result of the increasing log prior densities, rather than increasing likelihood values (see Figure 4.12). It is possible that these forms of priors may have constrained the weights too much in the initial stages of the simulation, making it difficult for the chains to move very freely throughout the weight space. For the data set III model, it can be seen in Figure 4.13 that none of the prior distributions assumed resulted in increasing log likelihood traces, indicating that the MCMC algorithm was unable to escape the local mode regardless of the prior distribution. Traces of the $\log p^*(\mathbf{w}|\mathbf{y})$ values of the individual chains are shown in Figures 4.14 and 4.15 for the models applied to data sets II and III, respectively. It can be seen in Figure 4.14 (a) that only one chain was able to escape the local mode when the uniform prior was assumed for the data set II model, whereas, for the data set III model, the $\log p^*(\mathbf{w}|\mathbf{y})$ traces of the individual chains remained flat throughout the simulation, as shown in Figure 4.13.

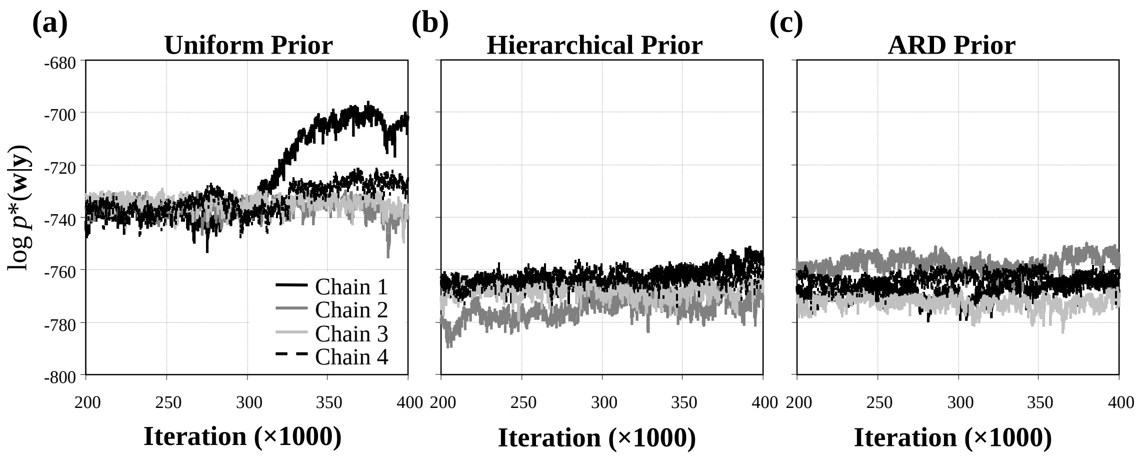

Figure 4.14 Individual $\log p^*(\mathbf{w}|\mathbf{y})$ traces resulting from the (a) uniform, (b) hierarchical and (c) ARD prior distributions when the MCMC algorithm with unfixed hyperparameters was used to train the poorly initialised data set II model.

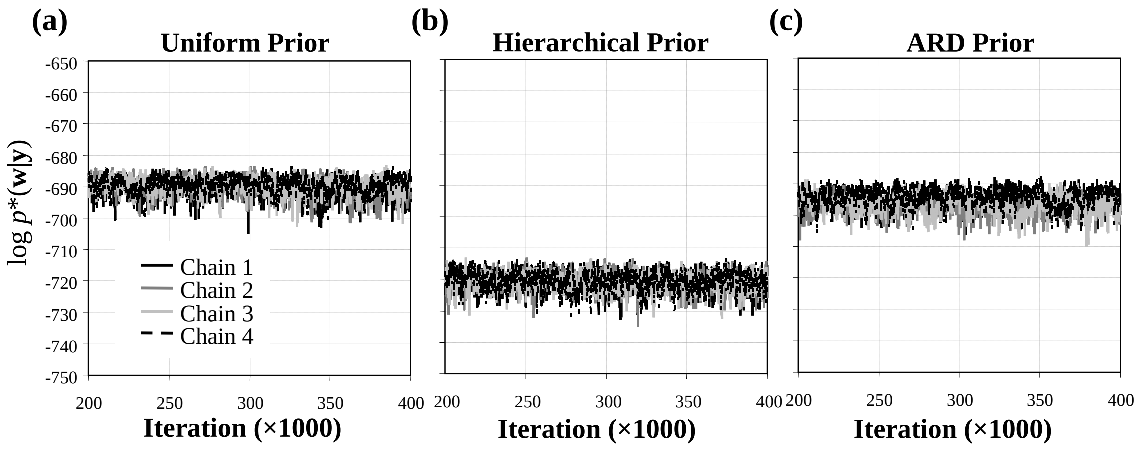

Figure 4.15 Individual $\log p^*(\mathbf{w}|\mathbf{y})$ traces resulting from the (a) uniform, (b) hierarchical and (c) ARD prior distributions when the MCMC algorithm with unfixed hyperparameters was used to train the poorly initialised data set III model.

For the data set II model, the log likelihood value at $\hat{\mathbf{w}}$ was reduced to -732.1 (0.6%), -817.2 (11%), -984.9 (26%) and -1103.3 (34%) by fixing $\sigma^2_{\mathbf{y}}$ equal to 0.5, 1.0, 2.0 and 3.0, respectively. Overall, it was found that the best results were obtained by fixing $\sigma^2_{\mathbf{y}} = 1.0$, indicating that the log likelihood need not be reduced by a large amount for the chains to move more freely in the vicinity of the local mode. Traces of the resulting mean $\log p^*(\mathbf{w}|\mathbf{y})$, $\log L(\mathbf{w})$ and $\log p(\mathbf{w})$ values when $\sigma^2_{\mathbf{y}}$ was fixed equal to 1.0 for $t_{\sigma^2_0} = 500$ are shown in Figure 4.16. As can be seen, the $\log L(\mathbf{w})$ traces resulting from the assumption of hierarchical and ARD priors significantly increased above the initial value, whereas this trace remained relatively flat for the uniform prior. These results show that, by fixing the hyperparameters of a regularisation type prior (i.e. hierarchical or ARD) for a short initial period, the chains are better able to move more freely around the search space, giving them a better ability to escape poor weight initialisations and discover new modes. Shown in Figure 4.17 are the $\log p^*(\mathbf{w}|\mathbf{y})$ traces of the individual chains, where it can be seen that the MCMC simulations using uniform and hierarchical priors had both converged to a stationary distribution (although about different modes), whereas the chains were still moving between modes at the end of the simulation when the ARD prior was used. These plots highlight that the stepped nature of the mean traces shown in Figure 4.16 was due to different chains finding new modes at different times. Overall, it can be seen that when the hierarchical prior was used, the chains were most successful at exploring the weight space, obtaining a mean $\log L(\mathbf{w})$ value of approximately -695.0 and converging to a stationary distribution within 200,000 iterations.

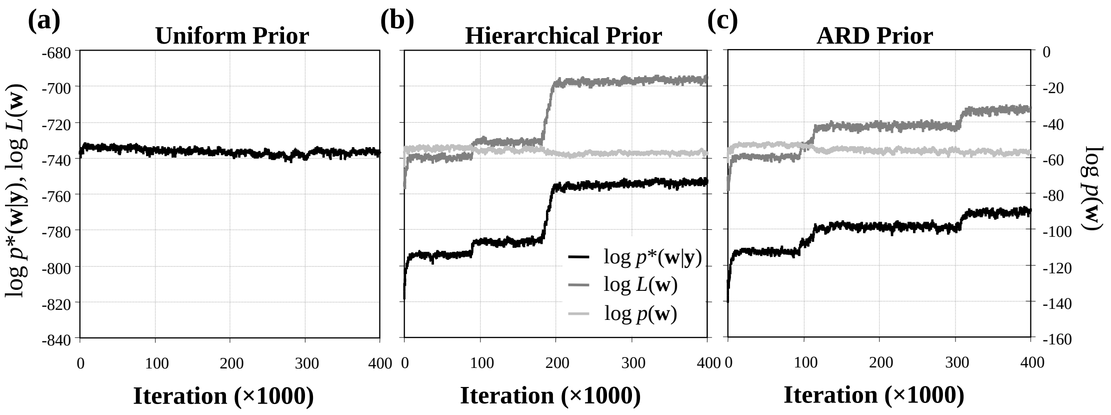

Figure 4.16 Mean $\log p^*(\mathbf{w}|\mathbf{y})$, $\log L(\mathbf{w})$ and $\log p(\mathbf{w})$ traces resulting from the (a) uniform, (b) hierarchical and (c) ARD prior distributions when the MCMC algorithm with initially fixed hyperparameters was used to train the poorly initialised data set II model.

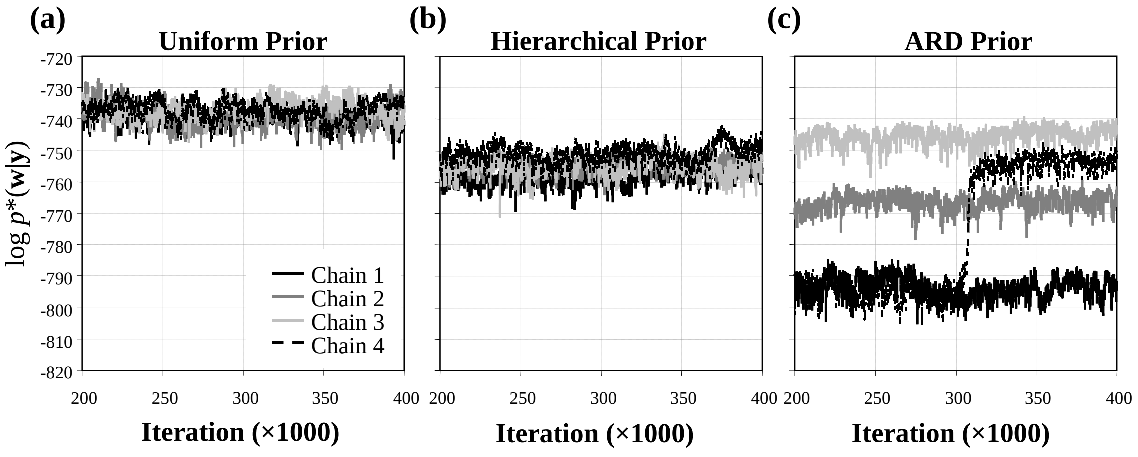

Figure 4.17 Individual $\log p^*(\mathbf{w}|\mathbf{y})$ traces resulting from the (a) uniform, (b) hierarchical and (c) ARD prior distributions when the MCMC algorithm with initially fixed hyperparameters was used to train the poorly initialised data set II model.

By fixing $\sigma^2_{\mathbf{y}}$ equal to 0.3, 0.5, 1.0, 2.0 and 3.0, the log likelihood value at $\hat{\mathbf{w}}$ for poorly initialised data set III model was reduced to -748.6 (9%), -903.5 (25%), -1225.6 (44%), -1592.1 (57%) and -1820.3 (63%), respectively. It was found that the best results were obtained when $\sigma^2_{\mathbf{y}}$ was fixed equal to 0.3, which resulted in a similar reduction in the log likelihood value as the best results obtained for data set II (approximately 10%). Traces of the resulting mean $\log p^*(\mathbf{w}|\mathbf{y})$, $\log L(\mathbf{w})$ and $\log p(\mathbf{w})$ values are shown in Figure 4.18, while the $\log p^*(\mathbf{w}|\mathbf{y})$ resulting from the individual MCMC chains are shown in Figure 4.19. As can be seen in Figure 4.18, the hierarchical prior distribution resulted in the greatest increase in $\log L(\mathbf{w})$ value above the initial value, indicating that, again, the MCMC algorithm was best able to escape the poor local mode when this form of prior was assumed. However, it can also be seen in Figure 4.19 that only one of the chains was able to move away from the initial weights and discover a better mode and that, by the end of the simulation, one of the chains resulting from the assumption of a uniform prior was also able to do this. Therefore, the MCMC simulation was run for another 400,000 iterations to determine whether the MCMC algorithm did, in fact, have the greatest ability to escape the poor local mode when a hierarchical prior was assumed. The mean MCMC output traces resulting from the longer simulation are shown in Figure 4.20, where it can be seen that this was indeed the case. The stepped nature of the mean traces shown in this Figure 4.16 was again found to be the result of different chains finding new modes at different times throughout the simulation.

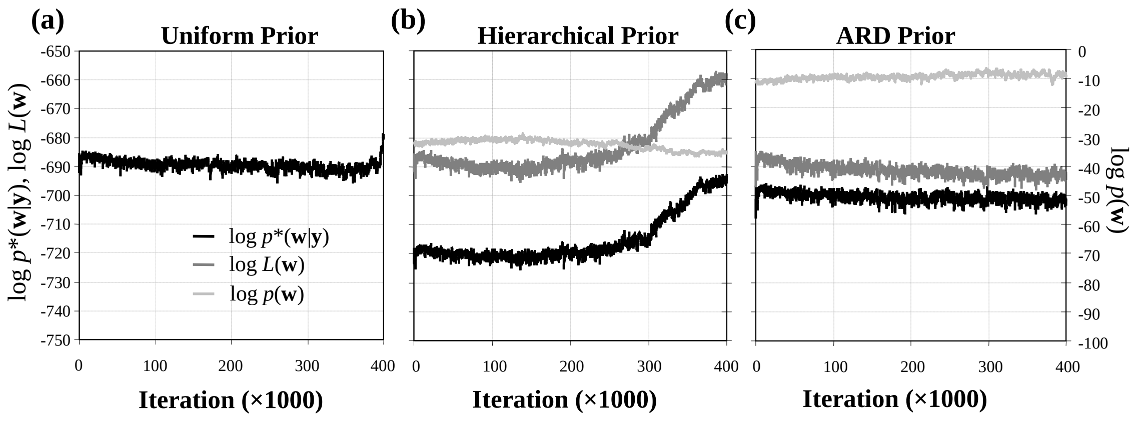

Figure 4.18 Mean $\log p^*(\mathbf{w}|\mathbf{y})$, $\log L(\mathbf{w})$ and $\log p(\mathbf{w})$ traces resulting from the (a) uniform, (b) hierarchical and (c) ARD prior distributions when the MCMC algorithm with initially fixed hyperparameters was used to train the poorly initialised data set III model.

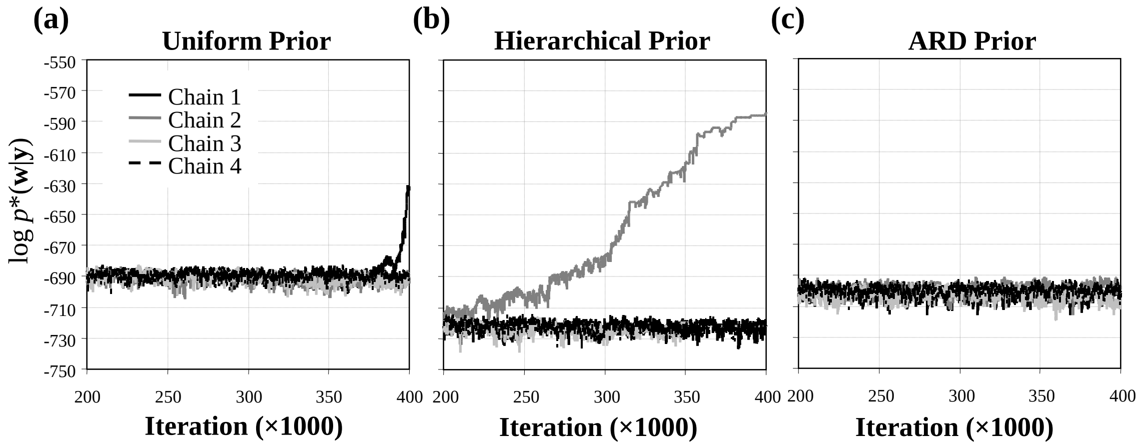

Figure 4.19 Individual $\log p^*(\mathbf{w}|\mathbf{y})$ traces resulting from the (a) uniform, (b) hierarchical and (c) ARD prior distributions when the MCMC algorithm with initially fixed hyperparameters was used to train the poorly initialised data set III model.

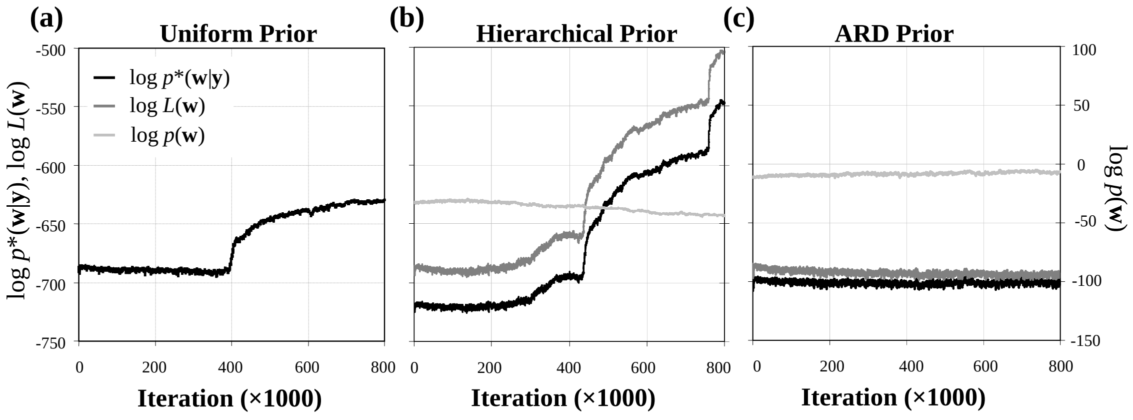

Figure 4.20 Mean $\log p^*(\mathbf{w}|\mathbf{y})$, $\log L(\mathbf{w})$ and $\log p(\mathbf{w})$ traces obtained for the data set III model with a longer simulation.

For both the data set II and data set III models, it was found that the standard deviations of the weights, calculated immediately after the initial period $t_{\sigma_0^2}$ for which the hyperparameters of the hierarchical prior had been fixed, were greater than those calculated when they were unfixed, confirming that the chains were able to move more freely around the weight space when the hyperparameters were fixed for a short time. However, although the MCMC algorithm, given the combination of a hierarchical prior and fixed hyperparameters, had some success in finding new modes, it was unable to find the maximum likelihood values obtained with the deterministic training algorithms. The MCMC algorithm was also initialised with the best weights obtained for the 3 hidden node ANN applied to data set II and the 4 hidden node ANN applied to data set III using the SCE-UA algorithm. Traces of the resulting mean $\log p^*(\mathbf{w}|\mathbf{y})$, $\log L(\mathbf{w})$ and $\log p(\mathbf{w})$ values are shown in comparison to the best results obtained for the poor weight initialisation in Figures 4.21 and 4.22, respectively.

In the attempt to improve the results obtained with a hierarchical prior and initially fixed hyperparameters, simulated annealing was also employed. Several general observations were made regarding the simulated annealing parameters for the models developed for data sets II and III. Overall, it was found that the best combination of simulated annealing parameters was $\varphi = 5.0 \times 10^{-5}$ and $T_0 = 15$, which were also the values used by *Neal* (1992). For initial temperatures of $T_0 = 5$ and $T_0 = 30$, the best results were obtained when $\varphi = 5.0 \times 10^{-5}$, indicating that simulated annealing was most successful when the temperature was cooled slowly. However, simulated annealing was found to be least successful when $T_0 = 50$, regardless of the value of φ. For $T_0 = 5$, the best

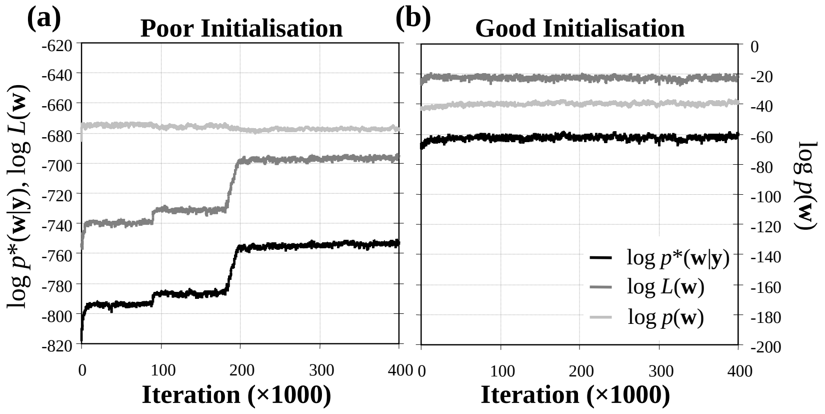

Figure 4.21 Mean $\log p^*(\mathbf{w}|\mathbf{y})$, $\log L(\mathbf{w})$ and $\log p(\mathbf{w})$ traces obtained for the data set II model with (a) a poor weight initialisation and (b) a good weight initialisation.

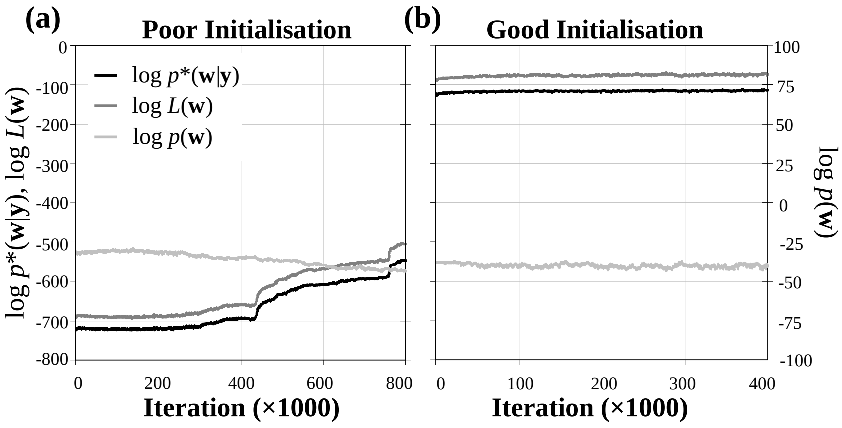

Figure 4.22 Mean $\log p^*(\mathbf{w}|\mathbf{y})$, $\log L(\mathbf{w})$ and $\log p(\mathbf{w})$ traces obtained for the data set III model with (a) a poor weight initialisation and (b) a good weight initialisation.

results obtained with $\varphi = 5.0 \times 10^{-5}$ were similar, although slightly worse than when no annealing was applied. It was found that the steps in the mean $\log p^*(\mathbf{w}|\mathbf{y})$ trace became smoother for the data set II model, whereas, for the data set III model, the transition between modes became less smooth than when no annealing was applied. The simulated annealing algorithm was somewhat successful with $T_0 = 30$ when applied to the data set II model, although it did not give as good results as when $T_0 = 15$, as the simulation had yet to converge within 400,000 iterations. On the other hand, the results obtained when $T_0 = 30$ for the data set III model were significantly worse than when no annealing was applied. Traces of the mean $\log p^*(\mathbf{w}|\mathbf{y})$, $\log L(\mathbf{w})$ and $\log p(\mathbf{w})$ values obtained with the best combination of simulated annealing parameters (i.e. $\varphi = 5.0 \times 10^{-5}$ and $T_0 = 15$) are shown in Figure 4.23 for (a) the data set II model and (b) the data set III model. In comparison to Figure 4.16 (b), it can be seen that simulated annealing resulted in a smoother transition between different modes when applied to the data set II model, whereas, for the data set III model, slightly faster location of new modes was achieved, as seen in comparison to Figures 4.18 (b) and 4.20 (b). However, overall, the best results obtained with simulated annealing were approximately the same as when annealing was not employed. The MCMC algorithm was still unable unable to find the maximum likelihood values obtained with the deterministic training algorithms, as shown in Figures 4.21 (b) and 4.22 (b).

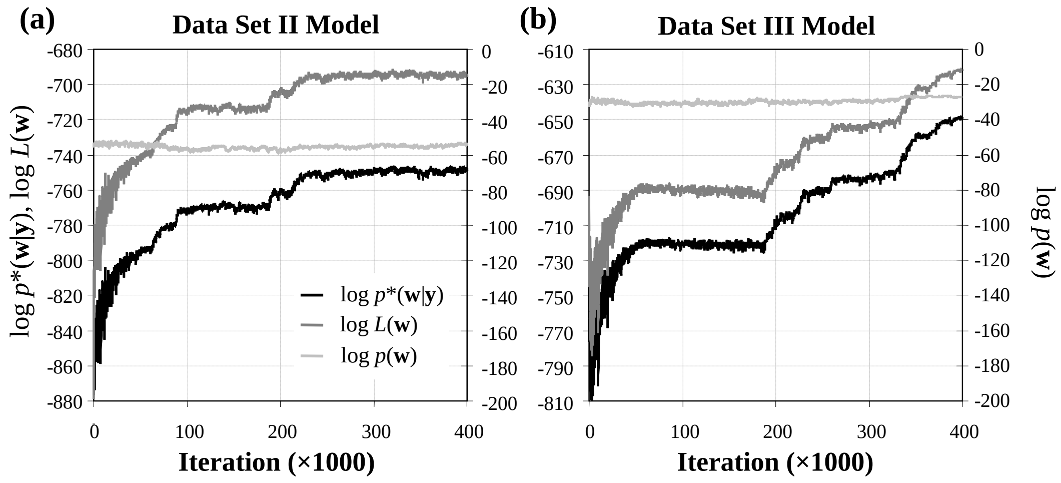

Figure 4.23 Mean $\log p^*(\mathbf{w}|\mathbf{y})$, $\log L(\mathbf{w})$ and $\log p(\mathbf{w})$ traces obtained using simulated annealing with $\varphi = 5.0 \times 10^{-5}$ and $T_0 = 15$ for (a) the data set II model and (b) the data set III model.

Prevention of overfitting

Shown in Figures 4.24 and 4.25 are the mean $\log p^*(\mathbf{w}|\mathbf{y})$, $\log L(\mathbf{w})$ and $\log p(\mathbf{w})$ traces resulting from each form of prior distribution when the MCMC algorithm, initialised with weights obtained when the SCE-UA training algorithm was stopped early, was applied to the 10 hidden node ANNs used for modelling data sets I and II, respectively. Also shown for comparison in these figures are the mean $\log L(\mathbf{w})$ traces obtained using the optimal ANN structures (i.e. 1 hidden node ANN for data set I and 3 hidden node ANN for data set II). Given the results presented in the previous section, the hyperparameters were fixed at $\sigma_{\mathbf{y}}^2 = 1.0$, $\sigma_{\mathbf{w}}^2 = 1.0$ for $t_{\sigma_0^2} = 500$. It can be seen by comparing the $\log L(\mathbf{w})$ traces with those obtained using the optimal ANN structures that, although the 10 hidden node models had the potential of becoming overtrained, overfitting did not begin during any time in the MCMC simulation using any of the prior distributions, including the noninformative uniform prior, which does not incorporate weight regularisation. It can also be seen that, for both the data set I and data set II models, the $\log p^*(\mathbf{w}|\mathbf{y})$

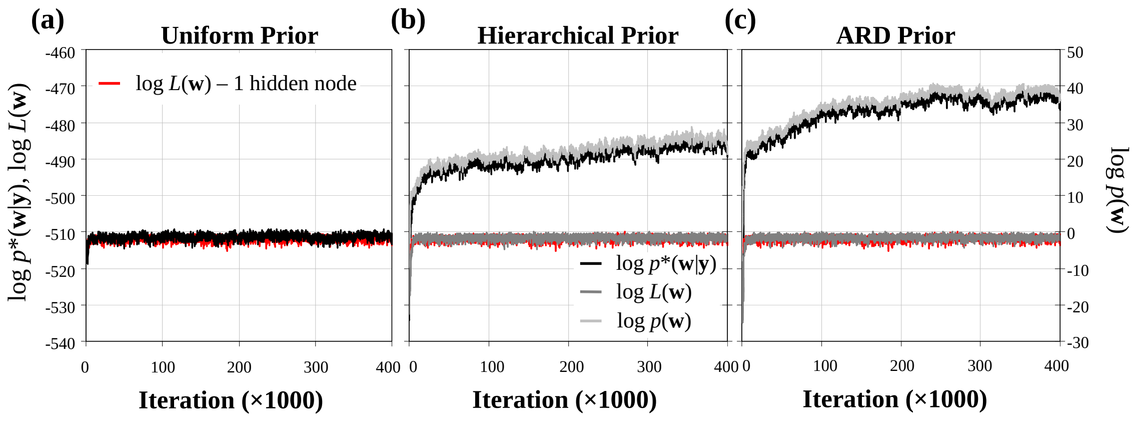

Figure 4.24 Mean $\log p^*(\mathbf{w}|\mathbf{y})$, $\log L(\mathbf{w})$ and $\log p(\mathbf{w})$ traces resulting from the (a) uniform, (b) hierarchical and (c) ARD prior distributions when the MCMC algorithm was used to train the 10 hidden node data set I model, initialised with weights obtained when overfitting was prevented.

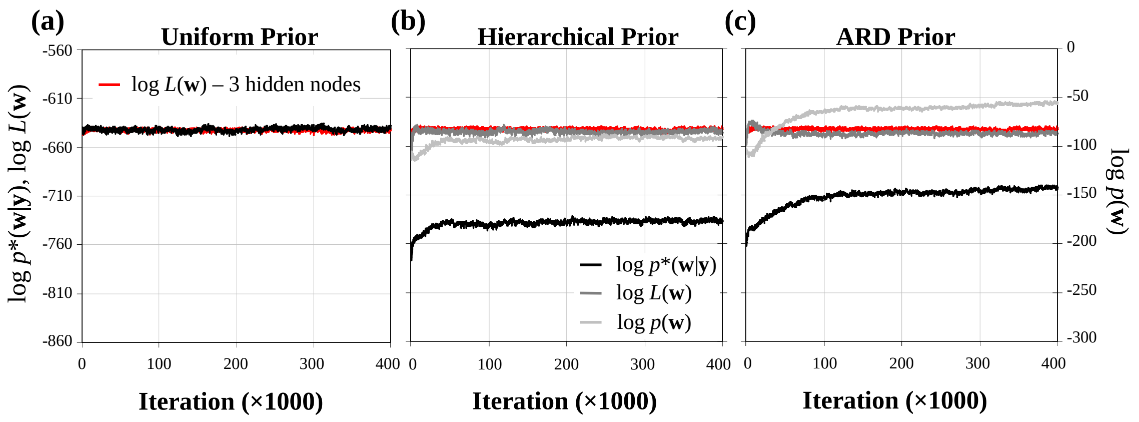

Figure 4.25 Mean $\log p^*(\mathbf{w}|\mathbf{y})$, $\log L(\mathbf{w})$ and $\log p(\mathbf{w})$ traces resulting from the (a) uniform, (b) hierarchical and (c) ARD prior distributions when the MCMC algorithm was used to train the 10 hidden node data set II model, initialised with weights obtained when overfitting was prevented.

traces were continually increasing throughout the simulations when hierarchical and ARD priors were assumed, as a result of this regularisation.

Figures 4.26 and 4.27 display the mean $\log p^*(\mathbf{w}|\mathbf{y})$, $\log L(\mathbf{w})$ and $\log p(\mathbf{w})$ traces resulting from each form of prior distribution when the MCMC algorithm was initialised with weights obtained when the SCE-UA training algorithm was run until convergence and applied to the data set I and data set II models, respectively. It can be seen that the $\log L(\mathbf{w})$ traces obtained using the 10 hidden node models started off significantly higher than those obtained using the optimal ANNs, but came back towards the optimal models' traces during the simulations. For the data set I model (Figure 4.26), this happened quickly within 9,000 and 32,000 iterations, using the hierarchical and ARD priors, respectively. For the data set II model (Figure 4.27), convergence to the optimal $\log L(\mathbf{w})$ traces was somewhat slower; however, in each case the results indicate that hierarchical and ARD priors are able to prevent overfitting, even when initialised with weights where overfitting had already occurred. On the other hand, although the $\log L(\mathbf{w})$ traces obtained using the uniform prior distribution also decreased slightly throughout the simulations, this process was slow, such that by the end of the simulation, the model was still overfitting the data, indicating that the uniform prior is not appropriate for preventing overfitting.

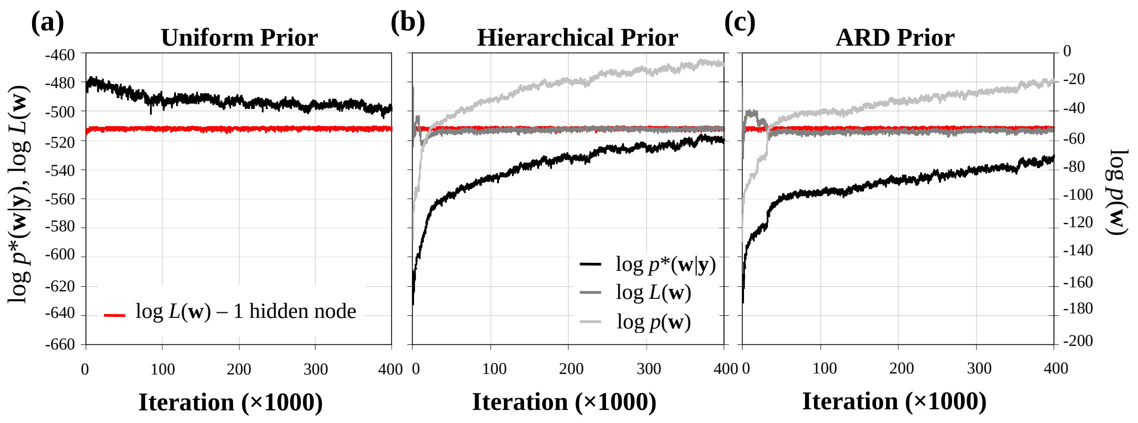

Figure 4.26 Mean $\log p^*(\mathbf{w}|\mathbf{y})$, $\log L(\mathbf{w})$ and $\log p(\mathbf{w})$ traces resulting from the (a) uniform, (b) hierarchical and (c) ARD prior distributions when the MCMC algorithm was used to train the 10 hidden node data set I model, initialised with weights obtained when the model had been overtrained.

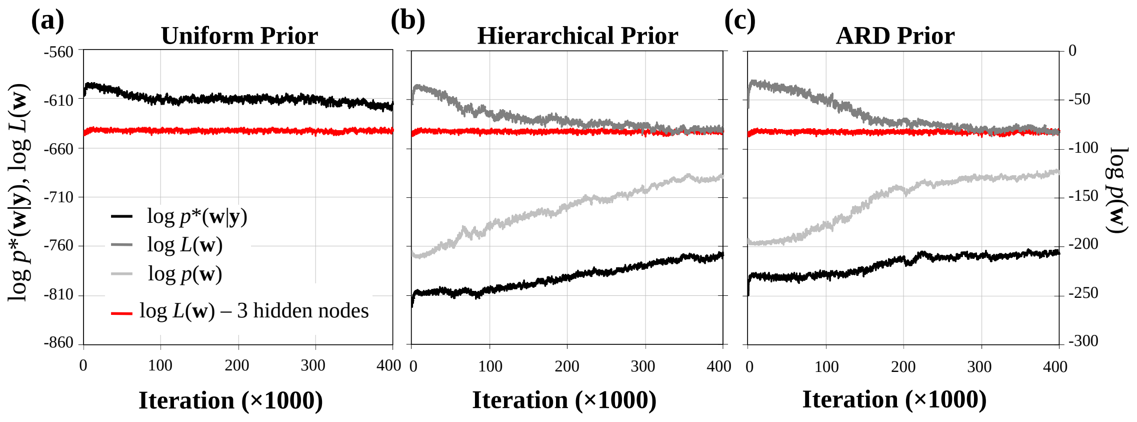

Figure 4.27 Mean $\log p^*(\mathbf{w}|\mathbf{y})$, $\log L(\mathbf{w})$ and $\log p(\mathbf{w})$ traces resulting from the (a) uniform, (b) hierarchical and (c) ARD prior distributions when the MCMC algorithm was used to train the 10 hidden node data set II model, initialised with weights obtained when the model had been overtrained.

The effect of each form of prior distribution on the magnitude of the weights was also investigated by inspecting plots of the average magnitudes of different weight groups throughout the MCMC simulations. These plots are shown in Figures 4.28 and 4.29, for the models applied to data sets I and II, respectively, and were obtained over the last 200,000 iterations of the MCMC simulations initialised with weights obtained when the SCE-UA algorithm was run until convergence. As expected, it was found that the magnitudes of the weights were not suppressed at all using the noninformative uniform prior, whereas the weights became significantly smaller when the hierarchical and ARD prior distributions were used. For the data set I model, the input-hidden layer weights had average magnitudes close to zero when hierarchical and ARD prior distributions were assumed. This is appropriate, since these inputs are all linear and, therefore, their input-hidden weights need to be small in order to lie on the linear part of the tanh hidden layer activation functions. However, it is important to note that, although the weights associated with these inputs lie close to zero with little variance, the inputs are not irrelevant to the model; thus, highlighting the shortcomings of ARD as an input selection or importance measure when the problem involves linear inputs. For the data set I model, the variance hyperparameters associated with the input weight groups had mean values of 0.925, 0.375 and 0.271 for inputs y_{t-1}, y_{t-4} and y_{t-9}, respectively, while, for the data set II model, the mean variance hyperparameters associated with inputs y_{t-1}, y_{t-4}, y_{t-9} and x_t were 6.603, 53.155, 2.357 and 31.622, respectively. These results confirmed the shortcomings of ARD, as the more important the input, the greater the value of σ^2_{w} should be. For data set

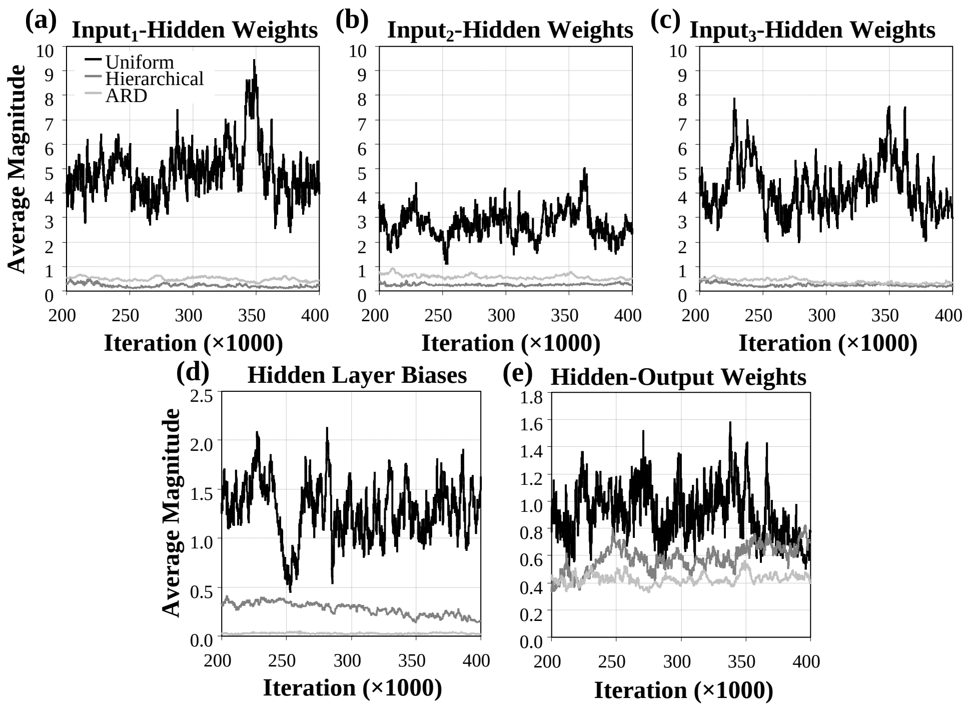

Figure 4.28 Average magnitude of the (a)-(c) input-hidden, (d) hidden layer bias and (e) hidden-output weights for the data set I model, given each form of prior.

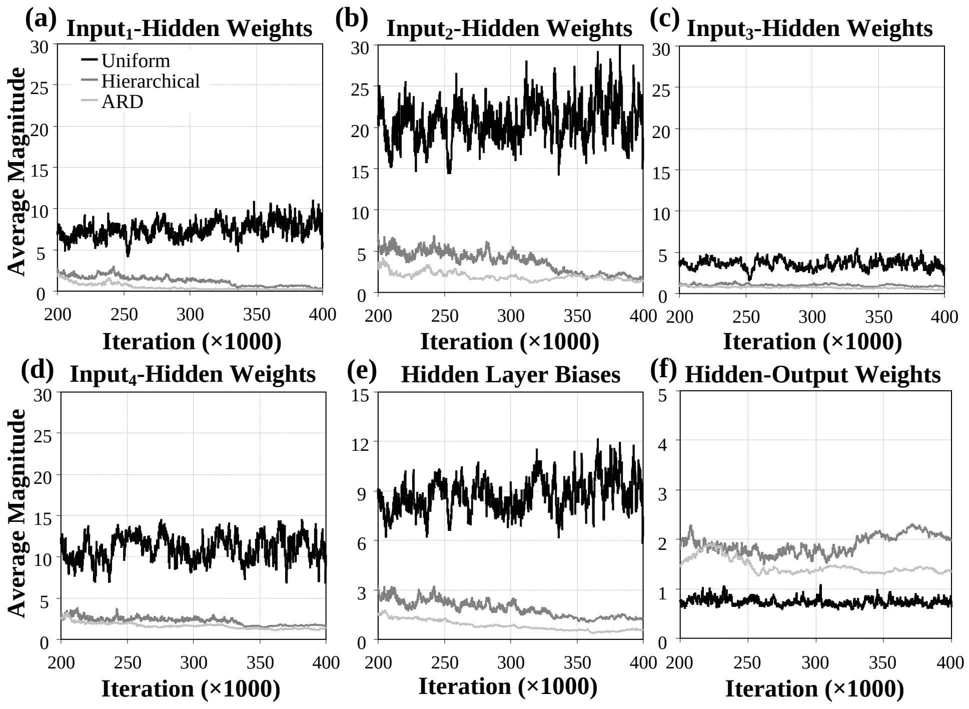

Figure 4.29 Average magnitudes of the (a)-(d) input-hidden, (e) hidden layer bias and (f) hidden-output weights for the data set II model, given each form of prior.

I, input y_{t-1} has the lowest importance, as does input x_t for data set II, which contradict the findings of the ARD.

Finally, using the mean predictions generated by each of the models developed with the MCMC algorithm, the SSE values were estimated for both the "measured" and "true" training data. The SSE values for the "true" data were used to determine the extent of overfitting, since if no overfitting had occurred, these values should be close to zero. The SSE values are given in Table 4.2, in comparison to the SSE values obtained with the optimised deterministic weights for the optimal ANN structures, the 10 hidden node ANNs when the training algorithm was stopped before overfitting had begun and when the algorithm was allowed to run until convergence. The SSE values for the optimal

Table 4.2 SSE values obtained for each model developed in the overfitting investigation.

ANN Model	"Measured" SSE	"True" SSE
DATA SET I		
Deterministic weights		
1 hidden node	*250.254*	*1.075*
10 hidden nodes - early stopped	249.647	1.960
10 hidden nodes - converged	215.657	37.184
Bayesian weights - 10 hidden nodes, early stopped initialisation		
Uniform prior	248.442	1.624
Hierarchical prior	250.377	**1.585**
ARD prior	250.237	1.636
Bayesian weights - 10 hidden nodes, converged initialisation		
Uniform prior	225.239	11.322
Hierarchical prior	250.325	**1.838**
ARD prior	250.720	2.570
DATA SET II		
Deterministic weights		
3 hidden nodes	343.447	*12.794*
10 hidden nodes - early stopped	319.786	35.298
10 hidden nodes - converged	319.210	34.817
Bayesian weights - 10 hidden nodes, early stopped initialisation		
Uniform prior	334.838	17.251
Hierarchical prior	343.488	15.438
ARD prior	346.975	**13.578**
Bayesian weights - 10 hidden nodes, converged initialisation		
Uniform prior	301.615	42.545
Hierarchical prior	322.080	24.748
ARD prior	325.377	**23.503**

ANN structures are shown in italics, as these are approximately the values expected if the models were not overtrained. The best “true” SSE values obtained using the Bayesian ANNs are highlighted by bold font and, as can be seen, the hierarchical prior was the most successful at preventing overfitting when applied to the data set I models, whether or not it had already occurred, whereas, the ARD prior was the most successful at achieving this when applied to the data set II models. However, as seen in Table 4.2, the SSE values obtained for the models trained with the MCMC algorithm under the assumptions of hierarchical and ARD prior distributions were very similar, indicating that the additional complexity of the ARD prior, over the hierarchical prior, is not warranted, given its ability to prevent overfitting.

4.4.3.2 Assessment of Evidence Estimators

Given the results of Section 4.4.3.1, the models considered in this investigation were trained with the MCMC algorithm using the hierarchical prior distribution, as this prior gave the best results in terms of prevention of overfitting, escaping local modes and minimising the complexity of the algorithm. To ensure convergence of the algorithm, the simulations were run for a total of 800,000 iterations ($t_F = 800,000$), where the first 600,000 were discarded ($t_b = 600,000$) and the final 200,000 iterations were assumed to be sampled draws from the posterior and were used in the evidence calculations.

The model evidence results for data set I, calculated using the G-D, C-J and -1/2BIC estimators, are given in Table 4.3 for the 10 different network sizes considered. It should be noted that these values were estimated based on the scaled model outputs and target data; therefore, there is a discrepancy between the mean -1/2BIC values presented here and the BIC values presented in Table 3.12, which were calculated based on unscaled data. The maximum evidence values estimated using each method are highlighted by bold italics and, as it can be seen, each of the methods correctly estimated that the 1 hidden node model had the greatest evidence. It can also be seen that each of the estimators gave similar evidence values for all of the 10 different network sizes, particularly the G-D and C-J estimators, as better seen in Figure 4.30. However, while the G-D and C-J estimates are very similar, they appear to be incorrect for some of the network sizes. It would be expected that any additional complexity over that required (in this case, 1 hidden node) would result in a reduction in the evidence. As it can be seen in Figure 4.30, this was not the case for the 6, 7 and 8 hidden node networks, which were all estimated to be more probable than the 5 hidden node model, with the 8 hidden node ANN having the greatest evidence of the three. It is apparent that the evidence of the 5 hidden node model was

Table 4.3 Evidence estimates for data set I ANN models.

Hidden Nodes	G-D Estimator	C-J Estimator	Mean -1/2BIC
1	***-528.328***	***-524.105***	***-530.787***
2	-529.071	-527.261	-547.129
3	-543.961	-544.036	-562.981
4	-559.238	-559.601	-577.994
5	-612.856	-609.773	-594.086
6	-592.344	-592.258	-610.465
7	-605.279	-601.050	-625.885
8	-580.660	-581.054	-641.425
9	-624.230	-618.985	-657.127
10	-687.408	-688.531	-674.091

underestimated, whereas the evidence of the 8 hidden node network was overestimated, resulting in an incorrect ordering of the models. The mean -1/2BIC estimates, on the other hand, indicate a more logical ordering of the models considered.

The Bayes factors calculated based on the -1/2BIC evidence estimates for the highest ranked model (Rank 1) against each other model considered (i.e. $BF_{Rank1,i}$) are presented in Table 4.4. It can be seen in this table, by comparison with Table 4.1, that the evidence in favour of a 1 hidden node ANN is very strong. While it is known that this result is correct, since the data are synthetic, in a real world study this would not be the case. Due to the difficulties associated with estimating the evidence, it is worthwhile to check the Bayes factor results against the marginal posterior weight distributions for the hidden-

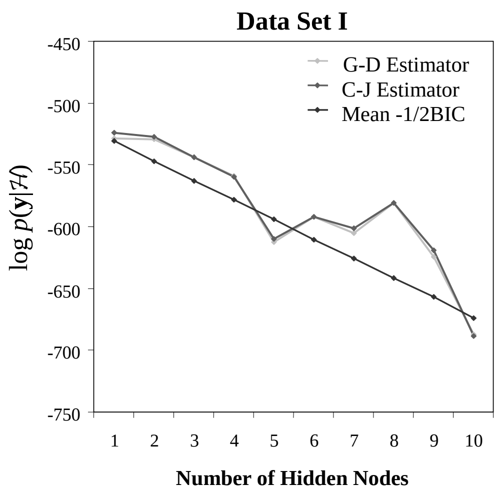

Figure 4.30 Evidence estimates for the $1, \ldots, 10$ hidden node ANNs applied to data set I.

Table 4.4 Log Bayes Factors in favour of the highest ranked model for data set I.

Rank	Hidden Nodes	$\log_e BF$ in Favour of Rank 1 Model
1	1	–
2	2	16.342
3	3	32.194
4	4	47.206
5	5	63.299
6	6	79.678
7	7	95.098
8	8	110.638
9	9	126.340
10	10	143.304

output weights, which can be used to determine whether all hidden nodes in the model are necessary, as discussed in Section 4.3.4.2. The marginal posterior distribution for the hidden-output weight of the 1 hidden node model is displayed in Figure 4.31. As this distribution does not include the value zero, it is indicated that this node is necessary to model data set I. On the other hand, the 95% highest density regions of the marginal posterior distributions shown in Figure 4.32 (a) and (b), which are the hidden-output weights of a 2 hidden node network, do include zero, indicating that at least one of the nodes is not necessary. The scatter plot in Figure 4.32 (c) does not pass through the origin; therefore, it can be determined that only one of the hidden nodes may be removed from the network. This leaves a 1 hidden node network, which means that the Bayes factor results were correct. However, it was known that a network with no hidden nodes was, in fact, optimal for this data set; yet, this was not determined by inspecting the marginal hidden-output distributions. When there is only one hidden node in an ANN,

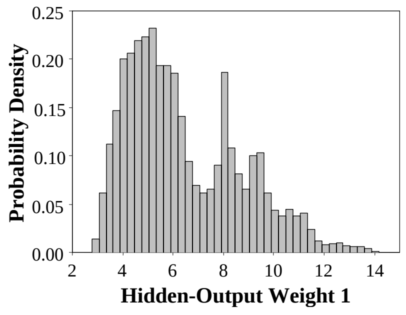

Figure 4.31 Marginal posterior hidden-output weight distribution for the 1 hidden node ANN.

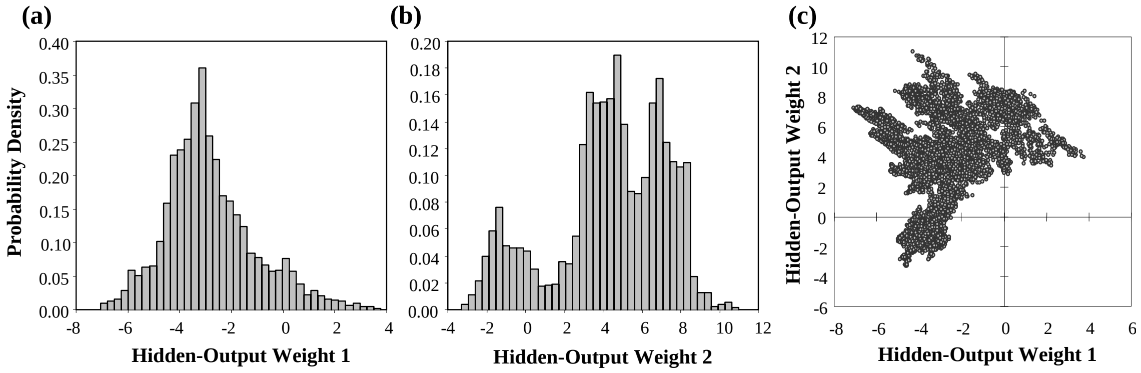

Figure 4.32 Marginal posterior distributions for (a) hidden-output weight 1 and (b) hidden-output weight 2, together with (c) the scatter plot of the joint hidden-output weight distribution.

the marginal posterior hidden-output weight distribution will never include zero unless there is no relationship between the model inputs and outputs, since the inputs would then be disconnected from the output. Similarly, when there are only two hidden nodes in the network, the joint distribution of the hidden-output weights will never pass through the origin for the same reason. While it is possible to build a network with no hidden layer, with the inputs connected directly to the output, this would result in a linear model and, as mentioned in Section 3.4.2.4, only nonlinear models were considered in this research.

Table 4.5 gives the model evidence results for data set II. Again, the values calculated using the different methods correctly estimated the 3 hidden node ANN to have the greatest evidence (highlighted in bold italics). The evidence values are plotted in Figure 4.33,

Table 4.5 Evidence estimates for data set II ANN models.

Hidden Nodes	G-D Estimator	C-J Estimator	Mean -1/2BIC
1	-760.390	-757.601	-764.402
2	-766.520	-766.606	-783.648
3	***-723.427***	***-717.256***	***-705.040***
4	-745.980	-745.118	-725.317
5	-771.298	-759.791	-745.451
6	-853.902	-844.695	-763.963
7	-803.790	-798.116	-783.247
8	-842.126	-839.281	-802.007
9	-896.673	-895.476	-822.855
10	-981.232	-982.163	-843.845

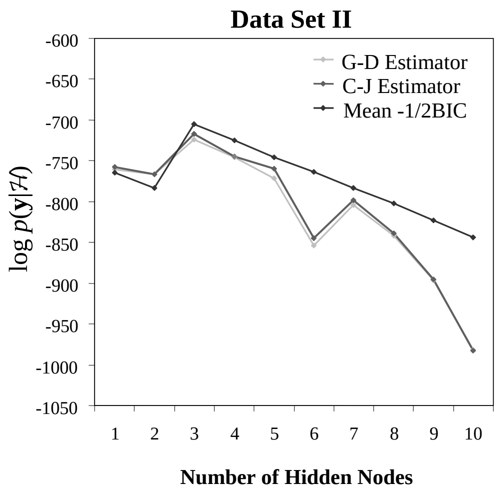

Figure 4.33 Evidence estimates for the $1, \ldots, 10$ hidden node ANNs applied to data set II.

where it appears that some of the evidence values calculated using the G-D and C-J estimators are erroneous, as there is no logical reason why the 6 hidden node network should be less probable than the 7 and 8 hidden node ANNs. As is the case for data set I, the mean -1/2BIC evidence estimates appear to be the most consistent and rational.

The $BF_{Rank1,i}$ results, calculated based on the -1/2BIC evidence estimates, are presented in Table 4.6. The evidence in favour of the 3 hidden node ANN is very strong according to the Bayes factors, which was verified by inspection of Figures 4.34 and 4.35, which display the marginal posterior distributions of the hidden-output weights of the 3 and 4 hidden nodes networks, respectively. In Figure 4.34, it can be seen that none of the distributions include zero, indicating that all of the nodes are necessary for mod-

Table 4.6 Log Bayes Factors in favour of the highest ranked model for data set II.

Rank	Hidden Nodes	$\log_e BF$ in Favour of Rank 1 Model
1	3	–
2	4	20.278
3	5	40.412
4	6	58.923
5	1	59.362
6	7	78.208
7	2	78.608
8	8	96.967
9	9	117.816
10	10	138.805

elling data set II, whereas the 95% highest density region of the distribution shown in Figure 4.35 (b) does include zero, which indicates that this node may be removed from the network without loss of predictive performance.

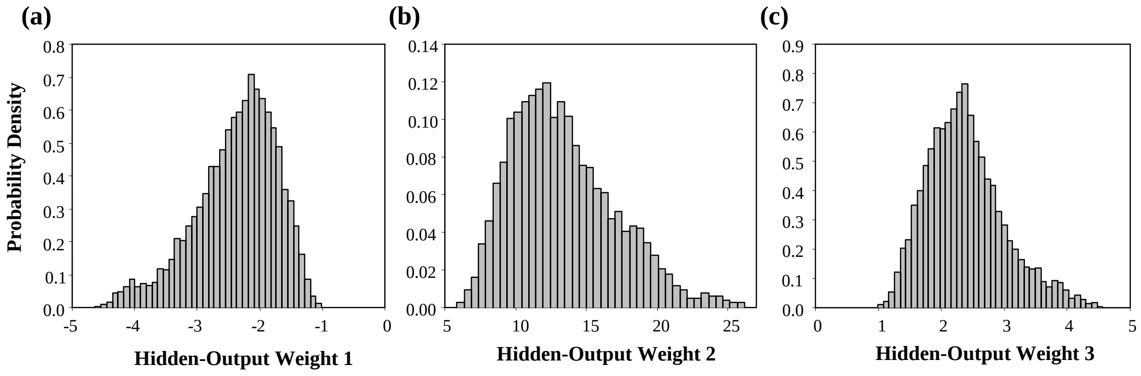

Figure 4.34 Marginal posterior hidden-output weight distributions for the 3 hidden node ANN applied to data set II.

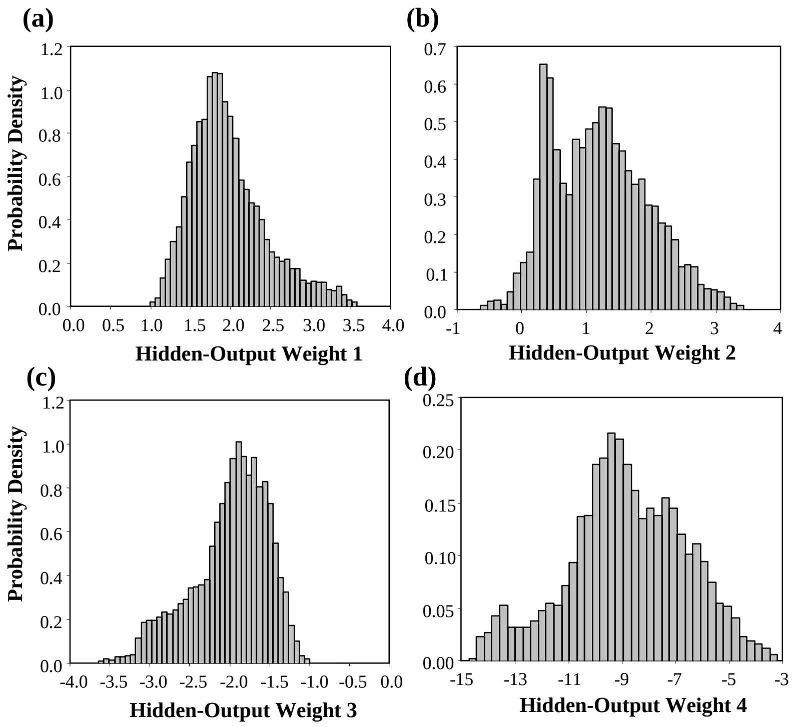

Figure 4.35 Marginal posterior hidden-output weight distributions for the 4 hidden node ANN applied to data set II.

Table 4.7 Evidence estimates for data set III ANN models.

Hidden Nodes	G-D Estimator	C-J Estimator	Mean -1/2BIC
1	-1181.973	-1178.764	-1187.835
2	-979.765	-974.182	-969.696
3	-807.071	-806.271	-785.296
4	-241.059	-235.593	-177.878
5	***-193.011***	***-193.503***	***-133.750***
6	-235.465	-236.286	-154.014
7	-303.281	-299.990	-181.012
8	-408.727	-408.916	-211.591
9	-404.002	-404.919	-243.774
10	-457.596	-455.242	-257.550

The evidence results obtained for data set III are presented in Table 4.7 and Figure 4.36. The highest evidence values estimated using each method (highlighted in bold italics in Table 4.7) correctly indicate that the 5 hidden node model provides the optimal complexity. Similar to the results obtained for data sets I and II, some of the G-D and C-J estimates seem erroneous. For example, the 8 hidden node ANN was estimated to be less probable than the 9 hidden node network by both methods and the 4 hidden node network was estimated as being more probable than the 6 hidden node ANN, which contradicts the results obtained in Chapter 3 and the evidence values estimated using the G-D and -1/2BIC methods. Again, the BIC estimator is the most consistent of the methods investigated.

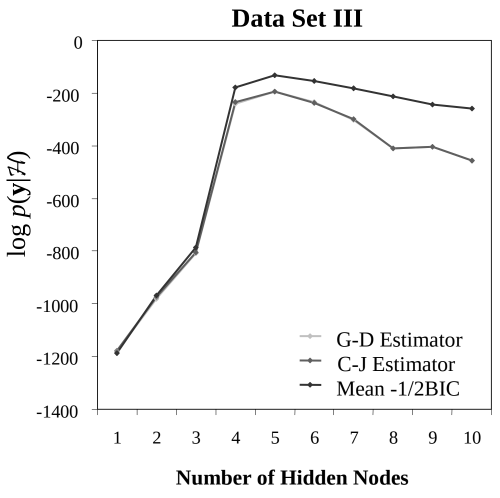

Figure 4.36 Evidence estimates for the $1, \ldots, 10$ hidden node ANNs applied to data set III.

Table 4.8 Log Bayes Factors in favour of the highest ranked model for data set III.

Rank	Hidden Nodes	$\log_e BF$ in Favour of Rank 1 Model
1	5	–
2	6	20.263
3	4	44.128
4	7	47.261
5	8	77.840
6	9	110.023
7	10	123.799
8	3	651.546
9	2	835.945
10	1	1054.084

The $BF_{Rank1,i}$ results presented in Table 4.8 indicate that there is very strong evidence in favour of the 5 hidden node ANN over the 6 hidden node network. However, inspection of the marginal posterior distributions of the hidden-output weights of the 5 and 6 hidden node ANNs, shown in Figures 4.37 and 4.38, respectively, indicates that this is not the case, as none of these distributions for either of the models include zero. There may be a

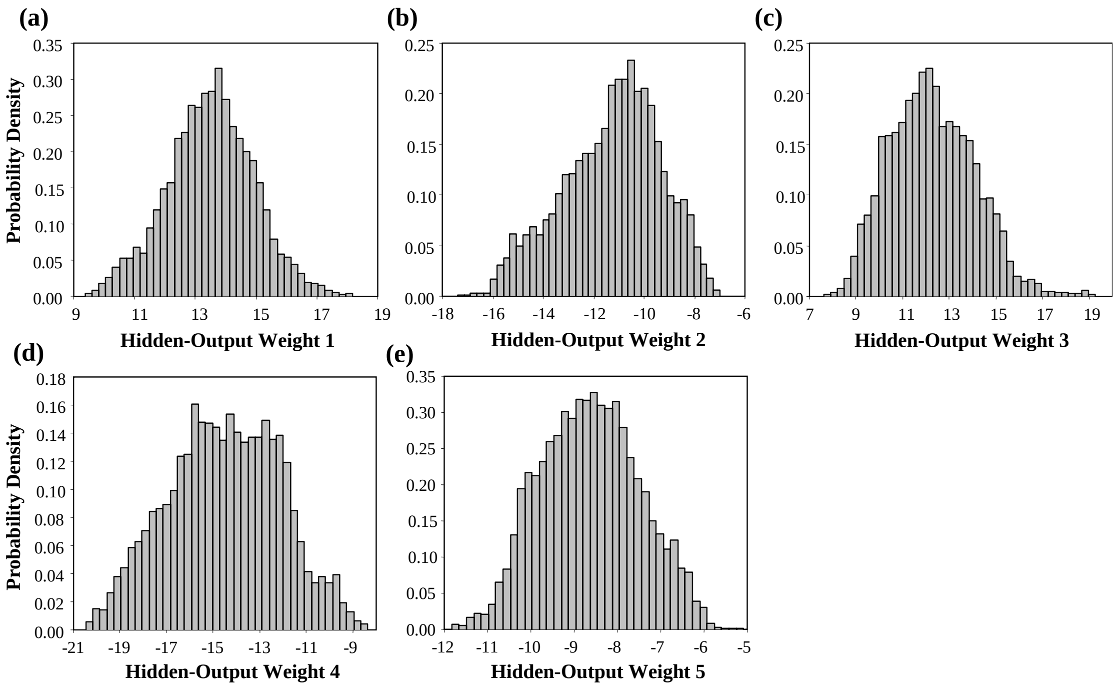

Figure 4.37 Marginal posterior hidden-output weight distributions for the 5 hidden node ANN applied to data set III.

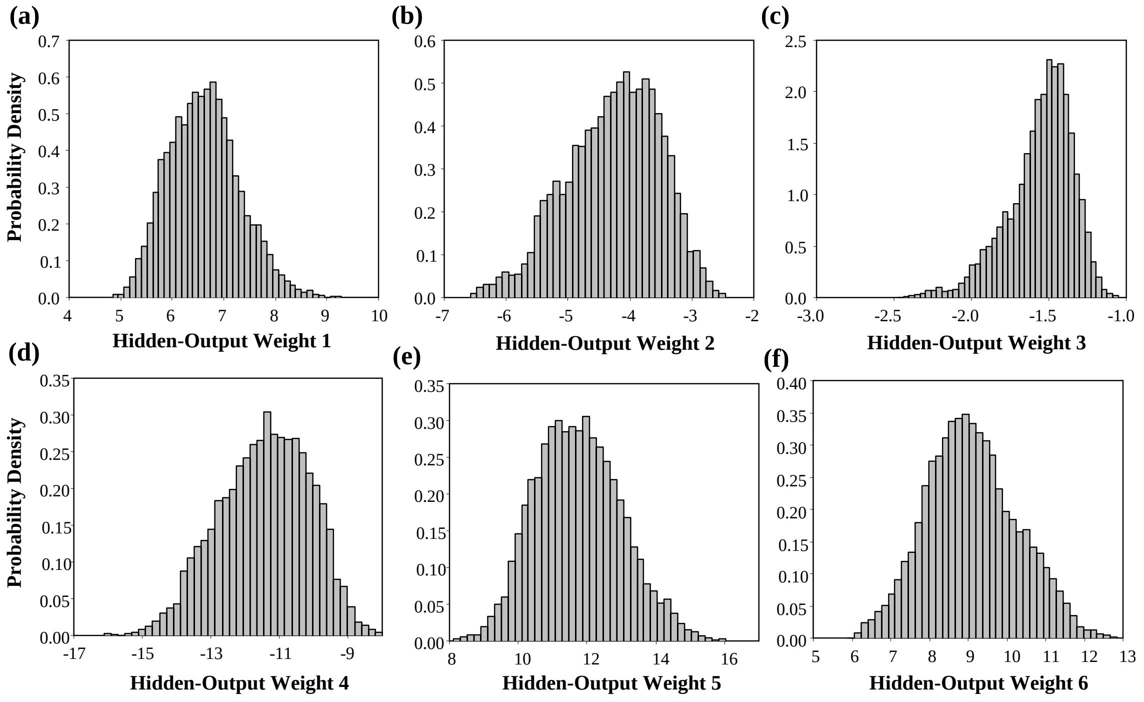

Figure 4.38 Marginal posterior hidden-output weight distributions for the 6 hidden node ANN applied to data set III.

number of reasons why the Bayes factor results suggest strong evidence in favour of the 5 hidden node ANN, when it is, in fact, difficult to choose between the 5 and 6 hidden node ANN models. For example, the interpretive scale given in Table 4.1 may not be appropriate for ANNs, where the addition of each hidden node increases the complexity of the model by several dimensions; the evidence approximation given by the -1/2BIC may be poor; or the MCMC algorithm may not have converged properly (although inspection of output MCMC traces indicated that it had). By inspection of the marginal posterior distributions for the hidden-output weights of the 7 hidden node model, shown in Figure 4.39, it was seen that only 6 hidden nodes were necessary, as the marginal posterior distribution displayed in subplot (e) included zero and was therefore considered redundant. It was concluded from these plots that the 6 hidden node ANN may be more appropriate for modelling data set III than the 5 hidden node ANN, as it is apparent that there may be some increase in model performance by including the additional hidden node.

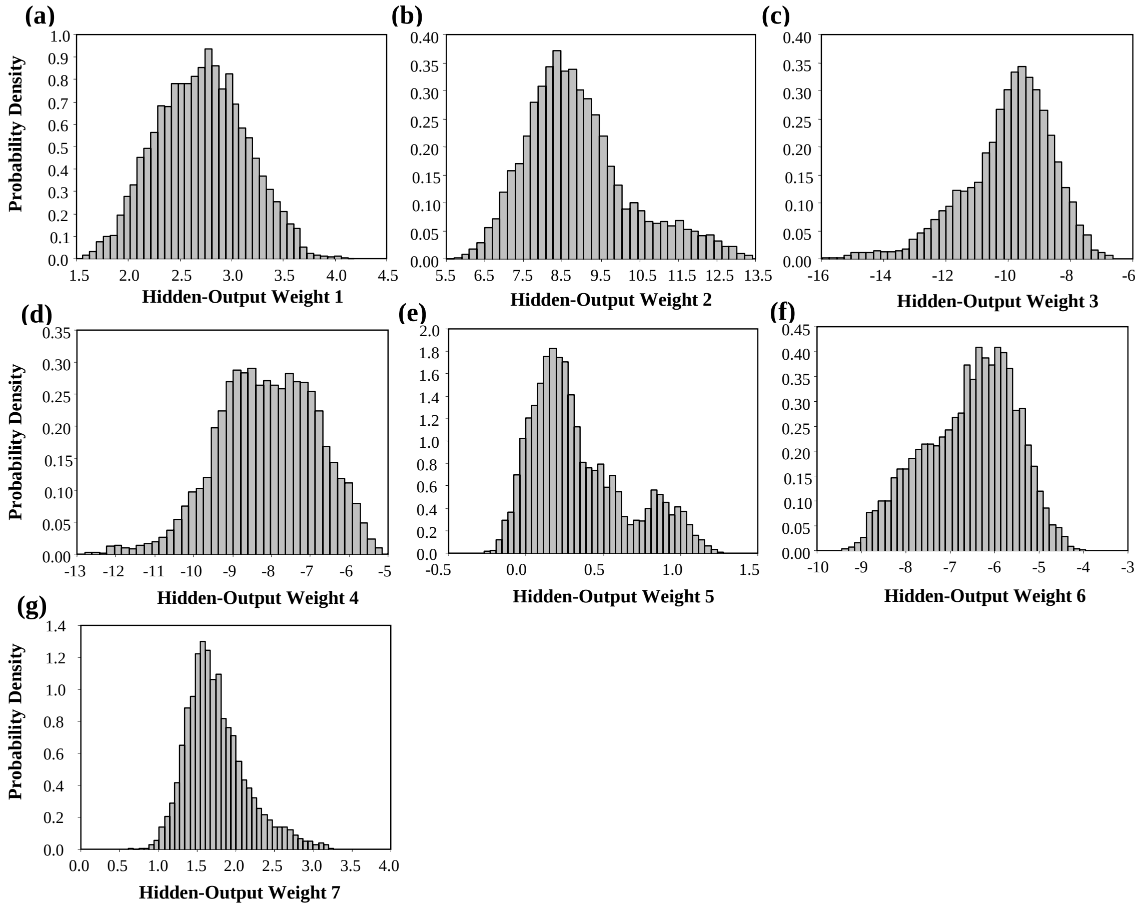

Figure 4.39 Marginal posterior hidden-output weight distributions for the 7 hidden node ANN applied to data set III.

4.4.4 Evaluation of Best Models

As with the deterministic ANNs developed in Chapter 3, the best models developed using the Bayesian framework were evaluated by assessing their performance against the “measured” and “true” training, testing and validation data subsets for synthetic data sets I, II and III. It should be noted that the testing data subset was unnecessary for the Bayesian training approach, as cross validation is not required in the Bayesian context; therefore, it could have been combined with the training data. However, in order to provide a fair comparison with the results obtained using the deterministic models, this was not done. Shown in Figure 4.40 are scatter plots of the mean model predictions and 95% prediction limits of the 1 hidden node ANN found most suitable for modelling data set I versus the “measured” and “true” training, testing and validation data. It can be seen in this figure that the mean predictions provide a very good fit to the “true” data and that the majority of

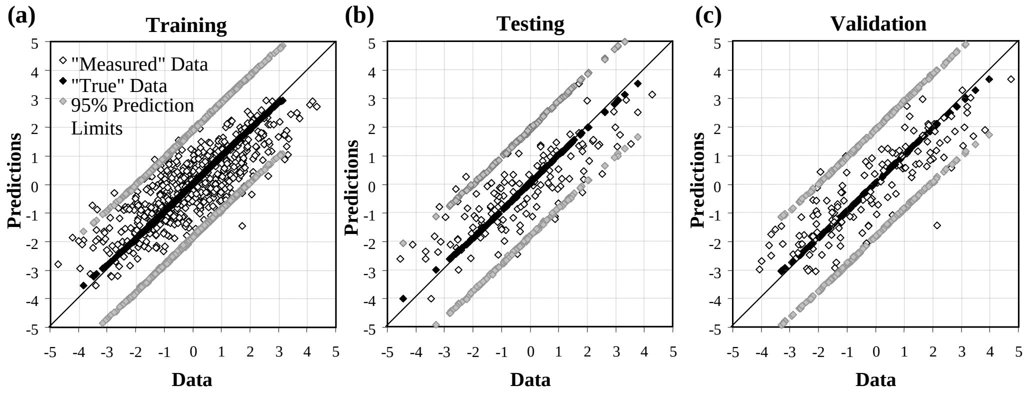

Figure 4.40 Scatter plots of the 1 hidden node ANN model mean predictions and 95% prediction limits versus "measured" and "true" (a) training, (b) testing and (c) validation data for data set I.

the "measured data" are contained within the 95% prediction limits. In fact, these limits contain 100% of the "true" data points and 94.8% of the "measured" data.

A time series plot of the mean model predictions and 95% prediction limits is shown in Figure 4.41 against the "measured" and "true" recombined training, testing and validation data. The first 100 points of this plot are shown in Figure 4.42 to better illustrate the model fit to the data. It can be seen that the mean predictions provide a near perfect fit to the "true" data and that the prediction limits, while relatively narrow, account for almost all of the "measured" data.

Shown in Figure 4.43 are the RI distributions for each input of the 1 hidden node ANN used for modelling data set I. These were calculated by evaluating the RI values for each of the weight vectors sampled from the posterior distribution using the modified Connection Weight Approach. The minimum, mean and maximum values of these distributions are summarised in Table 4.9 in comparison to the PMI-based RI estimates. It can be seen here that there is reasonable agreement between the mean RI values and the PMI-based estimates and that these estimates are each incorporated within the bounds of the estimated RI distributions, indicating that the model developed had approximated the underlying relationship well.

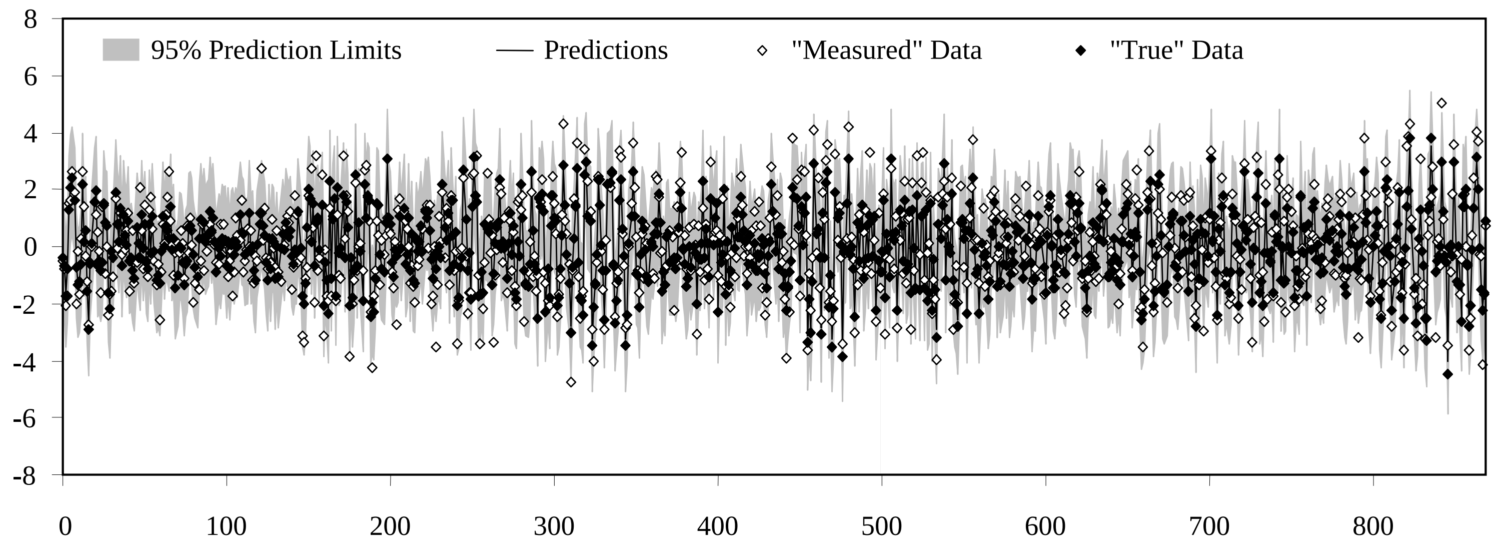

Figure 4.41 Plot of mean model predictions and 95% prediction limits against the combined training/testing/validation “measured” and “true” data for data set I.

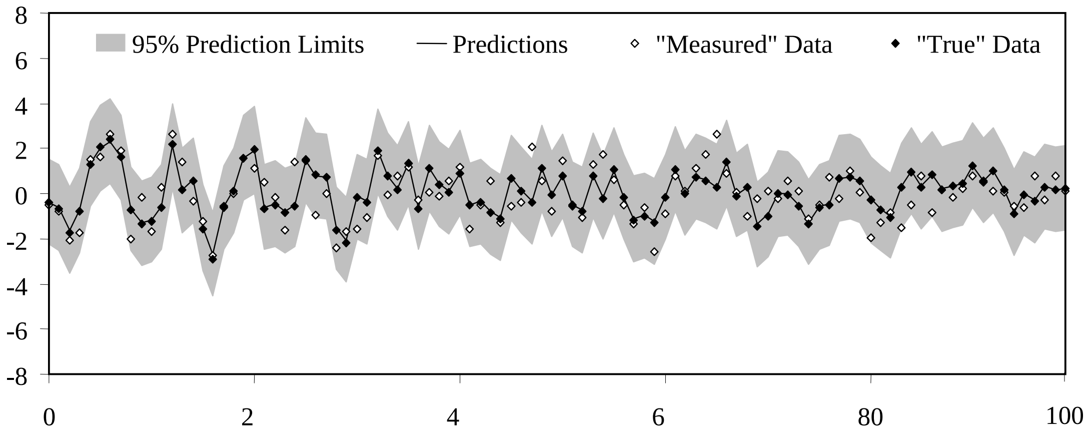

Figure 4.42 Plot of the first 100 mean model predictions and 95% prediction limits against the combined training/testing/validation "measured" and "true" data for data set I.

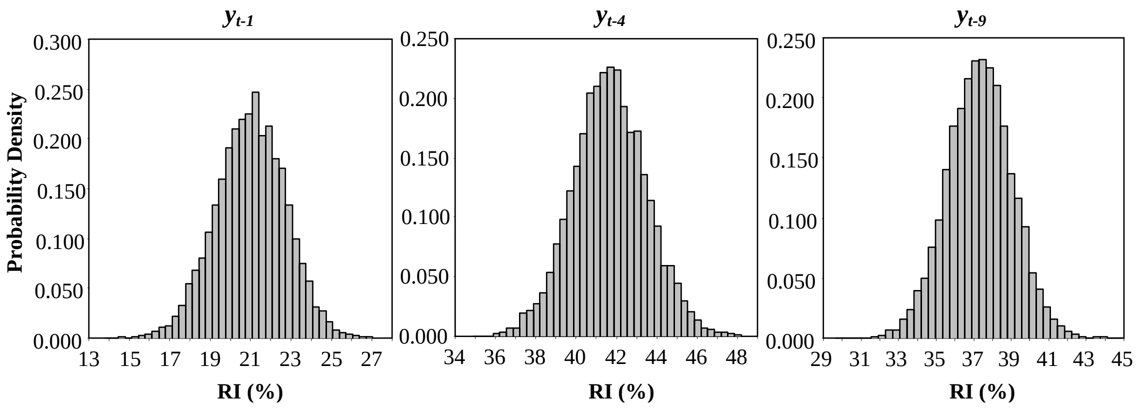

Figure 4.43 Estimated RI distributions for the inputs of data set I.

Table 4.9 Minimum, mean and maximum RI values (%) for the inputs of the 1 hidden node ANN developed for modelling data set I.

RI Estimation Method	y_{t-1}	y_{t-4}	y_{t-9}
Minimum	13.81	34.88	29.64
Mean	21.00	41.67	37.33
Maximum	27.04	48.20	44.90
PMI-based	*22.33*	*45.07*	*32.60*

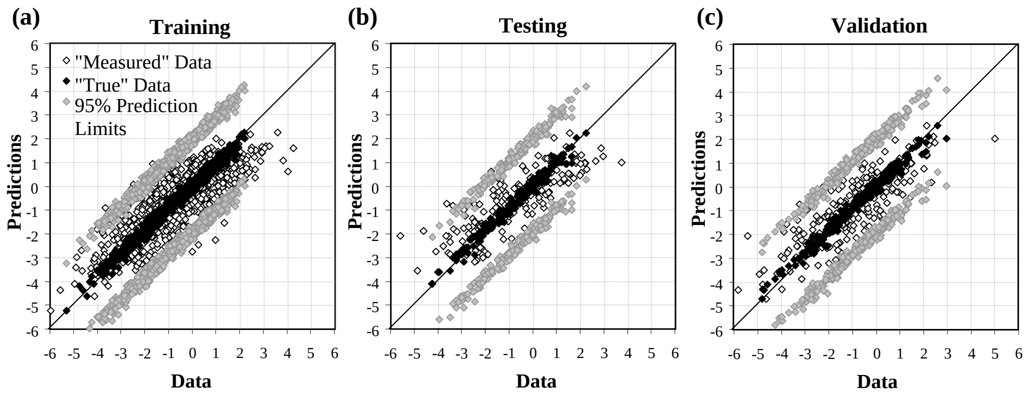

Figure 4.44 Scatter plots of the 3 hidden node ANN model mean predictions and 95% prediction limits versus "measured" and "true" (a) training, (b) testing and (c) validation data for data set II.

A 3 hidden node ANN was found to be best for modelling data set II and the scatter plots of the resulting mean model predictions and 95% prediction limits versus the "measured" and "true" training, testing and validation data are shown in Figure 4.44. In this case, the 95% prediction limits incorporate 100% of the "true" data points and 96.3% of the "measured" data.

Figures 4.45 and 4.46 show time series plots of the mean model predictions and 95% prediction limits against the "measured" and "true" data for the first 100 points of the recombined training, testing and validation data set and for the entire data set, respectively. It can be seen in Figure 4.45 that although the mean predictions do not accurately predict all of the "true" data points, all of these data are contained well within the 95% prediction limits.

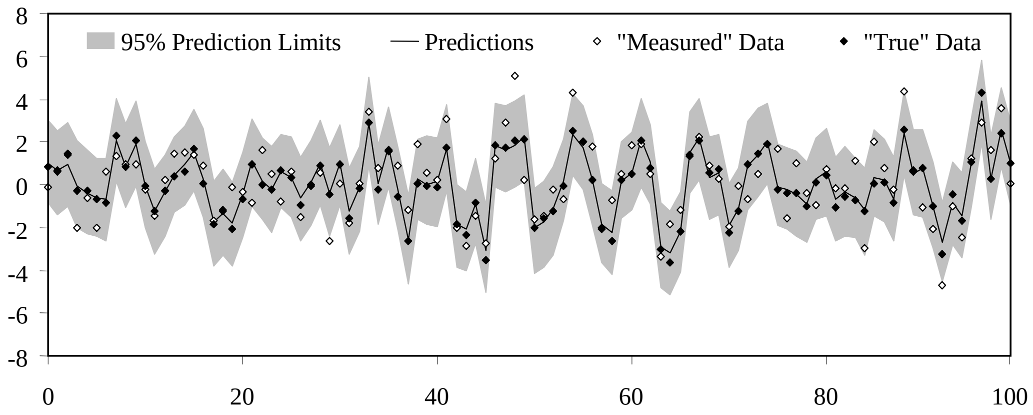

Figure 4.45 Plot of the first 100 mean model predictions and 95% prediction limits against the combined training/testing/validation "measured" and "true" data for data set II.

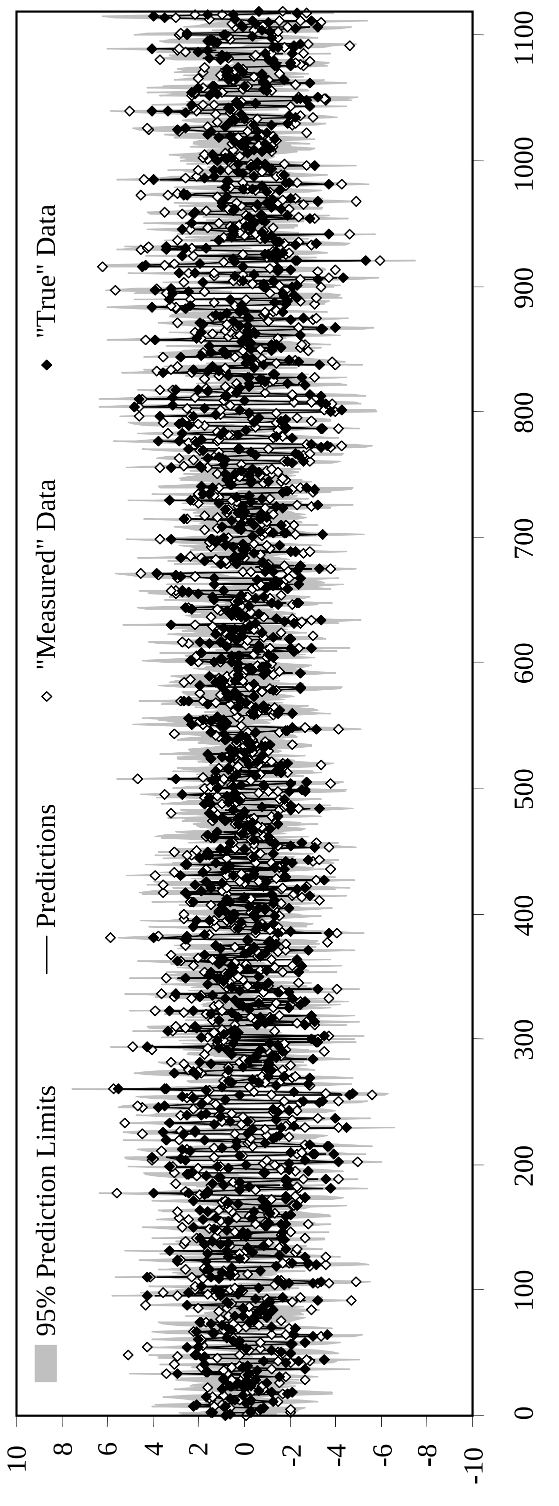

Figure 4.46 Plot of mean model predictions and 95% prediction limits against the combined training/testing/validation "measured" and "true" data for data set II.

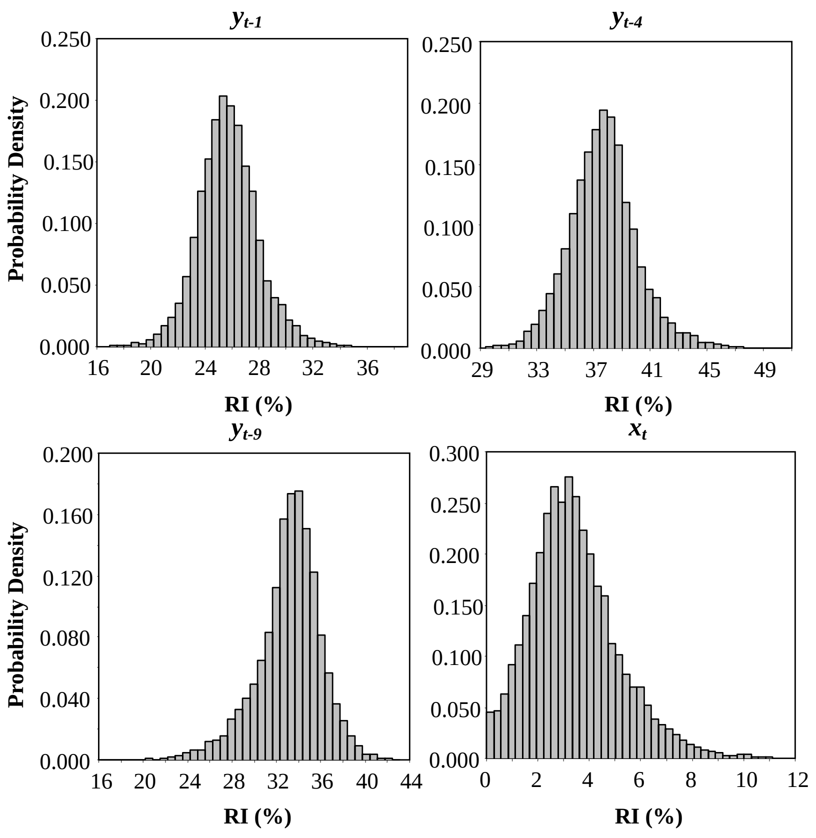

Figure 4.47 Estimated RI distributions for the inputs of data set II.

The RI distributions for each input of data set II, estimated with the modified Connection Weight Approach, are shown in Figure 4.47, while the minimum, mean and maximum RI values are summarised in Table 4.10. For inputs y_{t-1}, y_{t-4} and y_{t-9}, there is good agreement between the mean RI values and the PMI-based RI estimates, which are also given in Table 4.10. While the mean RI value is significantly less than the PMI-based RI estimate for input x_t, the maximum of the RI distribution is approximately equal to the PMI-based RI estimate; thus, it was considered that the model obtained a

Table 4.10 Mean RI values (%) for the inputs of the 3 hidden node ANN developed for modelling data set II.

RI Estimate	y_{t-1}	y_{t-4}	y_{t-9}	x_t
Minimum	16.93	29.39	16.15	0.00
Mean	25.73	37.71	33.14	3.42
Maximum	38.66	50.82	43.09	11.12
PMI-based	*21.62*	*36.13*	*30.82*	*11.44*

good approximation of the data-generating function.

Similar to the deterministic case, it was inconclusive using Bayesian model selection whether a 5 hidden node ANN or a 6 hidden node ANN was better for modelling data set III. Scatter plots of the 5 hidden node ANN model mean predictions and 95% prediction limits versus the "measured" and "true" data are shown in Figure 4.48, while Figure 4.49 shows the same plots for the 6 hidden node ANN model. The 95% prediction limits of both the 5 and 6 hidden node models include 100% of the "true" data and include 95.0% and 95.4% of the "measured" data, respectively.

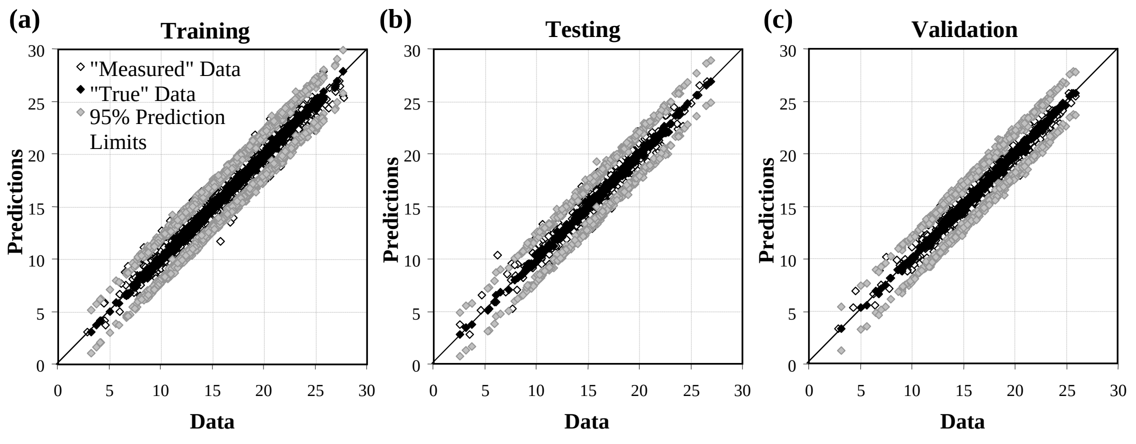

Figure 4.48 Scatter plots of the 5 hidden node ANN model mean predictions and 95% prediction limits versus "measured" and "true" (a) training, (b) testing and (c) validation data for data set II.

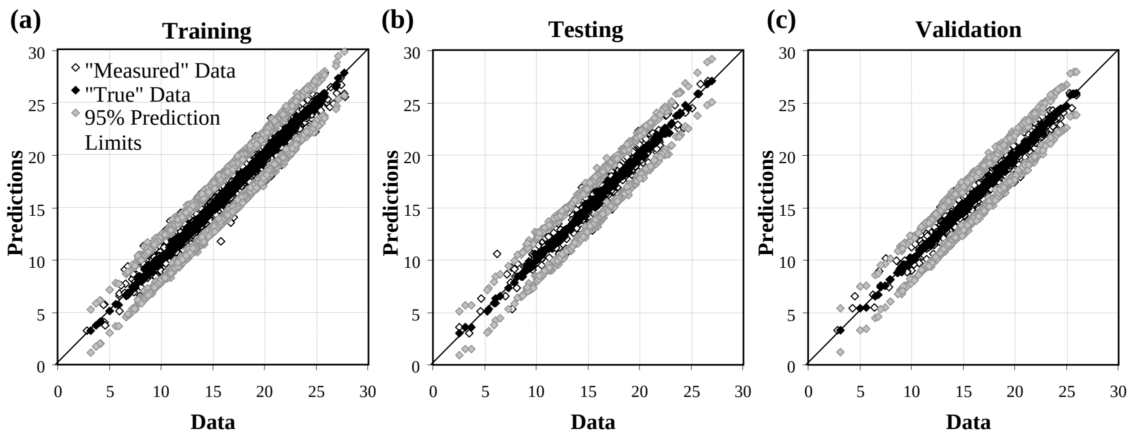

Figure 4.49 Scatter plots of the 6 hidden node ANN model mean predictions and 95% prediction limits versus "measured" and "true" (a) training, (b) testing and (c) validation data for data set II.

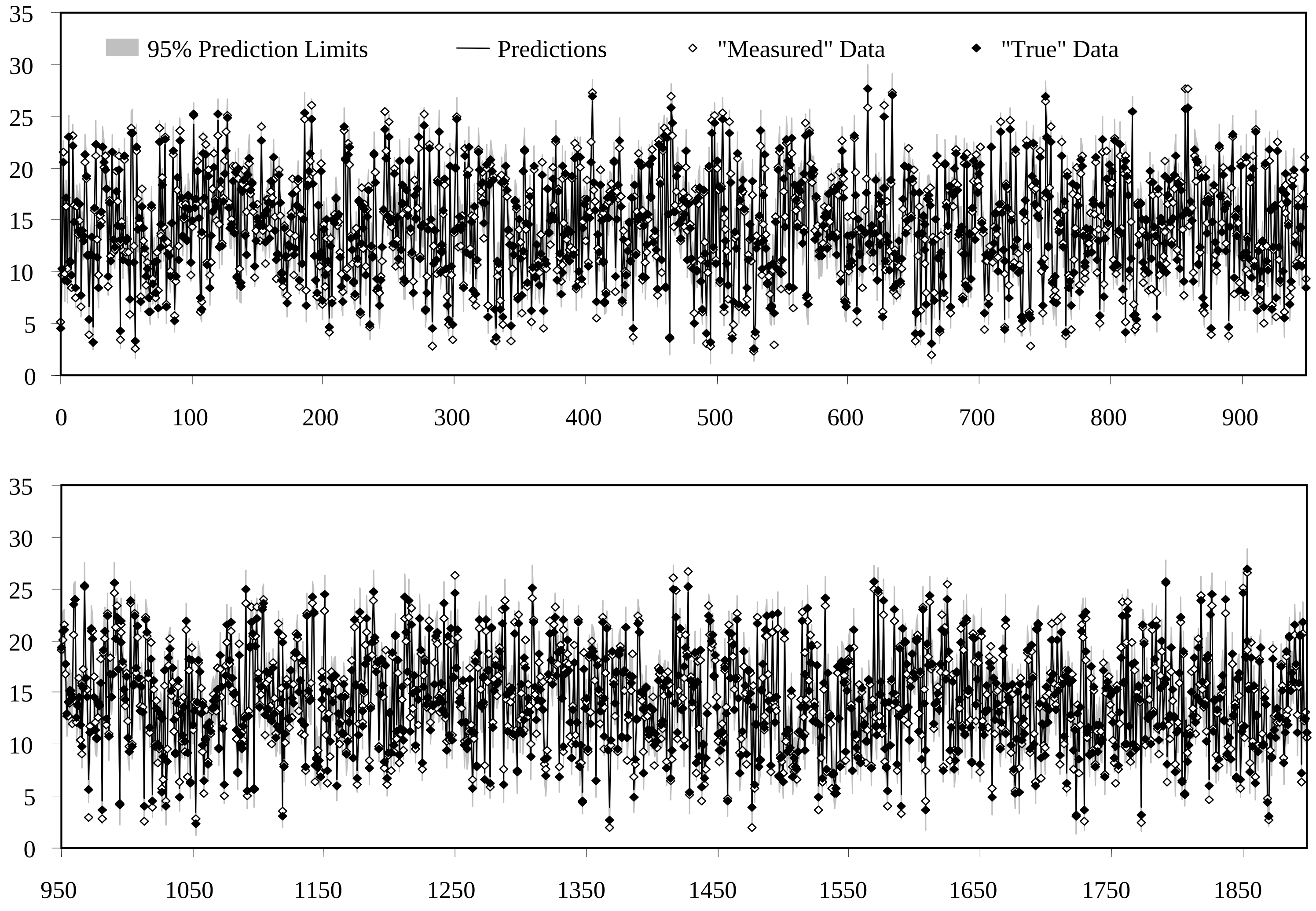

Figure 4.50 Plot of the mean predictions and 95% prediction limits against the combined training/testing/validation "measured" and "true" data for the 5 hidden node ANN applied to data set III.

Shown in Figures 4.50 and 4.51 are output plots of the 5 hidden node ANN model mean predictions and 95% prediction limits against the “measured” and “true” data for the entire recombined training, testing and validation data set and for the first 100 points of this data set, respectively. The prediction limits are relatively narrow about the mean predictions, yet the “true” data are still well contained within the bounds, indicating that the model predictions have little associated uncertainty. This is a result of the high signal-to-noise ratio of the data, which enabled the model to extract relatively noise free information from the data to estimate the underlying relationship. The output plots for the 6 hidden node ANN model are not shown, as they are almost identical to those presented in Figures 4.50 and 4.51.

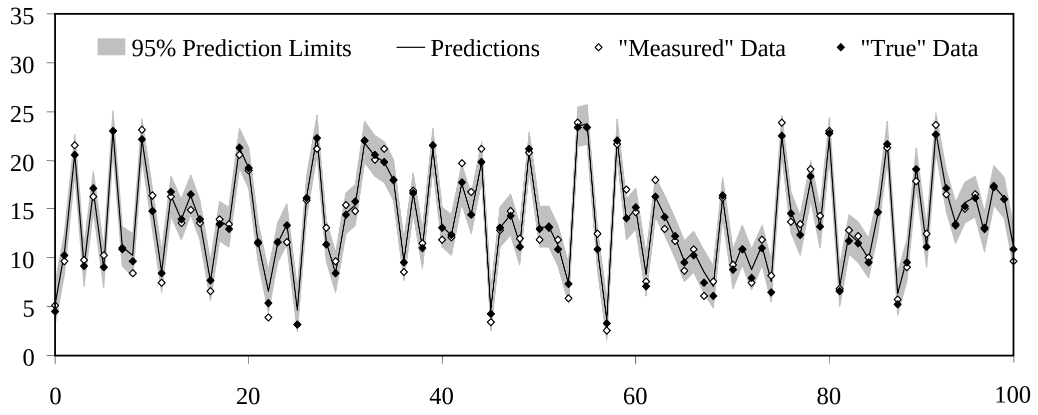

Figure 4.51 Plot of the first 100 mean predictions and 95% prediction limits against the combined training/testing/validation “measured” and “true” data for the 5 hidden node ANN applied to data set III.

The RI distributions for each input of data set III, estimated with the modified Connection Weight Approach using the weights of the 5 and 6 hidden node ANN weights, are shown in Figures 4.52 and 4.53, respectively. The minimum, mean and maximum RI values of these models are summarised in Table 4.11 together with the PMI-based RI estimates. It can be seen in this table that there is better agreement between the RI distributions estimated for the 6 hidden node ANN and the PMI-based RI estimates, indicating that this model obtained a better approximation of the data-generating function than the 5 hidden node ANN.

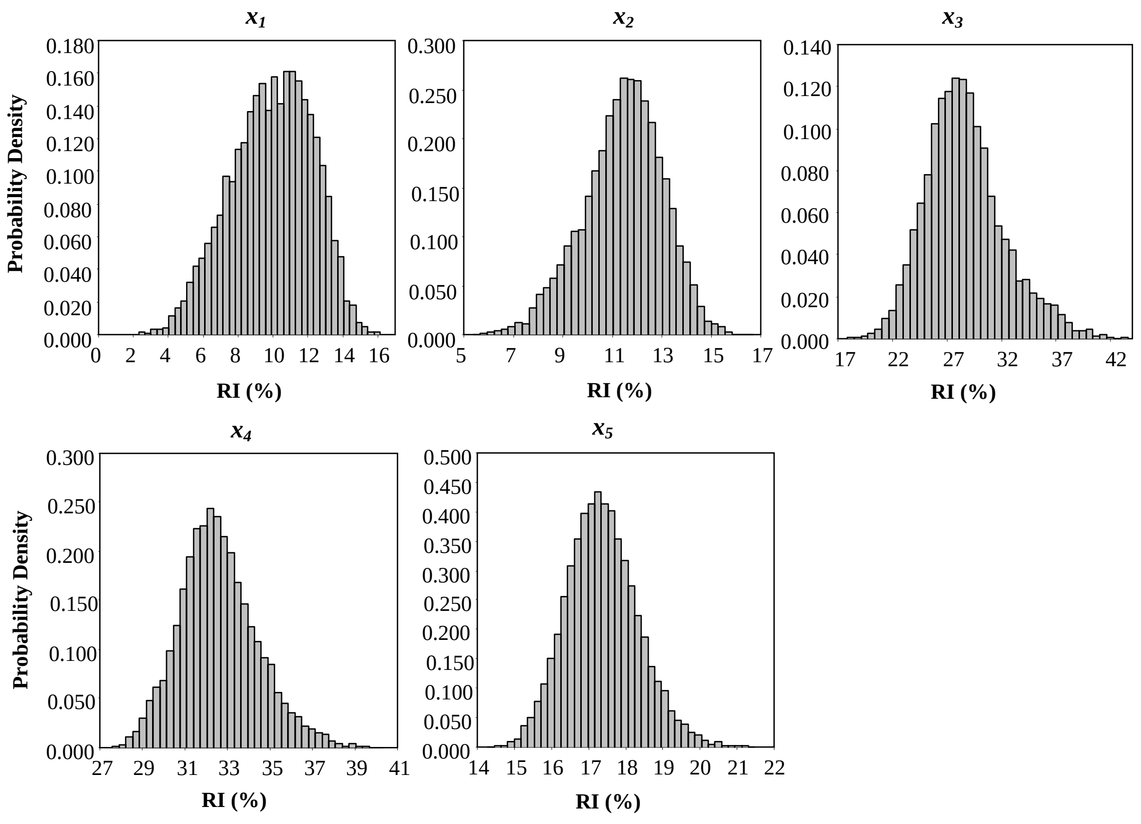

Figure 4.52 RI distributions for the inputs of data set III estimated using the 5 hidden node ANN weights.

Table 4.11 Minimum, mean and maximum RI values (%) for the inputs of the 5 and 6 hidden node ANN developed for modelling data set III.

RI Estimation Method	x_1	x_2	x_3	x_4	x_5
5 hidden nodes - minimum	0.06	3.80	16.55	26.22	13.33
5 hidden nodes - mean	9.39	10.91	29.43	32.83	17.42
5 hidden nodes - maximum	16.60	16.35	41.14	41.91	22.02
6 hidden nodes - minimum	13.42	27.71	0.00	21.97	11.46
6 hidden nodes - mean	19.80	34.19	3.31	27.47	15.23
6 hidden nodes - maximum	26.02	40.40	15.09	33.58	19.92
PMI-based	*20.07*	*22.65*	*10.65*	*30.80*	*15.83*

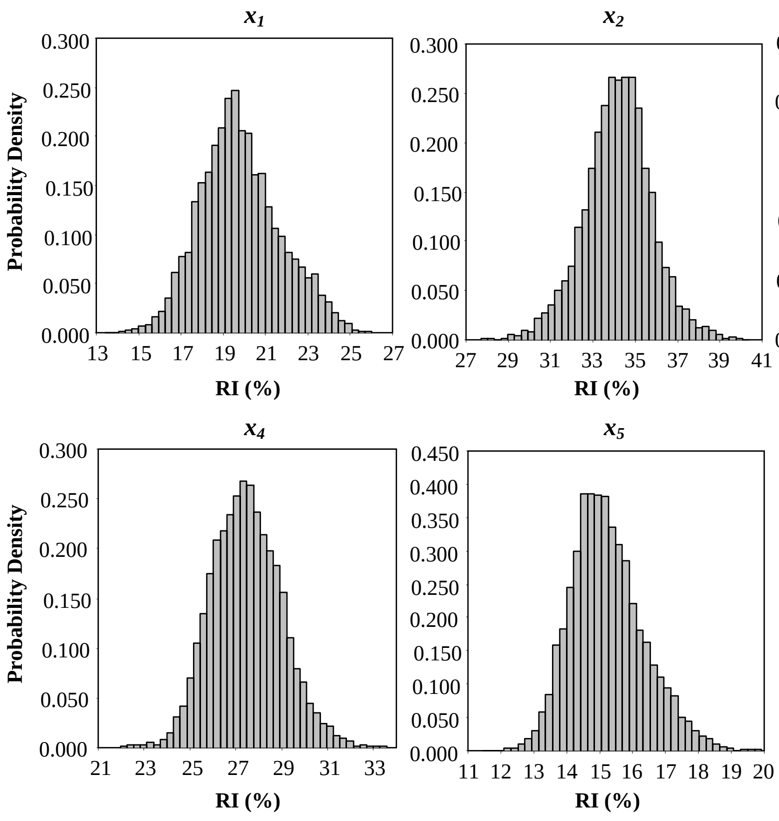

Figure 4.53 RI distributions for the inputs of data set III estimated using the 6 hidden node ANN weights.

Performance criteria are given in Table 4.12 to summarise the mean performance (i.e. calculated based on the mean predictions) of the ANN models developed when applied to the “measured” and “true” validation data for all three synthetic data sets. The results obtained using the deterministic development approach, as well as the actual $\sigma_{\mathbf{y}}^2$ and MAE values (shown in italics), are given for comparison. As can be seen, the mean performance of all of the ANN models developed using Bayesian methods was slightly better than the performance of the corresponding deterministic ANN models on the “true” validation data. This indicates that each of the data-generating functions was slightly better approximated using the Bayesian approach. Performance of the Bayesian and deterministic models on the “measured” validation data was similar; however, in general, the Bayesian models also performed slightly better on this data.

Table 4.12 Mean performance of the ANNs developed using Bayesian methods in comparison to the performance of the corresponding deterministic ANNs when applied to "measured" and "true" validation data.

Performance	Deterministic		Bayesian	
Measure	"Measured"	"True"	"Measured"	"True"
DATA SET I				
$\hat{\sigma}^2_{\mathbf{y}}$	1.020	0.010	1.104	0.008
MAE	0.814	0.075	0.817	0.070
RMSE	1.010	0.098	1.012	0.091
CE	0.703	0.996	0.702	0.997
Actual $\hat{\sigma}^2_{\mathbf{y}}$	*1.035*	*0.000*	*1.035*	*0.000*
Actual MAE	*0.816*	*0.000*	*0.816*	*0.000*
DATA SET II				
$\hat{\sigma}^2_{\mathbf{y}}$	0.922	0.059	0.917	0.055
MAE	0.766	0.190	0.760	0.182
RMSE	0.960	0.243	0.958	0.234
CE	0.682	0.974	0.684	0.976
Actual $\hat{\sigma}^2_{\mathbf{y}}$	*0.852*	*0.000*	*0.852*	*0.000*
Actual MAE	*0.733*	*0.000*	*0.733*	*0.000*
DATA SET III - 5 hidden node ANN				
$\hat{\sigma}^2_{\mathbf{y}}$	0.924	0.068	0.922	0.055
MAE	0.768	0.201	0.768	0.181
RMSE	0.961	0.262	0.960	0.234
CE	0.947	0.996	0.948	0.997
Actual $\hat{\sigma}^2_{\mathbf{y}}$	*0.869*	*0.000*	*0.869*	*0.000*
Actual MAE	*0.752*	*0.000*	*0.752*	*0.000*
DATA SET III - 6 hidden node ANN				
$\hat{\sigma}^2_{\mathbf{y}}$	0.909	0.059	0.905	0.050
MAE	0.769	0.185	0.766	0.173
RMSE	0.953	0.242	0.951	0.224
CE	0.948	0.997	0.949	0.997
Actual $\hat{\sigma}^2_{\mathbf{y}}$	*0.869*	*0.000*	*0.869*	*0.000*
Actual MAE	*0.752*	*0.000*	*0.752*	*0.000*

4.4.5 Conclusions

4.4.5.1 Bayesian Training and Prediction

It was seen that the best training results in terms of the estimated posterior weight distribution, given overfitted and poor initial conditions, were obtained when a hierarchical prior distribution was used. The use of an ARD type prior, although possibly more representative of actual prior knowledge, did not provide any significant additional benefits to the simulation and only served to increase the complexity of the algorithm, whereas the noninformative uniform prior, whilst retaining the simplicity of the algorithm, was unable to prevent overfitting or enable the chains to efficiently explore the search space. The hierarchical prior distribution enabled both the prevention and correction of overfitting when the MCMC algorithm was used to train overparameterised models. Furthermore, when the hyperparameters of the hierarchical prior were fixed for a short initial period, the results of the MCMC simulation, given poor initialisation of the weights, were substantially improved, with the chains being better able to explore the search space and discover new modes. On the other hand, simulated annealing was unable to achieve any significant improvement in the results of the MCMC simulation, as no further modes of higher posterior probability were discovered than when simulated annealing was not applied. It is possible that, with a greater number of iterations and more optimal values of φ and T_0, simulated annealing could have been more successful; however, it is considered that the increase in complexity of the MCMC algorithm as a result of the additional tuning parameters is not warranted. Therefore, simulated annealing will not be considered further in this research. Overall, it was concluded that, although helpful in "forgetting" initial conditions, the use of a hierarchical prior with initially fixed hyperparameters is no substitute for a good weight initialisation. It is therefore recommended that extra care be taken in finding appropriate weights $\widehat{\mathbf{w}}$ to initialise the MCMC algorithm. Thus, if the algorithm does become stuck in the vicinity of a local mode, there will at least be some confidence that it is a "good" mode (i.e. the best estimate of the maximum likelihood value given by a rigorous search algorithm that tries to thoroughly search the space). The use of a hierarchical prior with initially fixed hyperparameters will help to lessen the bias that may be caused by this initialisation.

4.4.5.2 Bayesian Model Selection

The -1/2BIC estimator of the ANN models' evidence values was found to be the most consistent and logical of the estimators investigated. For each of the three synthetic case studies considered, the -1/2BIC values followed a similar pattern to the theoretically more correct G-D and C-J estimators, yet did not suffer from the same instabilities due to the prior distribution assumed or the proposal distribution used to sample draws from the posterior distribution. Furthermore, a distribution of -1/2BIC values is obtained, which may be better to estimate the strength of the evidence for or against a given model (by assessing overlap, if any, of -1/2BIC distributions and the difference between maximum and minimum values). Inspection of marginal posterior hidden-output weight distributions was shown to be useful for checking the results obtained from the -1/2BIC Bayes factors, as these distributions indicate whether or not similar results could be obtained if one or more of these connections was set equal to zero; thus, removing it from the network.

4.4.5.3 Overall

The models developed using the Bayesian training and model selection framework performed slightly better than those developed using the deterministic approach presented in Chapter 3. The 95% prediction limits generated successfully accounted for approximately 95% of the "measured" data and 100% of the "true" data for each of the synthetic data sets, while the probabilistic RI values provided a better approximation of the true relative contributions of the models' inputs than the deterministic estimates did.