Chapter 1
Introduction

1.1 Motivation

Water resources engineering involves the planning, development and management of water resources in order to meet the water needs of society and the environment. Of overall concern is the hydrological cycle, as water moves from a natural source to its point of use and back, taking into account the environmental processes that act on the system during this cycle. Models play an integral part in water resources engineering; however, it is very difficult to simulate the behaviour of these natural systems due to the innumerable nonlinear and often poorly understood interactions that occur within them. Over the past 15 years, there has been a growing interest in artificial neural networks (ANNs) for simulating, forecasting and predicting many different aspects of the hydrological cycle, as it is often considered that they provide the best model of a water resource system in the face of these difficulties.

ANNs have the ability to extract a relationship between a number of causal input variables and a dependent variable of interest from available characterising data, without the need for restrictive assumptions about the relationship under study. Their nonlinear and flexible functional form makes it possible to model any continuous function to an arbitrary degree of accuracy, and as such ANNs are often considered to be universal function approximators. Furthermore, ANNs are relatively easy to use and have made it possible for water resources practitioners to model complex nonlinear processes without the need for sophisticated statistical techniques. The capability of ANNs to model hydrological and water resources variables has been demonstrated in numerous applications, including rainfall-runoff modelling (*Hsu et al.*, 1995; *Dawson and Wilby*, 1999; *Dibike and Solomatine*, 2001; *Anmala et al.*, 2000), streamflow prediction (*Zealand et al.*, 1999; *Imrie et al.*, 2000; *Yitian and Gu*, 2003; *Dolling and Varas*, 2003) and water quality forecasting (*Maier and Dandy*, 1996; *Zhang and Stanley*, 1997; *Whitehead et al.*, 1997; *Maier* *et al.*, 1998). A comprehensive review of such applications can be found in *ASCE Task Committee* (2000a,b), *Maier and Dandy* (2000a) and *Dawson and Wilby* (2001).

Despite the increasing use of ANNs in water resources modelling, they are still viewed with some scepticism by users of more conventional statistical and knowledge-based modelling methodologies. The main premise in support of ANNs is their ability to generalise solutions to new examples from previous ones. However, this ability is reliant upon the proper estimation of a governing set of parameters that characterise the underlying system. These parameters have no physical interpretation and their values must be determined by calibration with a finite set of noisy data, which itself is performed solely by minimisation of predictive error and does not provide any means of incorporating knowledge of the system into the model. Furthermore, calibration of ANNs is a multidimensional nonlinear optimization problem, which is far from being straightforward. Determining the optimum level of complexity required to model a given problem is one of the most difficult tasks in the development of an ANN. Therefore, ANNs typically contain many more free parameters, or degrees of freedom, than conventional statistical or conceptual models, and minimising the predictive error often results in a model that 'overfits' the calibration data. Alternatively, optimisation algorithms used for calibration may become trapped in one of the many local optima that typically exist on the complex error surface, rather than finding the global minimum. To make matters worse, once an ANN has been calibrated, any explanation of its internal behaviour is only revealed back to the user in the form of an "optimal" parameter vector, which is difficult to interpret as a functional relationship. Consequently, there is no direct way to validate the model in terms of its physical plausibility.

Given these problems, it is very difficult to use ANNs confidently in any context other than interpolation. However, when used for prediction and forecasting purposes, it is generally inevitable that the model will also be required to extrapolate. In order to increase the confidence in ANN predictions, and hence improve their usability in water resources applications, it is necessary to acknowledge and quantify the uncertainty associated with estimating appropriate parameter values. As stated by *Maier and Dandy* (2000a) in the concluding paragraph of their review on the use of ANNs for water resources modelling:

> A further challenge is the incorporation of uncertainty into ANN models. Until now, ANN models in the field of water resources have been almost exclusively deterministic. However, it is well documented that many water resources models are subject to inherent, model and parameter uncertainties. Consequently, techniques for dealing with uncertainty should be considered

in the development of ANN models.

Accordingly, the primary motivation of this research is to incorporate uncertainty into ANNs used for water resources modelling. In recent years, Bayesian methods have been increasing in popularity for this purpose in various fields, including water resources modelling with more traditional models. However, a full Bayesian framework has not yet been extended to ANNs used for water resources modelling and, therefore, this will be the main focus of this thesis.

In addition to providing a confidence measure for the predictions, it is expected that the Bayesian framework will also help to overcome the difficulties in selecting an "optimal" set of model parameters and will aid in the design of a model with the appropriate level of complexity. To properly investigate these advantages, the new Bayesian neural network methodology presented in this research is compared to a "state-of-the-art" deterministic ANN modelling approach. Therefore, current best practice deterministic ANN development methods will be reviewed, and where limitations are identified, improvements made, in order to devise a state-of-the-art deterministic approach. Furthermore, to demonstrate the advantages of the new Bayesian neural network methodology in a practical setting, two water resources case studies are considered. These include forecasting salinity and cyanobacteria concentrations in a river.

1.2 Research Objectives

The overall objective of this research is to use Bayesian methods to help overcome some of the limitations that prevent ANNs from becoming more widely accepted and reaching their full potential as reliable water resources models, namely the lack of consideration of prediction uncertainty, the difficulty in estimating appropriate parameter values, the difficulty in selecting the optimum complexity and the lack of an objective method to properly validate the model and interpret the relationship modelled. In order to meet this overall goal, a number of primary and secondary objectives will be addressed. The primary objectives include:

- The development of a Bayesian framework applicable to ANNs used for water resources modelling, that incorporates:
  - a method to identify the distribution of plausible model parameters; and
  - a method to identify the optimal (or near optimal) ANN structure for a given case study.

- Comparison of the Bayesian approach to state-of-the-art deterministic ANN development approaches to determine the advantages and limitations of the developed methods.
- Application of the Bayesian ANN development approach to real water resources case studies to demonstrate the practical advantages and limitations of the Bayesian methods.

The secondary objectives are those not directly related to the Bayesian framework including:

- The assessment, comparison and, if necessary, improvement of currently used deterministic ANN development methods to devise a state-of-the-art approach.
- The development of an objective validation framework based on the modelled relative contributions of the predictor variables in generating the response variable. This should involve the comparison and, if necessary, improvement of existing input importance measures for use in the framework.

1.3 Layout and Contents of Thesis

A background to this research is presented in Chapter 2. This includes background information on ANN concepts (Sections 2.2.1 to 2.2.3) and Bayesian methodology (Section 2.3.1), as well as a discussion of current practices, strengths and limitations in both of these areas resulting from a review of the relevant literature (Sections 2.2.4 to 2.2.5 and 2.3.3 to 2.3.6).

In Chapter 3, deterministic methods applied to ANNs used for water resources modelling are reviewed and, if necessary, improved, in order to devise a state-of-the-art deterministic ANN approach. Methods currently employed at each stage of the model development process are reviewed in Section 3.2 to determine best practice approaches and identify any areas requiring improvement or further assessment. The state-of-the-art approach adopted throughout this research is summarised in Section 3.3, together with the model development steps identified as requiring further analysis. Various methods for carrying out these steps are investigated and compared in Section 3.4 using three synthetic data sets. Finally, the conclusions made based on the investigations conducted are presented in Section 3.4.7

The new Bayesian ANN framework developed in this research is presented in Chapter 4. The 'training and prediction' component of the framework is discussed in Section 4.2, which includes a background to the use of Markov chain Monte Carlo (MCMC) methods for estimating parameter distributions (Section 4.2.1), a review of MCMC methods previously used for ANN training (Section 4.2.2) and, finally, an outline of the proposed Bayesian training and prediction approach (Section 4.2.3). In Section 4.3, the 'model selection' component of the framework is proposed, based on a review of available Bayesian model selection methods (Section 4.3.1) and those previously applied for selecting the optimum complexity of an ANN (Section 4.3.3). In Section 4.4, the details of each of these components are determined and the overall proposed methodology is assessed through investigations carried out on synthetic data.

In Chapters 5 and 6, the proposed Bayesian and state-of-the-art deterministic ANN development methods are compared when applied to two real-world water resources case studies, which involve forecasting salinity and cyanobacteria concentrations in a river, respectively. Both of these cases are considered to be good benchmark studies against which the relative merits of each of the ANN development approaches can be assessed in a real-world context.

Finally, the overall contributions, conclusions and recommendations of this research are given in Chapter 7.

Chapter 2
Research Background

2.1 WATER RESOURCES MODELLING

Models are required in almost all areas of water resources planning and management (*Wurbs*, 1995). They enable the response and behaviour of a system to be investigated under various physical conditions, which is generally either impossible or infeasible to do in the real world. Therefore, they play a vital role in areas such as river regulation and management, hydraulic infrastructure design and water quality protection.

Water resources modelling will inevitably involve the use of some type of hydrological model. The types of models currently used to model hydrological systems can be broadly categorised into three main groups: physically-based, conceptual and empirical (*ASCE Task Committee*, 2000a; *Dawson and Wilby*, 2001). Physically-based models aim to represent the underlying physics of the system by using a series of partial differential equations to describe, as best they can, the change of state (e.g. mass, energy) of the system over time. In theory, once developed, these models should have a wide range of applicability (e.g. on data outside the domain of those used to develop the model and on ungauged systems with known or assumed characteristics), as they are based on fundamental physical relationships. The parameters of physically-based models are directly related to catchment characteristics and, thus, give useful insight into the system under investigation. However, the development of a physically-based model is very data intensive, requiring catchment specific spatial and temporal data to describe the physical characteristics of the system (*Maier and Dandy*, 2000b; *ASCE Task Committee*, 2000a). These types of data are generally not available; therefore, in many cases intensive data collection programs are required which can be both costly and time consuming, while in other cases typical parameter values are selected from manuals or textbooks and may provide little reflection of the actual values (*Reckhow*, 1999). Furthermore, development of a physically-based model requires that all of the physical processes occurring within the system are understood sufficiently well to be described mathematically (*Maier and Dandy*, 2000b). However, there still remain many components of hydrological systems that are only poorly understood or are simply too complex to model accurately with mathematical relationships. Consequently, even the most complex physically-based models are extreme simplifications of reality (*Reckhow*, 1999). Additionally, by attempting to simulate all of the processes that occur within a system, physically-based models are subject to overparameterisation (*Beven*, 1989) and parameter redundancy (*de Vos and Rientjes*, 2005); thus the complexity of these models may, in fact, hinder model performance.

In order to overcome some of the limitations of physically-based models, conceptual models only aim to represent key components of the hydrological system using simplified descriptions of the physical mechanisms that occur (*de Vos and Rientjes*, 2005). Conceptual models are therefore simpler than their physically-based counterparts, requiring fewer parameters, and are also less data intensive. The hydrological system is commonly conceptualised as a series of interconnected water stores, where empirical relationships are used to describe the recharge and depletion processes that occur within and between them (*Kokkonen and Jakeman*, 2001). While still based on conservation of mass, albeit on a larger temporal and spatial scale, calibration of conceptual models with historical data is required to estimate the parameters that govern the empirical equations. In general, conceptual models do not exploit previously measured values of the output (i.e. streamflow), except for calibration purposes, and assume that observed values of the independent inputs (e.g. rainfall and evaporation), together with the model structure, are sufficient to describe the evolution of the system over time (*Toth and Brath*, 2002). However, errors in the measured data, together with the various simplifying assumptions made concerning the responses of the individual system components, introduce some level of error into the predicted response of the system components (*Beven*, 1989). This error is then propagated through the modelled evolution of the system and, thus, the lead time for predictions made by conceptual models is limited (*Toth and Brath*, 2002). Furthermore, both conceptual and physically-based models require continuous data sets and cannot handle gaps in the data. If data are missing, which is often the case, predicted values of the inputs must be used (*Heneker*, 2002), further adding to the errors introduced into the model.

Empirical models are data-driven with the aim being prediction rather than explanation (*Grant et al.*, 1997). In other words, these models are developed primarily to extract information contained in a set of observed data and use it to characterise system response, rather than directly attempting to represent the physical processes occurring within the hydrological system (*Kokkonen and Jakeman*, 2001; *Toth and Brath*, 2002). Empirical models are often referred to as "black-box" models, as inputs are presented to the model and outputs are generated, with little regard given to the actual mechanisms being modelled (*ASCE Task Committee*, 2000a; *Toth and Brath*, 2002). However, as these models do not require in depth consideration of the underlying physical laws, they do not suffer from the same drawbacks as physically-based and conceptual models, that arise due to an inadequate description of the physical processes. Furthermore, empirical models typically use historical values of the target variable as inputs and are thus less affected by error propagation through the model (*Toth and Brath*, 2002). Therefore, empirical models are suited to modelling complex systems where the underlying relationships are unknown or difficult to describe and where observed data are abundant (*Qi and Zhang*, 2001). Traditionally, the types of empirical techniques used for hydrological modelling have included Box-Jenkins time series methods, and linear and nonlinear regression (*Hipel and McLeod*, 1994; *Maier and Dandy*, 2000b). More recently, machine learning techniques with routes in artificial intelligence, which include, for example, model trees (MTs), support vector machines (SVMs) and artificial neural networks (ANNs), have proven to be successful empirical methods for many water resources applications (*Solomatine*, 2002). Of these techniques, ANNs are by far the most popular and have received considerable attention as a water resources modelling approach since the first publications in this field appeared in 1992 (*French et al.*, 1992; *DeSilet et al.*, 1992). The promising results of these initial studies led to numerous comparisons between ANNs and more conventional hydrological modelling methods, where in most cases it was shown that ANNs can perform at least as well as, if not better than, many traditional models (*Hsu et al.*, 1995; *Shamseldin*, 1997; *Tokar and Johnson*, 1999; *Dawson and Wilby*, 1999; *Toth et al.*, 2000; *Thirumalaiah and Deo*, 2000). As a result, the popularity of ANNs in the field of water resources modelling has increased dramatically since their first introduction (*Maier and Dandy*, 2000a).

2.2 Artificial Neural Networks (ANNs)

2.2.1 Background and Description

ANNs were first developed in the 1940s when they were originally designed to mimic the functioning of the brain (*McCulloch and Pitts*, 1943). However, it has only been in the last 20 or so years, since the development of new calibration techniques and the increase in computational power, that their popularity in various fields as prediction tools has blossomed (*Maier and Dandy*, 2000a; *de Vos and Rientjes*, 2005). ANNs are mathematical models composed of a number of highly interconnected processing units called "nodes" or "neurons", which, individually, carry out rather simple and limited computations. However, collectively as a network, complicated computations can be performed due to the connectivity between the nodes and the way in which information is passed through and processed within the network (*Flood and Kartam*, 1994). ANNs are nonparametric, or "model free", and as such are able to model both linear and nonlinear functions without prior specification of the functional form of the model (*Qi and Zhang*, 2001). Therefore, when used for prediction, which is the focus of this thesis, ANNs can be considered as a very general form of nonlinear regression model (*Castellano-Méndez et al.*, 2004).

There are many different types of ANNs in terms of structure and mode of operation (*Flood and Kartam*, 1994). These are generally classified according to network topology and type of connectivity between the nodes, the type of data used, the way in which the network 'learns' the underlying function and the computations performed by each node (*Sarle*, 2002). Multi-layer perceptrons (MLPs) are the most popular and widely used ANN structure for regression problems (*Cheng and Titterington*, 1994; *Zhang et al.*, 1998), including those relating to water quality and quantity (*Maier and Dandy*, 2000a; *Dawson and Wilby*, 2001). Therefore, MLPs are the focus of this research and throughout this thesis the term "ANN" will refer to an MLP unless stated otherwise.

2.2.2 Multi-Layer Perceptrons (MLPs)

Multi-layer perceptrons, as the name suggests, are made up of several layers of nodes. As shown in Figure 2.1, the nodes are arranged into an input layer, an output layer and one or more intermediate layers, called 'hidden' layers. Information in the form of a

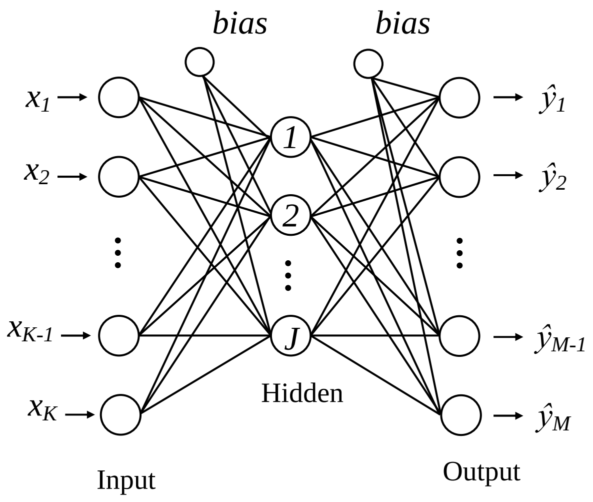

Figure 2.1 Layer structure of a multi-layer perceptron.

vector of observed data values is passed through the network, generally in a forward direction (feedforward), from one layer to the next. The input layer contains a node corresponding to each input variable and information is received at this layer as a vector of real values $\mathbf{x}^K = (x_1, \ldots, x_K)$. The values are then transmitted to each node in the next layer via weighted connections, where the weight determines the strength of the signal. At the nodes, the weighted values from the previous layer are summed together with a weighted bias. The result is then passed through a (possibly) nonlinear transfer, or activation, function to generate an activation level for that node. The activations are then transmitted to the subsequent layer and the process is continued until the information reaches the output layer. The activation level generated at the mth output node is the prediction $\hat{y}_m$. Figure 2.2 shows the operation of a single hidden layer node.

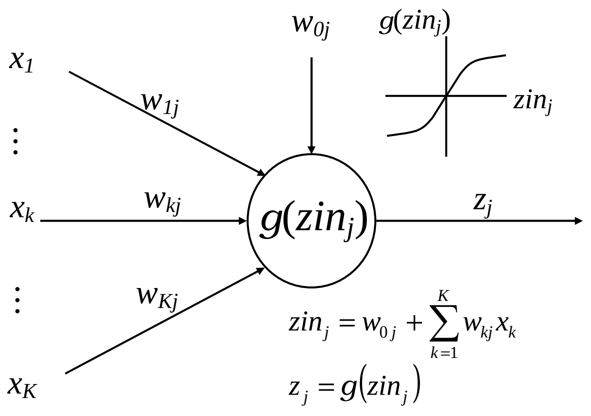

Figure 2.2 Operation of a single node.

The connection and bias weights, $\mathbf{w}$, are the free parameters of the network and, in order for the network to perform the desired function, appropriate values of these weights must be estimated. The complexity of an ANN is dependent on the number of weights, or dimension of the weight vector d, which is determined by the number of nodes in the network. The input and output layer nodes are fixed according to the number of input and output variables, respectively; however, the number of nodes in the hidden layers is flexible and must be specified by the modeller. Therefore, model complexity is adjusted by increasing or decreasing the number of hidden nodes. The complexity of the model is also dependent on the choice of activation function $g(\cdot)$, which may be any continuous differentiable function. The most commonly used activation functions are sigmoidal type functions, such as the hyperbolic tangent (tanh) and logistic functions (*Maier and Dandy*, 2000a).

2.2.3 Training

ANN “training” can be compared to the calibration of coefficients in statistical models (*Maier and Dandy*, 2001). During training the aim is to find values for the connection and bias weights so that the outputs produced by the ANN approximate the training data well. However, it is not sufficient to just reproduce solutions in the training data set; rather, a generalised solution applicable to all examples is required (*Flood and Kartam*, 1994). ANNs, like all mathematical models, work on the assumption that there is a real function underlying a system that relates a set of K independent predictor variables $\mathbf{x}^K$ to M dependent variables of interest $\mathbf{y}^M$. Therefore, the overall aim of ANN training is to infer an acceptable approximation of this relationship from the training data, so that the model can be used to produce accurate predictions when presented with new data. Thus, if the function relating the measured target data to the model inputs is given by:

$$\mathbf{y}^M = f\left(\mathbf{x}^K, \mathbf{w}\right) + \epsilon \tag{2.1}$$

where $f(\cdot)$ is the function described by the ANN, $\mathbf{w}$ is a vector of weights that characterise the data generating relationship and ϵ represents random measurement noise, the aim is to find the best estimate of the weight vector, which is denoted by $\hat{\mathbf{w}}$. This is typically done by iteratively adjusting and optimising the weights such that some function of the difference between the measured target data $\mathbf{y}^M$ and the predicted outputs $\hat{\mathbf{y}}^M$ (e.g. the sum squared error $E_{\mathbf{y}} = \frac{1}{2}\sum(\mathbf{y}^M - \hat{\mathbf{y}}^M)^2$) is minimised. The term “generalise” is used to imply that the functional form $f(\cdot)$ will not be explicitly revealed, but will instead be represented by the estimated weights $\hat{\mathbf{w}}$ (*ASCE Task Committee*, 2000a).

Following training, the performance and generalisation ability (‘generalisability’) of the ANN is checked by subjecting it to an independent validation data set.

2.2.4 Advantages

The increased popularity of ANNs for water resources modelling can be attributed to a number of factors. Firstly, as mentioned above, modelling hydrological variables with more conventional physically-based or conceptual models may be limited by a poor understanding of the complex interactions that are involved in the process (*ASCE Task Committee*, 2000a). Therefore, there are clear advantages to using empirical approaches such as ANNs over knowledge-based methods, as they have the ability to extract the input-output relationship from data without requiring an in-depth knowledge of the physics occurring within the hydrological system (*Zhang et al.*, 1998). Secondly, ANNs are able to model nonlinear relationships (*Hill et al.*, 1994). The majority of hydrologic processes are highly nonlinear in nature; however, when using empirical methods, water resources practitioners have typically only used simple linear regression or time series models (*Maier and Dandy*, 2000a). While there may be some advantages to using simple linear models in terms of implementation and interpretation, these models are limited by the fact that they are unable to capture the complex nonlinear nature of hydrological problems (*Zhang et al.*, 1998). Thirdly, ANNs have the ability to generalise solutions to new examples from previous ones. This makes ANNs relatively insensitive to noise in the data and, as inputs are considered as variables, without regard for their membership to a time series, ANNs can handle incomplete, discontinuous data sets (*Dawson and Wilby*, 2001; *Castellano-Méndez et al.*, 2004). Thus, ANNs have an advantage over both conceptual and Box-Jenkins time series models, which do not have this generalisation ability (*Zealand et al.*, 1999). Finally, ANNs are relatively easy to use and are a highly flexible modelling approach (*Maier and Dandy*, 2000a; *Gaume and Gosset*, 2003). The advantages presented thus far are not exclusive to ANNs, and in fact, sophisticated statistical methods capable of performing similar modelling tasks to those performed by ANNs (i.e. complex, nonlinear) have been available for many years prior to the introduction of ANNs (*Maier and Dandy*, 2000a). However, the need to prespecify the functional form of these model-based approaches has limited their use, since there are too many possible nonlinear patterns, making it very difficult to formulate an appropriate nonlinear model (*Zhang*, 2001); hence, the preference for simple linear statistical models (*Maier and Dandy*, 2000a). On the other hand, ANNs are model free, allowing them to model both linear and nonlinear relationships without requiring any restrictive assumptions about the functional form (*Qi and Zhang*, 2001), giving them more general appeal than less flexible models (*Zhang*, 2001). Furthermore, the complexity of ANNs can be easily adjusted by altering the number and configuration of hidden nodes and/or the types of activation functions used, and ANNs can be easily extended from being univariate models to multivariate models (*Maier and Dandy*, 2000a). In fact, the nonlinear and flexible functional form of ANNs makes it possible to model any continuous function to an arbitrary degree of accuracy (*Zhang et al.*, 1998), and as such ANNs are often considered to be 'universal function approximators' (*Cheng and Titterington*, 1994; *Bishop*, 1995; *ASCE Task Committee*, 2000b).

2.2.5 Issues and Limitations

Despite the increasing use of ANNs as hydrological models, a number of limitations and methodological issues have caused them to be viewed sceptically by users of conventional modelling methodologies (*Chatfield*, 1993a; *Gorr*, 1994; *Hill et al.*, 1994; *Gaume* *and Gosset*, 2003). During the development of an ANN there are many alternatives available to the modeller at each stage of the process (*Maier and Dandy*, 2000b). While this provides great flexibility, there are no well established specification or diagnostic tests, such as those commonly employed in traditional modelling, and therefore it is difficult to provide any confidence that the developed model has a plausible theoretical interpretation (*Refenes and Zapranis*, 1999). Failure to carefully consider 'good practice' model identification principles is the main reason for contradictory or inconclusive results about the predictive capability of ANNs in the literature (*Zhang*, 2001). However, due to the lack of ANN development theory, and perhaps encouraged by the ability of an ANN to develop a solution to a problem automatically, ANN users have commonly relied on arbitrary decisions for many key model development steps (*Flood and Kartam*, 1994; *Maier and Dandy*, 2000a). Based on a review of the current literature, it is considered that the most significant issues facing the wider acceptance of ANNs are generalisability, interpretability and uncertainty.

2.2.5.1 Generalisability

As discussed in Section 2.2.4, one of the main advantages of ANNs is the ability to infer a generalised solution to a problem (*Flood and Kartam*, 1994), yet, actually obtaining good generalisation can be very complicated (*Sarle*, 2002). Firstly, to achieve appropriate generalisation, the training data set must be of sufficient quality and quantity such that it contains enough reliable information relating the input variables to the targets (*ASCE Task Committee*, 2000a; *Tokar and Johnson*, 1999). This requires selection of the necessary model inputs, which, for a hydrological system, involves deciding which are the important causal variables and which time lagged values of these variables are necessary to account for the time structure in the data (*Maier and Dandy*, 2001). As the problems modelled by ANNs are typically poorly understood, selecting the correct inputs is a difficult task, yet it is also one that has a significant impact on the prediction ability of an ANN: the inclusion of unnecessary inputs can confuse the training process by adding irrelevant information, whereas omitting important inputs results in a loss of information that leads to poor prediction performance (*ASCE Task Committee*, 2000a). Furthermore, to achieve generalisation, the training data must be a representative sample of the population from which the data were generated. Due to the lack of physical principles incorporated into ANNs, they generally do not perform well on data outside the domain of that used to calibrate the model (i.e. extrapolation), as there is no information regarding the form of the solution surface outside of this region (*Flood and Kartam*, 1994; *Toth and Brath*, 2002). As stated by *Bienenstock and Geman* (1994):

> ...statisticians know that generalisation (good performance on samples not in the training set) depends almost entirely on the extent to which the training set is representative, and/or the structure of the problem happens to accommodate the models used. It is too much to expect statistical methods to "discover", by themselves, complex and nontrivial structure...

Consequently, it is important to include enough data in the training set such that extrapolation is avoided (*Sarle*, 2002; *ASCE Task Committee*, 2000b). However, this may not always be possible, as many hydrologic records do not go back far enough.

Secondly, generalisability is closely related to model complexity; thus the network geometry that provides the appropriate level of complexity for the problem under consideration must be selected. As the number of input and output nodes are fixed by the number of input and output variables in the model, network geometry is determined by the number of hidden layers and hidden layer nodes in the network. Selecting the optimal number and configuration of hidden layer nodes is one of the most critical and most difficult tasks in designing an ANN (*Qi and Zhang*, 2001). It is highly problem dependent and, although a number of 'rules of thumb' have been suggested in the literature that relate the number of training samples and the number of connection weights, there is no universally accepted theoretical basis to guide selection (*Maier and Dandy*, 2000a; *de Vos and Rientjes*, 2005). As a consequence, network geometries are commonly found using trial-and-error approaches (*Zhang*, 2001). In determining the number of free parameters of a network, a compromise is required between the ability to approximate the underlying function and the generalisation ability of the network (*Dawson and Wilby*, 2001). With too few hidden nodes, the network may not have sufficient free parameters to correctly estimate the complex relationship between inputs and outputs and as a result the underlying function is approximated poorly, or 'underfitted'. On the other hand, too many hidden nodes can lead to 'overfitting', where, rather than inferring the general underlying trend, noise and spurious features of the training data are learnt, which results in poor generalisation on samples not contained in the training data (*ASCE Task Committee*, 2000a; *Maier and Dandy*, 2000a; *Dawson and Wilby*, 2001). While overfitting also occurs in conventional models with many parameters, it is more common in ANNs due to the typically large number of parameters set to be estimated (*Zhang*, 2001). Furthermore, it has been suggested that model selection criteria commonly used to select the optimal level of complexity for conventional statistical models, such as Akaike's information criterion (AIC) and the Bayesian information criterion (BIC), over-penalise complexity in ANNs and emphasise overly simple models (*Qi and Zhang*, 2001).

Finally, good generalisation relies on appropriate estimation of the network weights. ANN training is a multidimensional nonlinear optimisation problem and obtaining good estimates of the network weights can be problematic. The nonlinearity and high dimensionality of the problem can lead to the existence of multiple local and flat optima on the solution surface (*van der Smagt and Hirzinger*, 1998). Training algorithms may become trapped in local minima rather than converging on the global solution and, although sophisticated global optimisation algorithms have been developed, there is still no algorithm that can guarantee global convergence in a reasonable amount of time (*Zhang et al.*, 1998). Furthermore, as discussed above, ANNs are susceptible to overfitting (also referred to as 'overtraining'), which results in weight estimates that do not provide a general representation of the underlying function. Various methods can be used during training to prevent overfitting. These can generally be categorised into two main groups: those that reduce the effective size of the network, and those that stop training before overfitting can occur. Methods belonging to the latter group are those most commonly used to avoid overfitting (*Sarle*, 1995). Such methods ensure that the network has sufficient flexibility (degrees of freedom) to fit the data accurately if it were allowed to train to convergence; however, to prevent overfitting, training is stopped early (*De Veaux et al.*, 1998). A typical approach employed to determine when to stop training is known as 'cross-validation' (*Ripley*, 1994), which involves the use of a test data set to determine when the network is beginning to overfit the training data (*Dawson and Wilby*, 2001). In the initial stages of training, errors for both the training and test data sets should decrease at approximately the same rate. When overfitting begins, the training errors continue to decrease but test set errors begin to rise (*ASCE Task Committee*, 2000a), hence training is stopped at this point. However, as noted by *Ripley* (1994), there are a number of difficulties associated with the use of cross-validation, including:

1. There is no guarantee that the path taken by the optimisation algorithm is sensible.

2. The test set error often rises and falls a number of times making it difficult to determine when the best point on the path has been reached.

3. The use of a test set wastes data.

In regard to this last point, partitioning the available data into a greater number of subsets reduces the amount of information contained in the training data. In some cases, there may not be sufficient data available to allow for a test set. In such cases, training may simply be stopped when the training error has reached a sufficiently small value or when it ceases to decrease significantly (*ASCE Task Committee*, 2000a; *Maier and Dandy*, 2000a); however, this requires the subjective choice of what is a 'sufficiently small' error.

Methods belonging to the former group include regularisation and pruning algorithms. These methods begin with a network that is flexible enough to fit the training data accurately and subsequently remove or disable unnecessary weights and/or nodes during training (*Maier and Dandy*, 2000a). Regularisation involves the use of a penalty term which is added to the error function in order to penalise model complexity and ensure smoother mappings (*Bishop*, 1995). The regularised error function to be minimised is therefore given by:

$$E_{regularised} = E_{\mathbf{y}} + \alpha E_{\mathbf{w}} \tag{2.2}$$

where $E_{\mathbf{y}}$ is the error on the data, $E_{\mathbf{w}}$ is the penalty term that measures model complexity and α is a regularisation coefficient that determines the influence of the penalty term on the solution. A commonly used regulariser is known as 'weight decay' and has the form (*Bishop*, 1995):

$$E_{\mathbf{w}} = \frac{1}{2}\sum_{i=1}^{d} w_i^2 \tag{2.3}$$

This term penalises large weights, forcing them to decay exponentially to zero, hence reducing the effective model complexity. However, determining the magnitude of the regularisation coefficient α is difficult, yet critical to the generalisation ability of the network (*Sarle*, 2002); if it is chosen to be too small, overfitting may still occur, whereas if it is chosen to be too large, underfitting is a problem (*Neal*, 1996a). It has been noted by *Anders and Korn* (1999) that the regularisation coefficients usually do not result from theoretical reasoning but are instead set in an ad hoc fashion, and moreover, may involve the use of a cross-validation, or testing, data set, which again results in inefficient use of the available data (*Bishop*, 1995). Furthermore, to ensure good generalisation, different types of weights in the network usually require different regularisation coefficients (*Bishop*, 1995). According to *Sarle* (2002), for a one hidden layer MLP, at the very least two different regularisation coefficients are required for the input-hidden and hidden-output weights. In general, pruning algorithms are used to remove weak connections (i.e. small absolute weights) (*Abrahart et al.*, 1999) or elements (weights or nodes) that have minimal effect on the error function (i.e. elements to which the error function is insensitive) (*Reed*, 1993). However, a question which then arises is at what threshold value should elements be removed from the network (*Olden and Jackson*, 2002)?

Ideally, achieving good generalisability would involve selecting an ANN of optimal complexity, where optimality is defined as the smallest network that adequately captures the underlying relationship, and then estimating its weights from a set of good quality training data, which properly represent the population from which the data were generated. However, when modelling water resources systems, there are too many factors that are unknown or cannot be controlled. When the system is poorly understood, it cannot be guaranteed that all important inputs are included in the model and that a network providing the optimal level of complexity will be selected. Furthermore, modellers of environmental systems often have to make do with available data, either due to time or monetary constraints, and these data may not be of sufficient quality and/or quantity to derive appropriate estimates for the weights. Currently used ANN development methods are either incapable of ensuring good generalisability, given these factors, or due to the lack of systematic guidelines, are not appropriately employed to develop the best model of the system given the available data.

2.2.5.2 Interpretability

ANNs are much less interpretable than other empirical modelling methods, such as traditional times series and regression models (*Hill et al.*, 1994). Any explanation of the internal behaviour of an ANN is only revealed back to the user in the form of an "optimal" weight vector, which is difficult to interpret as a functional relationship. Consequently, ANNs are frequently criticised for operating as "black-box" models (*ASCE Task Committee*, 2000a), where solutions to a problem are automatically generated with no consideration or explanation of the physical process being modelled (*Sudheer and Jain*, 2004; *Olden and Jackson*, 2002).

As the data used to develop ANNs contain important information about the physical process being modelled, it is generally implied that once an ANN has been trained and validated, the trained model represents the physical process of the system (*Sudheer*, 2005). However, a consequence of the training problems presented in the previous section is that many combinations of weights may result in similar network performance. Due to the poor interpretability of ANN weights, there is no way to directly distinguish which combination of weights best approximates the underlying relationship. While the calibration problems presented also apply to many conventional models to some extent, it is usually easier to validate the modelled function against a priori knowledge. Black-box models are generally undesirable as predictive models, as it is difficult to determine whether the model will behave correctly when presented with previously unseen data; in other words, it is hard to trust their reliability (*Benítez et al.*, 1997). Thus, due to their inability to explain, in a comprehensible way, the process by which the model outputs are generated, the utility of ANNs as prediction tools is limited (*Andrews et al.*, 1995).

Recently, efforts have been made to develop hybrid ANN/knowledge-based models which exploit the strengths of each individual approach (*See and Abrahart*, 2001; *Ganguly*, 2002; *Coulibaly et al.*, 2005). A common approach for developing a hybrid ANN is to use a physical model as the basis, with the ANN calculating unknown or immeasurable parameters (*Aguiar and Filho*, 2001). This way, aspects of the system that are well understood are described by mathematical equations, whereas the ANN can be used to estimate the unknown components (*Lee et al.*, 2002). An alternative approach is to use the ANN to model the residuals between the physical model and the target data. As such, the ANN models the gap in knowledge between the physical model and the actual process (*Côté et al.*, 1995). A further alternative is to use a weighted combination of the physical model and ANN outputs, where the weights assigned to each model are determined by considering the variances of the forecast errors of the individual models (*Coulibaly et al.*, 2005). Each of these methods results in a “grey-box” model of the system (*De Veaux et al.*, 1999), where the overall model has physical basis and interpretation, yet the limitations due to an inadequate description of the complex physical system are overcome using the strengths of data-based black-box methods. However, a requirement for the development of hybrid ANNs is that there exists a knowledge-based (physical or conceptual) model of the system.

In order to increase the transparency of ANNs and overcome their black-box image in the absence of a knowledge-based model, extraction of the knowledge locked up within a trained ANN has become an active and evolving discipline (*Tickle et al.*, 1998). Since the late 1980s, a number of methods have been proposed in the literature for interpreting what has been learnt by an ANN (*Montaño and Palmer*, 2003); however, there is currently no widely accepted method for doing this. The hidden units in ANNs can be thought of as representing “derived features” (*Craven and Shavlik*, 1997) of the modelled system. Therefore, recent attempts have been made in the field of hydrological modelling to understand the relationships represented by the hidden nodes by relating the outputs of the individual hidden neurons to components of the hydrological system (*Wilby et al.*, 2003; *Jain et al.*, 2004; *Sudheer and Jain*, 2004). However, because of the distributed nature of ANNs, individual hidden units generally do not correspond well with features in the problem domain, e.g. the baseflow component of a flood hydrograph. Rather, these physical components are likely to be encoded across a number of hidden nodes, and similarly, each hidden node may partially represent a number of different system components (*Craven and Shavlik*, 1997).

On a more general level, the methods used to interpret trained ANNs can usually be categorised as those that translate the function modelled by the ANN into a set of symbolic rules that are easier to interpret by the user (i.e. rule extraction methods) (*Benítez et al.*, 1997), and those that aim to understand the function modelled by quantifying the strength of the relationships between individual inputs and the output (i.e. input importance measures) (*Sarle*, 2002). *Andrews et al.* (1995) and *Tickle et al.* (1998) review the relative merits of the various rule extraction methods available for ANNs. With the exception of *Maier et al.* (2000, 2001), who applied a neurofuzzy approach for predicting riverine cyanobacteria concentrations, there appear to have been very few attempts to apply rule extraction methods to ANNs in the field of water resources modelling. Possible reasons for this could be some of those noted by *Tickle et al.* (1998), including that the computational complexity of rule extraction algorithms may be a limiting factor on what is achievable from these techniques. Overall, there have been limited attempts to interpret ANNs trained to model water resources variables; however, those that have been made have tended to favour methods that indicate the influence of input variables on the output, such as sensitivity analysis (*Maier et al.*, 1998; *Jeong et al.*, 2001; *Walter et al.*, 2001), saliency analysis (*Abrahart et al.*, 2001) and perturbation analysis (*Sudheer*, 2005). All available methods for quantifying the importance of ANN inputs have limitations, and in acknowledging this, *Sarle* (2002); *Olden and Jackson* (2002); *Olden et al.* (2004) and *Gevrey et al.* (2003) have reviewed and compared such methods. However, while each of these comparisons provides a good reference, it is considered that they are inadequate to indicate a ‘best’ measure. The comparison carried out by *Sarle* (2002) is based on a simple, nonlinear function with weights of an MLP specified to fit the function. However, the weights specified are unrealistic of those that would result if the ANN was trained in the usual manner and it is therefore considered that the results of the comparison do not fairly represent the relative performances of different input importance measures under normal circumstances. It was argued by *Olden et al.* (2004) that the comparison by *Gevrey et al.* (2003) was based on empirical data, however, to establish the accuracy of the different measures the true correlation structure of the data needs to be known. Therefore, they carried out their own comparison using synthetically generated data with known correlation structure. However, this data set was generated by a linear function, and as one of the complicating factors of using certain input importance measures arises due to the “squashing” effect that nonlinear activation functions have on the weights (*Sarle*, 2002), it is considered that this comparison is also inadequate. In addition, one of the factors that limits the use of most input importance measures is that the inputs are required to be independent, meaning that a method must be used either before or during training to ensure that only relevant independent inputs remain in the network (*Féraud and Clérot*, 2002).

It is unlikely that there will ever be a method to perfectly summarise the relationship modelled by an ANN without considering the actual input-output function computed. Nevertheless, in order to validate ANN models against a priori knowledge or gain information about the physical system from ANNs, the best way to interpret the relationship modelled by an ANN needs to be identified. Comparative investigations that have attempted to do this in the past have been limited due to the way in which the investigations were carried out and, consequently, there is still no recommended 'best' method for improving the interpretability of ANNs, which is necessary for them to overcome their "black-box" image.

2.2.5.3 Uncertainty

In spite of the black-box characteristics of ANNs and the difficulty associated with obtaining good generalisability, ANNs in the field of water resources modelling have been almost exclusively deterministic, with little regard given to the uncertainty associated with the predictions generated (*Maier and Dandy*, 2000a). It is widely accepted that water resources models are subject to uncertainty due to the stochastic nature of natural processes, the limitations of a finite calibration data set and the inability of models to accurately describe these complex processes (*Beck*, 1987; *Chow et al.*, 1988; *Kavetski et al.*, 2002). This is particularly the case for ANNs, whose parameter values are determined entirely by calibration, the problems of which were discussed in section 2.2.5.1.

Quantifying the uncertainty associated with water resources models is extremely important, as suppressing this information can create a false sense of security in the predictions generated. As a consequence, inappropriate design or management strategies may be implemented that can, in turn, result in significant social and economic loss (*Krzysztofowicz*, 2001). Thus, in order to increase the usability of ANNs in water resources modelling, the predictions need to be supported by an associated confidence measure, which indicates the quality of the predictions.

Prediction intervals may be used to express upper and lower limits between which a prediction is expected to lie at a given probability (*Chatfield*, 1993b). Such an interval measures the accuracy of an ANN's output with respect to the observed data (i.e. $\mathbf{y} -$

$\hat{\mathbf{y}}$) and should encompass a measure of the accuracy of the estimated system function (i.e. $f(\mathbf{x}, \mathbf{w}) - \hat{\mathbf{y}}$) and a measure of the accuracy of the observed data (i.e. ϵ in (2.1)) (*Heskes*, 1997). For a given model structure, estimating the prediction intervals is then a problem of estimating the distributions of ϵ and $\mathbf{w}$. It is generally assumed that ϵ is an independent normally distributed random variable with mean of zero and constant variance (i.e. $\epsilon \sim \mathrm{N}(0, \sigma_{\mathbf{y}}^2)$) (*Chatfield*, 1993b; *Tibshirani*, 1996; *De Veaux et al.*, 1998), therefore estimates are required for $\sigma_{\mathbf{y}}^2$ and $p(\mathbf{w}|\mathbf{y}, \mathbf{x})$, the conditional probability density function (PDF) of the weights given the observed data.

Several methods borrowed from nonlinear regression methodology have been applied to construct prediction intervals for ANNs. These are typically based on classical analysis (*White*, 1989; *Hwang and Ding*, 1997; *De Veaux et al.*, 1998), bootstrapping (*Tibshirani*, 1996; *Heskes*, 1997) and Bayesian methodology (*Buntine and Weigend*, 1991; *MacKay*, 1992a; *Neal*, 1992). Classical analysis techniques utilise a first-order linear approximation of the model function $f(\cdot)$ at the optimum weight values. This requires that the weights are uniquely identifiable, yet as pointed out by *Hwang and Ding* (1997), ANN weights are, by their nature, unidentifiable; therefore, classical methods are not directly applicable to estimate the uncertainty in the weight estimates. Nevertheless, they showed that the prediction limits calculated by this method were still asymptotically valid when the ANN was trained to convergence. However, *De Veaux et al.* (1998) demonstrated that standard classical methods were inaccurate when applied to estimating prediction intervals of ANNs. Firstly, they found that by ensuring convergence during training, overfitting occurred and, as a consequence, the limits were too narrow. Secondly, by stopping training early to prevent overfitting, it was found that $\sigma_{\mathbf{y}}^2$ was overestimated and the resulting prediction limits were too wide. Furthermore, the linear approximation requires calculation of the Hessian matrix (matrix of second derivatives of the error function with respect to the weights). This matrix can be nearly singular when the number of parameters is large, which leads to instability and prediction intervals that are much too wide in some cases (*De Veaux et al.*, 1998). To overcome these problems, they combined classical theory with weight regularisation to reduce the effective number of weights in the network.

Tibshirani (1996) compared the performance of a number of different methods for constructing prediction intervals based on classical techniques and on bootstrapping. Bootstrapping works by creating many pseudo-replicates of the training data set and re-evaluating new values of the network weights based on each different training set. The data are sampled *with replacement* to form training data sets of the same size as the available data. The standard error for the ith predicted value can then be calculated by:

$$s = \left\{ \frac{1}{B-1} \sum_{i=1}^{B} \left[f(\mathbf{x}_i, \hat{\mathbf{w}}^b) - f(\mathbf{x}_i, \cdot) \right]^2 \right\}^{1/2} \tag{2.4}$$

where B is the number of bootstrap samples, $\hat{\mathbf{w}}^b$ is the estimated weight vector for the bth sample and $f(\mathbf{x}_i, \cdot) = \sum_{i=1}^{B} f(\mathbf{x}_i, \hat{\mathbf{w}}^b)/B$. The advantages of this approach is that it is a distribution-free approach (*Chatfield*, 1993b) and does not require linearization of the modelled function, which may not be valid for a nonlinear model (*De Veaux et al.*, 1998). *Tibshirani* (1996) found that the bootstrap methods provided the most accurate estimates of the prediction variance and that the methods based on local linearization were often inaccurate due to convergence to local optima. Furthermore, it was found that prediction intervals could not be estimated by the classical techniques when the number of weights was large and weight decay was not used, due to difficulty in computing the Hessian matrix because of near singularities. However, a limitation of bootstrapping methods is that for each of the B samples, the ANN needs to be retrained. As B is typically in the range $20 \leq B \leq 200$ (*Tibshirani*, 1996) these methods are very computationally intensive (*Chatfield*, 1993b). Moreover, as noted by *Ossen and Rüger* (1998), local optima can lead to prediction intervals that are too wide.

Papadopoulos et al. (2001) compared the performance of classical, bootstrapping and approximate Bayesian methods for estimating prediction intervals when applied to an example where the distribution of ϵ was dependent upon the model inputs. Their results showed that the accuracy of the classical and bootstrapping methods was approximately equal. However, problems with calculating the Hessian matrix for classical methods were again reported in this study. Additionally, it was found that the bootstrapping method consistently overestimated the width of the prediction intervals, supporting the comment made by *Ossen and Rüger* (1998). Overall, the approximate Bayesian approach was found to give the most accurate prediction intervals. Unlike standard ANN approaches, Bayesian methodology is used to make predictions based on the posterior probability distribution of the weights $p(\mathbf{w}|\mathbf{y}, \mathbf{x})$, which explicitly accounts for the uncertainty associated with the weight estimates (*Bishop*, 1995). The method investigated by *Papadopoulos et al.* (2001) was considered to be *approximate* Bayesian because the posterior weight distribution was approximated by a Gaussian distribution. Exact Bayesian methods, on the other hand, do not make this assumption, rather, Markov chain Monte Carlo integration is used to estimate the distribution nonparametrically (*Neal*, 1992, 1996a). *Papadopoulos et al.* (2001) considered this exact Bayesian approach to be too computationally expensive and inappropriate for multidimensional, real-world applications.

Classical methods for estimating prediction limits are unsuitable for application to ANNs. These methods require a linear approximation of the nonlinear model function at the optimal weights, which, if not estimated correctly, can result in bounds that are either too narrow or too wide. Furthermore, calculation of the Hessian matrix required for the linearization may be difficult due to near singularities. Bootstrapping methods for estimating prediction limits can be time consuming and may be incorrect if affected by local minima in the error surface, resulting in bounds that are too wide. Approximate Bayesian methods may also be inappropriate due to the Gaussian approximation of the posterior weight distribution that is used. While an exact Bayesian approach for estimating prediction limits has previously been considered too computationally intensive, this method appears to be most suitable for ANNs and, with the increases in computer power, it is considered that this method warrants further investigation.

2.3 BAYESIAN METHODS

2.3.1 Background to Bayesian Methodology

The philosophy behind Bayesian inference is that any prior beliefs regarding unknown parameter values are updated based on new information contained in an observed set of data, to yield a posterior probability distribution of the parameters. This statement is usually referred to as *Bayes' theorem*. Put generally, if $\mathbf{y}$ is a set of observed data whose probability distribution depends on the values of a set of parameters θ, Bayes' theorem can then be used to infer the conditional distribution of the unknown θ given the observed data $\mathbf{y}$ as follows:

$$p(\theta|\mathbf{y},\mathcal{H}) = \frac{p(\mathbf{y}|\theta,\mathcal{H})p(\theta|\mathcal{H})}{p(\mathbf{y}|\mathcal{H})} \tag{2.5}$$

In this expression, the assumed model of the situation $\mathcal{H}$ is a conditioning statement upon which the probability for θ is based. The distribution $p(\theta|\mathbf{y}\mathcal{H})$, is called the *posterior* distribution of θ given the data $\mathbf{y}$, or simply, the "posterior". It describes what is known about the parameter values given knowledge of the data. On the other hand, $p(\theta)$, describes what is known about θ without knowledge of the data. It is known as the prior distribution of the parameters θ, or the "prior". The distribution $p(\mathbf{y}|\theta,\mathcal{H})$ is known as the "likelihood" of θ and it describes the information about θ contained in the data $\mathbf{y}$. It is this information that is used to update the prior distribution to obtain the posterior (*Box and Tiao*, 1973; *MacKay*, 1995a). The denominator $p(\mathbf{y}|\mathcal{H})$ is a normalising constant known as the 'evidence' of the model (*MacKay*, 1992a; *Rasmussen*, 2001) or the 'marginal likelihood' (*Chib and Jeliazkov*, 2001; *Titterington*, 2004) and is given by:

$$p(\mathbf{y}|\mathcal{H}) = \int p(\mathbf{y}|\theta,\mathcal{H})p(\theta|\mathcal{H})d\theta \tag{2.6}$$

As Bayesian inferences are based on *a priori* knowledge and conditional assumptions, they can be considered to be subjective. However, it is not possible to generalise about data without making assumptions (*MacKay*, 1995a), thus, conventional deterministic models are also subjective in a sense. The only difference is that Bayesian methodology makes explicit all assumptions that are made. As stated by *Lampinen and Vehtari* (2001), this is a considerable advantage of the Bayesian approach, as it gives a principled way to do inference when some of the prior knowledge is lacking or vague, so that one is not forced to guess values for attributes that are unknown.

2.3.2 Use of Bayesian Methods in Water Resources Modelling

In recent years, there has been an upsurgeance in the use of Bayesian methodology in various scientific fields (*Malakoff*, 1999), including hydrological and water resources modelling (*Kuczera*, 1983; *Kuczera and Parent*, 1998; *Beven and Binley*, 1992; *Romanowicz et al.*, 1994; *Reichert and Omlin*, 1997; *Omlin and Reichert*, 1999; *Bates and Campbell*, 2001; *Thiemann et al.*, 2001; *Kavetski et al.*, 2002; *Thyer et al.*, 2002; *Vrugt et al.*, 2003; *Marshall et al.*, 2004). While the importance of incorporating uncertainty analysis into water resources modelling has been emphasised by a number of authors (*Beck*, 1987; *Reckhow*, 1994; *Beven*, 1993; *Krzysztofowicz*, 2001), the limitations of classical statistical methods traditionally used for this purpose have also been noted (*Kuczera*, 1988; *Omlin and Reichert*, 1999; *Vrugt and Bouten*, 2002). In particular, classical methods relying on first-order linear approximations are typically unable to cope with the nonlinearity of hydrologic models (*Kuczera and Parent*, 1998), especially when the parameters are poorly identifiable (*Omlin and Reichert*, 1999).

Under the Bayesian paradigm, uncertainty in the model parameters is handled explicitly by estimating parameter distributions, rather than point values. Theoretically, this eliminates the need to linearly approximate the prediction intervals based on uniquely identified optimal parameter estimates, as the probability of a model's response y_{N+1}, given future input values $\mathbf{x}_{N+1}$ and the data used to calibrate the model $\mathbf{y}$, can be evaluated based on the entire posterior parameter distribution $p(\theta|\mathbf{y})$ by the integral:

$$p(y_{N+1}|\mathbf{x}_{N+1},\mathbf{y}) = \int p(y_{N+1}|\mathbf{x}_{N+1},\theta)p(\theta|\mathbf{y})d\theta \tag{2.7}$$

However, in practice, the high dimensionality of this integral makes its evaluation with conventional analytical or numerical integration techniques virtually impossible (*Kuczera*, 1988; *Marshall et al.*, 2004). Thus, approximate analytical solutions requiring the assumption of normally distributed parameters and either first-order (*Kuczera*, 1983, 1988) or second-order analysis (*Kuczera and Mroczkowski*, 1998) have been used in the past due to their computational efficiency. Again, these approximations can lead to a very poor approximation of prediction uncertainty for reasons noted.

With advances in computers, attention has now turned to more computationally intensive Monte Carlo based methods for estimating parameter uncertainty (*Kuczera and Mroczkowski*, 1998; *Gaume et al.*, 1998). Such methods include regionalised sensitivity analysis (*Hornberger and Spear*, 1981; *Beck*, 1987), generalised likelihood uncertainty estimation (GLUE) (*Beven and Binley*, 1992; *Beven*, 2001), and Markov chain Monte Carlo (MCMC) methods (*Kuczera and Parent*, 1998; *Bates and Campbell*, 2001; *Borsuk et al.*, 2001; *Vrugt et al.*, 2003; *Marshall et al.*, 2004). Each of these methods scan, in either a random or systematic way, the range of possible parameter vectors to identify those that give an acceptable result (*Gaume et al.*, 1998). After a sufficient number of trials, the acceptable parameter vectors sampled should converge to the posterior distribution of the parameters (*Beck*, 1987). However, as stated by *Brun et al.* (2001),

> Current applications of the Bayesian methodology, however, are still suffering from very time consuming calculations. They are restricted to models with short simulation times and a moderate number of parameters in order to keep computational costs reasonably low.

This is possibly the reason why the use of Bayesian methodology in water resources modelling has been limited to simple conceptual and data-based mechanistic models and traditional statistical models with few parameters.

2.3.3 Bayesian Neural Networks

Bayesian methodology has been applied to ANNs since the early 1990s, when it was first used by *Buntine and Weigend* (1991), *MacKay* (1992a) and *Neal* (1992). The use of 'Bayesian neural networks' has since been reviewed by *Bishop* (1995), *MacKay* (1995a), *Neal* (1996a), *Lampinen and Vehtari* (2001) and *Titterington* (2004). Unlike standard ANN approaches, the aim under the Bayesian framework is not to find a single "optimal" weight vector, but rather, it is to explicitly represent the uncertainty in the values of the weights by a posterior probability distribution.

MacKay (1992a, 1995a) describes two levels of Bayesian inference that can be performed in ANN modelling: the first level involves inference of the network weights under the assumption that the chosen ANN structure is "true", while the second level involves model comparison in light of the data and the estimated weight distributions. In this research, these two levels of inference will be considered in terms of ANN 'training and prediction' and 'model selection', respectively. The advantages of the Bayesian neural network framework, given the two levels of inference, are briefly summarised as follows (*Bishop*, 1995; *Neal*, 1992; *Rasmussen*, 1996):

- Uncertainty in both the model parameters and the predictions is handled explicitly. Predictive distributions are calculated by integrating the predictions from all possible weight vectors over the posterior weight distribution, thus allowing prediction intervals to be assigned and achieving better generalisation than standard ANN approaches (as overfitting is avoided to some extent).

- The Bayesian framework provides a natural interpretation for regularisation, allowing overfitting to be avoided in a consistent manner. Values of the regularisation coefficients (i.e. α in (2.2)) are selected automatically using only the training data; thus, a separate testing data set is not required.

- Complex networks can be used without fear of overfitting the data. Therefore, more structure can be learnt from the data, improving prediction accuracy. Furthermore, a relatively large number of regularisation coefficients can be used, which would not be computationally feasible if their values had to be found by cross-validation.

- ANNs of varying complexity can be compared in an objective and principled manner, using only the training data. The 'evidence' of an ANN $p(\mathbf{y}|\mathcal{H})$ (see Section 2.3.1) is the likelihood that the given model $\mathcal{H}$ is the "true" model, given the data $\mathbf{y}$. This term automatically penalises overly complex models and therefore can be used to select the optimum level of complexity for modelling the data.

- The relative importance of different inputs can be determined using the Automatic Relevance Determination (ARD) method of *MacKay* (1994) and *Neal* (1996a, 1998).

2.3.4 Training and Prediction

As discussed in Section 2.2.3, the aim of standard (deterministic) ANN training is to find a single optimal weight vector $\hat{\mathbf{w}}$ that provides the best fit of the model to the observed data $\mathbf{y}$. The aim of Bayesian training, on the other hand, is to infer the posterior probability distribution of the weights given the observed data $p(\mathbf{w}|\mathbf{y})$ (*Bishop*, 1995). This posterior weight distribution can then be used to make probabilistic predictions given new values of the model inputs. However, the first problem involves estimation of the posterior.

2.3.4.1 Bayesian Training (Posterior Weight Estimation)

When applied to estimate the weights of an ANN, Bayes' theorem, given by (2.5), can be written as:

$$p(\mathbf{w}|\mathbf{y},\mathbf{X},\mathcal{H}) = \frac{p(\mathbf{y}|\mathbf{w},\mathbf{X},\mathcal{H})p(\mathbf{w}|\mathbf{X},\mathcal{H})}{p(\mathbf{y}|\mathbf{X},\mathcal{H})} \tag{2.8}$$

The observed set of input data $\mathbf{X} = (\mathbf{x}_1^K, \ldots, \mathbf{x}_N^K)$ and the model structure $\mathcal{H}$ are the conditional assumptions upon which the probability measure for the weights is based (*MacKay*, 1995a). When only considering one model (as is the case when applying the first level of inference), it is common to drop these conditioning terms (*Bishop*, 1995) to simplify the notation. Furthermore, the normalising constant $p(\mathbf{y}|\mathbf{X},\mathcal{H})$ is irrelevant at this level of inference and as such it is commonly ignored (*MacKay*, 1992a). Therefore (2.8) can be simplified as follows:

$$p(\mathbf{w}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{w})p(\mathbf{w})}{p(\mathbf{y})} \propto p(\mathbf{y}|\mathbf{w})p(\mathbf{w}) \tag{2.9}$$

which states that the posterior weight distribution is proportional to the product of the likelihood of the weights and the prior weight distribution. Therefore, before inferences can be made about the posterior of the weights $p(\mathbf{w}|\mathbf{y})$, choices must be made about the likelihood function $p(\mathbf{y}|\mathbf{w})$ and prior weight distribution $p(\mathbf{w})$.

As the likelihood function $p(\mathbf{y}|\mathbf{w})$ is commonly regarded as a function of the weights $\mathbf{w}$ rather than as a function of the data $\mathbf{y}$, it may be written as $L(\mathbf{w}|\mathbf{y})$, or simply $L(\mathbf{w})$ (*Box and Tiao*, 1973; *Neal*, 1996a). For a set of N statistically independent observed data points, the likelihood is equal to:

$$\begin{aligned} L(\mathbf{w}) &= L(\mathbf{w}|y_1, \ldots, y_N) \\ &\propto p(y_1, \ldots, y_N|\mathbf{w}) = \prod_{i=1}^{N} p(y_i|\mathbf{w}) \end{aligned} \tag{2.10}$$

This function is defined up to a multiplicative constant, as it is only the relative value of the likelihood which is of importance. As discussed in Section 2.2.5.3, it is generally assumed that ϵ in (2.1) is normally and independently distributed with zero mean and constant variance $\sigma_{\mathbf{y}}^2$. Under this assumption, the likelihood function is given by:

$$L(\mathbf{w}) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\Pi\sigma_{\mathbf{y}}^2}} \exp\left\{-\frac{[y_i - f(\mathbf{x}_i,\mathbf{w})]^2}{2\sigma_{\mathbf{y}}^2}\right\} \tag{2.11}$$

In the simplest case, the prior probability distribution for each of the network weights is also assumed to be normal with zero mean and constant variance $\sigma_{\mathbf{w}}^2$ (*Ragg et al.*, 2002).

$$p(\mathbf{w}) = \prod_{i=1}^{d} p(w_i) = \prod_{i=1}^{d} \frac{1}{\sqrt{2\Pi\sigma_{\mathbf{w}}^2}} \exp\left(-\frac{w_i^2}{2\sigma_{\mathbf{w}}^2}\right) \tag{2.12}$$

This prior is selected based on the experience that the values of the weights can be positive and negative with equal probability and that the weights have a finite variance (*Thodberg*, 1996). It also restricts the complexity of the ANN, as small weight values are sought (*Ragg et al.*, 2002). By rewriting (2.9) as:

$$\begin{aligned} p(\mathbf{w}|\mathbf{y}) &\propto \exp\left\{-\left(\frac{1}{2\sigma_{\mathbf{y}}^2}\sum_{i=1}^{N}[y_i - f(\mathbf{x}_i,\mathbf{w})]^2 + \frac{1}{2\sigma_{\mathbf{w}}^2}\sum_{i=1}^{d} w_i^2\right)\right\} \\ &\propto \exp\left\{-\left(\frac{1}{\sigma_{\mathbf{y}}^2}E_{\mathbf{y}} + \alpha E_{\mathbf{w}}\right)\right\} \end{aligned} \tag{2.13}$$

it can be seen that the form of prior given by (2.12) is equivalent to weight decay (i.e. $E_{\mathbf{w}} = \frac{1}{2}\sum_{i=1}^{d} w_i^2$, see Section 2.2.5.1), with $\alpha = 1/\sigma_{\mathbf{w}}^2$; thus, regularisation is automatically incorporated into the Bayesian framework in the form of priors on the weights (*MacKay*, 1992a; *Titterington*, 2004).

The variance terms upon which both the likelihood function and prior distribution depend (i.e. $\sigma_{\mathbf{y}}^2$ and $\sigma_{\mathbf{w}}^2$, respectively) are generally referred to as 'hyperparameters' (*MacKay*, 1995a; *Neal*, 1996a; *Lampinen and Vehtari*, 2001), as they play an important role in estimating the posterior weight distribution, but ultimately play no part in the developed model. In a full Bayesian approach, no fixed values are used for any parameters or hyperparameters (*Lampinen and Vehtari*, 2001); therefore, both the weights $\mathbf{w}$ and the hyperparameters $\{\sigma_{\mathbf{y}}^2, \sigma_{\mathbf{w}}^2\}$ are estimated as follows:

$$p(\mathbf{w}, \sigma_{\mathbf{w}}^2, \sigma_{\mathbf{y}}^2|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{w}, \sigma_{\mathbf{y}}^2)p(\mathbf{w}|\sigma_{\mathbf{w}}^2)p(\sigma_{\mathbf{w}}^2, \sigma_{\mathbf{y}}^2)}{p(\mathbf{y})} \tag{2.14}$$

Estimation of the parameters is given a hierarchical treatment (*Lampinen and Vehtari*, 2001; *Ragg et al.*, 2002), where the lower level of the hierarchy is comprised of the weights $\mathbf{w}$, while the upper layer consists of the hyperparameters that control the distributions of the weights $\sigma_{\mathbf{w}}^2$ and the noise levels in the regression model $\sigma_{\mathbf{y}}^2$. The hyperparameters are then assigned their own priors or 'hyperpriors' ($p(\sigma_{\mathbf{w}}^2, \sigma_{\mathbf{y}}^2)$ in (2.14)), which are generally vague or 'noninformative' and enable the hyperparameter distributions to be determined automatically from the data (*Bishop*, 1995; *Neal*, 1996a; *Lampinen and Vehtari*, 2001). While the parameter $\sigma_{\mathbf{y}}^2$ is not strictly a 'hyperparameter', as it does not control the distribution of the lower level parameters (*Neal*, 1996a), it is commonly referred to as such in the Bayesian neural network literature and is treated in the same way as $\sigma^2_{\mathbf{w}}$ (*Neal*, 1992; *MacKay*, 1992a,b; *Buntine and Weigend*, 1991).

As discussed in Section 2.2.5.1, different groups of weights within the network require different regularisation coefficients. Therefore the prior in (2.12), based on the common hyperparameter $\sigma^2_{\mathbf{w}}$, is generally not a good choice for the network weights. One of the great advantages of the Bayesian framework is that a large number of hyperparameters (which correspond to regularisation coefficients) can be used since they are determined automatically (*Ragg et al.*, 2002). Therefore, a more general and flexible prior distribution is the product of G different normal distributions, where G is the number of different weight groups (i.e. $\mathbf{w} = \{\mathbf{w}_1, \ldots, \mathbf{w}_G\}$ and $\sigma^2_{\mathbf{w}} = \{\sigma^2_{\mathbf{w}_1}, \ldots, \sigma^2_{\mathbf{w}_G}\}$) (*Husmeier et al.*, 1999). This prior is written as follows:

$$p(\mathbf{w}) = \prod_{g=1}^{G} p(\mathbf{w}_g|\sigma^2_{\mathbf{w}_g}) = \prod_{g=1}^{G} (2\Pi\sigma^2_{\mathbf{w}_g})^{-d_g/2} \exp\left(-\frac{\sum_{i_g=1}^{d_g} w^2_{i_g}}{2\sigma^2_{\mathbf{w}_g}}\right) \tag{2.15}$$

where d_g is the dimension of the gth weight group. In the most extreme case, the number of groups would correspond to the number of weights. In other words, the prior distribution of each weight would have a different variance (*Titterington*, 2004). A more typical approach is to have four groups of weights corresponding to the input-hidden layer weights, the hidden layer biases, the hidden-output layer weights and the output layer biases (*Lampinen and Vehtari*, 2001). Another alternative is to further divide weights in the input-hidden layer into groups of weights exiting the same input. This is done in the Automatic Relevance Determination (ARD) method of *MacKay* (1994) and *Neal* (1996a, 1998). As the variance hyperparameters are automatically adapted during training, when equilibrium is reached they can be used to assess the relevance of the weights belonging to that group. The smaller the variance, the more tightly the weights are distributed around zero; hence, the less relevant they are to the model. Therefore, by treating the weights exiting the same input as a group, the relative importance of the input can be assessed by comparing the variance hyperparameter to those controlling other input weight groups (*Thodberg*, 1996; *Husmeier et al.*, 1999; *Vivarelli and Williams*, 2001).

2.3.4.2 Marginalization (Prediction)

Under the Bayesian paradigm, the predictive distribution of a new datum y_{N+1} is determined by integrating the predictions made by all of the weight vectors over the posterior distribution of the weights, as follows (*Eleuteri et al.*, 2002):

$$\begin{aligned} p(y_{N+1}|\mathbf{x}_{N+1},\mathbf{y}) &= \int p(y_{N+1},\mathbf{w}|\mathbf{x}_{N+1},\mathbf{y})d\mathbf{w} \\ &= \int p(y_{N+1}|\mathbf{x}_{N+1},\mathbf{w})p(\mathbf{w}|\mathbf{y})d\mathbf{w} \end{aligned} \tag{2.16}$$

Since only $p(\mathbf{w}|\mathbf{y})$ is required to make predictions, it is also necessary to integrate out the hyperparameters from (2.14):

$$\begin{aligned} p(\mathbf{w}|\mathbf{y}) &= \int p(\mathbf{w},\sigma^2_{\mathbf{w}},\sigma^2_{\mathbf{y}}|\mathbf{y})d\sigma^2_{\mathbf{w}}d\sigma^2_{\mathbf{y}} \\ &= \int p(\mathbf{w}|\sigma^2_{\mathbf{w}},\sigma^2_{\mathbf{y}},\mathbf{y})p(\sigma^2_{\mathbf{w}},\sigma^2_{\mathbf{y}}|\mathbf{y})d\sigma^2_{\mathbf{w}}d\sigma^2_{\mathbf{y}} \end{aligned} \tag{2.17}$$

The process of integrating out the unwanted parameters $\mathbf{w}$ and $\left\{\sigma^2_{\mathbf{w}},\sigma^2_{\mathbf{y}}\right\}$ is known as ‘marginalization’. However, for complex problems, the high dimensionality of these integrals makes marginalization analytically intractable (*Neal*, 1993; *Titterington*, 2004). In order to overcome this problem, two main approaches to marginalization have generally been followed in the Bayesian neural network literature, including:

1. Gaussian approximation of the posterior weight distribution about the most probably weight vector $\hat{\mathbf{w}}$ to enable analytical integration, as introduced by *Buntine and Weigend* (1991) and *MacKay* (1992a).

2. Numerical integration using Markov chain Monte Carlo methods, as introduced by *Neal* (1992)

With the first method, there has been some controversy regarding how the hyperparameters should be handled (*MacKay*, 1999). *Buntine and Weigend* (1991) introduced a method for analytically integrating out the hyperparameters before the Gaussian approximation is made. This is done by integrating $p(\mathbf{w}|\sigma^2_{\mathbf{w}})$ over $\sigma^2_{\mathbf{w}}$ and integrating $p(\mathbf{y}|\mathbf{w},\sigma^2_{\mathbf{y}})$ over $\sigma^2_{\mathbf{y}}$ to obtain $p(\mathbf{w})$ and $p(\mathbf{y}|\mathbf{w})$, respectively. The most probable weights are then found by maximising $p(\mathbf{y}|\mathbf{w})p(\mathbf{w})$ and the Gaussian posterior assumption is subsequently made about these weights. On the other hand, *MacKay* (1992a,b) integrates the posterior $p(\mathbf{y}|\mathbf{w},\sigma^2_{\mathbf{y}})p(\mathbf{w}|\sigma^2_{\mathbf{w}})$ over the weights to obtain $p(\mathbf{y}|\sigma^2_{\mathbf{w}},\sigma^2_{\mathbf{y}})$, which he terms the ‘evidence’ of the hyperparameters. A Gaussian approximation is again made for this evidence term, which is then maximised to find optimal values of $\sigma^2_{\mathbf{w}}$ and $\sigma^2_{\mathbf{y}}$. The Gaussian approximation is then made with the hyperparameters fixed at their optimal values. In a comparison of the two Gaussian approximation approaches, *MacKay* (1999) demonstrated that, from a predictive point of view, is it better to integrate over many weights rather than over few hyperparameters. Therefore, the 'evidence' framework has been the most widely adopted Gaussian approximation method and is highly influential in the Bayesian neural network literature(*Titterington*, 2004). However, for multi-layered ANNs, the posterior weight distribution is typically very complex and multi-modal and thus the assumption of a Gaussian weight distribution is generally not a good one (*Neal*, 1996a). This has been acknowledged by *MacKay* (1995a) who then assumes that the distribution is *locally* Gaussian around each mode and treats each mode as a separate model. However, this raises the question of how to properly handle the multiple modes when making predictions. Furthermore, the assumption of even a locally Gaussian distribution in the vicinity of the modes is sometimes questionable, particularly when the model is complex and the data available for training are limited (*Rasmussen*, 1996).

To avoid the need to make a Gaussian approximation of the posterior weight distribution, *Neal* (1992) introduced a Markov chain Monte Carlo (MCMC) implementation to sample from the posterior weight distribution. The use of a MCMC algorithm to estimate the posterior distribution involves the construction of a Markov chain of sampled weight vectors and hyperparameters, which, at equilibrium, has the target distribution $p(\mathbf{w}, \sigma^2_{\mathbf{w}}, \sigma^2_{\mathbf{y}}|\mathbf{y})$. By choosing conjugate inverse chi-square priors for the variance hyperparameters, the Gibbs sampler (*Gelman et al.*, 2004), which is the simplest MCMC algorithm, may be used to update the hyperparameters (*Neal*, 1996a; *Titterington*, 2004). However, the complexity of the likelihood of the weights prevents Gibbs sampling for the weights (*Titterington*, 2004). A commonly used MCMC algorithm used when the Gibbs sampler is not applicable is the Metropolis algorithm of *Metropolis et al.* (1953). However, as noted by *Neal* (1992), while it is possible to use the Metropolis algorithm to sample the weights, this algorithm can be slow to converge to the target distribution and it is difficult to determine whether or not convergence has been reached. Therefore, the approach promoted by *Neal* (1996a) involves the use of the hybrid Monte Carlo algorithm developed by *Duane et al.* (1987) to sample the weight vectors. This is an elaboration of the Metropolis algorithm that makes use of gradient information to speed convergence to the target distribution (*Neal*, 1992).

Sampling from the posterior $p(\mathbf{w}, \sigma^2_{\mathbf{w}}, \sigma^2_{\mathbf{y}}|\mathbf{y})$ then follows a two-step procedure. In the first step, the hyperparameters are held constant while the weights are sampled from the distribution:

$$\begin{aligned} p(\mathbf{w}|\sigma^2_{\mathbf{w}}, \sigma^2_{\mathbf{y}}, \mathbf{y}) &= \frac{p(\mathbf{y}|\mathbf{w}, \sigma^2_{\mathbf{y}})p(\mathbf{w}|\sigma^2_{\mathbf{w}})}{p(\mathbf{y}|\sigma^2_{\mathbf{w}}, \sigma^2_{\mathbf{y}})} \\ &\propto p(\mathbf{y}|\mathbf{w}, \sigma^2_{\mathbf{y}})p(\mathbf{w}|\sigma^2_{\mathbf{w}}) \end{aligned} \tag{2.18}$$

using the hybrid Monte Carlo method. In the second step, the weights are held constant while the hyperparameters are sampled from their respective full conditional distributions:

$$\begin{aligned} p(\sigma^2_{\mathbf{w}}|\mathbf{w},\mathbf{y},\sigma^2_{\mathbf{y}}) &= p(\sigma^2_{\mathbf{w}}|\mathbf{w}) \\ &\propto p(\mathbf{w}|\sigma^2_{\mathbf{w}})p(\sigma^2_{\mathbf{w}}) \end{aligned} \tag{2.19}$$

$$\begin{aligned} p(\sigma^2_{\mathbf{y}}|\mathbf{w},\mathbf{y},\sigma^2_{\mathbf{w}}) &= p(\sigma^2_{\mathbf{y}}|\mathbf{w},\mathbf{y}) \\ &\propto p(\mathbf{y}|\mathbf{w},\sigma^2_{\mathbf{y}})p(\sigma^2_{\mathbf{y}}) \end{aligned} \tag{2.20}$$

After the MCMC algorithm reaches equilibrium, the sampled weights and hyperparameters can be considered as samples from the posterior distribution. An advantage of the MCMC approach is that marginalization over the hyperparameters is automatically accomplished (*Hanson*, 1999). By sampling from the joint posterior distribution $p(\mathbf{w},\sigma^2_{\mathbf{w}},\sigma^2_{\mathbf{y}}|\mathbf{y})$, when the posterior of $p(\mathbf{w}|\mathbf{y})$ is determined using the sampled weights, the remaining hyperparameters $\{\sigma^2_{\mathbf{w}},\sigma^2_{\mathbf{y}}\}$ are automatically integrated out. The sampled weights may therefore be used directly to obtain samples from the predictive distribution for the targets of a test case (*Neal*, 1996a). To do this, the model outputs are first calculated with the given test inputs using the sampled weight states. Samples from the predictive distribution are then obtained by adding Gaussian noise to the outputs, with variance given by the sampled $\sigma^2_{\mathbf{y}}$ hyperparameter that corresponds to the weight state used to calculate the outputs.

2.3.5 Model Selection

Given a set of H competing models $\{\mathcal{H}_i; i = 1,\ldots,H\}$, the Bayesian framework can be used to infer the posterior probability that, of the H models, $\mathcal{H}_i$ is the "true" model of the system given the observed data (*MacKay*, 1995a; *Bishop*, 1995). At this level of inference, Bayes' theorem yields:

$$p(\mathcal{H}_i|\mathbf{y}) = \frac{p(\mathbf{y}|\mathcal{H}_i)p(\mathcal{H}_i)}{p(\mathbf{y})} = \frac{p(\mathbf{y}|\mathcal{H}_i)p(\mathcal{H}_i)}{\sum_{j=1}^{H} p(\mathbf{y}|\mathcal{H}_j)p(\mathcal{H}_j)} \tag{2.21}$$

where $p(\mathcal{H}_i)$ is the prior probability assigned to $\mathcal{H}_i$ and the likelihood $p(\mathbf{y}|\mathcal{H}_i)$ is the denominator from (2.5), or the 'evidence' of the model. Although, it is unlikely that any model will actually be "true", the Bayes' approach enables the relative merits of the competing models to be compared, which is worthwhile assuming that at least one of the models is approximately correct (*Wasserman*, 2000). It is also generally assumed that the prior probabilities assigned to the different models are approximately equal, as a model thought to be highly implausible would not even be considered in the comparison (*Neal*, 1993). Therefore, 2.21 can be simplified to:

$$p(\mathcal{H}_i|\mathbf{y}) = \frac{p(\mathbf{y}|\mathcal{H}_i)}{\sum_{j=1}^{H} p(\mathbf{y}|\mathcal{H}_j)} \propto p(\mathbf{y}|\mathcal{H}_i) \tag{2.22}$$

which states the relative probabilities of the competing models can be compared based on their evidence (*Bishop*, 1995).

As discussed in *MacKay* (1995a) and *Rasmussen* (2001), the evidence of a model automatically incorporates 'Occam's razor', which is a principle that states the preference for simple theories, through the effect of its prior on the weights. In terms of an ANN model, (2.6) is rewritten as:

$$p(\mathbf{y}|\mathcal{H}_i) = \int p(\mathbf{y}|\mathbf{w}, \mathcal{H}_i)p(\mathbf{w}|\mathcal{H}_i)d\mathbf{w}, \tag{2.23}$$

which can be interpreted as the probability that, given a weight vector randomly selected from the prior weight distribution, a particular set of observed data will be generated (*Rasmussen*, 2001). Simple models have relatively narrow prior weight distributions which only allow a limited range of data sets to be generated, whereas complex models have rather wide flat prior weight distributions that enable a greater variety of data sets to be generated (*MacKay*, 1992a). The aim is then to select the model that has the greatest probability of generating a given set of observed data $\mathbf{y}_{obs}$, or the strongest evidence $p(\mathbf{y}_{obs}|\mathcal{H})$. This is illustrated in Figure 2.3, where $\mathcal{H}_1$, $\mathcal{H}_2$ and $\mathcal{H}_3$ are three models of increasing complexity and the observed data set $\mathbf{y}_{obs}$ is shown as a single value of $\mathbf{y}$. In this figure, it is shown that $\mathcal{H}_1$ would be very unlikely to generate the observed data given the narrow predictive distribution $p(\mathbf{y}|\mathcal{H}_1)$. Likewise, it would be unlikely that the observed data would be generated at random by $\mathcal{H}_3$, due to the wide range of $p(\mathbf{y}|\mathcal{H}_3)$. Thus, $\mathcal{H}_2$ is the most probable model for the observed data set $\mathbf{y}_{obs}$, as it results in the maximum predictive probability, or the strongest evidence. Typically, the value of the evidence for any model will be extremely small, as any particular data set of significant size will have low probability even under the correct model (*Neal*, 1993). However, by considering the relative magnitude of these small probabilities, the data-based evidence term can be used to rank a number of competing models in order of plausibility without the need to specify a prior that gives preference to simple models (*MacKay*, 1995a).

Similar to the integrals in (2.16) and (2.17), for complex models the integral in (2.23) is analytically intractable. In his evidence framework, *MacKay* (1992a, 1995a) instead evaluates the integral:

$$p(\mathbf{y}|\mathcal{H}_i) = \int p(\mathbf{y}|\sigma^2_{\mathbf{w}}, \sigma^2_{\mathbf{y}}, \mathcal{H}_i)p(\sigma^2_{\mathbf{w}}, \sigma^2_{\mathbf{y}}|\mathcal{H}_i)d\sigma^2_{\mathbf{w}}d\sigma^2_{\mathbf{y}} \tag{2.24}$$

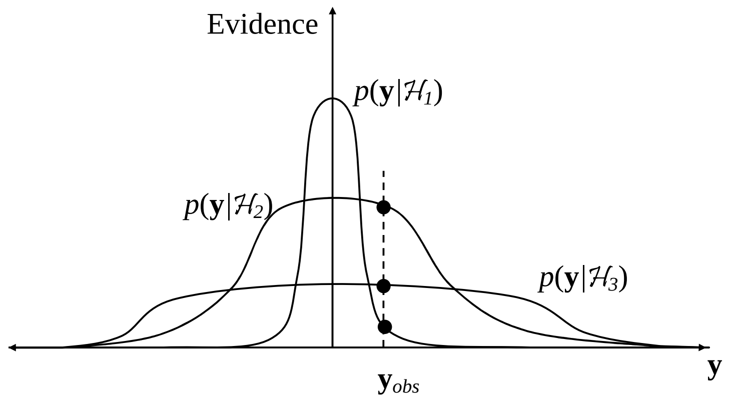
Evidence
$p(\mathbf{y}|\mathcal{H}_1)$
$p(\mathbf{y}|\mathcal{H}_2)$
$p(\mathbf{y}|\mathcal{H}_3)$
$\mathbf{y}_{obs}$
$\mathbf{y}$

Figure 2.3 Evidence incorporating Occam's razor.

by using the Gaussian approximation of $p(\mathbf{y}|\hat{\sigma}^2_{\mathbf{w}}, \hat{\sigma}^2_{\mathbf{y}}, \mathcal{H}_i)$, where $\hat{\sigma}^2_{\mathbf{w}}$ and $\hat{\sigma}^2_{\mathbf{y}}$ the optimal hyperparameter values used in estimating the posterior weight distribution (see Section 2.3.4.1), to obtain a well-defined $p(\mathbf{y}|\mathcal{H}_i)$ (*Titterington*, 2004).

Under the evidence framework, evaluation of the determinant of the Hessian matrix is required. As discussed in Section 2.2.5.3, the Hessian matrix of an ANN can be nearly singular and consequently, evaluation of the evidence can be very sensitive to errors in the small eigenvalues (*Bishop*, 1995). An alternative to this approach is that suggested by *Neal* (1994), where model complexity is not limited. Neal has argued that if a complete Bayesian analysis is performed without approximation and appropriate prior distributions are used on the weights, it is possible to use large networks without fear of overfitting the data. However, it may still be necessary to limit the complexity of an ANN to ensure that Gaussian assumptions for estimating the weights are valid or MCMC techniques achieve convergence in reasonable computational time (*Bishop*, 1995). A sophisticated MCMC approach for selecting the right level of complexity was developed by *Müller and Rios Insua* (1998), where the number of hidden nodes was treated as a random variable that was also estimated.

2.3.6 Limitations of Current Bayesian Neural Network Practices

While Bayesian techniques have been applied to ANNs (although rarely) for around 10-15 years, the complexity of ANNs makes it difficult to apply standard Bayesian methods that are increasing in popularity for other models (e.g. the standard Metropolis algorithm) (*Neal*, 1996a). Consequently, the majority of Bayesian techniques applied to ANNs in the past have employed complex statistics in order to overcome any complications. The majority of publications in this field have therefore been limited to the statistical, computer science and neural computing literature (*Titterington*, 2004). Most of the available ANN software does not allow for Bayesian analysis, and due to the difficulty associated with programming the available complicated techniques, Bayesian ANN training has not been adopted by water resources practitioners.

Considerable emphasis has been placed on achieving efficiency and statistical optimality when Bayesian methods have been developed for ANNs in the past. However, the difficulty in implementing these methods and their lack of adoption by water resources modellers indicates the need for a Bayesian ANN development framework that provides accurate results, while being relatively straightforward to code and implement.

2.4 CONCLUSION

Given the nonlinearity, complexity and limited physical understanding of the majority of processes that occur within water resource systems, ANNs may well be the best available tool for modelling such systems. However, ANNs have yet to become widely accepted and reach their full potential as models in the field of water resources engineering. Based on a review of the relevant literature, this is apparently due to three significant issues, namely generalisability, interpretability and uncertainty, which ANN modellers are poorly equipped to address, given currently available ANN development methods. As a result, ANNs continue to be viewed sceptically as a means for providing predictions that can be used confidently in water resources design and management applications. On the other hand, the Bayesian modelling paradigm appears to be a promising approach for dealing with these issues, whether directly or indirectly, and its application to more conventional models has been increasing in the field of water resources modelling. However, due to the complexity of ANNs, application of Bayesian methods for ANN development is not straightforward, as standard techniques are often infeasible or highly inefficient; thus, producing poor results. Consequently, the Bayesian techniques developed for ANNs, proposed primarily in the statistical and computer sciences literature, have generally employed complex statistics, which makes them difficult to code and implement. The lack of adoption of these sophisticated Bayesian methods in the field of water resources modelling clearly indicates the need for a Bayesian ANN development framework that is accessible to water resources modellers. As stated by *Maier and Dandy* (2000a),

> The primary focus should be on achieving good results, rather than statistical optimality, as this is one of the features that has attracted water resources modellers to ANNs in the first place.

Therefore, the aim of this research is to develop a relatively simple Bayesian framework that can be applied to ANNs in the field of water resources modelling, where the primary aim of the procedure is not statistical optimality, nor optimum efficiency, but rather good results and ease of programming and application. It is envisaged that the development of such a framework will enable significant advances to be made in the field of water resources modelling with ANNs, as ANNs will no longer be held back due to the lack of appropriate methods for addressing the modelling issues discussed in Section 2.2.5.

Chapter 3
State-of-the-Art Deterministic ANN Methodology

3.1 INTRODUCTION

Following almost a decade of reported applications of ANNs for modelling hydrological and water resources variables, three comprehensive state-of-the-art reviews were conducted on ANN modelling in this field (*ASCE Task Committee*, 2000a,b; *Maier and Dandy*, 2000a; *Dawson and Wilby*, 2001). Although it was evident from these reviews that ANNs had potential as a useful prediction and forecasting tool, one of the main conclusions of each review was that, for significant advances to be made in this field, a set of systematic guidelines would need to be established to aid the development of ANNs applied to hydrological and water resources modelling. In his recent thesis, *Bowden* (2003) attempted to address this issue by developing 'a robust methodology for the design and successful implementation of ANN models for the forecasting/prediction of water resources variables'. Therefore, the methods proposed in this chapter will attempt to build on this approach.

In this chapter, each step in the ANN development process is discussed, together with a review of the methods currently used for carrying out these steps and a summary of the methods proposed by *Bowden* (2003). Additionally, any limitations or shortcomings of current practices are identified and, if necessary, addressed through further assessment, comparison and simple modification of alternative existing methods when applied to synthetic data sets. The methods considered in this chapter are limited to conventional *deterministic* ANN methods, where a single optimum weight vector is sought and single-valued predictions are made. Furthermore, only methods applicable to feedforward MLPs with a single output variable will be considered; however, it is considered that these methods could be easily extended to ANNs with more than one output.

3.2 REVIEW OF THE CURRENT STATE-OF-THE-ART

As discussed in *Maier and Dandy* (2000a,b) and outlined in Figure 3.1, there are a number of main steps in the ANN development process. It can be seen in this figure that there are also a number of options available at each step and, while this provides great flexibility in ANN modelling, it also leaves the modeller faced with the difficult task of selecting the most suitable methods. In order to develop a robust methodology for ANNs, the research carried out by *Bowden* (2003) involved a review of the alternatives available at each step, identification of the limitations of available methods and, finally, the proposal of new (or modified) methods to enable ANNs to be developed in a systematic and

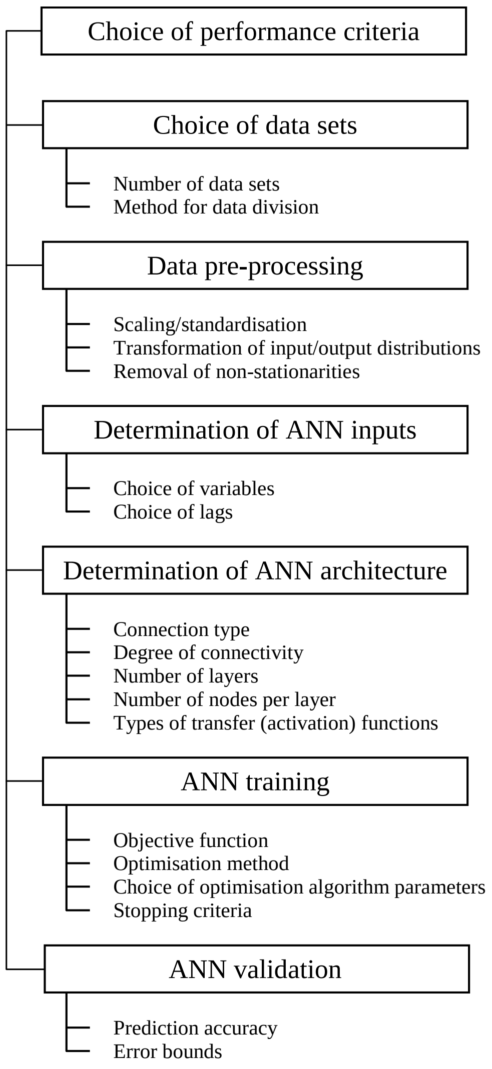

Figure 3.1 Main steps in the development of an ANN (source: adapted from *Maier and Dandy* (2000b).

consistent manner, where considered appropriate. However, while each of the ANN development steps was addressed to some extent by *Bowden* (2003), the main emphasis was placed on the choice of data sets, pre-processing and transformation of the data and selection of important network inputs. Since the choice of performance criteria, architecture selection, training and validation steps were not addressed in detail by *Bowden* (2003), improvements to these steps will be the main focus of Section 3.4.

3.2.1 Choice of Performance Criteria

3.2.1.1 Review of current practice

The first step in the ANN development process is the choice of performance criteria, as this determines how the model is assessed and will consequently affect many of the subsequent steps such as training and the choice of network architecture (*Maier and Dandy*, 2000a). Performance criteria may include measures of training and processing speed; however, the most commonly used performance criteria used in water resources modelling measure the prediction accuracy (*Bowden*, 2003).

Performance criteria which measure prediction accuracy generally measure the fit (or lack there of) between the model outputs $\hat{\mathbf{y}} = \{\hat{y}_i, \ldots, \hat{y}_N\}$ and the observed data $\mathbf{y} = \{y_i, \ldots, y_N\}$ by some error measure $E_{\mathbf{y}}$. They are used during training as *objective functions*, and after training to evaluate the trained ANN, where the criterion used for each purpose need not necessarily be the same. The most commonly used objective function, which is minimised during training, is the sum squared error (SSE) given by:

$$E_{\mathbf{y}} = \mathrm{SSE} = \frac{1}{2}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 = \frac{1}{2}\sum_{i=1}^{N}(y_i - f(\mathbf{x}_i, \mathbf{w}))^2 \tag{3.1}$$

The SSE provides an overall estimate of modelling performance (*Dawson and Wilby*, 2001), as it measures the total deviation of the modelled function from the observed data, as estimated by summing the squared model residuals (deviations between the observed data and the model outputs) over the entire data set. The reason for using squared residuals rather than their absolute values is that this allows the residuals to be treated as a continuous differentiable quantity, which is important if a gradient-based search technique is used to minimise the objective function during training (see Section 3.2.6). Use of the SSE as an objective function gives rise to the well known *least squares* parameter estimation method, which makes the assumption that the residuals are randomly distributed with zero mean and constant variance. It is also implicitly assumed that the target data are approximately normally distributed. While this is not a requirement of the least squares estimation method, under the assumption of independent Gaussian residuals, minimising the SSE is equivalent to maximising the likelihood of the weights $p(\mathbf{y}|\mathbf{w})$ when $\mathbf{y} \sim \mathrm{N}\left(f(\mathbf{x},\mathbf{w}),\sigma_{\mathbf{y}}^2\right)$ (see Section 2.3.4.1). Furthermore, the SSE can be very sensitive to outliers in the data, resulting in poor weight estimates. Extreme random values are uncommon under a Gaussian distribution; therefore, optimal results are achieved when the target data are approximately normally distributed.

While the SSE may be suitable as an objective function during training, this measure does not quantify the error in terms of the units of the target variable, nor does it take into account the size of the data set, which is appropriate if model performance is compared on different data sets. Furthermore, although the SSE indicates overall performance, it may not be adequate for assessing the model's ability to fit both low and peak events. Thus, it is also common to assess the predictive performance of a trained ANN using error measures that indicate the particular areas of model deficiency that are considered to be of most importance for a given problem (*Dawson and Wilby*, 2001). According to both *Dawson and Wilby* (2001) and *Bowden* (2003), the most commonly used error measures in water resources modelling include the root mean squared error (RMSE), the mean absolute error (MAE), the mean squared relative error (MSRE), the coefficient of determination (r^2) and the coefficient of efficiency (CE), given by (3.2) to (3.6), respectively.

$$\mathrm{RMSE} = \left[\frac{1}{N}\sum_{i=1}^{N}\left(y_i - \hat{y}_i\right)^2\right]^{1/2} \tag{3.2}$$

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}\left|y_i - \hat{y}_i\right| \tag{3.3}$$

$$\mathrm{MSRE} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{y_i - \hat{y}_i}{y_i}\right)^2 \tag{3.4}$$

$$r^2 = \left[\frac{\sum_{i=1}^{N}\left(y_i - \bar{y}\right)\left(\hat{y}_i - \tilde{y}\right)}{\sqrt{\sum_{i=1}^{N}\left(y_i - \bar{y}\right)^2\left(\hat{y}_i - \tilde{y}\right)^2}}\right]^2 \tag{3.5}$$

$$\mathrm{CE} = 1 - \frac{\sum_{i=1}^{N}\left(y_i - \hat{y}_i\right)^2}{\sum_{i=1}^{N}\left(y_i - \bar{y}\right)^2} \tag{3.6}$$

In (3.5) and (3.6), $\bar{y}$ is the mean of the observed data and $\tilde{y}$ is the mean of the corresponding model outputs.

The RMSE is also a measure of general model performance, but unlike the SSE, the sample size is taken into consideration and it returns an error value with the same units as the data. However, the RMSE (like the SSE) is sensitive to outliers and extreme values in the data, as squaring the residuals means that larger values can have a significantly greater influence on the overall statistic. Consequently, the RMSE is generally weighted towards the prediction of peak events, as these tend to be the poorest predictions (*Abrahart and See*, 1998). The MAE uses absolute values of the residuals, and is therefore less weighted towards fitting extremes in the data. *Legates and McCabe* (1999) note that the degree to which the RMSE exceeds the MAE is an indicator of the extent to which outliers (or variance in the residuals) exist in the data. It has also been suggested that measures of relative errors, such as the MSRE, which allow larger valued observations to have larger inherent errors, are more suited to measuring predictive performance on moderate values than the RMSE (*Karunanithi et al.*, 1994). However, the MSRE is inappropriate if the observed data contain values of zero. If the aim is to fit extreme values, a higher order error measure, where the residuals are raised to a higher even power (e.g. $(y_i - \hat{y}_i)^4$), could be used to place more emphasis on such extreme events (*Abrahart and See*, 1998).

The r^2 and CE criteria are dimensionless "goodness-of-fit" measures. They provide a useful relative assessment of model performance in comparisons between studies, since they are independent of the scale of data used. The r^2 criterion measures the linear correlation between the model outputs and the observed data and ranges from 0, for no correlation, to 1, for perfect correlation. However, *Legates and McCabe* (1999) note that the r^2 criterion is limited because it does not account for differences between the means and variances of observed data and the predicted outputs. Nevertheless, r^2 continues to be one of the most commonly used criteria to evaluate an ANNs performance. The CE criterion also provides a measure of the correlation between model outputs and the observed data, but unlike r^2 it is sensitive to differences in the observed and predicted means and variances. The value of CE can range from $-\infty$ in the worst case to 1 for perfect correlation.

Alternatively, performance measures may be considered which take into account the parsimony of the model. It can be expected that a more complex model will be able to fit data better than a model with fewer degrees of freedom, and hence, less flexibility. However, whether the increase in fit is justifiable given the available data and the increased effort required to develop the model should be considered (*Dawson and Wilby*, 2001). The two most commonly used performance measures that account for the complexity of a model while measuring its performance include Akaike's information criterion (AIC) and the Bayesian information criterion (BIC), which combine a measure of fit with a term that penalises model complexity as shown in (3.7) and (3.8):

$$\mathrm{AIC} = -2\log L(\hat{\mathbf{w}}) + 2d \tag{3.7}$$

$$\mathrm{BIC} = -2\log L(\hat{\mathbf{w}}) + d\log N \tag{3.8}$$

where $\log L(\hat{\mathbf{w}})$ is the maximised log likelihood function and d is the dimension of the weight vector (i.e. the number of weights in the model). These measures are generally calculated based on the training data, or *in-sample*, performance.

Given the wide range of performance measures available, *Dawson and Wilby* (2001) state that "the problem then becomes deciding which (if any) are most appropriate to a particular application". To address this problem, *Bowden* (2003) used an approach proposed by *Diskin and Simon* (1977) to compare a number of possible error measures in order to select the most suitable measure for a particular application. Given X possible performance criteria, the procedure applied by *Bowden* (2003) involved training and cross-validating an ANN X times, using each criterion in turn to measure the performance of the model on the test (cross-validation) data set. Training was stopped when the error on the test set, as measured by the given criterion, was minimised; thus it was considered that the weights obtained were "optimal" according to that criterion. The "optimal" weights obtained using each of the different criteria were then used to calculate the values of the other criteria for which the weights were not optimal. The criterion whose "optimal" weights resulted in the best values of the other criteria, overall, was then selected as the most suitable performance criterion for the given application and was used to evaluate the performance of the trained ANNs.

3.2.1.2 Limitations and Conclusions

Diskin and Simon (1977) originally applied the procedure used by *Bowden* (2003) to conceptual hydrologic models calibrated using the pattern search optimisation method, which does not require derivatives of the objective function for gradient descent, and thus nondifferentiable and discontinuous objective functions can be used. Therefore, the procedure proposed by *Diskin and Simon* (1977) involved calibrating the models using the different performance criteria as objective functions in order to obtain the optimal weights. However, *Bowden* (2003) adapted the method such that it could be applied to ANNs trained by backpropagation, which is based on gradient decent and does require that the objective function used is differentiable (see Section 3.4.2.1). Therefore, the ANN was trained based on the SSE over the training data, but training was stopped according to the performance on the test set, which was calculated by the various performance criteria investigated. However, it is considered that this procedure is flawed, as stopping training according to a different error measure than that used to define the error surface could result in a set of weights that is optimal according to neither criterion used. Even if a global training algorithm were used that does not require a differentiable objective function, such that the method could be applied as proposed by *Diskin and Simon* (1977), with numerous possible error measures to consider, this procedure could be extremely time consuming and, as it does not take into account the possibility of becoming trapped in local minima, the results could still be misleading. According to *Bishop* (1995), as long as the assumptions of the least squares method are approximately correct, the SSE is the most suitable, and simple, form of performance measure or objective function for solving the regression problem given by (2.1). Therefore, for the ANNs developed in this research, the SSE will be used during training to evaluate the error on both the training and test data sets.

A further limitation of the method proposed by *Bowden* (2003) is that a different performance criterion may be used to evaluate the predictive performance of an ANN for each different case considered. As noted by *Dawson and Wilby* (2001) in their review of ANNs used for hydrological modelling, the absence of a standard error measure for evaluating the performance of trained ANNs has led to a lack of objectivity and consistency in the way the predictive performance of an ANN is assessed. *Legates and McCabe* (1999) suggest that a complete assessment of model performance should include at least one relative error measure, such as CE or r^2 (although use of r^2 is warned against), and at least one absolute error measure, such as the RMSE or MAE, with additional supporting information such as a comparison between the observed and simulated mean and standard deviations. Therefore, the RMSE, MAE, r^2 and CE will be used to evaluate the performance of the trained ANNs developed in this research. The AIC and BIC will also be used to evaluate the generalisability of the ANN models based only on the training data results.

In the approach proposed by *Bowden* (2003), by only considering predictive accuracy, the physical plausibility of the model is disregarded. As discussed in Section 2.2.5.2, ANNs treated as black-boxes, where no consideration is given to the modelled function, are generally undesirable as predictive models, as it is difficult to trust their reliability. Therefore, it is considered that ANNs need to be evaluated not only in terms of their predictive performance, but also in terms of their ability to capture the underlying relationship. However, as there is no widely accepted method for interpreting what has been learnt by an ANN, this is an area that still requires further investigation. In this research, a number of input importance measures will be investigated for quantifying the strength of the modelled relationships between individual inputs and the output in order to determine which measure, if any, is most appropriate for assessing the relationship modelled by an

ANN.

3.2.2 Choice of Data Sets

3.2.2.1 Review of current practice

At a minimum, the available data need to be divided into two subsets; one for training and the other for independent validation of the trained model. However, in general, three data sets are required; namely a training, testing and validation set. As discussed in Section 2.2.5.1, cross-validation with an independent data set is commonly employed during training to prevent overfitting. However, the validation data must not be used in any context during the training and model selection process (*Maier and Dandy*, 2000a), therefore, a third independent data set is required. The same applies if a trial-and-error process is used to optimise the network architecture or to select the network inputs or parameters of the optimisation algorithm used. Therefore, the training data are used to find an optimal set of network weights, the testing data are used to select the best network during development and, if cross-validation is employed, to prevent overfitting, and the validation set is used to validate or confirm the generalisability of the selected model.

Traditionally, the data have been divided arbitrarily without giving consideration to the statistical properties of the respective data sets (*Maier and Dandy*, 2000a). However, the way in which the data are divided can significantly influence an ANN's performance (*Yapo et al.*, 1996; *Tokar and Johnson*, 1999). As it was also discussed in Section 2.2.5.1, there is no information provided to an ANN about the form of the solution surface other than that contained in the training data (i.e. there is no incorporation of physical concepts). Therefore, to achieve good generalisation of the data generating relationship, the training data must be a representative sample of the population from which the data were generated. Furthermore, it follows that if the training data must be representative of the data population to achieve generalisability, the toughest evaluation of generalisability is if the testing and validation data are also representative subsets. Also, an unrepresentative test set could bias the cross-validation procedure and the selection of the optimum network architecture.

While it is important for each of the data subsets to be representative of the data population, the proportion of samples to include in each of the subsets is also an important consideration. *ASCE Task Committee* (2000b) define an optimal training data set as "one that fully represents the modelling domain and has the minimum number of data pairs in training". This is because large sets of repetitive data can slow down training while only marginally improving network performance. However, due to the time and cost involved in data collection, in many practical circumstances the available data are limited (*Flood and Kartam*, 1994). Therefore, it is again important to consider the relative sizes of the subsets, in order to include the maximum amount of information in the training data set.

Bowden et al. (2002) and *Bowden* (2003) proposed two methods for systematically dividing the available data into statistically representative subsets. The first method involved the use of a genetic algorithm (GA) to minimise the difference in the statistics (mean and standard deviation) of the subsets and the second employed a self-organising map (SOM) (*Kohonen*, 1982) to cluster the data into groups of similar data patterns, such that samples from each group could be included in each of the subsets.

The GA used for dividing the data was designed to allocate the available data into training, testing and validation sets of prespecified proportions according to a set of pseudo random numbers. The decision variable being optimised was the pseudo random number seed used to generate the random sorting (i.e. the random seed used to determine the optimal allocation of data into subsets). The objective function minimised was the sum of the absolute difference in mean and standard deviation values for each input and output variable between each pair of the three subsets:

$$\begin{aligned}\text{Objective fn} \;=\; & \sum_{i=1}^{K+1} \Big\{ [\mu(i)_{train} - \mu(i)_{test}] + [\mu(i)_{test} - \mu(i)_{validation}] \\ & + [\sigma(i)_{train} - \sigma(i)_{test}] + [\sigma(i)_{test} - \sigma(i)_{validation}] \Big\} \qquad (3.9)\end{aligned}$$

where K is the number of inputs, and μ and σ are the mean and standard deviation of the input or output variable, respectively. To ensure that the maximum and minimum values of each variable were included in the training set, penalty constraints were added to the objective function. Penalty constraints were used, rather than manually removing extreme values from the data and placing them in the training set, as it was noted that there may be a trade-off between keeping the statistics of the training, testing and validation sets the same and ensuring that the extreme values are in the training set.

With the second data division method, the SOM was used to cluster the data by presenting the ANN input and output variables as the SOM's inputs. A SOM grid size was specified, where each cell in the grid represents a node in the Kohonen layer, and by training the SOM, similar data samples were clustered into each of the grids. This is illustrated in Figure 3.2, where each square represents a cluster and each dot represents a sample of data.

Using the method employed by *Bowden* (2003), three data records from each cluster were sampled and allocated to each of the training, testing and validation subsets. However, if a cluster only contained one record, this record was allocated to the training set.

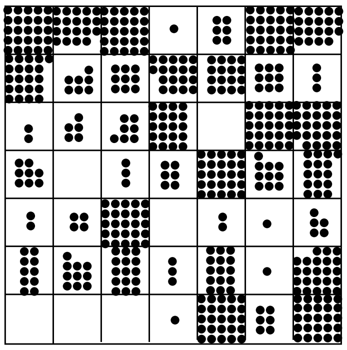

Figure 3.2 SOM data division

If a cluster contained two records, one record was placed in the training set and the other in the testing set. It was considered that an advantage of this technique over other data division methods is that it avoids the need to arbitrarily select which proportion of data to include in each subset and that it is capable of constructing a representative training data set using the minimum number of samples.

It was found that each method was capable of dividing the available data into statistically similar subsets and the predictive performance of the models developed using these subsets was significantly greater than when the data were divided arbitrarily.

3.2.2.2 Limitations and Conclusions

It is considered that each approach proposed by *Bowden* (2003) suffers from some limitations. Firstly, dividing the data with a GA can be very time consuming as many different combinations for arranging the data need to be compared to find the best allocation. In the simple example given by *Bowden* (2003), if there are 60 data samples that must be divided into training, testing and validation sets consisting of 40, 10 and 10 data samples respectively, then there are:

$$\frac{60!}{40! \times 10! \times 10!} = 7.7 \times 10^{20}$$

ways of arranging the data samples. In reality, it is unlikely that an optimal division of the data will be found within a reasonable time frame, although good results can still be obtained. More importantly, though, in the GA method described by *Bowden* (2003), the cross-over operator is unable to function as it should. Crossing over two random number seeds does not result in a set of random numbers that share the properties of the parents, thus the method as presented is only relying on selection and mutation to find the appropriate division of the data, which further slows the process (for details of the operation of a GA see Section 3.4.2.2).

Using the SOM data division method, the number of data samples allocated to each data subset depends on the Kohonen grid size specified. However, as noted by *Bowden* (2003), there is no theoretical principle for determining the optimum size of the Kohonen layer. According to *Shahin et al.* (2004), the grid size specified can have a significant impact on the results obtained using the SOM data division method, as the underlying assumption of the approach is that the data samples in one cluster provide the same information in high-dimensional space. *Bowden* (2003) stated that the grid size was selected such that it was large enough to ensure that the maximum number of clusters were formed from the available data. However, theoretically, the grid size could be specified large enough such that each grid only contained one sample of data, making it impossible to choose representative subsets. Furthermore, by only selecting one sample from each cluster for each data set, the amount of data used for ANN development is significantly reduced. In one of the case studies presented by *Bowden* (2003), 2005 available data samples were reduced to 147 samples using this method, with only 49 samples allocated to each data subset. Such a reduction in data may result in a significant loss of information, in which case the resulting training data set would not adequately represent the population of data. The amount of information that is lost through such data reduction depends on the intra-cluster variation. If this were large for even some of the clusters, important information may be omitted from the training set by only selecting one sample from each cluster.

Nevertheless, the SOM method appears to be a promising approach for systematically dividing the data into statistically similar subsets; thus, it is considered that no further investigation is required on data division methods in the present research. However, it is proposed that, within reason, the entire clustered data set should be divided into the three subsets, with 64% allocated to the training set, 16% to the testing set and 20% to the validation set, which are the proportions proposed by *Bowden* (2003) when using the GA data division method. Furthermore, in the present research, analytical measures will be used to aid selection of the optimal SOM grid size. As mentioned above, specifying a grid size that is too large can result in too many clusters containing single data points, making it difficult to choose representative subsets. Conversely, if the grid size is too small, there may be significant variation within the clusters. To determine the optimum number of clusters for data division, *Shahin et al.* (2004) used the average silhouette width $\bar{s}(k)$, which is an average measure of how well the data samples lie within the clusters they have been assigned to at the end of the clustering process. This measure is calculated by evaluating the silhouette value $s(i)$ for each data sample $i = 1, \ldots, N$, and taking the average over the data set. The silhouette value for a data sample i is given by (*Kaufman* *and Rousseeuw*, 1990):

$$s\left(i\right) = \frac{b\left(i\right) - a\left(i\right)}{\max\left\{a\left(i\right), b\left(i\right)\right\}} \tag{3.10}$$

where $a(i)$ is the average dissimilarity of sample i to all other samples in a cluster A; and $b(i)$ is the smallest average dissimilarity of sample i to all points in any cluster E different from A. If $s(i)$ is close to 1, the "within" dissimilarity $a(i)$ is smaller than the smallest "between" dissimilarity $b(i)$. Therefore it is considered that sample i has a strong membership to cluster A. The optimum number of clusters can be determined by choosing the number of clusters that maximises the value of $\bar{s}(k)$. However, for only a small number of clusters, the smallest "between" dissimilarity may be reasonably large, which can result in a high value of $\bar{s}(k)$, regardless of how similar the samples within a cluster are. Therefore, a "discrepancy" measure, which indicates the total "within" dissimilarity, will be used together with the average silhouette width to select the SOM grid size. This discrepancy measure is given by:

$$\text{Discrepancy} = \sum_{i=1}^{N} \left\| \{\mathbf{x}_i, y_i\} - \hat{\mathbf{W}}_i \right\| \tag{3.11}$$

where $\hat{\mathbf{W}}_i$ is the weight vector associated with the cluster to which the sample $\{\mathbf{x}_i, y_i\}$ is assigned. The smaller the discrepancy value, the better the samples "fit" to the clusters to which they have been assigned. However, if there are a large number of clusters containing a small number of samples, the discrepancy may be small, although the dissimilarity between the clusters may also be small. Consequently, the discrepancy value needs to be used in conjunction with the average silhouette width to select the SOM grid size.

3.2.3 Data Pre-Processing

3.2.3.1 Review of current practice

Data pre-processing involves transforming the data into a format that will enable easier and more effective processing by the ANN. This may involve rescaling, standardisation, de-trending, distribution transformation and removal of outliers. As discussed in *Bishop* (1995), being universal function approximators, ANNs should, in principle, be able to map raw input data directly onto the required output values; however, in practice this approach would normally give poor results, as the model is relied upon too much to find appropriate transformations. Therefore, pre-processing is an important step in ANN development.

The simplest and most commonly used form of pre-processing is linear transformation, which involves transforming the data such that the variables have similar values

(*Bowden*, 2003; *Bishop*, 1995) and includes rescaling and standardisation. Rescaling generally refers to the scaling of data between specified upper and lower bounds, whereas standardisation often refers to statistical standardisation where a measure of location (e.g. mean) is subtracted and the result is divided by a measure of scale (e.g. standard deviation). If x_i is the ith raw value of variable $\mathbf{x}$, the ith linearly transformed value $\acute{x}_i$ can be obtained by rescaling or by standardisation according to (3.12) and (3.13), respectively:

$$\acute{x}_i = \left(\frac{x^T_{high} - x^T_{low}}{x_{max} - x_{min}}\right) \cdot x_i + \left(\frac{x^T_{low}x_{max} - x^T_{high}x_{min}}{x_{max} - x_{min}}\right) \tag{3.12}$$

$$\acute{x}_i = \frac{x_i - x_{mean}}{x_{stdev}} \tag{3.13}$$

In (3.12), x_{max} and x_{min} are the maximum and minimum values of the untransformed variable, while x^T_{high} and x^T_{low} are specified upper and lower bounds which become the new maximum and minimum values of the transformed data, respectively. In (3.13), x_{mean} and x_{stdev} are the mean and standard deviation of variable $\mathbf{x}$, respectively.

Since different variables generally span different ranges, and because typical values do not necessarily reflect the relative importance of inputs in determining the output, transforming the inputs to a similar scale is particularly important in ensuring that each variable receives equal attention during training (*Maier and Dandy*, 2000a). *Sarle* (2002) recommends either scaling the inputs between $[-1, 1]$ or standardising them to a mean of zero and a standard deviation of one, as in (3.13), as it is important to centre the inputs around the origin in order to get good random initialisations of the weights. Recently, *Shi* (2000) suggested that if the input data are linearly scaled within some specified limits (e.g. [-1,1]), for some variables a large proportion of the data may be confined to a very small range (e.g. [-1,-0.95]), making it difficult to achieve a continuous mapping for such inputs on the entire input range. To overcome this, *Shi* (2000) proposed a distribution transformation method to transform the input data to uniform distributions on the range [0,1] using the cumulative distribution functions (CDF) of the input variables. If $F(\mathbf{x})$ is the CDF of input $\mathbf{x}$, the transformed data are obtained by $\acute{x}_i = F(x_i)$.

If bounded activation functions are used on the output layer (see Section 3.2.5), it is also necessary to scale the target data, such that they are proportionate with the limits of the activation function. For example, if the hyperbolic tangent activation function, which has the limits [-1,1], is used on the output nodes, the data should be scaled between -0.9 and 0.9 or -0.8 and 0.8 (*Bowden*, 2003). It is not recommended that the data be scaled to the extreme limits of the activation function, as this will likely cause the size of the weight updates to become extremely small, leading to the occurrence of flatspots in training (*Maier and Dandy*, 2000a). Furthermore, when scaling the target data to force the values into the range of the output activation function, it is important to use lower and upper bounds of the target variable, rather than the minimum and maximum values in the training set (*Sarle*, 2002). For example, the lower bound of a variable that can only take positive values is zero; therefore, x_{min} should be substituted with zero in (3.12). If an unbounded activation function (e.g. linear) is used at the output layer, it is not strictly necessary to rescale the target data. However, by ensuring that the input and output variables are of order unity, either through rescaling or standardisation, it can be expected that the weights will also be of order unity, which makes it easier to randomly initialise the weights appropriately (*Bishop*, 1995).

There has been some confusion as to whether further transformations, such as those used in traditional statistical modelling, need to be applied to the data when using an ANN. In assuming that the model residuals are normally distributed with a constant variance, the most commonly used methods in regression analysis also make the assumption that the target data are (approximately) normally distributed, either implicitly (e.g. least squares estimation) or explicitly (e.g. maximum likelihood estimation). In traditional statistical modelling, nonlinear mathematical functions, such as the square root, logarithm or inverse, are widely used to transform the data in order to reduce the non-normality and stabilise the variance in the data (*Osborne*, 2002). In traditional time series modelling it is also common to remove deterministic components in the data such as trends and seasonality (*Maier and Dandy*, 2000a). However, in a study by *Faraway and Chatfield* (1998), results of empirical trials indicated that there was no improvement in an ANN's performance when a logarithmic transformation was used and that the performance deteriorated when the seasonality was removed from the data.

Bowden et al. (2003) and *Bowden* (2003) investigated the effects of six different transformations on the performance of ANN models, as outlined below:

1. Linear Transformation - The inputs were linearly rescaled between -1.0 and 1.0, while the network outputs were rescaled between -0.8 and 0.8 to be commensurate with the limits of the hyperbolic tangent activation function used on the output layer node.

2. Logarithmic Transformation - A logarithmic transformation of both the inputs and outputs was performed.

3. Histogram Equalization Transformation - The distribution transformation method of *Shi* (2000) is dependent on fitting a probability density function (PDF) to each of

the input variables at an acceptable level of significance. To ensure an appropriate fit of the input PDFs, a discrete version of distribution transformation known as histogram equalization (*Looney*, 1997) was applied to the input data.

4. Kernel Transformation - Similar to the histogram equalization transformation above, this transformation was developed by *Bowden* (2003) to transform the input variables according to the distribution transformation method of *Shi* (2000). The method uses univariate kernels to approximate the PDF of each input series, which is then integrated to approximate the required CDF.

5. Seasonal Standardisation - Deterministic seasonality was removed from the inputs and outputs by subtracting a seasonally varying mean and dividing by a seasonally varying standard deviation.

6. Transformation to Normality - A two-step transformation to normality was proposed by *Bowden* (2003), combining the histogram equalization transformation to compute the CDF of an input series (uniform distribution between 0 and 1) with an approximation of the inverse Gaussian CDF (*Beasley and Springer*, 1977) to produce the corresponding normal deviates.

When applied to a water resources case study, *Bowden* (2003) found that the models developed using the linear, histogram equalization and kernel transformations performed significantly better than those developed using the logarithmic, seasonal and normality transformations for the training, testing and validation sets. It was concluded that the latter three transformations distorted the original relationships between variables in a way that was not beneficial to ANN learning. It was also found that, while the models developed using data transformed by histogram equalization and kernel transformation had (slightly) superior performance when tested on data within the training domain, these models were not as robust as the models developed using linearly transformed data when tested on data outside of the training domain. Analysis of the residuals produced by the ANN models trained using linearly transformed data showed that the assumptions of least squares error model were satisfied, indicating that the data did not require any further transformations other than linear rescaling.

3.2.3.2 Limitations and Conclusions

It is recommended by *Osborne* (2002) that all data transformations should be utilised with care and never without a clear reason. In the investigation carried out by *Bowden* (2003), there was no clear reason to transform the data in some cases. For example, seasonality was removed from exogenous input variables when it was not necessary; similarly, logarithmic and normality transformations were applied to input data when there was no need, as discussed below. Data transformations should be applied in order to comply with the requirements of the modelling technique used. In traditional time series modelling, it is necessary to remove seasonality from the output time series, as it may conceal the true underlying movement in the data. However, it is not necessary to remove the seasonality from other external input variables included in the model. When using linear regression, it is important to transform both input and target data to normality to achieve constant variance in the residuals and linearity in the equation. However, transformations should not be applied simply to achieve linearity when using nonlinear modelling techniques. Furthermore, transformations to normality should only be applied when the data or resulting model residuals are substantially non-normal, as regression models are generally robust, to some extent, to violation of the assumption of normally distributed residuals (*Osborne and Waters*, 2002). It is more important that the distribution of the data be approximately symmetrical and not have a heavy tail than to be normally distributed (*Masters*, 1993). As data transformations can alter the fundamental nature of the data, it is possible that unnecessary transformations may have distorted the results of the investigation carried out by *Bowden* (2003).

Following the recommendations of *Osborne* (2002), a more systematic approach to data pre-processing will be used in this research, with transformations only applied to data when there is a clear reason. Due to the general function mapping abilities of ANNs, less emphasis has to be placed on careful optimisation of data pre-processing than in traditional linear regression or time series modelling (*Bishop*, 1995). However, as discussed in Section 3.2.3, there are well established reasons why some pre-processing, particularly linear transformations, can significantly improve the performance of an ANN. In this research, all inputs and outputs will be standardised to have a mean of 0 and a standard deviation of 1, as recommended by *Sarle* (2002), except if a bounded activation function is used on the output layer, in which case the target data will be scaled to be commensurate with the limits of this function. Furthermore, once the model is fitted, diagnostic checking of the residuals will be carried out to determine whether the assumptions of the regression model specified by (2.1) have been met. These assumptions are:

1. The mean of ϵ is zero;

2. The variance of ϵ is constant;

3. ϵ is statistically independent; and

4. ϵ is normally distributed.

If any of these assumptions are significantly violated, a nonlinear transform of the target data, such as the logarithm, square root or inverse, will be considered.

3.2.4 Determination of ANN Inputs

3.2.4.1 Review of current practice

Determining what the important inputs are for a given problem can be one of the most critical decisions in ANN model development, as the inputs contain the important information required to define the data-generating relationship. This can also be one of the most difficult tasks in water resources modelling because many of the hydrological and environmental processes acting upon these systems are poorly understood. As water resource systems vary in space and time, potentially important inputs may include observations of causal variables at different locations and time lags, as well as lagged observations of the dependent variable of interest. Therefore, the number of potentially important inputs can be large. However, the inclusion of unnecessary inputs is undesirable, as such inputs do not provide any useful information about the underlying relationship, but increase the size and complexity of the network, making the task of extracting important information from the data difficult. On the other hand, omitting key inputs results in a loss of important information, which can be detrimental to the predictive performance of an ANN.

In the past, selection of important inputs has been given relatively little attention, as it has been considered that, being a data-driven modelling approach, ANNs should determine automatically which inputs are critical (*Maier and Dandy*, 2000a). However, as mentioned above, presenting a large number of inputs to an ANN increases the size and complexity of the model; thus, slowing processing time, reducing interpretability and increasing the potential of overfitting. Therefore, there are considerable advantages in using analytical techniques to help select important inputs. According to *Bowden* (2003), there are only a small number of recent papers that treat input determination as an important step in ANN development. The methods used in these papers were classified into five broad groups by *Bowden* (2003); *Bowden et al.* (2005a), as discussed below.

1. Methods that rely upon the use of *a priori* knowledge of the system being modelled. Selecting important inputs for ANNs usually always requires some degree of *a priori* knowledge, as it is necessary to use some judgement to select a set of potentially important inputs. However, relying solely on *a priori* knowledge for input determination requires a good understanding of the system being modelled

(*ASCE Task Committee*, 2000a). Inspection of time series plots of potential inputs and outputs can help in selecting important inputs (*Maier and Dandy*, 1996), as can existing physically-based or conceptual models (*Zealand et al.*, 1999; *Shamseldin*, 1997) or previous studies of the system or similar systems (*Thirumalaiah and Deo*, 2000; *Wei et al.*, 2001). However, if the system is less well understood, the use of analytical techniques, in conjunction with *a priori* knowledge, would be beneficial (*Maier and Dandy*, 2000a).

2. Methods based on linear cross-correlation. Cross- and auto-correlation analysis has been popular for selecting appropriate inputs (including lagged observations of the target variable) for ANNs in the field of water resources modelling (*Fernando and Jayawardena*, 1998; *Imrie et al.*, 2000; *Coulibaly et al.*, 2000; *Lekkas et al.*, 2001; *Sudheer et al.*, 2002). This method uses the strength of the cross-correlations between potential input variables and the output to select important causal inputs or the strength of autocorrelations to select appropriate lags. However, a significant disadvantage of cross-correlation analysis is that it is only able to detect linear dependence between two variables and is therefore not optimal for selecting inputs of nonlinear relationships.

3. Methods that utilise a heuristic approach. Using such approaches, different combinations of inputs are trialled in order to find the combination that results in the best model performance. A stepwise selection approach is commonly employed to avoid consideration of all subsets of inputs. Stepwise selection can be applied in a forward manner, where, given a set of selected inputs, the input that improves the model's performance most is added to the final model, beginning with the best single input. Alternatively, backward selection may be used, where an ANN is first developed with the set of all potentially important inputs, and inputs that reduce the performance least when deleted are sequentially removed from the model. *Tokar and Johnson* (1999), *Luk et al.* (2000) and *Maier and Dandy* (2001) used a heuristic input selection approach. The disadvantage of these approaches is that they are computationally intensive, requiring the ANN to be retrained each time a new combination of inputs is trialled. Furthermore, heuristic model selection methods are based on trial-and-error, and consequently, there is no guarantee that they will find the globally best subset of inputs.

4. Methods that extract knowledge contained within the trained ANN. Methods such as sensitivity analysis, saliency analysis and partitioning of connection weights are

designed to quantify the relative importance of inputs in trained ANNs (*Olden et al.*, 2004). Of these methods, sensitivity analysis is the most commonly used for input selection in the field of water resources modelling (*Maier and Dandy*, 1996, 1997; *Liong et al.*, 2000; *Wei et al.*, 2001). The main disadvantage of these approaches is that if all inputs in the set of potentially important inputs are not statistically independent, the effects of dependent inputs generally cannot be separated (*Sarle*, 2002). Additionally, it can be difficult to determine the statistical significance of input variables using these approaches; thus, subjective judgement is often required to determine at what threshold value inputs should be removed or retained in the network (*Olden and Jackson*, 2002).

5. Methods that use various combinations of the above four approaches. According to *Bowden* (2003), a number of papers also report the combined use of some of the above approaches.

Bowden (2003) and *Bowden et al.* (2005a) proposed two input determination approaches; one *model-based* and one *model-free*. Model-based approaches rely on the modelled input-output relationship to determine the dependence of the output on each input variable, while model-free approaches use some statistical measure of dependence (e.g. correlation) to determine the strength of the relationship between each input and the output. The model-based input determination approach proposed by *Bowden* (2003) involved the use of a genetic algorithm (GA) together with a general regression neural network (GRNN), which is a type of supervised feedforward ANN with the advantages of having a fixed architecture and being relatively quick to train. Using this hybrid GAGRNN approach, the GA is employed to evolve the GRNN model with the optimal set of inputs, determined according to the corresponding predictive error. To begin the algorithm, a population of GRNN models is randomly initialised, each with a different subset of input variables, as depicted by a binary string. For example, if there are K potentially important inputs, the string will have length K and if the kth value of the string is equal to one, the kth input is included, otherwise, if it equals zero, it is not included. The GRNN models are then trained and the corresponding predictive error is calculated for each GRNN. Selection, crossover and mutation operators are then used to evolve the population of GRNN models over a number of generations to obtain the GRNN model with the optimal inputs (i.e. that producing the smallest predictive error). This method requires that the set of potentially important inputs optimised by the GAGRNN are statistically independent; therefore, an input preprocessing stage was used prior to the GAGRNN to reduce the original set of inputs to a set of independent inputs. For this, *Bowden* (2003) used principle component analysis (PCA) and SOM clustering techniques.

The model-free input selection approach proposed by *Bowden* (2003) was an adaptation of the stepwise partial mutual information (PMI) input selection procedure developed by *Sharma* (2000). The PMI criterion is a measure of (linear or nonlinear) partial dependence between an independent variable $\mathbf{x}$ and a dependent variable $\mathbf{y}$ and is given by:

$$\mathrm{PMI} = \frac{1}{N}\sum_{i=1}^{N} \log_e \left[\frac{f_{\mathbf{x}'\mathbf{y}'}\left(x_i', y_i'\right)}{f_{\mathbf{x}'}\left(x_i'\right) f_{\mathbf{y}'}\left(y_i'\right)} \right] \tag{3.14}$$

where $f_{\mathbf{x}'}\left(x_i'\right)$ and $f_{\mathbf{y}'}\left(y_i'\right)$ are the marginal PDFs of $\mathbf{x}'$ and $\mathbf{y}'$, respectively, and $f_{\mathbf{x}'\mathbf{y}'}\left(x_i', y_i'\right)$ is the joint (bivariate) PDF of $\mathbf{x}'$ and $\mathbf{y}'$. The PMI is a "partial" measure because it depends on the inputs already selected and measures any additional dependence a new input can add to the existing prediction model. In (3.14), $\mathbf{x}'$ and $\mathbf{y}'$ represent the residual information in variables $\mathbf{x}$ and $\mathbf{y}$, once the effect of the existing predictor(s) has been taken into consideration.

Briefly, the stepwise PMI input selection algorithm is carried out as follows (*Sharma*, 2000; *Bowden et al.*, 2005a):

1. Identify the set of potentially important inputs using *a priori* knowledge.

2. Estimate the PMI score between the dependent variable $\mathbf{y}$ and each of the potential inputs, conditional on the existing predictor(s), using (3.14).

3. Identify the potential input with the highest PMI score.

4. Create a set of randomised samples of the potential input identified in step 3 by randomly bootstrapping the input series to remove the dependence that existed between that input and the dependent variable. Estimate the 95th percentile randomised sample PMI score for this potential input.

5. If the PMI score for the identified potential input is higher than the 95th percentile randomised sample PMI score, select the input as an important predictor and remove it from the set of potential inputs. If the PMI score is less than the 95th percentile randomised sample PMI score, go to step 7.

6. Repeat steps 2–5.

7. Stop once all important predictors have been selected.

In *Bowden* (2003), the two input selection approaches were applied to a number of synthetic case studies with known dependence attributes. It was found that the model-free stepwise PMI approach was able to correctly select the important input variables for each test case, whereas the GAGRNN was only able to do this for the simplest of the synthetic data sets, and only when the SOM clustering technique was used to reduce the dimensionality of the original set of inputs. Although the SOM-GAGRNN method did not always select the actual model inputs, it was found that, overall, the models developed using this input selection method were able to achieve similar predictive performance to those developed using the stepwise PMI input selection method. It was therefore concluded by *Bowden* (2003) that both approaches are suitable for ANN input selection when predictive performance is the primary aim. However, as it was demonstrated that only the stepwise PMI algorithm was able to identify the actual model inputs to the test problems, it was considered that this method should be used when insight into the underlying process is of utmost importance.

3.2.4.2 Limitations and Conclusions

As discussed in *Bowden* (2003), both the stepwise PMI and GAGRNN input selection methods suffer from a number of limitations. Using the PCA-GAGRNN approach, it is not possible to determine the exact inputs to the system; rather, the important principal components (PCs) are selected using the GAGRNN, where each PC is a linear combination of all of the original inputs. Therefore, is likely that even important PCs contain extraneous inputs. Furthermore, only linear dependence between variables is considered in PCA; hence, this technique is not optimal for nonlinear relationships. The SOM-GAGRNN approach is an improvement over the PCA-GAGRNN technique, as the SOM is able to account for nonlinear relationships and does not combine the inputs in any way. However, using the SOM input reduction method, it cannot be guaranteed that the actual model inputs will be included in the reduced set of independent inputs, as an input that is highly correlated with the actual input may be selected if it is closest to the cluster centre. Moreover, like all model-based approaches, the resulting set of inputs selected using the GAGRNN is dependent upon the model developed. If the model does not properly represent the relationships between the candidate inputs and the output, the resulting set of selected inputs is likely to be suboptimal. Being model-free, the stepwise PMI approach is not dependent upon a modelled relationship and thus has an advantage over the GAGRNN approach. Furthermore, it does not require any input preprocessing step, as it is able to account for redundant inputs. However, to estimate $f_{\mathbf{x}'}(x_i')$, $f_{\mathbf{y}'}(y_i')$

and $f_{\mathbf{x}'\mathbf{y}'}(x'_i, y'_i)$ in (3.14), a kernel density estimator is used, which requires specification of a "bandwidth" to prescribe the smoothness of the estimated density. *Bowden* (2003) and *Sharma* (2000) used the Gaussian reference bandwidth (*Silverman*, 1986), which may be unsuitable for highly non-Gaussian data, leading to the selection of a sub-optimal set of model inputs. Nevertheless, as it was demonstrated that the stepwise PMI input selection technique was able to correctly select the model inputs for the test cases in *Bowden* (2003), this approach will be used in the current research. An additional advantage of this approach is that the magnitude of the PMI scores give useful information regarding the relative importance of each input.

3.2.5 Determination of ANN Architecture

3.2.5.1 Review of current practice

In the regression equation given by (2.1), the function $f(\cdot)$ is determined by the network architecture, which, in turn, is defined by the number of input and output nodes, the number and configuration of hidden layer nodes, the connectivity between the nodes and the types of activation functions used within the network. Therefore, it is the network architecture which determines model complexity. Only fully connected feedforward MLPs are considered in this research, where the input and output nodes are fixed according to the number of input and output variables included in the model, respectively. Therefore, determination of an appropriate ANN architecture, and thus model complexity, comes down to selecting the number and configuration of hidden layer nodes and choosing which activation functions to use on the hidden and output layers.

Activation functions are needed to introduce nonlinearity into an ANN. Any nonlinear function is capable of this; however, when a gradient descent type search algorithm is used for training, the activation function must be continuous and differentiable (*Sarle*, 2002). Sigmoidal activation functions, such as the logistic sigmoid or the hyperbolic tangent (tanh), given by (3.15) and (3.16), respectively (where zin is the summed input to a node), are most commonly used on the hidden layer nodes (*Maier and Dandy*, 2000a). There may be some practical advantage to using tanh activation functions rather than logistic, as empirically, the tanh function has been found to give faster convergence of the training algorithm (*Maier and Dandy*, 1998a). Although sigmoidal functions may also be used on the output layer, the linear, or identity, activation function, given by (3.17), is commonly used, as this function does not restrict the range of the possible outputs to the range that would be attainable if a bounded function were used (*Bishop*, 1995). However, there are sometimes good reasons to use a bounded activation function at the output nodes;

for example, if there are known upper and lower bounds for the target variable. The tanh, logistic and linear activation functions are illustrated in Figure 3.3 where it can be seen that the tanh and logistic activation functions are bounded on the ranges [-1,1] and [0,1], respectively, whereas the linear function is unrestricted.

$$g(zin) = \frac{1}{1+e^{-zin}} \tag{3.15}$$

$$g(zin) = \frac{e^{zin} - e^{-zin}}{e^{zin} + e^{-zin}} \tag{3.16}$$

$$g(zin) = zin \tag{3.17}$$

An ANN with sigmoidal hidden units and linear output units is not limited to modelling smooth nonlinear functions. A sigmoidal hidden node may approximate a linear hidden node by arranging all of the weights feeding into the node to be very small, such that the summed input lies on the linear part of the sigmoid curve. Similarly, a sigmoidal hidden unit may approximate a step function by setting all of the weights feeding into it to very large values (*Bishop*, 1995). In fact, it has been shown that a one hidden layer network with this configuration of activation functions can essentially approximate any continuous functional mapping to arbitrary accuracy, provided that the number of hidden nodes is sufficiently large (*Cybenko*, 1989; *Hornik et al.*, 1989; *Bishop*, 1995).

The flexibility in ANN architecture determination primarily lies in selecting the number and configuration of hidden layer nodes, which, in turn, determine the number of weights in the model. However, as discussed in Section 2.2.5.1, this is one of the most critical and difficult tasks in designing an ANN. Generally, the number of hidden layers is fixed, then the number of nodes in each hidden layer is chosen (*Maier and Dandy*,

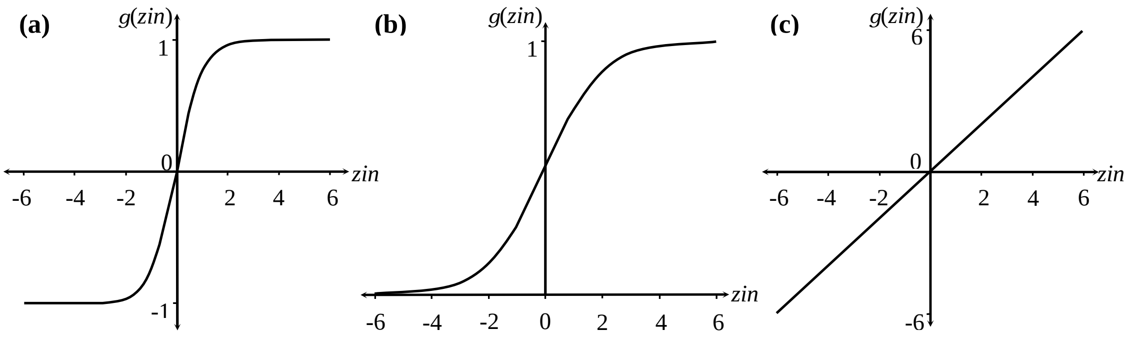

Figure 3.3 Typical activation functions (a) tanh, (b) logistic and (c) linear or identity.

2000a). As mentioned above, only one hidden layer is required to approximate any continuous function, thus, most studies only utilise one hidden layer (*Bowden*, 2003). However, *Cheng and Titterington* (1994) note that even though a one hidden layer network may be adequate, the number of hidden nodes required can be prohibitive. *Flood and Kartam* (1994) argue that it may be better to use two hidden layers to provide the greater flexibility necessary to model complex shaped solution surfaces, as a two hidden layer ANN can yield an accurate approximation with fewer weights than a one hidden layer network (*Bebis and Georgiopoulos*, 1994). However, in an empirical study by *de Villiers and Barnard* (1992), it was found that there was no difference in the optimal performance of one or two hidden layer ANNs with the same complexity, which led to the conclusion that there is no reason to use two hidden layer networks in preference to one hidden layer networks "in all but the most esoteric applications". Furthermore, the use of two hidden layers can exaccerbate the problem of local minima on the solution surface, making them more difficult to train (see Section 3.2.6) (*Sarle*, 2002; *de Villiers and Barnard*, 1992).

Although the above results are useful in selecting the number of hidden layers, they do not provide any guidance in selecting the number of hidden nodes. Theoretically, the optimal number of hidden nodes is that which results in the smallest network able to adequately capture the underlying relationship in the data. However, in the past, the selection of hidden nodes has been somewhat arbitrary, as the optimal number of hidden nodes is highly problem dependent; yet, there exists no systematic model selection method to ensure the optimal network will be chosen. A balance is required between having too few hidden nodes such that there are insufficient degrees of freedom to adequately capture the underlying relationship (i.e. the data are underfitted), and having too many hidden nodes such that the model fits to noise in the individual data points, rather than the general trend underlying the data as a whole (i.e. the data are overfitted). This is illustrated in Figure 3.4, which shows an example of an ANN (a) generalising well to the underlying trend in the data, (b) overfitting the data and (c) underfitting the data. The generalisability of the models shown in Figures 3.4 (b) and (c) would be poor, as a result of having too much and too little flexibility to fit the data, respectively. This can be explained by considering the bias-variance tradeoff, as discussed in *Bishop* (1995), where the model error is decomposed into the sum of the *bias* squared plus the *variance*. Bias results from a network function that is on average different from the regression function, whereas variance results from a network function that is overly sensitive to the particular data set. The model in Figure 3.4 (b) has negligible bias as the model fits that data perfectly, however, this model would generalise poorly due to a large variance in the error when applied to a different

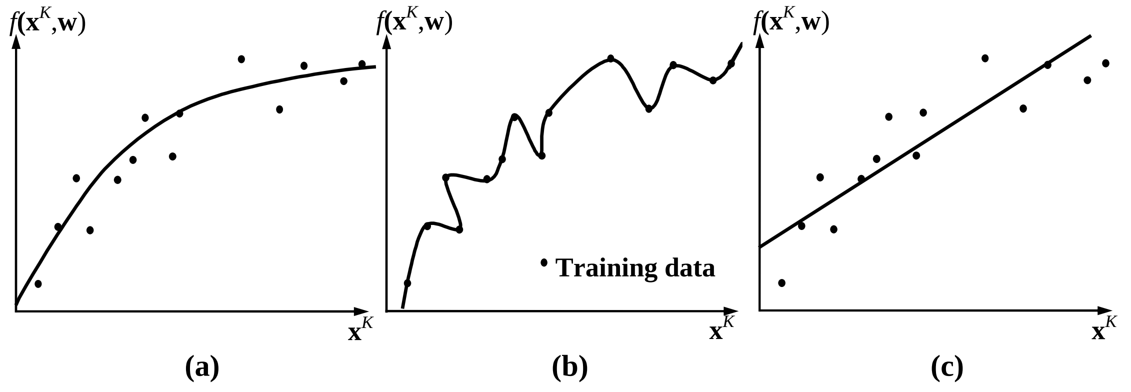

Figure 3.4 Example of an ANN (a) estimating the general underlying trend in the data, (b) overfitting the data and (c) underfitting the data.

data set. On the other hand, the model in Figure 3.4 (c) is insensitive to the data set to which it is applied and therefore has negligible variance. However, this model would also generalise poorly as a result of having a large bias across all data sets. This highlights the need to optimise the number of hidden nodes (and hence model complexity) such that there is a balance between the bias and variance and the best generalisability is achieved.

Numerous techniques have been suggested to make decisions regarding network architecture less arbitrary, such as pruning and construction algorithms, statistically based comparison procedures and 'rules of thumb' (*Bebis and Georgiopoulos*, 1994; *Anders and Korn*, 1999; *Qi and Zhang*, 2001; *Basheer and Hajmeer*, 2000). *Maier and Dandy* (2000a, 2001) discuss a number of general guidelines that have been proposed in the literature to help select the optimal number of hidden nodes by relating the number of training samples to the size of the network. For example, *Masters* (1993) suggests that there should be twice the number of training samples as there are weights in the network, whereas *Weigend et al.* (1990) suggest that this ratio should be 10:1. More formally, by analysing the asymptotic gain in generalisation error when early stopping is performed, it has been shown by *Amari et al.* (1997) that if the number of training samples is greater than 30 times the number of network weights, overfitting does not occur. In selecting the number of hidden nodes J, *Maier and Dandy* (2001) suggest taking the smaller of the values obtained using the upper limits for J given by *Hecht-Nielsen* (1987) and *Rogers and Dowla* (1994) in (3.18) and (3.19), respectively,

$$J \leq 2K + 1 \tag{3.18}$$

$$J \leq \frac{N}{K+1} \tag{3.19}$$

where N is the number of training samples and K is the number of inputs. However, while these guidelines may provide an upper limit, in many cases good performance can be obtained with fewer nodes. Therefore, the optimal geometry is generally not obtained using these guidelines.

The most commonly used method for selecting the number of hidden layer nodes is by trial-and-error (*Basheer and Hajmeer*, 2000; *Maier and Dandy*, 2000a), where a number of networks are trained, while the number of hidden nodes is systematically increased or decreased until the network with the best generalisability is found. The generalisability of an ANN can be estimated by evaluating its 'out-of-sample' performance on an independent test data set using some general measure of fit (e.g. RMSE, CE, MAE given in Section 3.2.1). In the statistical literature, this is known as cross-validation. However, cross-validation with an independent data set may not be practical if there are only limited available data, since the test data cannot be used for training. Furthermore, if the test data are not representative of the same population as the training data, the evaluation may be biased. Alternatively, information criteria which measure 'in-sample' fit (i.e. fit to the training data) but penalise model complexity, such as the AIC or BIC discussed in Section 3.2.1, can be used to estimate the generalisability of an ANN. As this method does not require the use of a test data set, all of the available data can be used for training; however, it has been suggested that the commonly used information criteria may overly penalise ANN complexity, leading to the selection of models that are too simplistic (*Qi and Zhang*, 2001).

Although the trial-and-error selection of hidden nodes is straightforward, it can be inefficient, as many networks may have to be trained before an acceptable one is found (*Reed*, 1993). In order to overcome the tedious manual search for the optimal number of hidden layer nodes, a number of pruning and construction algorithms, as well as evolutionary approaches, have been proposed to automate the selection procedure (*Reed*, 1993; *Yao*, 1999). Generally, pruning methods attempt to find an optimal network size by starting with a large network and systematically reducing it by eliminating weights or nodes that do not significantly contribute to the model fit. There are two main categories of pruning methods: those that prune weights and/or nodes according to some sensitivity measure (e.g. the change in model performance if a weight/node is removed from the network, or the partial derivative of the error function with respect to a given weight); and those that *effectively* prune weights by modifying the error function to include a term that penalises large weights (i.e. regularisation as discussed in Section 2.2.5.1) (*Reed*, 1993). Construction algorithms, on the other hand, begin with a minimal network and add new layers, nodes or connections as required during training. Small networks have a tendency to become trapped in local minima (see Section 3.2.6), therefore, the addition of new hidden nodes can change the shape of the weight space and help to escape the local minima (*Bebis and Georgiopoulos*, 1994). However, finding the optimal network geometry with these unit-by-unit evolution methods is not guaranteed and termination criteria used (e.g. all weights provide a 'significant' contribution to model fit) lack clear statistical meaning (*Vila et al.*, 1999).

Bowden (2003) used an evolutionary backpropagation MLP (EBMLP) to determine the most suitable network architecture, which was implemented using the commercially available software package, NeuroGenetic Optimizer (NGO) (*BioComp Systems Inc.*, 1998). An EBMLP combines a genetic algorithm with a feedforward MLP architecture, trained using the backpropagation algorithm (see Section 3.4.2.1), to find the optimal network architecture. The NGO software offers a number of features which enabled multiple hidden layers to be considered, an upper limit to be placed on the number of nodes in each hidden layer, a preference to be set for simpler models, linear, logistic or hyperbolic tangent activation functions to be utilised for each hidden and output node, the use of cross-validation during training, the selection of backpropagation parameters and the choice between two alternative methods for each of the three main GA operators: selection, crossover and mutation (see Section 3.4.2.2). Furthermore, the effect of different weight initialisations was taken into account during the evolutionary process, as the same architecture could be initialised numerous times throughout the algorithm with different initial weights.

The search space for an optimal ANN architecture is infinitely large, since the number of possible nodes and connections is unbounded (*Yao*, 1999). However, to define a more limited and reasonable search space, the NGO software allows upper limits of 8, 16, 32, 64, 128 or 256 to be placed on the number of nodes in each hidden layer. *Bowden* (2003) used (3.18) to determine the theoretical upper limit for the number of hidden nodes and then used this value to guide the choice of upper limit adopted using the NGO software. *Bowden* (2003) also allowed for two hidden layers to be considered to determine whether a second hidden layer was required. The same setting was used to define the upper limit for the number of nodes in each hidden layer. Additionally, the three available transfer functions were all considered for each of the hidden and output nodes, giving rise to a very large search space.

The GA selection method discards poor chromosome strings throughout the evolution; therefore, with each generation, the population needs to be refilled. The NGO software enables two alternative strategies to be used for refilling the population, including cloning the survivors of the selection process and randomly creating new population members. The former strategy has the advantage of faster convergence; however, premature convergence may result if the population is too small. By introducing new members into the population, the latter refill method avoids search stagnation. *Bowden* (2003) considered each of these refill strategies.

The results obtained by *Bowden* (2003) indicated that the EBMLP was sensitive to the type of refill strategy used during the GA optimisation, where it was apparent that cloning was the best method to refill the population. It was observed that while this refill strategy resulted in a decrease in the diversity of the networks obtained, it allowed for faster convergence to a near-optimal network configuration. The results also indicated that many networks with different configurations were capable of providing similar performance, and as a result, it was concluded that there can be no guarantee that the true optimal architecture was found.

3.2.5.2 Limitations and Conclusions

It is acknowledged that evolutionary approaches can be an effective way to conduct a directed search for an optimal solution within a large search space. However, in the case studies presented by *Bowden* (2003), it is considered that the search space was made much larger than necessary by allowing a great amount of flexibility in the architectures considered, when there are well established reasons to apply stricter constraints. For example, in the salinity forecasting case study presented by *Bowden* (2003), by placing an upper limit of 32 hidden nodes on each of two hidden layers allowed for and by enabling one of three possible activation functions for each hidden and output node, it was calculated that there were 1.3×10^8 possible network architectures. However, as discussed in Section 3.2.5, there are good reasons to use tanh hidden nodes and linear output nodes, and if these constraints were set, the search space would have been reduced considerably (1089 possible network configurations) without decreasing the possibility of finding the optimal architecture.

Additionally, the error surface of a small network is more complicated than that of a large network; thus, small networks are more susceptible to becoming trapped in local minima than larger ones (*Bebis and Georgiopoulos*, 1994). Therefore, there is the possibility of smaller networks being discarded in preference for larger ones using evolutionary architecture selection methods. This is particularly the case for the EBMLP used by *Bowden* (2003), as it is based on the local backpropagation training algorithm, which has limited capabilities for escaping local minima. The effect of different weight initialisations may be taken into account during the evolutionary process if the same architecture is initialised numerous times; however, if smaller networks are discarded in the early generations as a result of becoming trapped in local minima, there is little possibility that the same architecture will be initialised more than once, particularly using the cloning refill strategy. For example, using a trial-and-error architecture selection approach, a good place to start would be a one hidden layer network with one tanh hidden node and one linear output node. However, using the EBMLP, this configuration would initially have a 1 in 1.3×10^8 possibility of being considered. There is evidence in the results presented by *Bowden* (2003) that small networks may have been disregarded during the search, as the top 10 performing networks obtained using each of the population refill methods were reasonably large, containing between 11 and 27 nodes in the first hidden layer and between 7 and 16 nodes in the second hidden layer.

A further limitation of the EBMLP is that the results are dependent upon many user defined parameters. For example, parameters must be selected for the backpropagation training algorithm, the GA optimisation algorithm, as well as the overarching EBMLP parameters, such as constraints on the number of hidden layers and nodes and the types of activation functions enabled. While this provides much flexibility to the user, it may result in a suboptimal architecture being selected. It can be difficult to select the parameters of just one of these components to come up with an optimal solution; thus, the difficulty is increased when all three components are combined.

In this research, a trial-and-error architecture selection procedure will be used, beginning with a minimal network and systematically increasing the number of hidden nodes until it is considered that no significant improvement in model performance can be gained by the addition of further nodes. The networks considered will be limited to single hidden layer nets with tanh activation functions on the hidden layer nodes. If the target data have known upper and lower bounds, a bounded activation function, such as the logistic sigmoid or the hyperbolic tangent, will be considered for the output nodes; otherwise, linear output units will be used. It is acknowledged that a trial-and-error search for the optimum network may be tedious; however, by conducting a constrained and systematic search of the possible network architectures, it is considered that a near optimal network can be found in similar or less time than a less constrained evolutionary search. However, the problem remaining with trial-and-error architecture selection is that there is no widely accepted method for evaluating the generalisability of the trialled networks, as cross-validation with an independent data set reduces the available data for training and may give biased results if the test set is not properly representative. In addition, the use of performance criteria that penalise model complexity may overly penalise ANNs, leading to the selection of networks that are too simplistic. Therefore, in this research, further investigations will be conducted into how to best evaluate the generalisability of an ANN in order to select the optimal architecture.

3.2.6 ANN Training

3.2.6.1 Review of current practice

Training an ANN was discussed briefly in Section 2.2.3. It is the process by which the weights of an ANN are estimated, by using an iterative procedure to minimise a predetermined error, or objective, function, such as the SSE given by (3.1). Therefore, ANN training is essentially a nonlinear least squares problem, which can be solved using standard nonlinear least squares methods. However, like all complex nonlinear optimisation problems, training an ANN is generally not straightforward, as the error surface is typically a highly nonlinear function of the weights, complicated with many minima and saddlepoints (*Cheng and Titterington*, 1994; *Bishop*, 1995), as shown in Figure 3.5, in which points A and B are *local minima*, point C is the *global minimum* (i.e. the smallest value of the error function) and point D is a saddle point.

Local or global optimisation algorithms may be used to train an ANN. Local methods search for an optimum solution in a downhill direction from their initial position on the error surface. While such methods can be an effective way of optimising the weights of feedforward networks, they are susceptible to becoming trapped in local minima in the error surface. Therefore, as can be seen in Figure 3.5, the location of the initial weights

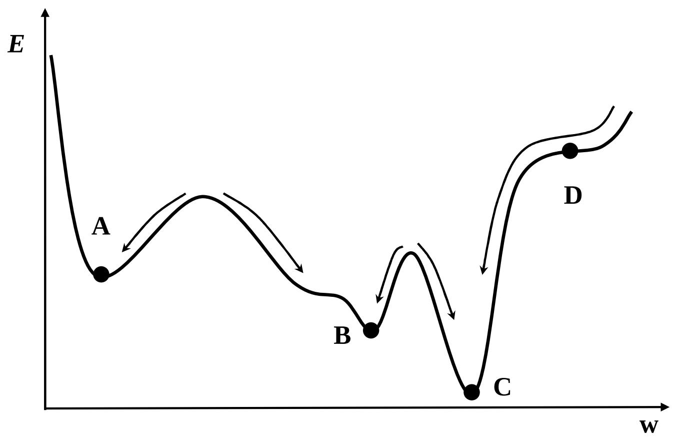

Figure 3.5 Examples of different local minima on the error surface.

can have a significant impact on the local minimum found. On the other hand, global methods employ directed random search techniques to allow the simultaneous search for an optimum in several directions. Such algorithms are therefore less sensitive to weight initialisations, as they have an increased ability to escape local minima. However, global methods are generally more computationally intensive than local algorithms. Therefore, the suitability of a particular training method is generally a compromise between computation cost and performance (*Maier and Dandy*, 2000a).

Using local optimisation methods, the general form of the weight updates is:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \gamma_t \mathbf{d}_t \tag{3.20}$$

where γ_t is the stepsize, $\mathbf{d}_t$ is the vector which defines the direction of descent and t is the iteration number (*Maier and Dandy*, 2000a). Either first-order or second-order local search methods are available, where the essential difference between the two is the determination of the vector $\mathbf{d}_t$, which determines the convergence rate and computational complexity (*Maier and Dandy*, 2000a).

First-order methods are based on a linear approximation of the error function about the current weight state $\mathbf{w}_t$. They use steepest, or gradient, descent to search the error surface for a minimum solution, where $\gamma_t \mathbf{d}_t$ is proportional to the local negative gradient of the error surface, calculated by taking the first partial derivatives of the error function with respect to $\mathbf{w}_t$. The *backpropagation* algorithm, also known as the *generalized delta rule* (*Rumelhart et al.*, 1986), is a first-order local search method that is by far the most widely used method to train feedforward MLPs (*Maier and Dandy*, 2000a). The main contribution of this algorithm is that it provides a computationally efficient method for evaluating the partial derivatives of the error function by propagating the computed error between the model output and the target data backwards through the network. The calculated derivatives are then used to update each of the weights in the network according to (3.21):

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla E_{\mathbf{w}_t} \tag{3.21}$$

where $\nabla E_{\mathbf{w}_t}$ denotes the gradient of the error function at point $\mathbf{w}_t$ and η is known as the *learning rate*. The learning rate η has an important impact on the ultimate performance of ANNs trained with backpropagation, as it directly affects the size of the steps taken in weight space (*Maier and Dandy*, 1998a). A choice of η that is too small can lead to very slow convergence of the algorithm and can increase the likelihood of becoming trapped in a local minimum. A larger learning rate will speed convergence; however, if η is too large, the algorithm may overshoot the optimum solution and fall into possibly

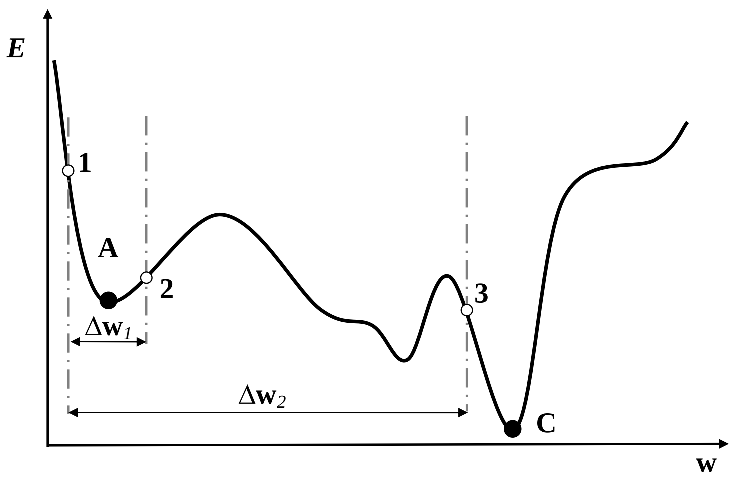

Figure 3.6 The effect of the learning rate used for backpropagation.

divergent oscillations. The effect of η is illustrated in Figure 3.6, where it can be seen that a small value of η could result in a small step from point 1 to point 2, from which point the algorithm would then continue to jump from one side of point A to the other, with the local minimum at point A being the best solution that would be found. Alternatively, with a larger value of η, a large step from point 1 could result in the set of weights at point 3, which is in the vicinity of the global minimum point C; however, the next step would likely overshoot C with the algorithm possibly falling into an oscillatory trap.

A number of alternatives have been proposed to enhance the backpropagation algorithm by reducing the influence of a fixed learning rate. The most popular of these includes a *momentum* term in the weight update formula as follows:

$$\Delta \mathbf{w}_t = -\eta \nabla E_{\mathbf{w}_t} + \phi \Delta \mathbf{w}_{t-1} \tag{3.22}$$

where ϕ is the momentum parameter. Adding momentum has the effect of adding inertia to the steps taken in weight space, which can either act to increase or decrease the size of the steps taken in weight space. In Figure 3.7, two situations are shown in which the addition of momentum helps the backpropagation algorithm converge on a better solution than it would if standard backpropagation was used. Figure 3.7 (a) shows an example of a situation where the standard backpropagation algorithm would oscillate between A and B; however, by using momentum to account for the inertia caused by the previous step in the opposite direction, the size of successive steps taken in weight space is gradually reduced, enabling the algorithm to find the minimum solution. Conversely, Figure 3.7 (b) shows a situation where the momentum term increases the size of the steps taken in weight space, enabling the algorithm to more efficiently transverse regions on the error surface with a small gradient. If the standard backpropagation algorithm was used in this case, the size of the steps taken would become successively smaller with the reducing gradient, with the algorithm possibly converging in the flat region between points A and B where the gradient is virtually zero. Therefore, as can be seen, the inclusion of momentum generally leads to a significant improvement in the backpropagation algorithm (*Bishop*, 1995). Other variations of first-order local search methods include the delta-bar-delta algorithm (*Jacobs*, 1988) and Rprop (*Riedmiller*, 1994). The main differences between the variety of local gradient based search methods include the information used to modify the step size, the parameters that are modified and whether the parameters are modified globally or individually for each node (*Maier and Dandy*, 2000a).

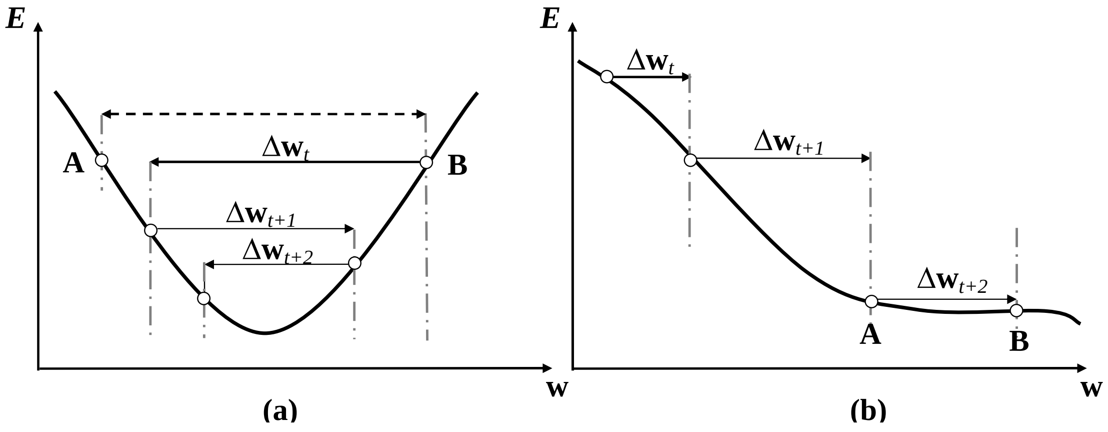

Figure 3.7 The impact of momentum in backpropagation.

While first-order optimisation algorithms are simple to implement, a problem with these methods is that, for most points in weight space, the local negative gradient does not point to the minimum of the error function. Therefore, an indirect route is taken towards the minimum, as illustrated in Figure 3.8, which results in slow convergence (*Bishop*, 1995). Second-order local optimisation methods attempt to overcome this by using second-order information about the error surface to take a more direct route towards the minimum. These methods are based on a local quadratic approximation of the error function about the current weight state $\mathbf{w}_t$, and use the local Hessian matrix $\mathbf{H}$ (matrix of second partial derivatives of the error function with respect to $\mathbf{w}_t$) to provide information about the curvature of the surface. The optimum weight update is therefore given by:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \mathbf{H}^{1}_{\mathbf{w}_t} \nabla E_{\mathbf{w}_t} \tag{3.23}$$

which, unlike first-order methods, results in a step directly towards the minimum (*Bishop*, 1995). The formula given by (3.23) is the classical Newton algorithm and the vector

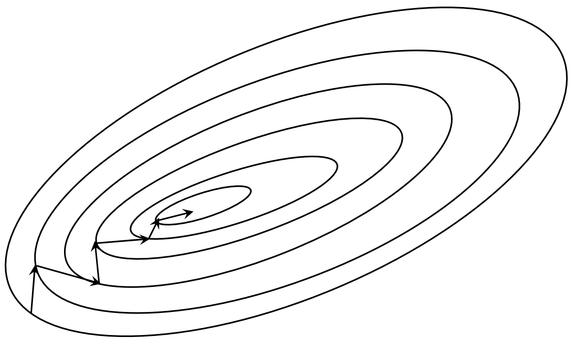

Figure 3.8 Indirect route taken towards the minimum of the error surface, where the ellipses represent contours of the surface.

$-\mathbf{H}^{1}_{\mathbf{w}_t}\nabla E_{\mathbf{w}_t}$ is known as the Newton direction. However, as the quadratic approximation of the error function is not exact, the Newton algorithm is iterative and as such, requires evaluation of the local Hessian matrix at each step. For all but the smallest networks, this can be extremely computationally demanding and places serious restrictions on the use of this method (*Maier and Dandy*, 2000a). Furthermore, the Hessian must be non-singular if it is to be inverted, which is not always the case, particularly if the network is large, as discussed in Section 2.2.5.3. Various modifications of the Newton algorithm are available that make use of different amounts of second-order information and thus have different computational demands. The QuickProp algorithm (*Fahlman*, 1989), conjugate-gradient methods, quasi-Newton methods and the Levenberg-Marquardt algorithm are examples of such algorithms listed in increasing order of use of second-order information. The amount of second-order information used can then be directly related to the size of the networks to which these algorithms may be applied (*Sarle*, 2002).

Because both first- and second-order local training algorithms tend to converge on the local minimum solution in the region of their starting point, finding the global minimum is dependent upon a fortuitous initialisation of the weights. To more consistently obtain a set of globally optimal weights, global optimisation methods, which have strategies to help them escape from local minima, can be used for ANN training (*Sexton et al.*, 1999a). The simplest global training method, known as *multistart*, involves the use of a local algorithm started from several points distributed over the whole weight space. A limitation of this method, however, is its lack of efficiency, as the same minimum solution can be determined several times, rather than thoroughly searching different solutions. Furthermore, the number of necessary starting points is generally unknown and problem dependent (*Sexton et al.*, 1998). Stochastic global search techniques, including evolutionary programming (EP) (*Fogel*, 1999), simulated annealing (*Kirkpatrick et al.*, 1983) and genetic algorithms (GAs) (*Goldberg*, 1989) provide alternative solutions to the optimisation problem. *Sexton et al.* (1998, 1999a,b); *Gupta and Sexton* (1999); *Sexton and Dorsey* (2000) and *Sexton and Gupta* (2000) carried out a number of comparisons between the backpropagation, genetic algorithm and simulated annealing algorithms for training ANNs, applied to both synthetic and real-world case studies. It was found that GAs are able to reliably and consistently outperform the backpropagation algorithm (*Sexton et al.*, 1998; *Gupta and Sexton*, 1999; *Sexton and Dorsey*, 2000; *Sexton and Gupta*, 2000). Additionally, simulated annealing was found to outperform backpropagation (*Sexton et al.*, 1999b); however, it was also observed that GAs are able to systematically obtain superior solutions to simulated annealing (*Sexton et al.*, 1999a).

Bowden (2003) compared six different training algorithms for estimating the weights of an ANN, including four first-order and two second-order local optimisation methods. The first-order methods included different variations of the delta rule, namely backpropagation (the generalised delta rule), the normalized cumulative delta (NCD) rule, the delta-bar-delta (DBD) algorithm, and the extended delta-bar-delta (EDBD) algorithm, while the second-order methods included QuickProp and a variation of this algorithm called MaxProp. To provide a fair comparison of the training algorithms, the ANNs were initialised with 30 different sets of weights for each training algorithm.

The results of the comparison carried out by *Bowden* (2003) showed that there was large variation between the performance of the models obtained using the different first-order training algorithms, as measured by the average error calculated using 30 different weight initialisations. The best results were obtained using the EDBD algorithm, followed by backpropagation, while the worst results were obtained using the NCD algorithm. It was also found that the generalisability of the ANNs developed using the second-order methods was inferior to that of the models developed using the first-order methods, which was consistent with the results obtained by *Maier and Dandy* (1999), who found that, depending on the size of the steps taken in weight space, first-order methods are able to escape local minima, whereas second-order methods are not. Overall, it was found that the results obtained using the QuickProp algorithm using the 30 different weight initialisations had the greatest variability, indicating that this algorithm was least able to converge on the same local minimum in the error surface.

3.2.6.2 Limitations and Conclusions

The investigation carried out by *Bowden* (2003) was limited by the ANN software used ((*NeuralWare*, 1991)), which did not enable the use of global training techniques. Although it has been shown that local optimisation algorithms, in particular backpropagation, can be used to successfully train an ANN (*Maier and Dandy*, 1999), results are often inconsistent due to the sensitivity of these algorithms to the initial weights. Therefore, it is difficult to place a reasonable degree of confidence in the solutions obtained using local search methods. While, there is currently no training algorithm that can guarantee the global solution of the network will be found in a reasonable amount of time (*Zhang et al.*, 1998), it has been demonstrated that global optimisation techniques, such as GAs, are able to more consistently converge on a near optimal solution. Therefore, in this research, the training performance of the most widely used backpropagation algorithm will be compared to two global optimisation techniques, namely a GA and the shuffled complex evolution (SCE-UA) algorithm developed by *Duan et al.* (1992). The SCE-UA algorithm has not yet been used to train ANNs, but has been found to be both effective and efficient in finding the global optimum in numerous other hydrological modelling studies (*Franchini et al.*, 1998; *Freedman et al.*, 1998; *Thyer et al.*, 1999).

3.2.7 ANN Validation

3.2.7.1 Review of current practice

An ANN may achieve almost perfect “in-sample” performance, which is evaluated according to the fit between the model outputs and the sample of data that it was trained on. However, before the model can be used to generate predictions or simulate data, it needs to be validated, which is usually done by evaluating its “out-of-sample” performance, or generalisability when applied to an independent set of validation data, using the performance criteria chosen (see Section 3.2.1) (*Maier and Dandy*, 2000a). To ensure appropriate validation of the developed ANN model, it is vital that the validation data were not used in any capacity during training and model selection.

Bowden (2003) did not consider ANN validation apart from selecting the performance criteria used to evaluate out-of-sample performance (see Section 3.2.1) and dividing the data to obtain a statistically representative validation data set (see Section 3.2.2).

3.2.7.2 Limitations and Conclusions

The main limitation of the standard method for validating ANNs is that no consideration is given to the physical plausibility of the estimated relationship. Rather, it is generally assumed that if an ANN model has good out-of-sample performance, it represents the physical process of the system (*Sudheer*, 2005). However, as discussed in Section 3.2.6, local minima may exist on the error surface, which may result in many combinations of weights having similar network performance. Knowing this fact does not provide great confidence in ANN predictions; however, if it can be demonstrated that a trained ANN has captured knowledge of the underlying system, the model can be applied with greater confidence. Therefore, in this research, validation of the ANNs developed will be carried out by evaluating out-of-sample performance on an independent set of validation data and by assessing the relationship modelled using the relative contributions of the model inputs in predicting the output. The relative contributions of the inputs as modelled by the ANN can then be compared to expert or *a priori* knowledge of the system, correlation or mutual information measures, or other data mining methods when there is little or no *a priori* knowledge of the system.

3.3 Summary of Approach Adopted and Further Investigations Required

Following the review of the current state-of-the-art ANN development process given in the preceding section, the methods adopted in this research for carrying out each step of the deterministic ANN development process are summarised below, together with any limitations of the current methodology and areas requiring further investigation:

Choice of data sets: The SOM data division approach proposed by *Bowden et al.* (2002); *Bowden* (2003) will be used to divide the available data into training, testing, and validation subsets. However, rather than allocating one data point from each cluster into each of the data subsets as suggested by *Bowden* (2003), the entire available data set will be divided into the respective subsets, with 64% of the data allocated to training, 16% allocated to testing and 20% allocated to validation. The average silhouette width $\bar{s}(k)$ will be used in conjuction with the discrepancy measure given by (3.11) to determine the appropriate size of the kohonen layer (i.e. SOM grid size) used to cluster the data.

Data pre-processing: In this research, data pre-processing will only be applied as necessary. Initially, all inputs and outputs will be standardised to have a mean of 0 and a standard deviation of 1, except if a bounded activation function is used on the output layer, in which case the target data will be scaled to be commensurate with the limits of this function. Once the model is fitted, diagnostic checking of the residuals will be carried out to determine whether the assumptions of the regression model, given in Section 3.2.3.2, have been met. If the assumptions are not met (e.g. model residuals are significantly non-Gaussian), nonlinear transformations including the logarithm, inverse and square root of the data will be considered in an attempt to improve the model.

Determination of ANN inputs: The stepwise PMI input selection approach proposed by *Bowden* (2003); *Bowden et al.* (2005a) will be adopted in this research. However, this method will first be applied to a number of synthetic data sets to verify that it is able to correctly select the important inputs.

Determination of ANN architecture: A trial-and-error procedure will be used to select the optimum number of hidden layer nodes for a given problem based on the best generalisability. The ANNs considered will be limited to single hidden layer networks with tanh hidden nodes and linear output nodes. As there is currently no widely accepted method for evaluating generalisability in order to select the optimum ANN size, this issue will be further investigated in this research.

ANN training: The optimisation performance of the backpropagation, GA and SCE-UA training algorithms will be compared in this research. The global optimisation method SCE-UA has not before been used to train an ANN, therefore, the investigation will aim to determine whether this algorithm is appropriate for training ANNs, and which algorithm is most suitable for training ANNs out of this new global method, the widely used local optimisation algorithm backpropagation, and the most commonly used global optimisation algorithm, the GA.

Choice of performance criteria and ANN validation: The commonly used performance criteria RMSE, MAE, r^2 and CE will all be used to evaluate the performance of the trained ANNs developed in this research. The AIC and BIC will also be used to evaluate the generalisability of the ANN models based only on the training data results, while taking into account the complexity of the models. To validate the physical plausibility of the models developed, the relationship modelled by the ANNs will be assessed by evaluating the relative contributions of the model inputs in predicting the output. However, as there is currently no widely accepted method for quantifying the relative importance of the ANN inputs, a number of input importance measures will be investigated in this research to determine which measure, if any, is most appropriate for assessing the relationship modelled by an ANN.

3.4 Further Investigation of Deterministic ANN Development Methods using Synthetic Data

3.4.1 Synthetic Data Sets

Three different synthetic data sets (data sets I, II and III) were used to compare and assess the methods investigated in the following sections. The use of synthetically generated data enabled the methods to be properly assessed and compared without the complication of other uncertainties, such as unknown important inputs or an unknown error model, and without being affected by data limitations. More importantly, if real data were used to assess the input importance measures investigated, as discussed in Section 3.4.4, it would not be possible to check the accuracy of the measures in relation to the true contributions. Furthermore, it was possible to generate "true" and "measured" target data, where the "true" data were generated by the model function without the addition of a random noise component and the "measured" data were obtained by corrupting the "true" data with random "measurement" errors, $\epsilon \sim \mathrm{N}(0,1)$.

The data sets were generated with different degrees of nonlinearity, noise levels and sizes, to represent the variability of cases that could be encountered in a real forecasting situation. Additionally, each data set represents a different type of forecasting problem, including time series forecasting, where inputs are past observations of the data series (i.e. $y_t = f(y_{t-1}, \dots, y_{t-K})$); causal forecasting, where inputs are independent predictor variables (i.e. $y = f(x_1, \dots, x_K)$); and in-between, where inputs include past values of the data series and independent variables (i.e. $y_t = f(y_{t-1}, \dots, y_{t-P}, x_1, \dots, x_{K-P})$). To quantify the degree of nonlinearity in the data sets, a multivariate linear regression model was fitted to the noise-free, or "true" data. The fit of the linear model to the data was evaluated using the coefficient of determination r^2, given by (3.5), where the closer r^2 is to one, the more linear the data are, and vice versa. To evaluate the noise levels in the data, the signal-to-noise ratio was evaluated as follows:

$$\lambda = \frac{\sigma^2_{signal}}{\sigma^2_{noise}} \tag{3.24}$$

where σ^2_{signal} is the variance of the "true" data and σ^2_{noise} is the variance of the added noise component ϵ.

The generated data were divided into training, testing and validation data subsets using the SOM data division method discussed in Section 3.2.2. In order to find the optimal grid size for each data set, the average silhouette widths and discrepancy values (see Section 3.2.2.2) were compared for SOM grid sizes ranging from 1×2 to 12×12. Once clustered, 64% of the data samples were allocated to the training subset, 16% were allocated to the testing subset and the remaining 20% were allocated to the validation subset, ensuring that at least one sample from each cluster was allocated to each subset where possible.

3.4.1.1 Data Set I

The autoregressive model of order nine (i.e. AR(9)), given by (3.25), was used to generate 870 data points to make up data set I. This model, also used by *Sharma* (2000) and *Bowden et al.* (2005a) for the generation of synthetic data, is a *linear time series* model, including only past observations of the data as inputs.

$$y_t = 0.3y_{t-1} - 0.6y_{t-4} - 0.5y_{t-9} + \epsilon \tag{3.25}$$

An r^2 value of 0.999 was obtained by fitting a linear regression model to the “true” data, indicating that the generated data are linear as expected. The signal-to-noise ratio of the “measured” data set was $\lambda = 1.92$, which indicates that the data are fairly noisy (i.e. the strength of the signal is less than twice the strength of the noise).

A histogram showing the probability density of the data is given in Figure 3.9, where it can be seen that the data are approximately normally distributed, indicating that only linear rescaling (standardisation) of the data was necessary. It was found that a 1×6 SOM grid size was optimal for clustering this data set, which resulted in 6 clusters all containing more than 3 samples. From these clusters, 557 samples were allocated to the training data subset, 139 were allocated to the testing subset and 174 were allocated to the

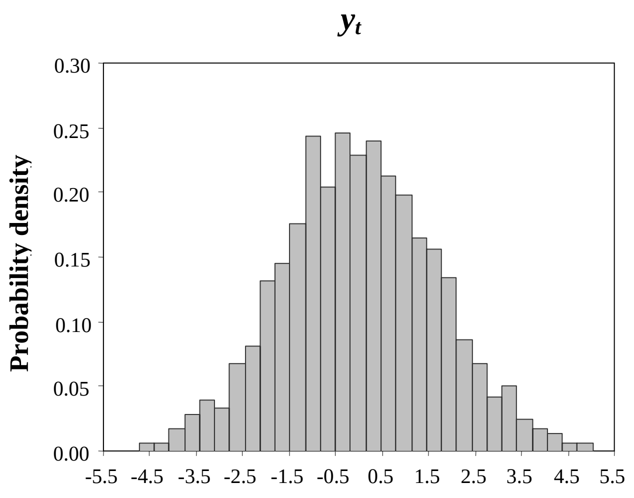

Figure 3.9 Probability density of response variable y_t for data set I.

validation subset.

3.4.1.2 Data Set II

Data set II, consisting of 1120 data points, was also generated by the AR(9) time series model, but with the linear addition of an independent nonlinear component $\sin(2x_t)$, as shown in (3.26), where x is an independent random variable uniformly distributed between $-\pi$ and π.

$$y_t = 0.3y_{t-1} - 0.6y_{t-4} - 0.5y_{t-9} + \sin(2x_t) + \epsilon \tag{3.26}$$

An r^2 value of 0.864 was obtained for this data set when a linear regression model was fitted to the "true" data, indicating that the generated data are reasonably linear with a nonlinear component that the linear model cannot account for. The signal-to-noise ratio of the "measured" data set was $\lambda = 3.39$, which indicates that the noise levels in the data are moderate (i.e. the strength of the signal is greater than three times the strength of the noise).

The probability density of the data is shown in Figure 3.10. Similar to data set I, the data are approximately normal; thus no further preprocessing was required apart from linear rescaling. For this data set, it was also found that a 1×6 SOM grid size was optimal for clustering the data and again, each of the 6 clusters contained greater than 3 data samples. From these clusters, 717 samples were allocated to the training subset, another 179 to the testing subset and the remaining 224 to the validation subset.

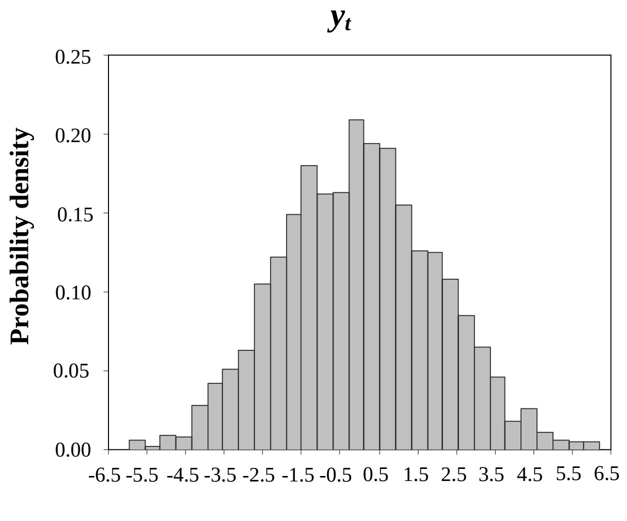

Figure 3.10 Probability density of response variable y_t for data set II.

3.4.1.3 Data Set III

1900 data points were generated by (3.27) to make up data set III. This function, suggested by *Friedman* (1991), describes a linear combination of both nonlinear and linear functions of independent random variables.

$$y = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + \epsilon \tag{3.27}$$

An r^2 value of 0.775 was obtained by fitting a linear regression model to the "true" data, indicating that data set III is the least linear of the data sets considered. The signal-to-noise ratio of the "measured" data set was $\lambda = 23.94$, which is significantly greater than the signal-to-noise ratios of data sets I and II. As the strength of the signal is almost 24 times the strength of the noise, the data are considered to contain little noise.

The probability density of the response variable y is shown in Figure 3.11. Again, the distribution of the data is approximately symmetric and did not require a nonlinear transformation. A 1×8 SOM grid size was found to be optimal for this data set, resulting in 8 clusters all containing greater than 3 data samples. From the clusters, 1215, 304 and 380 samples were allocated to the training, testing and validation subsets, respectively.

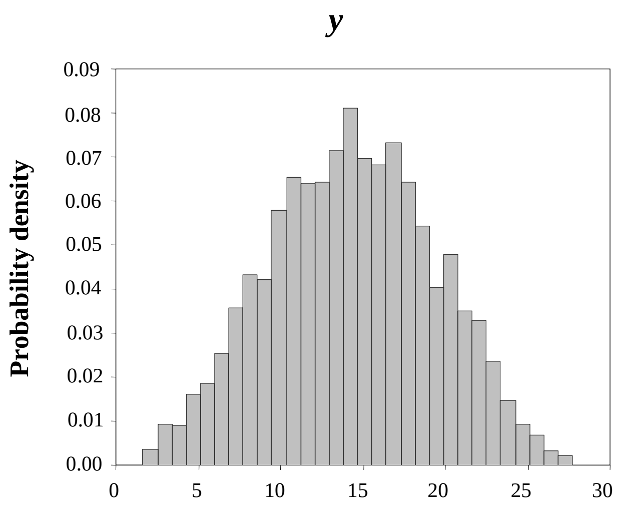

Figure 3.11 Probability density of response variable y for data set III.

3.4.2 ANN Training - Comparison of Training Algorithms

In this section, a description of each of the algorithms compared in terms of their training abilities is given, together with a description of the investigation undertaken. The back-propagation (BP) algorithm was included in the comparison as this is the most commonly used method to train MLPs, as discussed in Section 3.2.6, and provides a benchmark against which to evaluate other methods (*Dawson and Wilby*, 2001). Genetic algorithms (GAs) are a commonly explored alternative to using BP for training ANNs (*Sexton and Gupta*, 2000), and thus, were also included in the comparison. As mentioned in Section 3.2.6, GAs differ from traditional optimisation techniques by searching for an optimum from a population of points, rather than from a single point. They are also based on evaluations of the objective function, rather than its derivative or auxiliary information, and use probabilistic transition rules rather than deterministic rules (*Goldberg*, 1989). Although it has been found that GAs often provide better results when compared to gradient based methods such as backpropagation, they do not have the ability to fine-tune a solution, meaning that convergence can occur at a point where the gradient is not zero. It has been recognised that the efficiency of evolutionary algorithms, such as GAs, can be improved if they are combined with a local search method (*Yao*, 1999). This allows the evolutionary component of the algorithm to locate promising regions in the search space while the local search method is used to find the optima of these regions. The shuffled complex evolution - University of Arizona (SCE-UA) algorithm, developed by *Duan et al.* (1992, 1993), combines the strengths of a global evolutionary optimisation method with those of the local downhill simplex search method of *Nelder and Mead* (1965). The algorithm was designed primarily to deal with the "peculiarities encountered in conceptual watershed model calibration" and is based on the synthesis of four concepts, which are said to make the algorithm effective, robust, flexible and efficient (*Duan et al.*, 1994). These concepts are: (1) the combination of deterministic and probabilistic approaches; (2) systematic evolution of a 'complex' of points spanning the parameter space, in the direction of global improvement; (3) competitive evolution; and (4) complex shuffling. As stated in Section 3.2.6.2, the SCE-UA algorithm has not yet been used to train ANNs; therefore, it was included in the comparison to determine whether or not it is appropriate for ANN training.

3.4.2.1 Backpropagation (BP)

Shown in Figure 3.12 is a schematic of the BP algorithm. As can be seen, there are four main steps carried out during this algorithm, which are described as follows:

STEP 1: To initialise the algorithm, the weights are generally set to zero-mean random values. Choosing an appropriate size for the initial weights can have an important effect on training performance, as large weights may 'saturate' the nodes, causing the derivatives of the activation functions to be small and the error surface to be

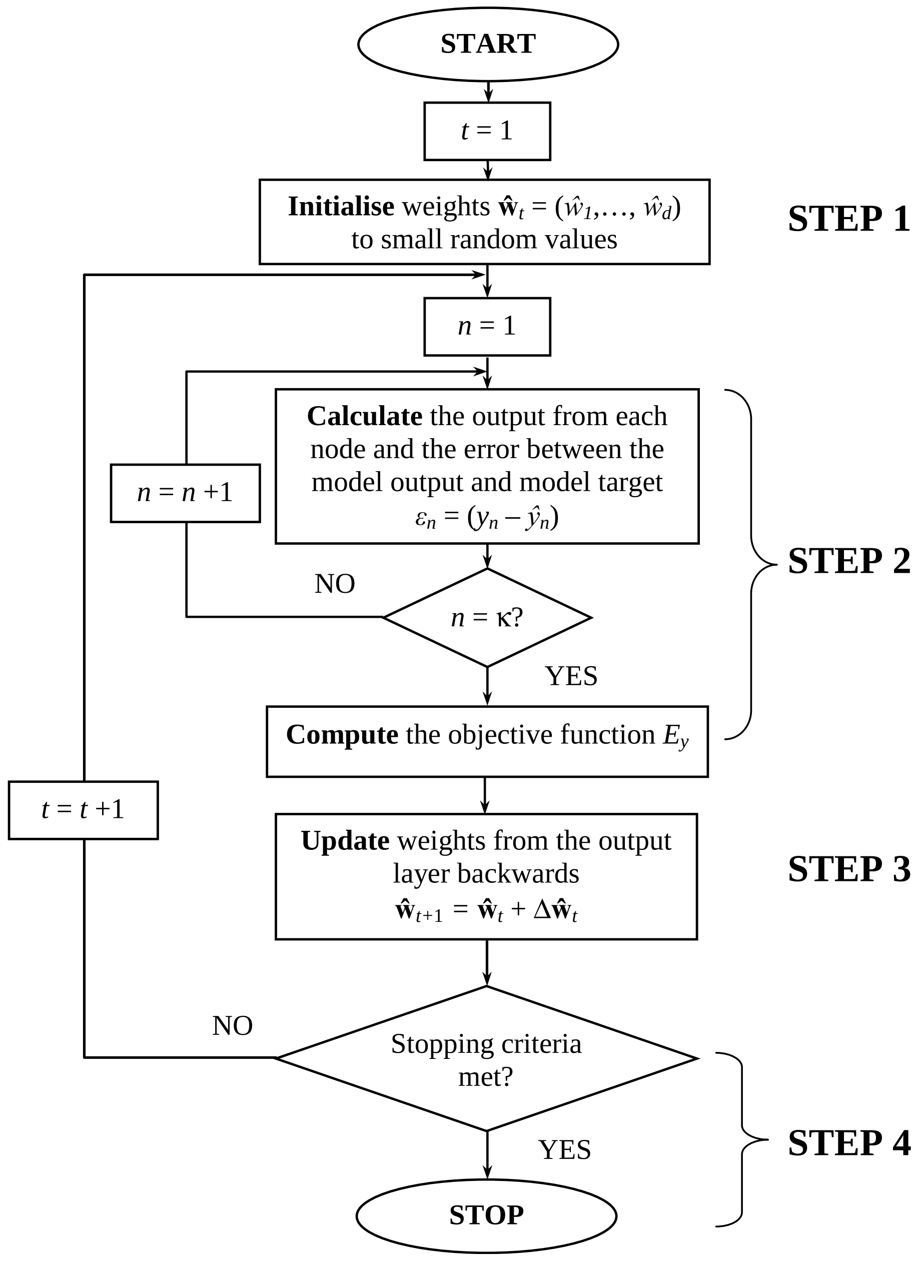
START
$t = 1$
Initialise weights $\hat{\mathbf{w}}_t = (\hat{w}_1, \ldots, \hat{w}_d)$ to small random values
STEP 1
$n = 1$
Calculate the output from each node and the error between the model output and model target $\varepsilon_n = (y_n - \hat{y}_n)$
$n = n + 1$
NO
$n = \kappa$?
YES
STEP 2
Compute the objective function E_y
$t = t + 1$
Update weights from the output layer backwards $\hat{\mathbf{w}}_{t+1} = \hat{\mathbf{w}}_t + \Delta\hat{\mathbf{w}}_t$
STEP 3
NO
Stopping criteria met?
YES
STEP 4
STOP

Figure 3.12 Schematic of the BP algorithm, outlining the main steps carried out

flat, and as a result, training is slow. On the other hand, if the initial weights are too small, the error propagated backwards to update the input-hidden layer weights will be small (see equations (3.30) and (3.31)); therefore, adaptation of these weights will be slow. It is therefore desirable that the summed inputs to sigmoidal activation functions be of order unity (*Bishop*, 1995). Thus, if the input variables are rescaled to an order of one, it is appropriate that the size of the weights is also of order unity.

STEP 2: This step involves the forward propagation of information through the network, where a defined number of training samples is presented to the network and the model outputs are evaluated. The number of samples presented to the network between weight updates is known as the *epoch size* and is denoted by κ in Figure 3.12. If the epoch size is equal to one (i.e. the weights are updated after each training pattern has been presented to the network), the weight updates are said to be *incremental*. If the epoch size is set equal to the size of the training set (i.e. the entire training set is processed in-between weight updates), the network is said to operate in *batch* mode. The epoch size can also be set to some intermediate number so that the network operates between incremental and batch modes. There are a number of advantages in presenting a number of training samples to the network before the weights are updated, as are there in updating the weights following the processing of each training sample. In batch mode, the weights are adapted based on the global error over the whole data set, rather than on the local minimum for the particular pattern being considered. On the other hand, updating the weights incrementally causes the search through weight space to become stochastic (i.e. computing the case-wise error function is equivalent to using an objective function that has been corrupted by noise (*Sarle*, 2002)), increasing the ability of the algorithm to escape from local minima in the error surface. *Maier and Dandy* (1998a) investigated the effect of the epoch size on training and found that, while the predictive ability of the network was unaffected by epoch size, training was much faster when a smaller epoch size was used. It was concluded that there was no advantage in using larger epoch sizes; therefore, in this research, incremental learning was used.

STEP 3: In this step, the model error is propagated backwards through the network in order to evaluate the partial derivatives of the error function and update the weights. Depending on whether a node is in the output layer or a hidden layer, the required weight update can be derived with different expressions. The first step is to calculate the weight updates for the hidden-output layer weights, which is done according to:

$$\Delta\hat{w}_{jm}(t) = \eta\delta_m z_j + \phi\Delta\hat{w}_{jm}(t-1) \tag{3.28}$$

where z_j is the the output from hidden node j. The value of δ_m is given by:

$$\delta_m = (y - \hat{y})\, g'_m(\hat{y}in) \tag{3.29}$$

where $\hat{y}in$ is the summed input to the output node (i.e. $\hat{y}in = \hat{w}_{0m} + \sum_{j=1}^{J}\hat{w}_{jm}z_j$). The weight updates for the input-hidden node weights are calculated as follows:

$$\Delta\hat{w}_{kj}(t) = \eta\delta_j x_k + \phi\Delta\hat{w}_{kj}(t-1) \tag{3.30}$$

where δ_j is given by:

$$\delta_j = g'_j(zin_j)\sum_{m=1}^{M}\hat{w}_{jm}\delta_m \tag{3.31}$$

and zin_j is the summed input to hidden node j (i.e. $zin_j = \hat{w}_{0j} + \sum_{k=1}^{K}\hat{w}_{kj}x_k$). Thus, the value of δ for a hidden node is calculated by propagating the δs backwards from nodes higher in the network. For an ANN with one hidden layer and a single output, the single value of δ_m is propagated backward to evaluate the δ values for each hidden node.

As discussed in Section 3.2.6, choosing an appropriate value for the learning rate η can be difficult, as small values result in slow convergence and increase the potential of becoming trapped in local minima, while large values can lead to oscillatory behaviour of the algorithm. Furthermore, if the error surface of an ANN contains many local optima, the optimal learning rate will change during the course of training. For incremental training, which was used in this research, the learning rate *must* be slowly reduced during training to guarantee convergence of the BP algorithm (*Sarle*, 2002). Therefore, an adaptive, or dynamic, learning rate was used in this research. To achieve faster learning with a small learning rate, a relatively large momentum term ϕ can be used; however, to ensure convergence of the BP algorithm, this value must be less than 1.0. The BP parameter values used in this research are further discussed in Section 3.4.2.4

STEP 4: Steps 2 and 3 are repeated for many iterations until a specified stopping criterion has been met. There are numerous criteria upon which training may be stopped, such as stopping after a fixed number of iterations; stopping when the training error falls below a specified value or when the relative change in the error falls below some tolerance value; or when the test set error begins to increase when applying cross-validation. The stopping criteria used in this research will be further discussed in Section 3.4.2.4.

3.4.2.2 Genetic Algorithms (GAs)

GAs are inspired by the Darwinian process of natural selection and survival of the fittest, where an initial random population is evolved over a number of generations by selectively sharing information among the best or 'fittest' solutions. The main steps in the GA training process, as outlined in Figure 3.13, are described as follows:

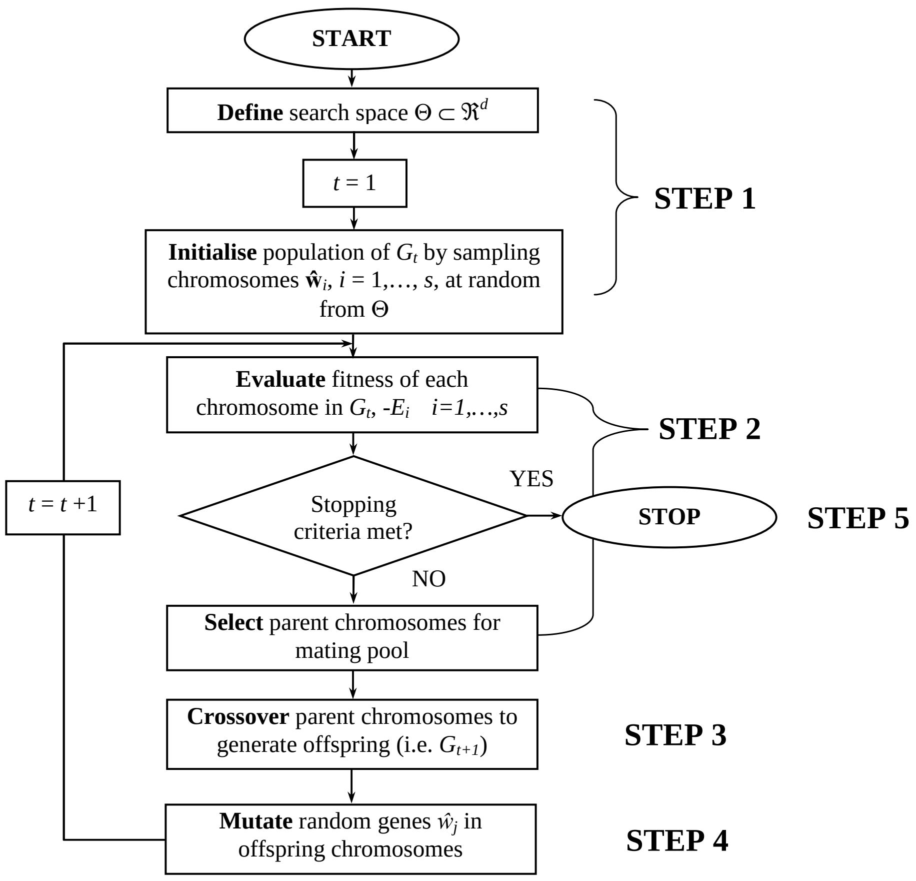
START
Define search space $\Theta \subset \Re^d$
$t = 1$
Initialise population of G_t by sampling chromosomes $\hat{\mathbf{w}}_i$, $i = 1,\ldots, s$, at random from Θ
STEP 1
Evaluate fitness of each chromosome in G_t, $-E_i$ $i=1,\ldots,s$
STEP 2
Stopping criteria met?
YES
NO
STOP
STEP 5
$t = t+1$
Select parent chromosomes for mating pool
Crossover parent chromosomes to generate offspring (i.e. G_{t+1})
STEP 3
Mutate random genes $\hat{w}_j$ in offspring chromosomes
STEP 4

Figure 3.13 Schematic of a GA outlining the main steps performed.

STEP 1: Using a GA, a possible solution to the optimisation problem is represented in the form of a string, called a 'chromosome'. Each chromosome is then made up of a number of elements, called 'genes', which contain encoded values of the variables being optimised. If real-valued encoding is used for the ANN training problem (as opposed to binary encoding, for example), each gene in a chromosome is one connection weight and, therefore, each chromosome represents a weight vector. To initiate the GA training process, the search space Θ needs to be defined and an initial population of chromosomes $G_1 = \{\hat{\mathbf{w}}_1, \ldots, \hat{\mathbf{w}}_s\}$ is generated randomly within this space, where the size of the population s *must* be an even number.

STEP 2: In the second step, the fitness of each individual chromosome is evaluated and a set of 'parent' chromosomes are selected. During the GA process, the aim is to evolve solutions with greater fitness (i.e. maximise fitness). Therefore, in this research, the fitness of each chromosome was calculated using the negative of the model error function (i.e. $\mathrm{fitness}_i = -E_i = -\mathrm{SSE}_i$); thus, the smaller the model error, the fitter the chromosome was considered to be. The parent chromosomes are chromosomes selected from the population that will contribute offspring to the next generation G_{t+1}. This operator, called *selection*, is analogous to the natural process of survival of the fittest, where the chromosomes compete, based on their fitness, to fill the population of parent chromosomes, called the *mating pool*. Different types of selection operator may be used to fill the mating pool. In this research, a selection operator known as 'tournament selection' was used, where pairs of chromosomes are randomly competed against one another and the winner (chromosome with the best fitness) is selected as a parent chromosome. The size of the mating pool needs to be the same as the initial population; therefore, each chromosome competes in a (random) tournament twice. As a result, fitter chromosomes may be included in the mating pool twice, whereas other less fit chromosomes may not be included at all.

STEP 3: Once the parent chromosomes have been selected, a genetic *crossover* operator is applied between pairs of parents to produce offspring, which form the next generation of chromosomes. The parents are paired by randomly selecting two chromosomes from the mating pool, without replacement. There are a number of different forms of crossover operator, all of which are designed to exchange or combine the information contained in the parent chromosomes. In this research, the two child staggered average crossover operator (*Vítkovský et al.*, 2000) was applied, as this operator has been designed to exploit the continuous nature of the weights, as opposed to alternative crossover operators that were designed for binary encoded chromosomes. One random crossover point was used, as shown in Figure 3.14, which illustrates the two child average crossover operator. Generally, when one-point crossover is applied, a crossover point is selected uniformly at random, and the portions of the parent chromosomes from the crossover point to the end are swapped, producing two new offspring. However, using the two child average crossover operator, the first offspring chromosome is produced by taking the average values of corresponding genes of a pair of parent chromosomes up until the crossover point, while the original genes are used from the crossover point to the end of the chromosome. Conversely, the second offspring chromosome is produced by using the original genes up until the crossover point, while taking the average of the parent genes from the crossover point to the end of the chromosome, as shown in Figure 3.14. The crossover operator is assigned a probability, or *crossover rate*, which determines whether or not crossover between a pair of parents will occur. Because crossover among parent chromosomes is a common natural process (*Caudill*, 1991), it is traditionally given a relatively high probability ranging from 0.6 to 1.0 (*Elbeltagi et al.*, 2005).

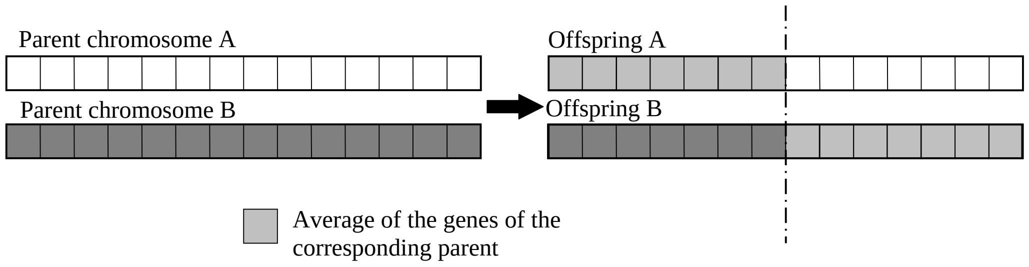

Figure 3.14 Two child average crossover operator.

STEP 4: *Mutation*, which is the occasional random alteration of the value of a gene (*Goldberg*, 1989), is the final step in the generation of offspring chromosomes. This operator ensures that the evolution does not become trapped in unpromising regions of the search space by introducing new information into the search. Similar to the selection and crossover operators, there are a number of alternative mutation operators available. The step size mutation operator (*Vítkovský et al.*, 2000) was used in this research, because, like the crossover operator used, it too was designed for continuous variables. This operator randomly mutates the value of a gene to within one *stepsize* of the current value as follows:

$$gene' = gene + \tau \times u \tag{3.32}$$

where $gene'$ is the mutated value of the gene, τ is the stepsize, or maximum incremental change, allowed for a gene and $u \sim U(-1, 1)$. The parameter τ should be selected according to the magnitude of the optimisation variables and the sensitivity of the objective function to these variables. A *mutation rate* is also assigned to the mutation operator, but unlike the crossover rate, the mutation rate is applied to a chromosome on a gene by gene basis. The bulk of a GA's processing power can be attributed to selection and crossover; therefore, mutation plays a secondary role in the algorithm (*Goldberg*, 1989). As mutation in nature is a rare process, the mutation rate is generally set to a small value (e.g. less than 0.1) (*Elbeltagi et al.*, 2005). However, according to *Eiben et al.* (1999), the use of rigid parameters that do not change their values is in contrast to the dynamic adaptive nature of a GA. They claim that different parameter values may be optimal at different stages of the evolutionary process, where large mutation steps can be useful in early generations for helping to explore the search space, whereas in later generations, small mutation steps might be needed to help to fine tune chromosomes. Therefore, in this research, a constant mutation rate was applied with a dynamic stepsize parameter. The parameter values used in this research are further discussed in Section 3.4.2.4.

STEP 5: Steps 2 to 4 are repeated for many generations and, like backpropagation, the final step in the algorithm involves determining when to stop training. The stopping criteria used in this research are discussed in detail in Section 3.4.2.4.

3.4.2.3 Shuffled Complex Evolution (SCE-UA)

The SCE-UA algorithm involves randomly selecting a population of points from the feasible search space, which are then divided into several communities, or *complexes*. The complexes are evolved independently, through a 'reproduction' process, where each member in a complex is a potential 'parent' with the ability to participate in the reproduction process. At periodic stages of the evolution, the entire population is shuffled before points are reassigned to complexes (i.e. the communities are mixed and new communities formed). This promotes the sharing of information gained by each community in order to direct the entire population toward the neighbourhood of a global optimum. A schematic of the SCE-UA algorithm is shown in Figure 3.15, outlining the main steps carried out during the algorithm. These are discussed further as follows:

STEP 1: To initialise the process, the feasible search space is defined by placing upper and lower limits on the weights, and a random sample of *points* $\hat{\mathbf{w}}_1, \ldots, \hat{\mathbf{w}}_s$ is generated within this space. The size of the sample s is equal to the number of complexes

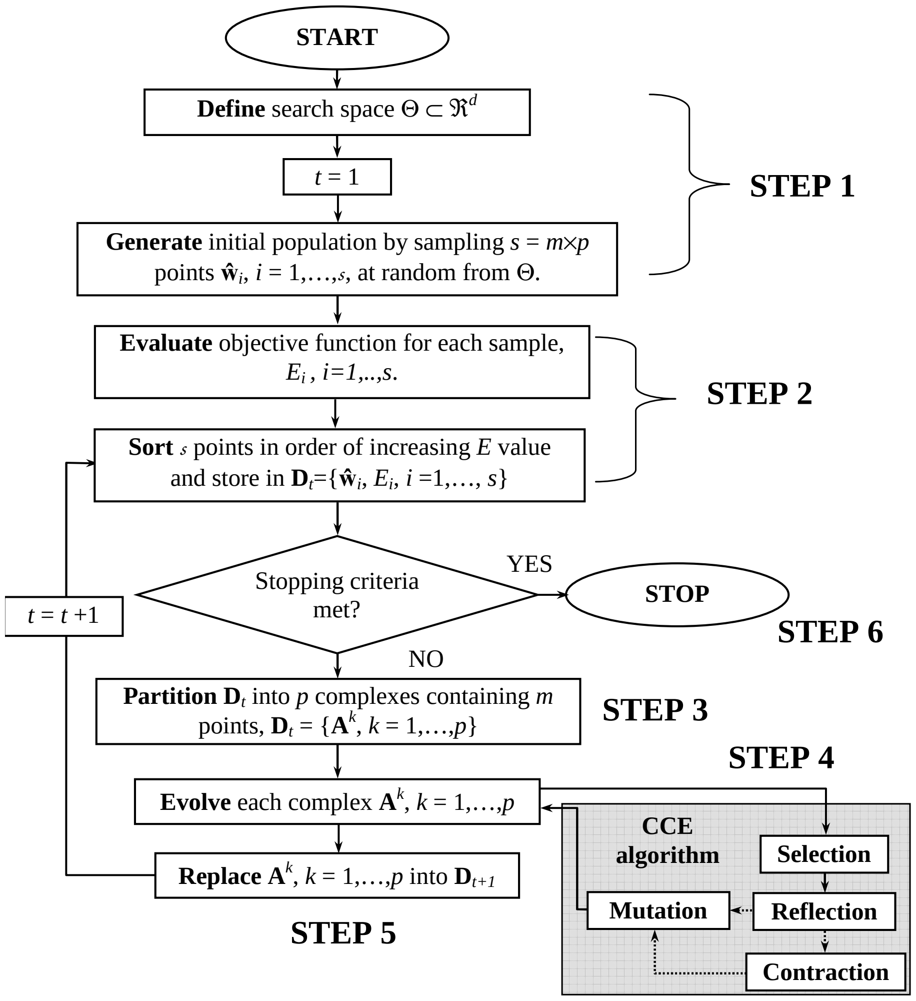
START
Define search space $\Theta \subset \Re^d$
t = 1
Generate initial population by sampling $s = m \times p$ points $\hat{\mathbf{w}}_i$, i = 1,…,s, at random from Θ.
STEP 1
Evaluate objective function for each sample, E_i, i=1,..,s.
Sort s points in order of increasing E value and store in $\mathbf{D}_t=\{\hat{\mathbf{w}}_i, E_i, i=1,\ldots,s\}$
STEP 2
Stopping criteria met?
YES
STOP
STEP 6
NO
$t = t + 1$
Partition $\mathbf{D}_t$ into p complexes containing m points, $\mathbf{D}_t = \{\mathbf{A}^k, k = 1,\ldots,p\}$
STEP 3
STEP 4
Evolve each complex $\mathbf{A}^k$, k = 1,…,p
CCE algorithm
Selection
Reflection
Mutation
Contraction
Replace $\mathbf{A}^k$, k = 1,…,p into $\mathbf{D}_{t+1}$
STEP 5

Figure 3.15 Schematic of SCE algorithm outlining the main steps carried out.

p, multiplied by the number of points in each complex m (i.e. $s = m \times p$).

STEP 2: The objective function E (e.g. the SSE given by (3.1)) is evaluated for each point. The s points are then sorted in order of increasing error function value and stored in the array $\mathbf{D} = \{\hat{\mathbf{w}}_i, E_i, i = 1, \ldots, s\}$, such that $i = 1$ represents the weight vector with the smallest error function value.

STEP 3: The array $\mathbf{D}$ is partitioned into p complexes $\mathbf{A}^1, \ldots, \mathbf{A}^p$, each containing m points, such that the first complex contains every $[p(j-1)+1]$ ranked point, the second complex contains every $[p(j-1)+2]$ ranked point, and so on, where $j = 1, \ldots, m$ (i.e. $\mathbf{A}^k = \left\{\hat{\mathbf{w}}_j^k, E_j^k | \hat{\mathbf{w}}_j^k = \hat{\mathbf{w}}_{k+p(j-1)}, E_j^k = E_{k+p(j-1)}, j = 1, \ldots, m\right\}$).

STEP 4: In this step, the complexes are evolved using the competitive complex evolution

(CCE) algorithm (*Duan et al.*, 1992). In this algorithm, a number of subcomplexes are selected from a complex, where a subcomplex acts as a pair of parents, although it may contain more than two members. A probability is assigned to the members of the complex such that better points have a greater chance of becoming parents, similar to the selection operator described for the GA. The downhill simplex method (*Nelder and Mead*, 1965) is then applied to each subcomplex to produce most of the offspring, where reflection and contraction processes are applied to direct the evolution in an improvement direction. Offspring are also occasionally randomly introduced to ensure that the evolution does not become trapped in an unpromising region. This is analogous to the mutation operator used in a GA. Each new offspring produced by a subcomplex then replaces the worst point in the subcomplex.

STEP 5: Once the complexes have been evolved, they are shuffled and reformed by combining all of the points in the evolved complexes into a single population, sorting the population in order of increasing objective function value and, finally, repartitioning the population into p complexes according to the procedure described in Step 3.

STEP 6: Steps 2 to 5 are repeated until a stopping criterion has been met and this step involves checking whether or not this has occurred. The stopping criteria used in this research are discussed in detail in Section 3.4.2.4.

3.4.2.4 Investigation

The optimal number of hidden nodes necessary for modelling data sets I, II and III was initially assumed to be unknown; therefore, each algorithm was used to train 10 different sized networks, containing between 1[1] and 10 hidden nodes, for each of the three synthetic data sets. It was considered that, within this range, there would be networks containing too few, too many and the optimum number of hidden nodes for the given case studies. This enabled the optimisation abilities of the training algorithms to be assessed under such different conditions of model specification. To investigate the robustness of the algorithms, each network was initialised with five different sets of weights. This resulted in 50 networks being developed for each data set, using each training algorithm, or a total of 150 ANN models for each data set. To initialise each of the algorithms, the initial weights were randomly generated from a normal distribution with zero mean and unit

[1]This was the minimum number of hidden nodes considered in this research, as ANNs with no hidden nodes result in linear models; thus defeating the purpose of using an ANN, rather than a linear regression model.

variance to achieve summed inputs to the hidden nodes of the order unity. Both the GA and the SCE-UA algorithm required that upper and lower bounds be placed on the weights in order to define the feasible search space. For each algorithm, these bounds were set to [-10,10] in order to easily accommodate the initial weight values and to not be overly restrictive or flexible for the algorithms to function properly.

Each of the algorithms required a number of user-defined parameters to be set. For the BP algorithm, these included the initial and final values of the dynamic learning rate, η_0 and η_F, and the momentum rate ϕ. The GA required that a population size s, a crossover rate ρ_{cross}, a mutation rate ρ_{mut}, and initial and final values of the dynamic stepsize, τ_0 and τ_F be specified. The SCE-UA algorithm has a number of parameters that require specification, including the number of complexes p; the number of points in a complex m; the number of points in a subcomplex q; the number of consecutive offspring generated by each subcomplex α; and the number of evolution steps taken by each complex β. However, *Duan et al.* (1993, 1994) provide default values for all of these parameters, except for p, which is highly dependent upon the complexity of the problem. These default values, given in Table 3.1, were used in this research; thus, only the parameter p required specification.

Table 3.1 Default parameter values for SCE-UA algorithm

Parameter	Default value
m	$2d+1$
q	$d+1$
α	1
β	$2d+1$

To set the user-defined parameters used in the investigation, each algorithm was used to train three networks containing 2, 6 and 10 hidden nodes, when applied to data set II, which was considered to have intermediate nonlinearity and noise properties of the data sets considered. The different sized networks were used to ensure parameters were selected that were suitable for the range of ANN sizes considered. Parameter values were varied between specified ranges in order to find the best configuration for each training technique. The ranges investigated were specified according to values typically used for these algorithms, apart from the GA mutation rate, for which larger values than typical were included, since the form of the mutation operator used is less random than other types of mutation. The parameters that resulted in the most effective and efficient training runs were selected and used in a further comparison of the algorithms. The values adopted

Table 3.2 User-defined parameters adopted for BP, GA and SCE-UA training algorithms

Parameter	Value Adopted	Range Investigated
BP		
Initial learning rate η_0	0.005	0.001, 0.005, 0.01, 0.05, 0.1, 0.2
Final learning rate η_F	0.0001	0.0001, 0.0005, 0.001
Momentum rate ϕ	0.6	$0.5, 0.6, \ldots, 0.9$
GA		
Population size s	20	10, 20, 50, 100, 500
Probability of crossover ρ_{cross}	0.7	$0.5, 0.6, \ldots, 0.9$
Probability of mutation ρ_{mut}	0.2	0.001, 0.005, 0.1, 0.2, 0.3, 0.4
Initial stepsize τ_0	0.2	0.1, 0.2, 0.5, 1
Final stepsize τ_F	0.001	0.0001, 0.001, 0.002, 0.005
SCE-UA		
No. of complexes p	d	$0.5d$, d, $2d$

for the user-defined parameters of each training algorithm are shown in Table 3.2, together with the ranges investigated.

The stopping criteria used in the investigation were set to achieve convergence, or near convergence, of the algorithms to within a specified tolerance value. Initially, the stopping criterion used for each algorithm was that given by *Thyer et al.* (1999) for the SCE-UA algorithm:

$$\frac{\left|SSE(\hat{\mathbf{w}})_{t+1}^{L} - SSE(\hat{\mathbf{w}})_{t}^{L}\right|}{\left|SSE(\hat{\mathbf{w}})_{t}^{L}\right|} < \text{tol} \tag{3.33}$$

where the superscript L denotes the lowest SSE value and "tol" was set equal to 1×10^{-6}. In this research, the algorithms were stopped after meeting this criterion for 20 consecutive iterations. However, in the preliminary runs used to select the user-defined parameters, it was found that this criterion alone resulted in prohibitively long training times when the SCE-UA algorithm was applied to the 10-hidden node network. Therefore, an additional stopping criterion was included, where training was stopped after 10 million error function evaluations using the GA and SCE-UA algorithm, and after 10 million forward passes of the entire training set using BP, regardless of whether (3.33) had been met. Thus, an algorithm stopped according to this criterion would not necessarily have converged to a stationary point of the error surface.

The stopping criterion given by (3.33) was also used to determine when to reduce the dynamic learning and stepsize parameters used in the BP algorithm and GA, respectively.

For these algorithms, (3.33) was not used as a stopping criterion until the final values of the dynamic parameters were reached. Instead, using the BP algorithm, the learning rate η was reduced by 10% each time the criterion was met, whereas, with the GA, the stepsize τ was reduced by 10% when the criterion was met for 10 consecutive generations. When and by how much to reduce these values was determined heuristically when selecting the user-defined parameters for the algorithms.

Finally, as the models developed in this investigation were also used in further investigations on architecture selection and model validation, cross-validation with a test data set was employed during training and the weights resulting in the minimum test set error (i.e. the point at which overtraining is considered to begin) were saved, and the corresponding model outputs computed so that they could be used in further investigations.

3.4.3 Determination of ANN Architecture - Assessment of Model Selection Criteria

The generalisation abilities of the models developed using the best weight initialisations, as described in the previous section, were evaluated to investigate how to best select the optimal number of hidden nodes in a network for a given problem. Generalisability was measured by computing the *out-of-sample* performance on the test data set when training was stopped early (i.e. when the saved weights resulting in the minimum test set error were used to compute the model outputs) using the RMSE, MAE, r^2, CE, AIC and BIC given by equations (3.2), (3.3), (3.5), (3.6), (3.7) and (3.8), respectively. It was also measured by computing the *in-sample* AIC and BIC values for the training data when training was allowed to run until convergence (or until the maximum number of function evaluations was reached).

For data set I, it was known that an ANN containing no hidden layer nodes was the optimal structure, as the data were generated by a linear function. Therefore, the “optimal” network sizes, as indicated by the various generalisability measures, could be evaluated against this knowledge. For data sets II and III, however, the optimal network structure was unknown. Nevertheless, as the data were synthetically generated, the level of noise added to the training data was known. Therefore, the MAE and variance of the models’ residuals obtained when the networks were trained to convergence could be compared to the MAE and variance computed for the actual training data noise, to determine what size network was necessary for an appropriate mapping of the data.

3.4.4 ANN Validation - Assessment of Input Importance Measures

A number of methods have been proposed in the literature for determining the strengths of the input variable contributions in predicting ANN outputs based on the optimised network weights. As the weights of a trained ANN control the interactions that occur between the model inputs and the output, the importance of each input can be determined by the strength and direction of the connection weights between them. For example, an input will have a positive impact on the output, through a given hidden node, if the input-hidden and hidden-output weights are of the same sign (i.e. both positive or both negative), whereas an input will have an inhibitory effect on the output through the hidden node if the signs of the input-hidden and hidden-output weights are opposite. While no method can perfectly summarise the information contained in the weights without considering the actual input-output function computed by the ANN, the question still remains as to which of the measures available for estimating input importance, if any, provides the best interpretation of the modelled function. To address this question, *Sarle* (2002); *Gevrey et al.* (2003) and *Olden et al.* (2004) compared a number of available methods for quantifying the importance of ANN inputs. However, while each of these comparisons provides a good reference, it is considered that limitations in the investigations prevent the selection of an overall 'best' measure for reasons discussed in Section 3.4.4.5.

The Connection Weight Approach (*Olden et al.*, 2004) and Garson's measure of relative importance (*Garson*, 1991), which were two of the most promising methods for assessing ANN input contributions identified in the comparisons conducted by *Olden et al.* (2004) and *Sarle* (2002), were further investigated in this research for their ability to accurately quantify ANN input importance. Additionally, given the known limitations of these methods (discussed in Sections 3.4.4.1 and 3.4.4.2), two new methods, based on modifications of the existing approaches, were also investigated and compared to the existing methods. Details of the methods compared are discussed in the following sections. It should be noted that when using any of the methods presented to quantify input contributions, standardising the input variables is extremely important in order to remove the effects of measurement scale and ensure that the importance of each input variable is reflected in its variability relative to the other inputs.

3.4.4.1 The Connection Weight Approach

The Connection Weight Approach was found to provide the best overall methodology for quantifying ANN input importance in the comparison conducted by *Olden et al.* (2004). This method is based on the sum of the products of input-hidden and hidden-output connection weights, or 'overall connection weight' (*Olden and Jackson*, 2002). With reference to Figure 3.16, the overall connection weight (OCW) of input $\mathbf{x}_k$ can be calculated by:

$$OCW_{\mathbf{x}_k} = \sum_{j=1}^{J} w_{I_k,H_j} \times w_{H_j,O} \tag{3.34}$$

While the OCW values themselves are rather meaningless as a model validation measure, they may be used to determine the relative importance (RI), or relative contribution, of each input in predicting the output as follows:

$$RI_{\mathbf{x}_k} = \frac{OCW_{I_k}}{\sum_{i=1}^{K} |OCW_{I_i}|} \times 100\% \tag{3.35}$$

The RI values can then be compared to any *a priori* knowledge of the data-generating relationship, or statistical mutual information measures, to assess how well the model has explained the true interactions that take place between model inputs and outputs.

The main limitation of the Connection Weight Approach is that it does not account for the "squashing" effect of hidden layer activation functions (*Sarle*, 2002). The amount of squashing increases with the magnitude of the summed input to a hidden node zin_j, as illustrated in Figure 3.17. Thus, for large values of zin_j, the computed relative importance measures are unlikely to accurately describe the modelled input-output relationships. However, as the summed input to a node depends on all of the input, weight and bias values feeding into that node, accounting for these complexities would require the use of the actual input-output function computed by the ANN (*Sarle*, 2002).

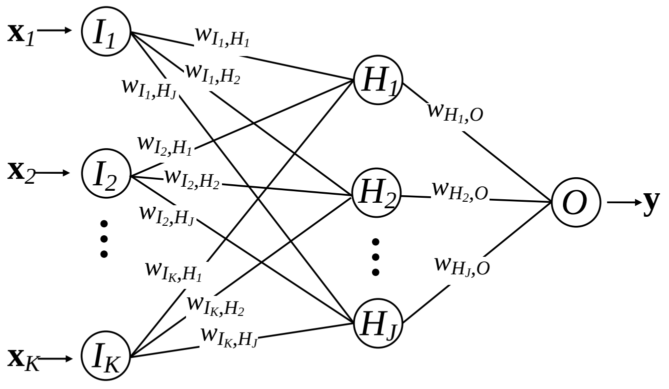

Figure 3.16 Example ANN.

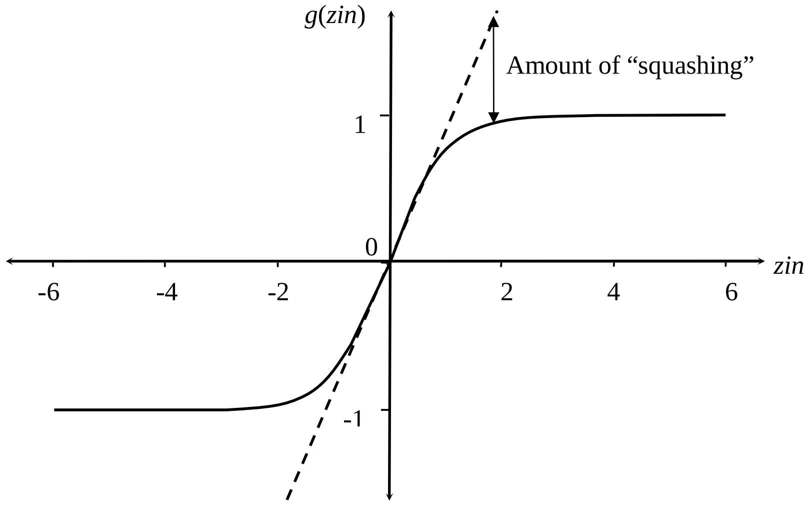

Figure 3.17 The squashing effect of nonlinear activation functions.

3.4.4.2 Garson's Measure of Relative Importance

Garson's measure of relative importance (*Garson*, 1991) was one of the earliest methods proposed for quantifying the relative contributions of ANN inputs and has been used in a number of studies, particularly in the field of ecological modelling, for extracting information from trained ANNs (*Brosse et al.*, 1999; *Aurelle et al.*, 1999; *Gozlan et al.*, 1999). This measure is calculated by partitioning the hidden-output layer connection weights into components associated with each input node, or, in other words, partitioning the sum of effects on the output layer into input node shares. With reference to Figure 3.16, the RI of input $\mathbf{x}_k$ can be calculated by:

$$\begin{aligned} RI_{\mathbf{x}_k} &= \frac{\sum_{j=1}^{J}\left(\frac{|w_{I_k,H_j}|}{\sum_{k=1}^{K}|w_{I_k,H_j}|}\times|w_{H_j,O}|\right)}{\sum_{i=1}^{K}\left[\sum_{j=1}^{J}\left(\frac{|w_{I_i,H_j}|}{\sum_{k=1}^{K}|w_{I_k,H_j}|}\times|w_{H_j,O}|\right)\right]}\times 100\% \\ &= \sum_{j=1}^{J}\left[\frac{|w_{I_k,H_j}|}{\sum_{k=1}^{K}|w_{I_k,H_j}|}\times\frac{|w_{H_j,O}|}{\sum_{i=1}^{J}|w_{H_i,O}|}\right]\times 100\% \end{aligned} \tag{3.36}$$

As can be seen from this equation, Garson's measure is the sum of products of normalised weights. By normalising the weights, the effect of squashing is accounted for to some extent, as the excessive influence of large weights is diminished (*Sarle*, 2002).

The main limitation of Garson's measure is that, because it uses absolute values of the weights, the signs of the input contributions are not taken into account, which can result in misleading RI values. For example, if an input has a positive impact on the output through one hidden node and an inhibitory effect on the output through another hidden node, the overall impact of the input should be somewhere in between (i.e. the overall contribution of an input is diminished if it has counteracting impacts through individual hidden nodes). However, as Garson's measure only accounts for the magnitude of the impacts through different hidden nodes, and not the direction, counteracting impacts are added together to strengthen the overall contribution.

3.4.4.3 Modification of the Connection Weight Approach

As the original Connection Weight Approach is limited by not accounting for the effects of squashing, the modified Connection Weight Approach presented in this thesis involves the use of a modified OCW measure that does account for squashing to some extent by using the hidden layer activation functions to "squash" the input-hidden node weights as follows:

$$OCW_{\mathbf{x}_k} = \sum_{j=1}^{J} g\left(w_{I_k,H_j}\right) \times w_{H_j,O} \tag{3.37}$$

where $g(\cdot)$ is the activation function used on the hidden layer nodes. If the input data are standardised, large weights feeding into the hidden nodes would be the primary cause, overall, for large summed inputs into the nodes, and hence, significant amounts of squashing. Therefore, by squashing the input-hidden node weights using the hidden layer activation functions, the influence of excessively large weights is removed. While it is acknowledged that this will still not result in an accurate representation of the modelled function, as the size of the input-hidden node weights are not considered in relation to the other weights feeding into the same hidden node, it should result in an improved representation of the relative contributions of the various ANN inputs.

3.4.4.4 Modification of Garson's Measure

A modified version of Garson's measure is also introduced in this research. As the main limitation of the original Garson's method is due to the use of absolute values of the weights, the proposed method calculates the relative importance of an input using normalised values of the raw weights as follows:

$$RI_{\mathbf{x}_k} = \sum_{j=1}^{J} \left[\frac{w_{I_k,H_j}}{\sum_{k=1}^{K} \left|w_{I_k,H_j}\right|} \times \frac{w_{H_j,O}}{\sum_{i=1}^{J} \left|w_{H_i,O}\right|} \right] \times 100\% \tag{3.38}$$

A limitation of this method, like all of the other methods discussed above, is that it does not take into account the effect of the bias weights. A large bias feeding into a hidden node may distort the values of the normalised input-hidden node weights, resulting in misleading RI values.

3.4.4.5 Investigation

In order to appropriately compare the measures discussed above in their ability to quantify ANN input contributions, the limitations of the comparisons conducted by *Sarle* (2002), *Gevrey et al.* (2003) and *Olden et al.* (2004) were first identified, such that the same inadequacies could be avoided in this research. The comparison carried out by *Sarle* (2002) was based on a simple additive model given by:

$$\begin{aligned} y \;=\; & 0.1\tanh(100x_1 + 0.25) - 0.1\tanh(100x_1 - 0.25) - 0.1 \\ & +0.1\tanh(100x_2 + 1.5) - 0.1\tanh(100x_2 - 1.5) \\ & +\tanh(x_3) \end{aligned} \tag{3.39}$$

Rather than training an ANN to fit this function, a 5 hidden node MLP with the weights specified in Table 3.3 was used by *Sarle* (2002). Various input importance measures were then compared in their ability to estimate the correct RI values corresponding to the three inputs x_1, x_2 and x_3. However, it can be seen that each hidden node H_j is only a function of one input plus a bias term, which is unrealistic of the way relationships are modelled by ANNs. Such a comparison of input importance measures is incomplete, as these measures may be sensitive to the effects of several inputs feeding into one hidden node; yet, this would not have been properly investigated. Therefore, it is considered that the results of the comparison conducted by *Sarle* (2002) do not fairly represent the relative performances of different input importance measures under normal circumstances.

The comparison conducted by *Gevrey et al.* (2003) was based on empirical ecological data, where a 5 hidden node ANN was initialised and trained by BP to fit the data 10 times. The mean input contributions calculated by each of the input importance measures

Table 3.3 ANN weights specified to describe the function given by (3.39) in the comparison of input importance measures carried out by *Sarle* (2002).

considered were assessed, together with the standard errors of the measures to evaluate their stability. The major shortcoming of this comparison was that the true input contributions in predicting the output were unknown, meaning that the accuracy of the different measures investigated could not be established by comparison with true values. Furthermore, as noted by *Olden et al.* (2004), using the standard errors of the input importance measures calculated from the 10 ANNs developed does not indicate the stability of the input importance measures; rather, it assesses the differences among variable contributions arising solely from different initial connection weights, which is an issue of optimisation stability.

To overcome the weaknesses of the comparison conducted by *Gevrey et al.* (2003), *Olden et al.* (2004) used simulated data with known input-output relationships in their comparison of ANN input importance measures. However, the major limitation of this study was that the synthetic data were generated by a linear function. To model a linear function with an ANN, the weights and biases feeding into a sigmoidal hidden node are generally very small, such that the summed input to the node lies on the linear part of the sigmoidal curve near the origin (*Bishop*, 1995). Therefore, the effect of squashing on the input importance measures would not have been properly investigated in the study by *Olden et al.* (2004), as the summed inputs to the hidden nodes would not have been large enough to experience any significant squashing effects.

The limitations of previous studies conducted to compare the accuracy of various input importance measures, summarised in Table 3.4, were considered when designing the investigation conducted in this research. The methods described in Sections 3.4.4.1 to 3.4.4.4 were assessed when applied to ANNs of different sizes, trained by the BP, GA and SCE-UA algorithms, to fit data sets I, II and III, which exhibit varying degrees of nonlinearity and complexity. The use of different network sizes and training algorithms was important to assess the accuracy and precision (degree of variation in accuracy) of

Table 3.4 Limitations of previous studies conducted to compare input importance measures.

Comparison	Limitations
Sarle (2002)	Unrealistic ANN weights, only considers a single input feeding into each hidden node.
Gevrey et al. (2003)	Empirical data used, not possible to validate the accuracy of the input importance measures. Evaluation of stability reflects optimisation stability rather than stability of the methods.
Olden et al. (2004)	Linear data used, performance of input importance measures when "squashing" by the hidden layer occurs was not considered.

the input importance measures when information was distributed through the ANNs in different ways, and not simply when the optimal network configuration was used, and to assess their sensitivity to the optimised weights obtained by different training methods.

While the use of synthetic data meant that the characteristics of each data set were known, the exact relative input contributions could only be determined directly (from the function coefficients) for data set I. If $M(\mathbf{y}, \mathbf{X})$ is a measure that quantifies the importance of K input variables $(\mathbf{x}_1, \ldots, \mathbf{x}_K)$ jointly for the dependent variable $\mathbf{y}$, then the relative importance of an explanatory variable $\mathbf{x}_k$ is defined by its contribution to the joint importance measure as follows:

$$M(\mathbf{y}, \mathbf{X}) = \sum_{k=1}^{K} M(\mathbf{y}, \mathbf{x}_k) \tag{3.40}$$

where $M(\mathbf{y}, \mathbf{x}_k)$ is a partial importance measure for input $\mathbf{x}_k$ (*Soofi et al.*, 2000). Therefore, to quantify the relative input contributions for data sets II and III, the stepwise PMI input selection procedure (see Section 3.2.4) was applied to the data sets, as the PMI criterion calculated using this method provides a model-free measure of either linear or nonlinear partial dependence between an independent variable and a dependent variable. The PMI scores for each input, corresponding to when the input was selected as most important, conditional on the existing predictors (i.e. when each input had the highest PMI score, as determined in step 3 of the stepwise PMI procedure described in Section 3.2.4), were then used to estimate RI values for the inputs according to:

$$RI_{\mathbf{x}_k} = \frac{M(\mathbf{y}, \mathbf{x}_k)}{M(\mathbf{y}, \mathbf{X})} \times 100\% \tag{3.41}$$

where $M(\mathbf{y}, \mathbf{x}_k) = \mathrm{PMI}_{\mathbf{x}_k}$ and $M(\mathbf{y}, \mathbf{x}_k)$ is calculated by (3.40). However, *Sharma* (2000) suggests that caution should be used when relying on the PMI scores, as these values can be sensitive to the calculation of the marginal and joint probability densities in (3.14). Furthermore, the PMI scores may be sensitive to the amount of data, the noise levels in the data, and the order in which the input variables are selected as being important (*Soofi et al.*, 2000). Nevertheless, it is considered that RI values estimated using the PMI scores may still provide a suitable guide for comparing the different input importance measures described in Sections 3.4.4.1 to 3.4.4.4. To verify that this was the case, the PMI algorithm was run applied to data set I and the RI values estimated from the PMI scores obtained were compared to the RI values calculated using the coefficients of (3.25).

In order to discount the effects of suboptimal network performance (e.g. due to inappropriate training or an inappropriate model structure) in the comparison, only the models *best representing the data*, developed using each training algorithm and containing a greater or equal number of hidden nodes to that identified as being optimal in the model selection investigation detailed in Section 3.4.3, were used in the comparison of the input importance measures. The models considered to best represent the data were determined based on how similar the estimated MAE and residual variance values, calculated when the models were applied to the training and testing data sets, were to the corresponding values calculated based on the actual random noise added to the simulated data.

Two methods of evaluation were used to assess the accuracy of the methods for quantifying input importance. The first was Gower's similarity coefficient (GSC) (*Gower*, 1971), which was used to compare the *order* of input importance as estimated by the methods investigated to the order of input importance estimated by the PMI procedure (referred to as *actual* order of importance for the purposes of this discussion). GSC was also used in the comparison conducted by *Olden et al.* (2004). The similarity of two individuals i and j (e.g. estimated order of importance and actual order of importance) may be compared on a characteristic k (e.g. order of importance of input k) by assigning a score s_{ijk} according to:

$$s_{ijk} = 1.0 - \frac{|x_{ik} - x_{jk}|}{r_k} \tag{3.42}$$

where x_{ik} is the value of individual i and x_{jk} is the value of individual j for characteristic k. The value of r_k is given by the range of values possible for the kth characteristic. For example, if there are three inputs which can have a possible order of importance between 1 and 3, the value of r_k is 2 (i.e. 3 - 1). To determine the overall similarity of individuals i and j, having $k = 1, \ldots, \nu$ characteristics, the following equation is used:

$$GSC_{ij} = \frac{\sum_{k=1}^{\nu} s_{ijk} W_{ijk}}{\sum_{k=1}^{\nu} W_{ijk}} \tag{3.43}$$

where W_{ijk} is a weight assigned to the kth characteristic score. In this research, all of the weight values were set to 1.0. A GSC value of one indicates exact similarity between the individuals, whereas a value of zero indicates no similarity.

While the similarity between the estimated and actual order of input importance provides a good assessment of the input importance measures, this evaluation does not take into account the similarity between the estimated and actual relative *magnitudes* of importance. Therefore, the second method of evaluation was based on the RMSE (given by (3.2)) between the absolute estimated RI values and the RI values estimated by the PMI approach. Absolute RI values were used in the evaluation, as the PMI approach does not give directions of the input-output relationships.

3.4.5 Results

3.4.5.1 Comparison of Training Algorithms

The training algorithm comparison results are summarised in Tables 3.5, 3.6 and 3.7 for data sets I, II and III, respectively. These tables give the best, worst, mean and standard deviation mean squared error (MSE) results obtained by initialising each training algorithm five times for each network size. The MSE values ($\mathrm{MSE} = \mathrm{SSE}/N$) are presented rather than the SSE values, as taking the mean of the error cancels the effect the training set size has on the results. The best results obtained for each network size are highlighted by bold italics. These include the smallest best, average and standard deviation MSE values obtained for a given network size, as well as the smallest, or *least*, worst MSE value. These results are also given for the *overall* performance of the training algorithms, averaged over all of the network sizes. Extended training results are presented in Appendix A.

Inspection of Table 3.5 shows that all three algorithms performed similarly when applied to data set I. However, while none of the algorithms performed obviously better than the others, it appears that the SCE-UA algorithm was more consistent in obtaining good solutions. The GA performed slightly worse than the other algorithms in training the models to fit the data; however, this was probably due to its poor 'fine-tuning' abilities, as the MSE values obtained were generally consistent (indicated by the small standard deviations) and were not significantly greater than those obtained using the BP and SCE-UA algorithms.

From the results obtained when the algorithms were applied to data set II (shown in Table 3.6), it is apparent that, while the BP algorithm seems better able to obtain a 'best' solution, particularly for the larger network sizes, it is also the least robust to different initial conditions, as suggested by the overall MSE standard deviation, which was the largest obtained of the three algorithms. Overall, the SCE-UA algorithm performed consistently better than the BP algorithm and the GA, as indicated by the smallest overall average MSE, the overall 'least worst' MSE and the smallest overall standard deviation. Again, the GA did not perform as well as the other two algorithms, although, nor did it perform significantly worse.

The training results obtained for data set III (Table 3.7) were very similar to the results obtained for data set II. Again, the BP algorithm was better able to obtain a best solution on the larger network sizes, but was generally not very robust to different initial conditions, as indicated by the relatively large MSE standard deviations. The SCE-UA performed the best on the smaller networks, as well as performing consistently well overall, obtaining the smallest overall MSE average and standard deviation values. While the

Table 3.5 Summary of training set MSE results for data set I.

Hidden	Best			Worst			Average			Standard Deviation		
Nodes	BP	GA	SCE-UA	BP	GA	SCE-UA	BP	GA	SCE-UA	BP	GA	SCE-UA
1	0.453	0.450	***0.448***	0.453	0.450	***0.448***	0.453	0.450	***0.448***	***0.000***	***0.000***	***0.000***
2	***0.438***	***0.438***	***0.438***	0.448	***0.438***	***0.438***	0.444	***0.438***	***0.438***	0.005	***0.000***	***0.000***
3	0.434	0.436	***0.426***	0.443	0.442	***0.436***	0.439	0.439	***0.433***	0.004	***0.002***	0.004
4	***0.418***	0.430	0.422	0.433	0.435	***0.431***	***0.427***	0.433	***0.427***	0.006	***0.002***	0.004
5	0.417	0.427	***0.412***	0.431	0.442	***0.427***	0.424	0.433	***0.420***	***0.005***	0.006	0.006
6	0.412	0.418	***0.411***	***0.418***	0.431	0.423	***0.414***	0.425	0.418	***0.003***	0.006	0.005
7	***0.398***	0.408	0.401	***0.412***	0.430	0.416	0.408	0.422	***0.407***	***0.006***	0.008	0.007
8	0.400	0.404	***0.390***	0.414	0.424	***0.401***	0.405	0.413	***0.397***	***0.005***	0.007	***0.005***
9	***0.377***	0.404	0.391	0.406	0.424	***0.400***	***0.394***	0.412	0.396	0.013	0.009	***0.003***
10	***0.374***	0.395	0.386	***0.396***	0.428	0.405	***0.387***	0.406	0.397	0.008	0.013	***0.007***
Overall	***0.412***	0.421	0.413	0.425	0.434	***0.423***	0.420	0.427	***0.418***	0.006	0.005	***0.004***

Table 3.6 Summary of training set MSE results for data set II.

Hidden	Best			Worst			Average			Standard Deviation		
Nodes	BP	GA	SCE-UA	BP	GA	SCE-UA	BP	GA	SCE-UA	BP	GA	SCE-UA
1	0.641	0.640	***0.639***	0.641	0.641	***0.639***	0.641	0.641	***0.639***	***0.000***	***0.000***	***0.000***
2	***0.558***	0.609	0.629	***0.631***	0.634	0.632	***0.602***	0.625	0.630	0.040	0.011	***0.001***
3	0.485	0.515	***0.480***	0.629	0.629	***0.621***	0.578	0.564	***0.536***	0.061	***0.052***	0.077
4	0.478	0.481	***0.471***	0.589	0.625	***0.477***	0.519	0.529	***0.473***	0.042	0.057	***0.002***
5	***0.463***	0.474	0.465	0.536	***0.493***	0.506	0.498	0.483	***0.475***	0.034	***0.007***	0.018
6	***0.458***	0.459	0.459	0.515	0.526	***0.463***	0.475	0.478	***0.461***	0.023	0.028	***0.002***
7	***0.438***	0.456	0.450	***0.468***	0.482	0.492	***0.454***	0.465	0.462	0.012	***0.010***	0.017
8	***0.433***	0.453	0.447	0.484	0.487	***0.455***	***0.450***	0.469	***0.450***	0.020	0.015	***0.004***
9	***0.420***	0.446	0.445	0.488	0.475	***0.462***	***0.441***	0.456	0.452	0.027	0.013	***0.008***
10	***0.414***	0.445	0.433	***0.446***	0.460	0.470	***0.429***	0.452	0.447	0.012	***0.007***	0.014
Overall	***0.479***	0.498	0.492	0.543	0.545	***0.522***	0.509	0.516	***0.503***	0.027	0.020	***0.014***

GA was generally fairly robust when applied to data sets I and II, it was found to be the least robust when applied to data set III, often obtaining ‘worst’ MSE values that were significantly greater than the ‘best’ MSE values for a given network size. This is possibly due to the fact that the algorithm parameters were selected according to the GA’s performance when applied to data set II, and perhaps these were not the most suitable when applied to data set III.

Overall, it can be concluded that the BP algorithm performed the best on the larger networks (e.g. containing greater than 7 hidden nodes), whereas the SCE-UA algorithm performed well most consistently and was the least sensitive to initial conditions. One of the reasons why the BP algorithm may have performed better on the large networks is that for these networks the SCE-UA algorithm was stopped according to the second stopping criterion applied (i.e. after 10 million function evaluations, see Section 3.4.2.4) and had not converged to within the specified tolerance value. Another possible reason why the BP algorithm performed well on these networks is that the error surface of large networks is less complicated by local minima than those of smaller networks; thus, a local optimisation algorithm may perform as well as, if not better than, a global method, as it would not be as likely to become trapped in local minima. Additionally, global optimisation methods tend to be lacking in local fine-tuning abilities, as their main purpose is to provide a thorough, yet less concentrated, search of the error surface. Therefore, the BP algorithm could be expected to have better training performance than a global method when the error surface is concave and uncomplicated by local minima. To illustrate the different error surfaces of small and large networks, the error surface of a 2 hidden node ANN applied to data set I is shown in Figure 3.18 (a), while the error surface of a 10 hidden node ANN applied to data set I is shown in Figure 3.18 (b). Both of these plots were obtained by altering weights w_{I_1,H_1} and $w_{H_1,O}$ (i.e the weights from the first input to the first hidden node and from the first hidden node to the output), while fixing all of the other weights equal to their optimal values. This gives an extremely simplified view of the error surface; however, it would be impossible to view the true surface with all of the weights changing in multidimensional space. In reality the error surface would be much more complex than those shown here for illustrative purposes. Nevertheless, the plots illustrate how a local optimisation method might perform poorly on a smaller network with a more complicated error surface, than on a larger one with an error surface containing a single minimum, or continuum of minimum error, as shown in Figure 3.18 (b). Not surprisingly, the BP algorithm had the poorest performance of the three training algorithms when applied to the 2 hidden network, whereas it had the best performance when applied

Table 3.7 Summary of training set MSE results for data set III.

Hidden	Best			Worst			Average			Standard Deviation		
Nodes	BP	GA	SCE-UA	BP	GA	SCE-UA	BP	GA	SCE-UA	BP	GA	SCE-UA
1	3.543	3.540	***3.538***	3.543	3.541	***3.538***	3.543	3.541	***3.538***	***0.000***	***0.000***	***0.000***
2	2.403	2.414	***2.356***	3.030	2.554	***2.356***	2.604	2.496	***2.356***	0.256	0.074	***0.000***
3	***1.419***	1.449	1.656	2.004	2.524	***1.656***	***1.646***	1.983	1.656	0.243	0.488	***0.000***
4	0.653	0.737	***0.600***	1.652	1.858	***1.640***	1.051	***1.009***	1.218	0.546	***0.482***	0.563
5	0.559	0.603	***0.526***	0.635	0.738	***0.566***	0.579	0.653	***0.539***	0.033	0.050	***0.017***
6	0.521	0.536	***0.518***	0.582	0.693	***0.567***	0.554	0.613	***0.530***	0.026	0.056	***0.021***
7	0.516	0.549	***0.511***	0.558	0.639	***0.517***	0.535	0.598	***0.513***	0.019	0.042	***0.003***
8	***0.509***	0.540	0.511	0.582	0.631	***0.537***	0.530	0.582	***0.517***	0.030	0.036	***0.011***
9	***0.502***	0.539	0.516	***0.523***	0.572	0.524	***0.514***	0.554	0.519	0.008	0.013	***0.004***
10	***0.489***	0.531	0.520	***0.506***	0.557	0.550	***0.498***	0.541	0.537	***0.006***	0.011	0.013
Overall	***1.112***	1.144	1.125	1.362	1.431	***1.245***	1.205	1.257	***1.192***	0.117	0.125	***0.063***

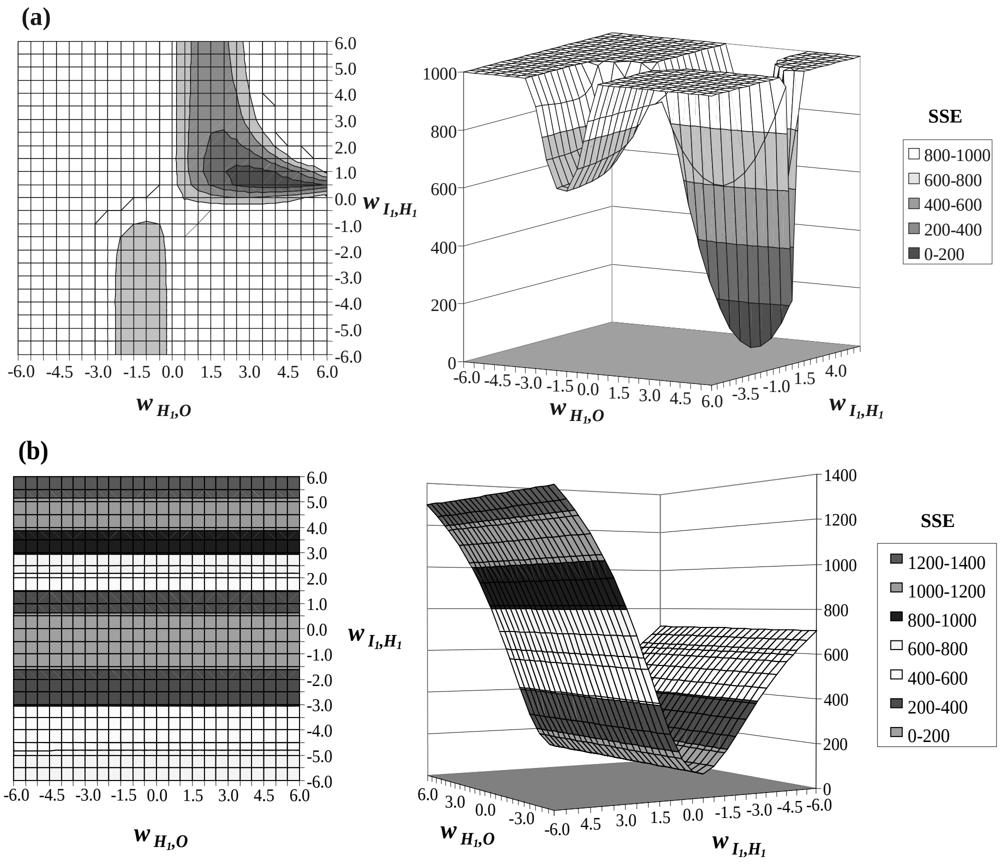

Figure 3.18 The error surface of (a) a 2 hidden node ANN, and (b) a 10 hidden node ANN, applied to data set I.

to the 10 hidden node network (see Table 3.5).

While the SCE-UA algorithm was found to have the overall best and most consistent optimisation performance, a significant limitation of this algorithm was found to be the time required for training, particularly on the larger networks. The average training times in minutes for all three algorithms are presented in Table 3.8, where it can be seen that the SCE-UA algorithm required significantly longer training times than the BP algorithm and the GA, taking between 6 and 30 times longer to converge on the 10 hidden node ANNs than BP or the GA, which had similar training times.

3.4.5.2 Assessment of Model Selection Criteria

The residual variance $\hat{\sigma}_{\mathbf{y}}^2$ and MAE results calculated based on the training set outputs obtained from the best weight initialisation for each network size are presented in Tables 3.9, 3.10 and 3.11 for data sets I, II and III, respectively. It is reiterated that these values were calculated based on the model outputs when the networks were trained to convergence

Table 3.8 Average training times in minutes for the BP, GA and SCE-UA algorithms.

Hidden	Data set I			Data set II			Data set III		
Nodes	BP	GA	SCE	BP	GA	SCE	BP	GA	SCE
1	0.68	0.36	0.09	0.16	0.47	0.25	0.38	0.63	0.42
2	1.16	3.06	0.26	0.42	1.82	1.02	1.24	2.89	2.33
3	1.60	2.24	0.98	2.27	5.97	4.78	2.58	4.32	13.12
4	4.28	4.30	5.54	1.62	7.41	9.94	4.27	5.70	38.59
5	4.88	5.68	10.23	7.08	7.58	25.76	6.60	6.97	73.47
6	6.53	8.17	15.11	15.74	9.35	31.68	4.45	7.02	138.67
7	9.75	8.83	42.77	13.14	8.20	79.40	7.68	8.33	202.65
8	9.45	13.50	112.89	16.69	12.09	126.20	9.53	7.91	244.82
9	11.43	14.63	197.04	26.02	12.31	160.87	9.27	11.33	272.17
10	13.79	20.14	124.10	20.06	12.11	174.98	9.93	11.06	302.18

(or until the maximum number of function evaluations had been reached). For data set I, the variance and MAE values calculated based on the actual random noise added to the simulated training data (henceforth, referred to as *actual* noise variance $\sigma^2_{\mathbf{y}}$ and *actual* MAE) were equal to 0.900 and 0.767, respectively. In Table 3.9, the smallest network able to model the data such that the *estimated* $\hat{\sigma}^2_{\mathbf{y}}$ and MAE were approximately equal to the corresponding actual values is framed. According to these criteria, it can be seen that an ANN containing 1 hidden node was sufficient for modelling data set I. In actual fact, an ANN model containing no hidden nodes would have been sufficient, as discussed in Section 3.4.3; however, a 1 hidden node network was the smallest network considered in the investigation.

Table 3.9 Optimal number of hidden nodes for modelling data set I, indicated by the variance $\hat{\sigma}^2_{\mathbf{y}}$ and MAE values of the model residuals.

Hidden	$\hat{\sigma}^2_{\mathbf{y}}$			MAE		
Nodes	BP	GA	SCE	BP	GA	SCE
1	***0.905***	***0.899***	***0.897***	***0.766***	***0.764***	***0.764***
2	0.877	0.877	0.877	0.748	0.748	0.747
3	0.868	0.872	0.852	0.744	0.746	0.738
4	0.836	0.860	0.843	0.727	0.743	0.734
5	0.833	0.854	0.825	0.736	0.742	0.717
6	0.824	0.837	0.821	0.723	0.732	0.725
7	0.796	0.816	0.802	0.715	0.722	0.712
8	0.799	0.809	0.781	0.713	0.715	0.705
9	0.754	0.808	0.782	0.695	0.718	0.702
10	0.748	0.790	0.773	0.682	0.704	0.707

Table 3.10 Optimal number of hidden nodes for modelling data set II, indicated by the variance $\hat{\sigma}_{\mathbf{y}}^2$ and MAE values of the model residuals.

Hidden	$\hat{\sigma}_{\mathbf{y}}^2$			MAE		
Nodes	BP	GA	SCE	BP	GA	SCE
1	1.282	1.281	1.279	0.907	0.908	0.907
2	1.117	1.219	1.259	0.857	0.890	0.898
3	***0.971***	1.030	***0.961***	***0.790***	0.821	***0.785***
4	0.956	0.963	0.942	0.786	0.784	0.779
5	0.925	0.948	0.931	0.777	0.778	0.768
6	0.916	0.918	0.917	0.768	0.771	0.765
7	0.876	0.911	0.900	0.748	0.767	0.766
8	0.867	0.907	0.893	0.750	0.767	0.755
9	0.840	0.892	0.889	0.735	0.760	0.757
10	0.829	0.889	0.867	0.728	0.754	0.756

For data set II, the actual variance $\sigma_{\mathbf{y}}^2$ and MAE of the training data noise were 0.983 and 0.797, respectively. The smallest network able to model the data such that the estimated $\hat{\sigma}_{\mathbf{y}}^2$ and MAE values approximated the corresponding actual values is framed in Table 3.10. It can be seen that a 3 hidden node network was sufficient for modelling the data, as indicated by the results obtained when the network was trained using the BP and SCE-UA algorithms. It does not matter that a 3 hidden node network trained using the GA was unable to fit the data sufficiently well, as this is a function of optimisation ability, rather than the appropriateness of the model structure. It can be seen that networks containing fewer than 3 hidden nodes were unable to fit the data appropriately when trained with any of the three training algorithms, as indicated by the larger estimated $\hat{\sigma}_{\mathbf{y}}^2$ and MAE values.

The actual $\sigma_{\mathbf{y}}^2$ and MAE values for data set III were 1.048 and 0.816, respectively. In Table 3.11, it can be seen that a network containing 5 to 6 hidden nodes was appropriate for modelling this data set, as indicated by the framed values. It is considered that a 5 hidden node ANN is probably sufficient for modelling the data, as the estimated $\hat{\sigma}_{\mathbf{y}}^2$ and MAE values were approximately equal to the actual values when the network was trained using the SCE-UA algorithm. However, it is apparent that a network of this size is difficult to train appropriately and a network containing 6 hidden nodes had better data mapping abilities.

Table 3.11 Optimal number of hidden nodes for modelling data set III, indicated by the variance $\hat{\sigma}_{\mathbf{y}}^2$ and MAE values of the model residuals.

Hidden	$\hat{\sigma}_{\mathbf{y}}^2$			MAE		
Nodes	BP	GA	SCE	BP	GA	SCE
1	7.175	7.169	7.165	2.053	2.051	2.051
2	4.866	4.888	4.771	1.747	1.756	1.737
3	2.874	2.935	3.353	1.320	1.330	1.484
4	1.322	1.493	1.216	0.908	0.955	0.868
5	1.133	1.221	***1.066***	0.838	0.866	***0.816***
6	***1.055***	1.085	***1.049***	***0.814***	0.825	***0.813***
7	1.045	1.113	1.034	0.810	0.833	0.805
8	1.031	1.094	1.034	0.807	0.832	0.808
9	1.018	1.090	1.044	0.802	0.832	0.814
10	0.991	1.074	1.053	0.794	0.819	0.814

The *in-sample* AIC and BIC results obtained when the ANNs were trained until convergence are given in Tables 3.12, 3.13 and 3.14 for data sets I, II and III, respectively. The minimum values of these criteria (indicating the best generalisability) obtained using each of the three training algorithms are highlighted by bold italics and the minimum values overall are indicated by framed values. The overall best values indicate the 'optimal' model structure according to each criterion. From an inspection of Tables 3.12, 3.13 and 3.14, it can be seen that the overall minimum BIC values correctly indicated the optimal network sizes for data sets I, II and III. It was interesting to note that only the minimum in-sample BIC obtained when the networks were trained using the SCE-UA algorithm

Table 3.12 In-sample AIC and BIC results for data set I.

Hidden	AIC			BIC		
Nodes	BP	GA	SCE	BP	GA	SCE
1	1539.9	1536.2	1534.9	***1565.9***	***1562.1***	$\boxed{\boldsymbol{1560.8}}$
2	1532.2	***1532.1***	1532.0	1579.8	1579.7	1579.6
3	1536.8	1538.9	***1525.9***	1606.0	1608.1	1595.1
4	1525.5	1541.6	1530.3	1616.3	1632.4	1621.1
5	1533.7	1547.4	1528.0	1646.1	1659.9	1640.5
6	1537.8	1546.0	1535.6	1671.8	1680.1	1669.6
7	1530.1	1542.2	1532.3	1690.1	1697.9	1687.9
8	1540.6	1547.0	1527.4	1717.9	1724.3	1704.7
9	$\boxed{\boldsymbol{1518.2}}$	1556.3	1538.0	1717.1	1755.3	1736.9
10	1523.2	1554.1	1541.8	1743.7	1774.7	1762.4

Table 3.13 In-sample AIC and BIC results for data set II.

Hidden	AIC			BIC		
Nodes	BP	GA	SCE	BP	GA	SCE
1	2220.8	2220.0	2218.9	2252.8	2252.0	2250.9
2	2134.1	2196.6	2219.5	2193.5	2256.0	2279.0
3	2045.9	2088.1	2038.4	***2132.7***	2175.0	$\boxed{\textit{\textbf{2125.3}}}$
4	2047.0	2051.9	***2036.6***	2161.3	***2166.2***	2150.9
5	2035.4	2053.0	2039.8	2177.2	2194.8	2181.6
6	2028.7	***2042.0***	2041.5	2170.5	2211.1	2210.6
7	2008.7	2048.8	2039.5	2177.8	2245.4	2236.1
8	2024.9	2057.0	2046.4	2248.9	2281.1	2270.4
9	$\boxed{\textit{\textbf{2014.3}}}$	2057.0	2055.0	2265.7	2308.5	2306.5
10	2016.7	2067.3	2048.9	2295.6	2346.2	2327.9

indicated that a 5 hidden node ANN was optimal, whereas the BIC values obtained when the ANNs were trained using BP and the GA suggest that 6 hidden nodes are best. This is in agreement with the conclusion made based on Table 3.11 that a 5 hidden node ANN may be optimal, but a better data mapping can be more easily obtained with a 6 hidden node ANN. It can also be seen that the AIC values were unable to correctly identify the optimum number of hidden nodes, selecting larger networks than necessary in each case. The AIC values were also found to be much more sensitive to the optimum solution obtained by training, as indicated by the large variation in AIC values obtained using the three different training algorithms.

These results indicate that the in-sample BIC adequately penalises complexity in order

Table 3.14 In-sample AIC and BIC results for data set III.

Hidden	AIC			BIC		
Nodes	BP	GA	SCE	BP	GA	SCE
1	5843.2	5842.3	5841.6	5884.0	5883.1	5882.4
2	5385.6	5391.0	5361.6	5462.1	5467.5	5438.1
3	4759.6	4785.1	4946.9	4871.8	4897.3	5059.1
4	3829.9	3977.9	3728.4	3977.9	4125.9	3876.4
5	3656.3	3747.2	3582.7	3840.0	3930.9	$\boxed{\textit{\textbf{3766.4}}}$
6	3583.7	***3617.9***	3576.8	***3803.1***	***3837.4***	3796.2
7	3586.9	3662.6	***3574.2***	3842.1	3917.7	3829.3
8	3584.4	3655.8	3587.6	3875.3	3946.6	3878.5
9	3582.1	3666.3	3613.5	3908.7	3992.8	3940.1
10	$\boxed{\textit{\textbf{3550.3}}}$	3662.1	3637.9	3876.9	4024.4	4000.2

to balance model fitting and model parsimony. However, whether this is true in all cases is not clear from the results obtained in this investigation, as it was found that the in-sample BIC values calculated for all of the models developed were very similar whether training was run until convergence or stopped early according to the test set error, as shown in Figure 3.19. This suggests that the degree to which the models were overfitted was never great enough to have a significant impact on the BIC values obtained. Therefore, these results may not be reflective of complex real-world problems with noisy data, where the degree of overfitting that is possible may be significantly greater than observed in this investigation. This issue requires further investigation.

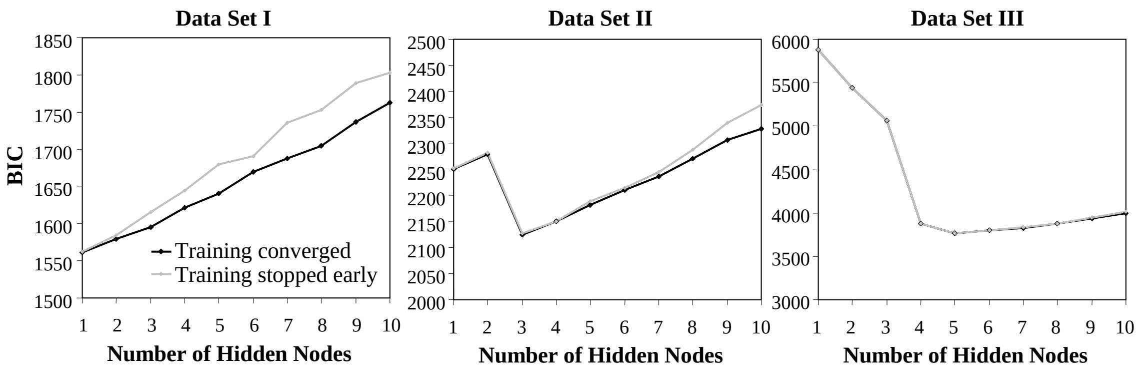

Figure 3.19 In-sample BIC results when training was stopped early and run to convergence.

Shown in Tables 3.15, 3.16 and 3.17 are the out-of-sample results of the model selection investigation. Again, the best values of these criteria (i.e. minimum RMSE, MAE, AIC and BIC values and maximum r^2 and CE values) obtained using each of the three training algorithms are highlighted by bold italics and the best values overall are indicated by framed values. By inspecting these three tables it can be seen that, while the RMSE, MAE, r^2 and CE criteria are reasonably consistent with one another, they all indicate that a larger than necessary network is optimal for each of the three data sets. Furthermore, there is quite large variation among these values for the different training algorithms, indicating that they can be quite sensitive to the solution obtained during training. The AIC and BIC seem better able to correctly select the optimum number of hidden nodes, with the AIC correctly selecting the optimum network for data sets I, II and III, and the BIC correctly selecting the optimum network for data sets I, and III. However, it is apparent that the BIC may overly penalise model complexity when used to assess out-of-sample performance, as can be seen most clearly when used to evaluate the generalisability of the ANN models developed for data set II (Table 3.16).

Table 3.15 Out-of-sample performance results for data set I.

Hidden	RMSE			MAE			AIC			BIC			r^2			CE		
Nodes	BP	GA	SCE	BP	GA	SCE	BP	GA	SCE	BP	GA	SCE	BP	GA	SCE	BP	GA	SCE
1	1.054	1.049	1.047	0.856	0.846	0.845	***424***	***423***	$\boxed{\textbf{\textit{422}}}$	***442***	***440***	$\boxed{\textbf{\textit{440}}}$	0.675	0.676	0.678	0.669	0.672	0.673
2	1.049	1.047	1.045	0.851	0.849	0.845	433	432	432	465	464	464	0.675	0.675	0.676	0.672	0.673	0.674
3	1.056	1.045	1.046	0.859	0.846	0.851	445	442	442	492	489	488.9	0.674	0.676	0.675	0.668	0.674	0.674
4	1.054	1.045	1.045	0.855	0.844	0.845	454	452	452	516	514	514	0.672	0.676	0.676	0.669	0.674	0.674
5	1.056	1.041	1.043	0.848	0.842	0.845	465	461	461	541	537	538	0.668	0.679	0.677	0.668	0.677	0.676
6	1.056	***1.039***	1.042	0.854	0.839	$\boxed{\textbf{\textit{0.830}}}$	475	470	471	566	561	562	0.674	***0.679***	0.677	0.667	***0.678***	0.676
7	1.060	1.044	1.043	0.847	0.841	0.845	488	481	481	596	587	587	0.666	0.675	0.677	0.665	0.675	0.675
8	1.052	1.046	1.043	0.856	***0.835***	0.843	493	492	491	614	613	612	0.676	0.675	0.676	0.670	0.673	0.676
9	1.063	1.045	$\boxed{\textbf{\textit{1.037}}}$	0.863	0.836	0.844	506	502	499	642	637	635	0.669	0.677	$\boxed{\textbf{\textit{0.682}}}$	0.663	0.674	$\boxed{\textbf{\textit{0.679}}}$
10	***1.047***	1.040	1.043	***0.843***	0.843	0.831	512	510	511	662	660	661	***0.676***	0.678	0.677	***0.673***	0.677	0.675

Table 3.16 Out-of-sample performance results for data set II.

Hidden	RMSE			MAE			AIC			BIC			r^2			CE		
Nodes	BP	GA	SCE	BP	GA	SCE	BP	GA	SCE	BP	GA	SCE	BP	GA	SCE	BP	GA	SCE
1	1.172	1.168	1.174	0.923	0.924	0.922	579	578	579	601	$\boxed{\boldsymbol{600}}$	***602***	0.561	0.558	0.561	0.550	0.553	0.549
2	1.073	1.155	1.166	0.852	0.912	0.912	559	585	589	***600***	627	630	0.650	0.579	0.568	0.623	0.563	0.555
3	1.020	1.048	1.013	0.819	0.819	0.812	***553***	***563***	$\boxed{\boldsymbol{550}}$	613	623	611	0.672	0.656	0.682	0.659	0.640	0.664
4	1.005	1.023	1.001	0.803	0.817	0.802	560	566	558	639	646	638	0.685	0.673	0.686	0.669	0.657	0.672
5	***0.997***	1.012	0.997	***0.792***	0.810	0.800	569	574	569	668	673	668	***0.694***	0.681	0.694	***0.675***	0.665	0.674
6	1.007	***0.993***	1.002	0.804	***0.795***	0.800	584	579	583	702	697	701	0.677	$\boxed{\boldsymbol{0.697}}$	0.689	0.668	***0.677***	0.671
7	1.001	1.004	1.004	0.801	0.802	0.812	594	595	595	731	732	732	0.693	0.686	0.680	0.672	0.670	0.670
8	1.004	1.006	$\boxed{\boldsymbol{0.987}}$	0.800	0.813	$\boxed{\boldsymbol{0.791}}$	607	608	601	763	764	757	0.685	0.682	***0.695***	0.670	0.669	$\boxed{\boldsymbol{0.681}}$
9	1.003	1.001	0.998	0.810	0.814	0.795	619	618	617	794	794	793	0.686	0.685	0.689	0.670	0.671	0.674
10	1.012	1.004	1.010	0.808	0.805	0.808	634	631	634	829	826	828	0.683	0.685	0.681	0.664	0.670	0.666

Table 3.17 Out-of-sample performance results for data set III.

Hidden	RMSE			MAE			AIC			BIC			r^2			CE		
Nodes	BP	GA	SCE	BP	GA	SCE	BP	GA	SCE	BP	GA	SCE	BP	GA	SCE	BP	GA	SCE
1	2.780	2.786	2.779	2.083	2.085	2.080	1495	1497	1495	1525	1526	1525	0.595	0.595	0.597	0.595	0.593	0.595
2	2.254	2.274	2.226	1.812	1.825	1.791	1382	1388	1375	1438	1443	1431	0.735	0.729	0.740	0.733	0.729	0.740
3	1.818	1.856	1.774	1.447	1.464	1.428	1266	1279	1251	1348	1360	1333	0.828	0.820	0.835	0.827	0.819	0.835
4	1.172	1.258	1.158	0.931	0.991	0.917	1014	1057	1007	***1122***	1165	1114	0.929	0.918	0.930	0.928	0.917	0.930
5	1.106	1.171	1.075	0.878	0.928	0.855	***993***	1027	$\boxed{\textit{\textbf{975}}}$	1126	***1161***	$\boxed{\textit{\textbf{1109}}}$	0.936	0.928	0.940	0.936	0.928	0.939
6	1.099	1.111	1.072	0.888	0.881	0.856	1003	***1010***	988	1163	1169	1148	0.937	0.936	0.940	0.937	0.935	0.940
7	1.095	1.116	$\boxed{\textit{\textbf{1.070}}}$	0.873	0.876	$\boxed{\textit{\textbf{0.848}}}$	1015	1026	1001	1201	1212	1186	0.937	0.936	$\boxed{\textit{\textbf{0.941}}}$	0.937	0.935	$\boxed{\textit{\textbf{0.940}}}$
8	1.090	1.076	1.080	0.863	***0.864***	0.866	1026	1018	1021	1238	1230	1232	0.938	***0.940***	0.940	0.938	***0.939***	0.939
9	***1.070***	1.090	1.083	***0.855***	0.867	0.869	1029	1040	1036	1267	1278	1274	***0.940***	0.938	0.939	***0.940***	0.938	0.938
10	1.076	***1.085***	1.080	0.855	0.876	0.854	1032	1051	1049	1270	1315	1312	0.940	0.939	0.940	0.939	0.938	0.939

The models containing the 'optimal' number of hidden nodes, as identified using the various generalisation measures, were applied to the independent validation data sets in order to assess the generalisability of the models when applied to data not used during training. These results are given in Tables 3.18, 3.19 and 3.20 for data sets I, II and III, respectively. For data set I, the models developed containing 1 hidden node, selected as optimal by the in-sample BIC and out-of-sample AIC and BIC, performed the best on the independent validation data, as highlighted in Table 3.18. The actual noise added to the simulated validation data had a variance of $\sigma_{\mathbf{y}}^2 = 1.035$ and an average absolute magnitude of $\mathrm{MAE} = 0.816$. It can be seen that the estimated $\hat{\sigma}_{\mathbf{y}}^2$ and MAE values for the models containing 1 hidden node closely approximated the actual values, indicating good generalisability of the models.

For data set II, the models developed containing 3 hidden nodes, selected as optimal by the in-sample BIC and out-of-sample AIC, had the best performance on the independent validation data. The actual $\sigma_{\mathbf{y}}^2$ and MAE values for the noise added to these data were 0.852 and 0.733, respectively. It can be seen in Table 3.19 that the estimated $\hat{\sigma}_{\mathbf{y}}^2$ and MAE values for the models containing 3 hidden nodes best approximated the corresponding actual values, indicating that this model had the best generalisability of the models selected by the various criteria.

The 7 hidden node ANN model, selected as the optimal network size by the out-of-sample RMSE, MAE, r^2 and CE, had the best performance on the independent validation data of data set III, as seen in Table 3.20. For this data set, the actual $\sigma_{\mathbf{y}}^2$ and MAE values for the added noise were 0.869 and 0.752, respectively. It can be seen that the estimated $\hat{\sigma}_{\mathbf{y}}^2$ and MAE values for the 7 hidden node ANN best approximate the actual

Table 3.18 Validation set results for data set I.

Performance	Number of hidden nodes				
Measure	1[a]	1[b]	6[c]	9[d]	9[e]
$\hat{\sigma}_{\mathbf{y}}^2$	1.025	***1.023***	1.052	1.099	1.542
RMSE	1.013	***1.011***	1.026	1.048	1.242
MAE	0.816	***0.815***	0.826	0.843	0.953
AIC	***513***	***513***	568	605	664
BIC	***532***	***532***	666	751	810
r^2	0.702	***0.704***	0.695	0.681	0.586
CE	0.702	***0.703***	0.694	0.681	0.552

[a]Trained until convergence with SCE-UA, selected as best by in-sample BIC
[b]Trained with SCE-UA, training stopped early by cross-validation, selected as best by out-of-sample AIC and BIC
[c]Trained with SCE-UA, training stopped early by cross-validation, selected as best by out-of-sample MAE
[d]Trained with SCE-UA, training stopped early by cross-validation, selected as best by out-of-sample RMSE, r^2 and CE
[e]Trained until convergence with BP, selected as best by in-sample AIC

Table 3.19 Validation set results for data set II.

Performance	Number of hidden nodes					
Measure	1[a]	3[b]	3[c]	6[d]	8[e]	9[f]
$\hat{\sigma}^2_{\mathbf{y}}$	1.229	0.922	***0.912***	0.940	0.973	0.953
RMSE	1.108	0.960	***0.955***	0.969	0.986	0.976
MAE	0.873	0.766	***0.760***	0.776	0.792	0.780
AIC	696	655	***653***	696	728	735
BIC	720	720	***718***	822	895	923
r^2	0.583	0.686	***0.689***	0.679	0.667	0.672
CE	0.576	0.682	***0.685***	0.676	0.664	0.671

[a]Trained with GA, training stopped early by cross-validation, selected as best by out-of-sample BIC
[b]Trained with SCE-UA, training stopped early by cross-validation, selected as best by out-of-sample AIC
[c]Trained until convergence with SCE-UA, selected as best by in-sample BIC
[d]Trained with GA, training stopped early by cross-validation, selected as best by out-of-sample r^2
[e]Trained with SCE-UA, training stopped early by cross-validation, selected as best by out-of-sample RMSE, MAE and CE
[f]Trained until convergence with BP, selected as best by in-sample AIC

values, indicating that this model had the best generalisability. It can also be seen that the 5 hidden node models, selected as the optimal network size by the in-sample BIC and out-of-sample AIC and BIC, achieved a similar, although slightly worse overall fit to the data. As suggested from an inspection of Table 3.11, an ANN containing 5 to 6 hidden nodes was optimal for this data set, which is possibly why the results for the 5 hidden nodes models were slightly worse on the validation data than those for the 7 hidden node model.

Table 3.20 Validation set results for data set III.

Performance	Number of hidden nodes			
Measure	5[a]	5[b]	7[c]	10[d]
$\hat{\sigma}^2_{\mathbf{y}}$	0.927	0.927	***0.894***	0.994
RMSE	0.963	0.963	***0.945***	0.997
MAE	0.770	0.771	***0.756***	0.791
AIC	1122	***1122***	1136	1204
BIC	1264	***1263***	1333	1456
r^2	0.947	0.947	***0.949***	0.944
CE	0.947	0.947	***0.949***	0.944

[a]Trained until convergence with SCE-UA, selected as best by in-sample BIC
[b]Trained with SCE-UA, training stopped early by cross-validation, selected as best by out-of-sample AIC and BIC
[c]Trained with SCE-UA, training stopped early by cross-validation, selected as best by out-of-sample RMSE, MAE, r^2 and CE
[d]Trained until convergence with BP, selected as best by in-sample AIC

Overall, these results indicate that the *out-of-sample* AIC and *in-sample* BIC are the most suitable criteria for ANN model selection. It was apparent that by accounting for model complexity as well as fit, the AIC was less sensitive to factors such as the testing data used or the weights obtained during training than other out-of-sample generalisability measures. However, the penalty given to model complexity was not sufficient to use the AIC criterion as an in-sample measure of generalisability. On the other hand, for the synthetic data sets considered in the investigation, it was found that the BIC criterion was able to penalise model complexity sufficiently, in order to select the most appropriate model structures based on in-sample performance. It is claimed that a major advantage of using in-sample criteria is that a test data set is not required. From Tables 3.18, 3.19 and 3.20, it is apparent that this is the case, as the optimal models selected using the in-sample BIC had similar generalisability whether training was stopped early or allowed to run to convergence. However, as mentioned previously, it is uncertain whether the in-sample BIC will adequately penalise model complexity in order to select the optimal model structure, given different degrees of overfitting potential, which is a function of the complexity of the problem being modelled, the amount of available data and the noise levels in the data.

3.4.5.3 ANN Validation

The results from applying the stepwise PMI input selection procedure to data sets I, II and III are given in Table 3.21. It was found that this method was correctly able to select only the important inputs from a set of 15 potentially important inputs for each synthetic data set, verifying the approach as an input selection method. However, to verify the approach as a suitable method for quantifying input importance, the RI values estimated for data set I were compared to the actual (absolute) RI values of the simulated data. The magnitudes of the contributions of y_{t-1}, y_{t-4} and y_{t-9} in predicting the output y_t have the ratio of $3 : 6 : 5$, as seen in (3.25); thus, the actual RI values of the inputs are 21.43%, 42.86% and 35.71%, respectively. It can be seen in Table 3.21 that the RI values for data set I, estimated based on the PMI scores, are a good approximation to the corresponding actual RI values. Furthermore, when applied to data set II, it can be seen that the PMI-based RI values correctly estimated similar proportions for these inputs. For data set III, it was known that the RI value of input x_4 should be approximately twice that of input x_4 and that the RI values of inputs x_1 and x_2 should be approximately equal (see (3.27)). The PMI method was able to correctly estimate all of these proportions, validating the procedure as an appropriate method for quantifying the relative importance of inputs.

Table 3.21 PMI results for data sets I, II, and III.

Input	PMI Score	RI (%)
Data set I		
y_{t-1}	0.184	22.33
y_{t-4}	0.372	45.07
y_{t-9}	0.269	32.60
Data set II		
y_{t-1}	0.203	21.62
y_{t-4}	0.339	36.13
y_{t-9}	0.289	30.82
x_t	0.193	11.44
Data set III		
x_1	0.185	20.07
x_2	0.209	22.65
x_3	0.098	10.65
x_4	0.284	30.80
x_5	0.146	15.83

As the optimal network configuration for data set I (of those investigated) was found to contain 1 hidden node (see Section 3.4.5.2), the four relative input importance measures investigated were calculated for all 10 network sizes (i.e. containing between 1 and 10 hidden nodes) using the models that best represented the data for each network size. Figure 3.20 displays the mean RI values, averaged over the RI values calculated for each of the 10 networks, obtained by (a) the BP algorithm; (b) the GA; (c) the SCE-UA algorithm; and (d) averaging over the results obtained for all three training algorithms. Error bars are also displayed in this figure, showing the standard deviations of the RI values, which indicate the precision of the methods. It can be seen in this figure that each of the methods for assessing input importance performed reasonably well in terms of accurately estimating the RI values estimated using the PMI procedure, and that none of the methods performed obviously better than the others. The different degrees of precision in the measures obtained by the three training algorithms can also be clearly seen in this figure, with reasonably large variation in the RI values when the weights were obtained by the BP algorithm and very small variation when the weights were estimated using the SCE-UA algorithm. This confirms that the SCE-UA training algorithm most consistently obtained good solutions of the three training algorithms investigated.

Table 3.22 summarises the evaluation of each method's accuracy in estimating the relative magnitudes and orders of input importance when applied to data set I. These are given in terms of the mean and standard deviation RMSE and GSC values calculated based

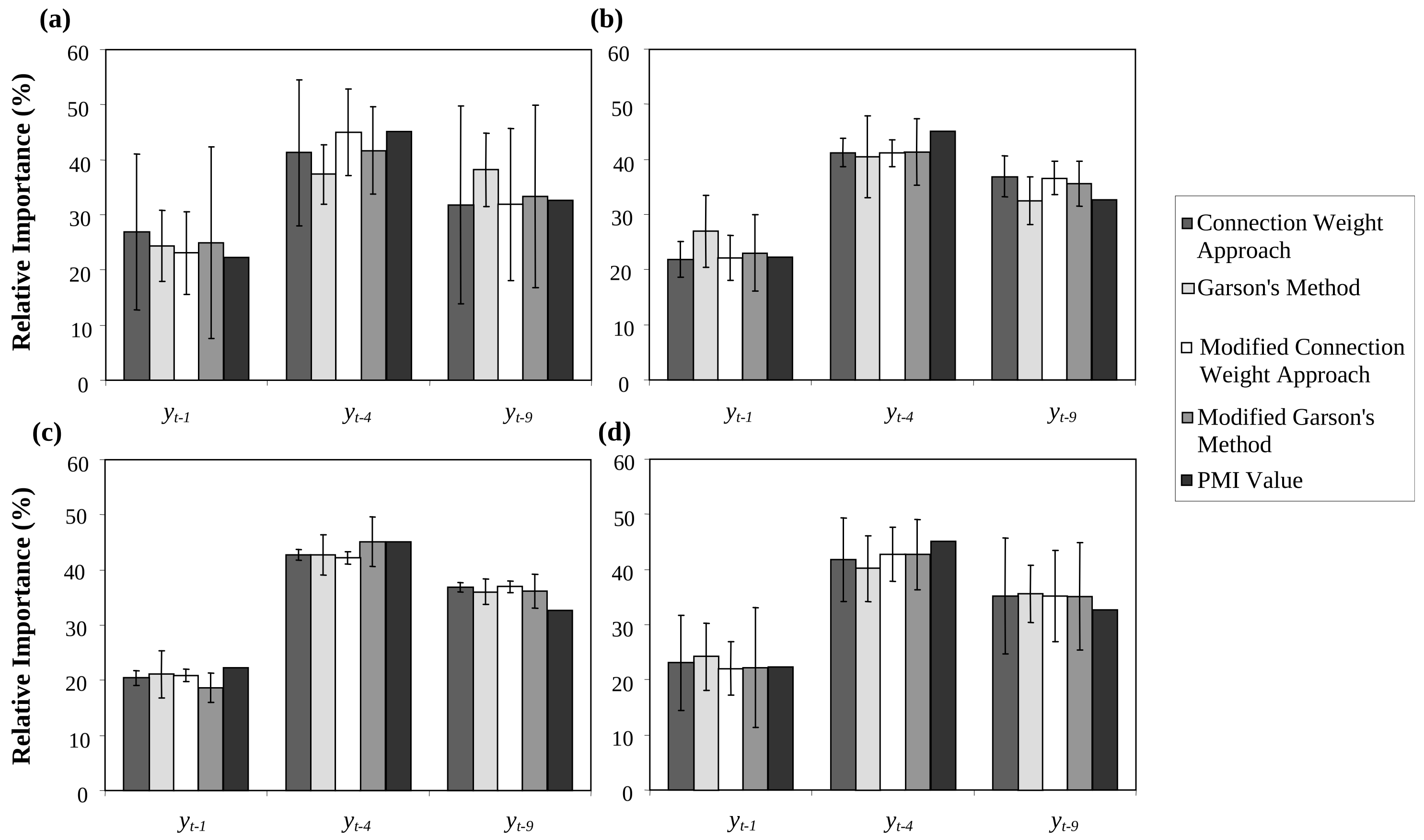

Figure 3.20 Average RI values estimated using the four different input importance measures when applied to Data Set I. Results were obtained based on 10 ANN sizes trained by (a) BP; (b) GA and (c) SCE-UA. Average results obtained by all of the algorithms are given in (d). Error bars represent the standard deviations of the estimated RI values.

Table 3.22 Evaluation of input importance measures when applied to data set I.

Input Importance Estimation Method	RMSE Mean	RMSE St. Dev.	GSC Mean	GSC St. Dev.
		BP		
Connection Weight Approach	13.110	7.416	0.700	0.189
Garson's Method	***7.195***	***4.009***	0.767	0.225
Modified Connection Weight Approach	7.941	5.748	***0.833***	***0.176***
Modified Garson's Method	10.421	10.004	0.800	0.233
		GA		
Connection Weight Approach	***4.183***	1.863	0.900	0.161
Garson's Method	6.163	3.566	0.833	0.283
Modified Connection Weight Approach	4.302	***1.239***	***0.967***	***0.105***
Modified Garson's Method	4.965	3.933	0.900	0.225
		SCE-UA		
Connection Weight Approach	***3.137***	***0.419***	***0.800***	0.155
Garson's Method	3.966	1.424	0.600	***0.000***
Modified Connection Weight Approach	3.239	0.630	0.767	0.103
Modified Garson's Method	4.033	2.019	0.783	0.075
		Overall		
Connection Weight Approach	6.810	6.239	0.867	0.188
Garson's Method	5.775	***3.383***	0.856	0.226
Modified Connection Weight Approach	***5.161***	3.879	***0.933***	***0.136***
Modified Garson's Method	6.473	6.733	0.900	0.199

on the 10 different network sizes. The best results (i.e. minimum RMSE mean, RMSE standard deviation and GSC standard deviation and maximum GSC mean) are highlighted by bold italics. It can be seen in this table that, while similar results were obtained using each of the input importance measures, the overall best results were obtained using the modified Connection Weight Approach.

The optimal number of hidden nodes necessary for modelling data set II was found to be 3 (see Section 3.4.5.2); therefore, the four relative input importance measures were calculated for networks containing 3 or more hidden nodes (8 network sizes) using the models that best represented the data for each network size. The mean and standard deviation RI values obtained from the 8 networks developed by (a) the BP algorithm; (b) the GA; (c) the SCE-UA algorithm; and (d) by averaging over the results obtained for all three training algorithms, are displayed in Figure 3.21. It can be seen in this figure that the two modified methods (modified Connection Weight Approach and modified Garson's method) were most accurate in estimating the PMI-based RI values, particularly in esti-

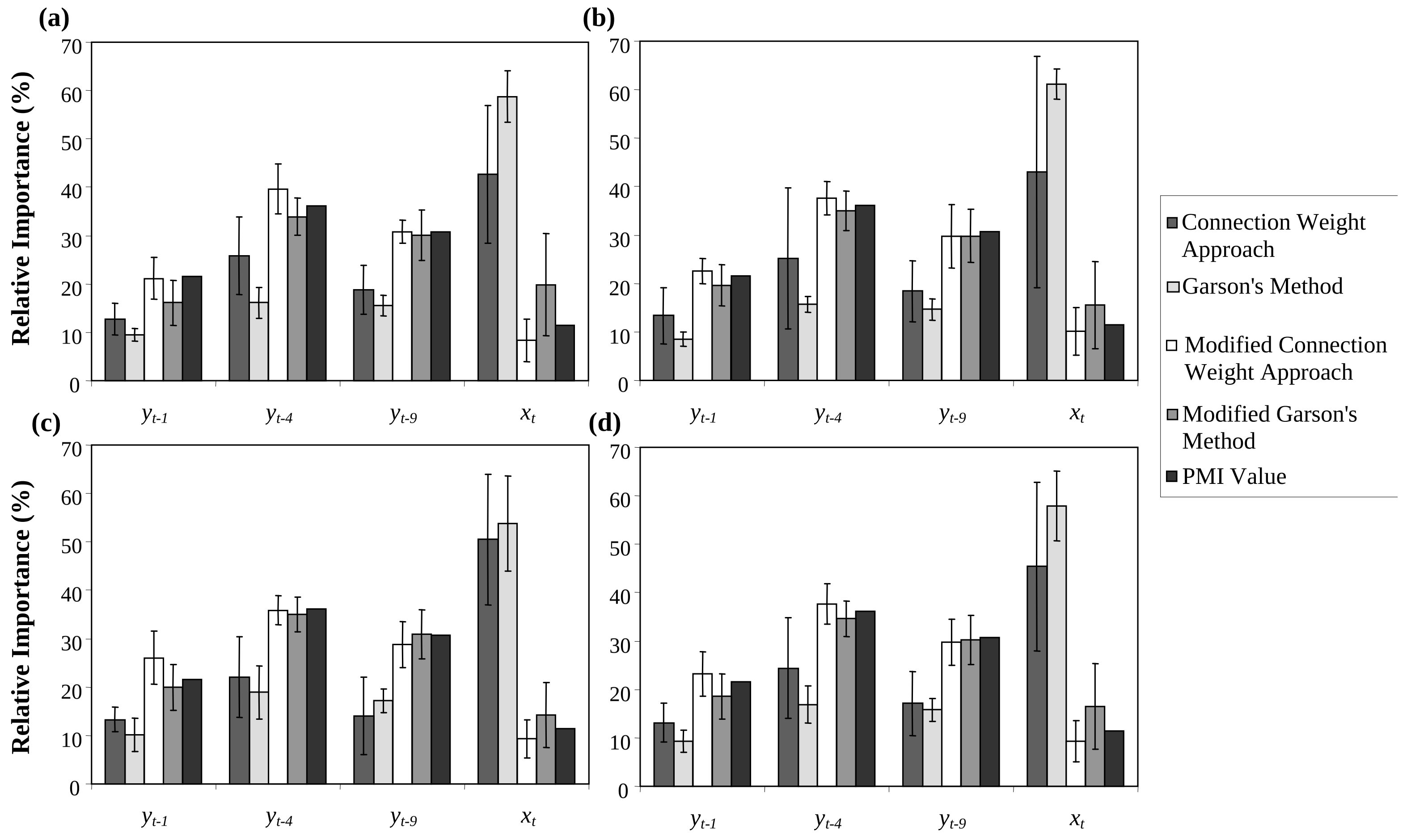

Figure 3.21 Average RI values estimated using the four different input importance measures when applied to Data Set II. Results were obtained based on 8 ANN sizes trained by (a) BP; (b) GA and (c) SCE-UA. Average results obtained by all of the algorithms are given in (d). Error bars represent the standard deviations of the estimated RI values.

mating the importance of the independent nonlinear input x_t. It can also be seen that, as well as being inaccurate, the original Connection Weight Approach was the least stable of the input importance measures, as suggested by the large error bars obtained for the RI values.

Table 3.23 summarises the evaluation of each method's accuracy in estimating the relative magnitudes and orders of input importance when applied to data set II. These results were calculated based on the 8 different network sizes considered and, again, the best results are highlighted by bold italics in this table. It can be seen the overall best results were again obtained using the modified Connection Weight Approach; however, it can also be seen that both the modified Connection Weight Approach and the modified Garson's method were significantly more accurate than the original methods when applied to data set II.

It was found that an ANN containing at least 5 hidden nodes was necessary for modelling data set III (see Section 3.4.5.2). Therefore, the four relative input importance

Table 3.23 Evaluation of input importance measures when applied to data set II.

Input Importance	RMSE		GSC	
Estimation Method	Mean	St. Dev.	Mean	St. Dev.
		BP		
Connection Weight Approach	18.290	8.153	0.583	0.126
Garson's Method	27.490	3.122	0.500	***0.000***
Modified Connection Weight Approach	***3.605***	***2.955***	***1.000***	***0.000***
Modified Garson's Method	6.895	4.431	0.833	0.199
		GA		
Connection Weight Approach	20.880	9.967	0.583	0.154
Garson's Method	28.843	***1.738***	0.500	***0.000***
Modified Connection Weight Approach	***3.953***	2.273	***0.958***	0.077
Modified Garson's Method	4.913	3.926	0.896	0.177
		SCE-UA		
Connection Weight Approach	23.107	7.823	0.479	0.059
Garson's Method	24.628	5.572	0.500	***0.000***
Modified Connection Weight Approach	***4.013***	3.003	***0.938***	0.124
Modified Garson's Method	4.902	***1.575***	0.896	0.086
		Overall		
Connection Weight Approach	20.759	8.552	0.549	0.125
Garson's Method	26.987	4.069	0.500	0.226
Modified Connection Weight Approach	***3.857***	***2.647***	***0.965***	***0.085***
Modified Garson's Method	5.570	3.512	0.875	0.157

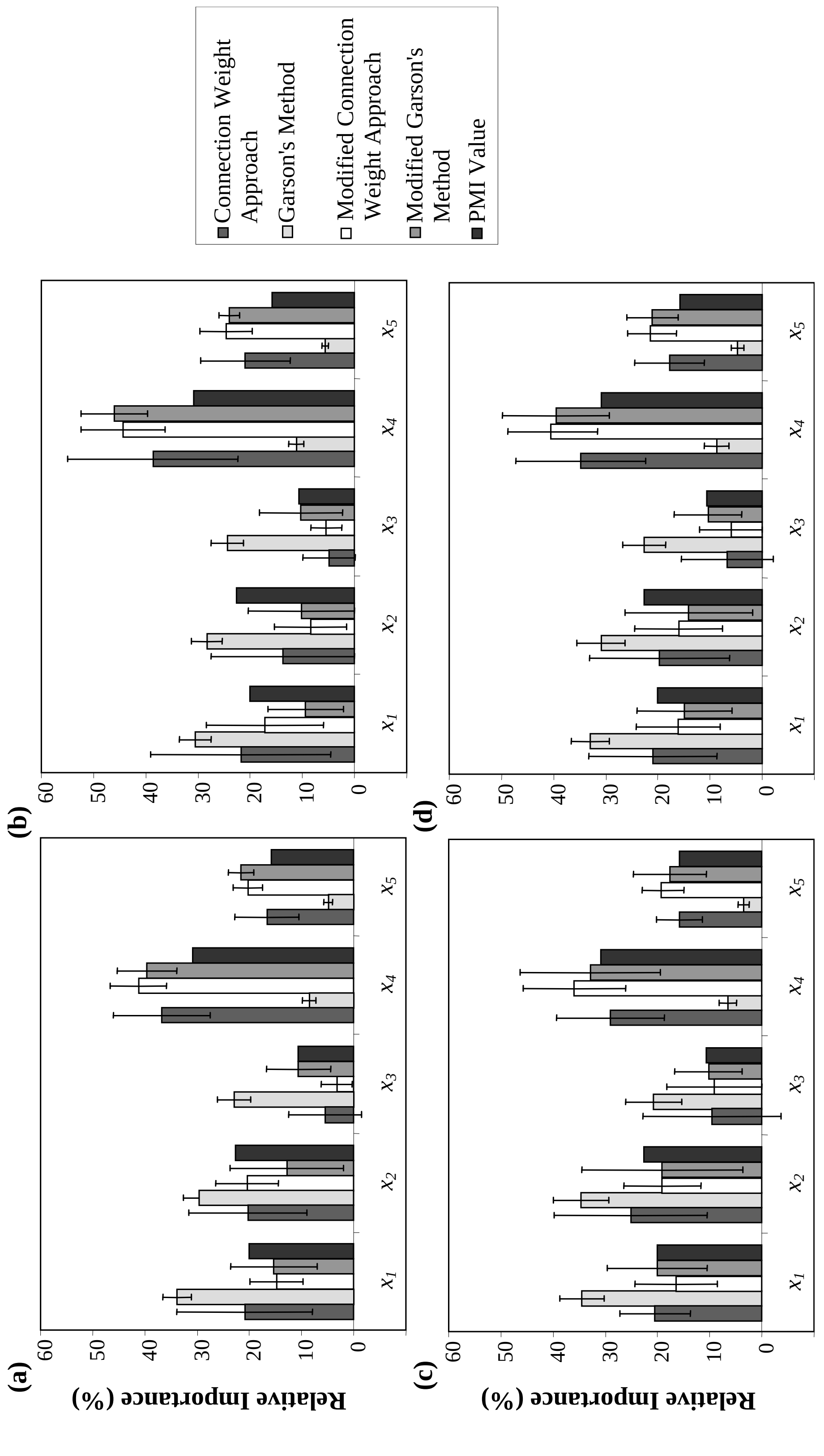

Figure 3.22 Average RI values estimated using the four different input importance measures when applied to Data Set III. Results were obtained based on 6 ANN sizes trained by (a) BP; (b) GA and (c) SCE-UA. Average results obtained by all of the algorithms are given in (d). Error bars represent the standard deviations of the estimated RI values.

measures were calculated for networks containing 5 or more hidden nodes (6 network sizes) using the models that best represented the data for each network size. Figure 3.22 shows the mean and standard deviation RI values obtained from the 5 networks developed by (a) the BP algorithm; (b) the GA; (c) the SCE-UA algorithm; and (d) by averaging over the results obtained for all three training algorithms, in comparison to the PMI-base RI estimates. It can be seen in this figure that the two modified approaches were again reasonably accurate in estimating the PMI-based RI values; however, it is apparent that the original Connection Weight Approach most accurately estimated the PMI-based RI estimates on average. Yet, it can also be seen that this method was the least precise, or most sensitive to the weights obtained for the different network sizes and by the different training algorithms. Garson's method was the most stable method; however, it was also the least accurate for quantifying the magnitudes of input importance.

The mean and standard deviation RMSE and GSC values obtained by applying the four input importance measures to data set III are given in Table 3.24. These results

Table 3.24 Evaluation of input importance measures when applied to data set III.

Input Importance	RMSE		GSC	
Estimation Method	Mean	St. Dev.	Mean	St. Dev.
		BP		
Connection Weight Approach	8.998	3.700	***0.833***	0.137
Garson's Method	14.337	***0.653***	0.600	***0.000***
Modified Connection Weight Approach	***7.353***	2.878	***0.833***	0.052
Modified Garson's Method	8.915	3.474	0.750	0.152
		GA		
Connection Weight Approach	12.710	5.070	***0.750***	0.055
Garson's Method	13.002	***0.661***	0.583	***0.041***
Modified Connection Weight Approach	***11.476***	4.061	***0.750***	0.122
Modified Garson's Method	12.288	2.679	0.717	0.117
		SCE-UA		
Connection Weight Approach	9.159	3.727	***0.800***	0.155
Garson's Method	15.834	***1.175***	0.600	***0.000***
Modified Connection Weight Approach	***7.813***	2.188	0.767	0.103
Modified Garson's Method	9.829	2.991	0.783	0.075
		Overall		
Connection Weight Approach	10.289	4.334	***0.794***	0.121
Garson's Method	14.391	***1.441***	0.594	***0.024***
Modified Connection Weight Approach	***8.881***	3.507	0.783	0.099
Modified Garson's Method	10.344	3.231	0.750	0.115

were calculated based on the 6 different network sizes considered. Overall, it can be seen that the modified Connection Weight Approach most accurately approximated the relative magnitudes of input importance, whereas the original Connection Weight Approach most accurately estimated the order of input importance. However, it can also be seen that the mean GSC values obtained for the original and modified Connection Weight Approaches were very similar, but large standard deviations of the RMSE and GSC were obtained using the Connection Weight Approach, suggesting that this approach is sensitive to the weights obtained.

Shown in Table 3.25, are the *overall* average and standard deviation RMSE and GSC values, based on the results obtained for the four input importance measures when applied to data sets I, II and III. It can be seen in this table that, overall, the modified Connection Weight Approach most accurately estimated the order and magnitude of input importance. Garson's measure was found to be the least sensitive to the weights obtained for various network sizes and with different training algorithms; however, it was also the least accurate. The modified Connection Weight Approach was found to be the next most stable method for quantifying the importance of ANN inputs. Furthermore, it can be seen that the overall accuracy of both of the modified input importance measures was greater than that of the original methods.

Table 3.25 Overall evaluation of input importance measures.

Input Importance	RMSE		GSC	
Estimation Method	Mean	St. Dev.	Mean	St. Dev.
Connection Weight Approach	12.619	6.375	0.737	0.145
Garson's Method	15.718	***2.964***	0.650	***0.083***
Modified Connection Weight Approach	***5.966***	3.344	***0.894***	0.106
Modified Garson's Method	7.462	4.492	0.842	0.157

3.4.6 Evaluation of Best Models

The performance of the best models developed for modelling each synthetic data set was evaluated based on the "measured" and "true" training, testing and validation data. For data set I, the best model contained 1 hidden node and was trained with the SCE-UA algorithm. Scatter plots of the resulting model predictions versus the "measured" and "true" training, testing and validation data are shown in Figures 3.23 (a), (b) and (c), respectively, and the model performance results are summarised in Table 3.26 in comparison to the actual σ_y^2 and MAE values (shown in italics). It can be seen in Figure 3.23 that the

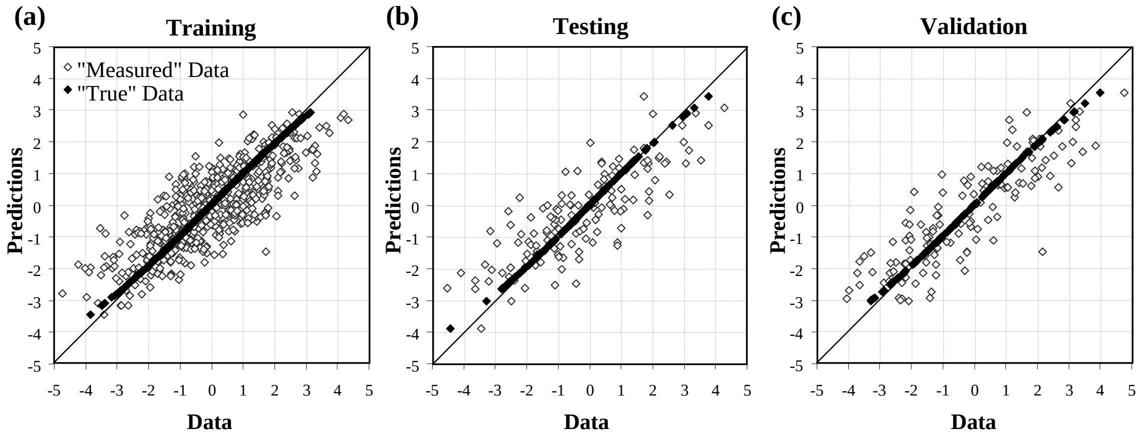

Figure 3.23 Scatter plots of the 1 hidden node ANN model predictions versus "measured" and "true" (a) training, (b) testing and (c) validation data for data set I.

Table 3.26 Performance of 1 hidden node ANN developed for modelling data set I.

Performance	"Measured" Data			"True" Data		
Measure	Train	Test	Validation	Train	Test	Validation
$\hat{\sigma}^2_{\mathbf{y}}$	0.899	1.102	1.020	0.005	0.010	0.010
MAE	0.764	0.849	0.814	0.057	0.072	0.075
RMSE	0.948	1.050	1.010	0.072	0.098	0.098
CE	0.635	0.671	0.703	0.997	0.996	0.996
Actual $\sigma^2_{\mathbf{y}}$	*0.900*	*1.112*	*1.035*	*0.000*	*0.000*	*0.000*
Actual MAE	*0.767*	*0.900*	*0.816*	*0.000*	*0.000*	*0.000*

model has predicted the "true" data very well, even though it was trained on the noisy "measured" data. However, it can also be seen that the high and low data values were slightly under- and overpredicted, respectively, for each data subset. Nevertheless, the results presented in Table 3.26 indicate that the model has good generalisability across each of the three subsets.

Shown in Figure 3.24 is a time series plot of the model predictions against the "measured" and "true" recombined training, testing and validation data, where it appears as though the model has obtained a near perfect fit to the "true" data, confirming the ability of the model to generalise to the underlying trend in the data.

The RI values of the 1 hidden node ANN inputs estimated using the modified connection weight method are presented in Table 3.27 in comparison to the corresponding PMI-based RI estimates. Given that both of these methods are approximations of the

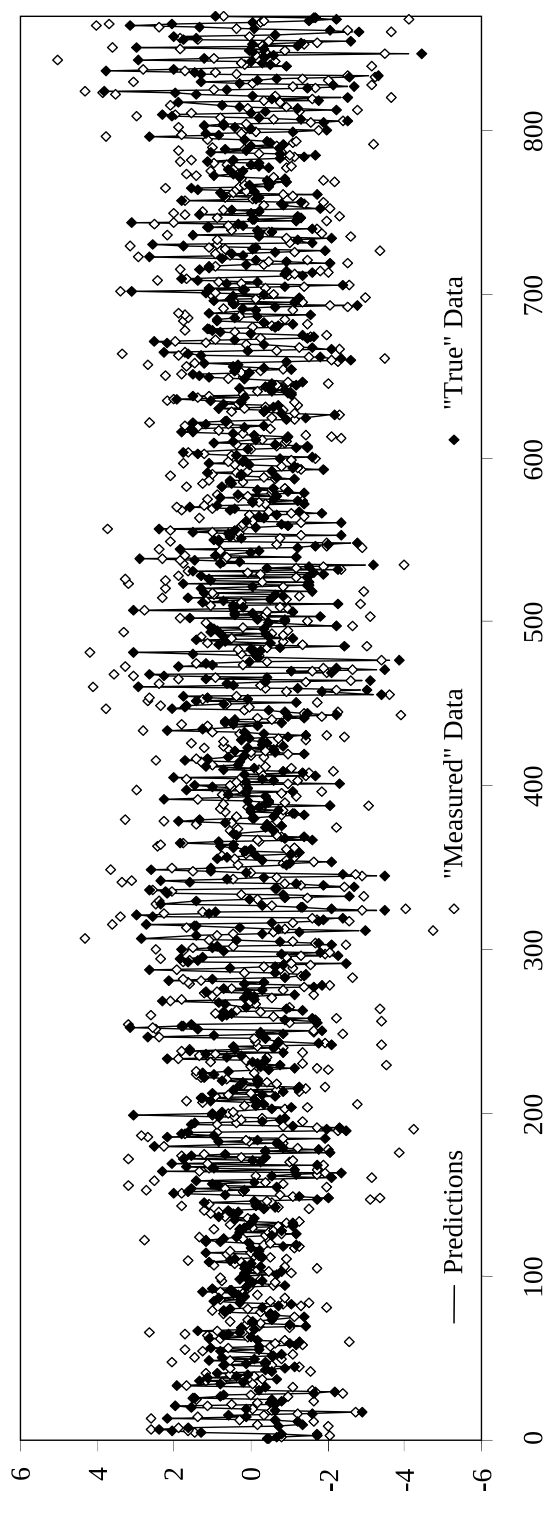

Figure 3.24 Time series plot of model predictions against the combined training/testing/validation “measured” and “true” data for data set I.

Table 3.27 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}
Modified connection weight	21.21	41.40	37.39
PMI-based	22.33	45.07	32.60

actual input-to-output relationships and there is general agreement between the values, it is considered that the model approximated the underlying relationship well. However, considering that the data are linear, it is acknowledged that an ANN without a hidden layer would probably have resulted in a better approximation

For data set II, the best model developed contained 3 hidden nodes and was trained with the SCE-UA algorithm. Scatter plots of the resulting model predictions versus the “measured” and “true” data are shown in Figure 3.25, while the model performance results are presented in Table 3.28. It can be seen in Figure 3.25 that a reasonably good fit to the “true” data was obtained using this model; however, the fit was not as good as that obtained for data set I, which can be seen by the larger amount of scatter about the straight lines. The results in Table 3.28 indicate that while the model has reasonable generalisability, it may have overfitted the training data slightly, as seen by comparing the actual σ_{y}^2 and MAE values to the corresponding estimated values for the “measured” training data subset. This may be the reason for the slightly worse fit to the underlying trend in the data, which is represented by the “true” data. Shown in Figure 3.26 is a time series plot of the model predictions against the “measured” and “true” data for the

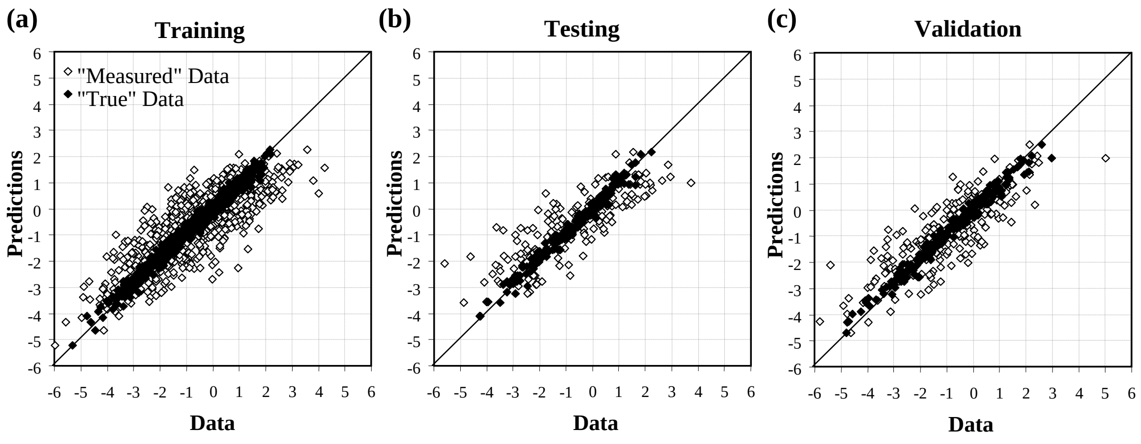

Figure 3.25 Scatter plots of the 3 hidden node ANN model predictions versus “measured” and “true” (a) training, (b) testing and (c) validation data for data set II.

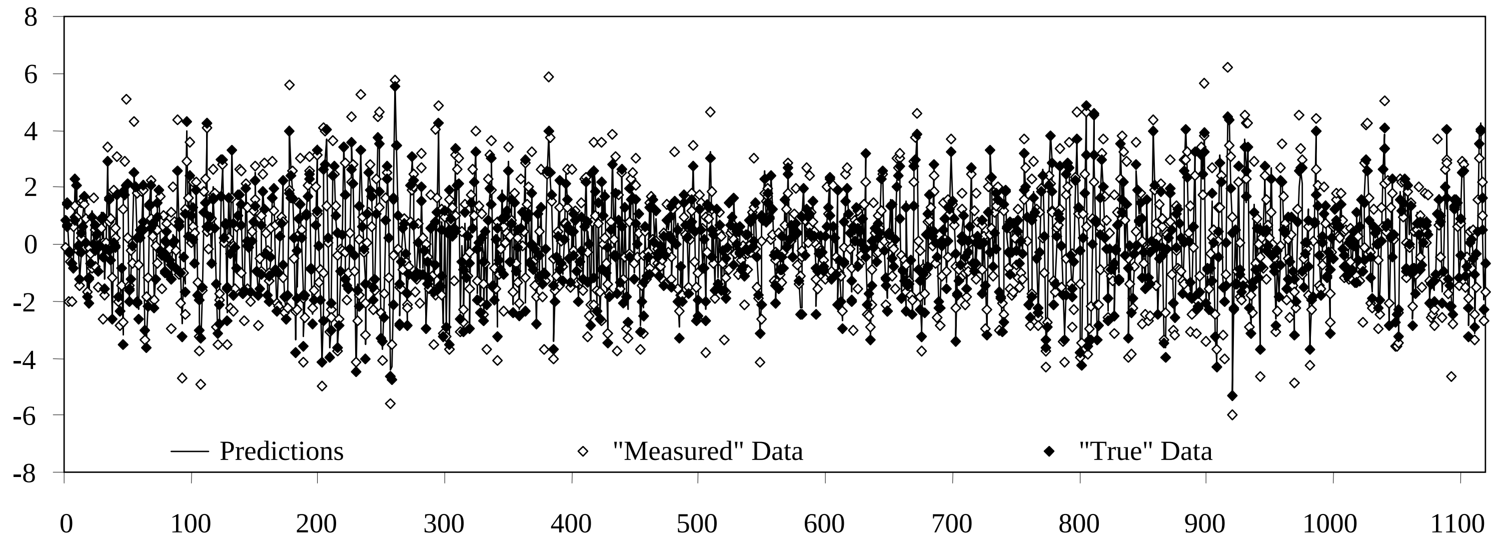

Figure 3.26 Time series plot of model predictions against the combined training/testing/validation “measured” and “true” data for data set II.

Table 3.28 Performance of 3 hidden node ANN developed for modelling data set II.

Performance	"Measured" Data			"True" Data		
Measure	Train	Test	Validation	Train	Test	Validation
$\hat{\sigma}^2_{\mathbf{y}}$	0.963	1.025	0.922	0.042	0.045	0.059
MAE	0.788	0.812	0.766	0.160	0.163	0.190
RMSE	0.981	1.013	0.960	0.205	0.211	0.243
CE	0.652	0.664	0.682	0.979	0.974	0.974
Actual $\sigma^2_{\mathbf{y}}$	*0.983*	*0.982*	*0.852*	*0.000*	*0.000*	*0.000*
Actual MAE	*0.797*	*0.793*	*0.733*	*0.000*	*0.000*	*0.000*

entire recombined training, testing and validation data set. It is apparent in this figure that the model was able to generalise reasonably well to the underlying trend, although a number of the "true" data points were not fitted to accurately, particularly the lower values, indicating that the function modelled was slightly incorrect. The RI values given in Table 3.29 confirm this.

Table 3.29 RI values (%) for the inputs of the 3 hidden node ANN developed for modelling data set II.

RI Estimation Method	y_{t-1}	y_{t-4}	y_{t-9}	x_t
Modified connection weight	27.31	37.86	32.74	2.09
PMI-based	21.62	36.13	30.82	11.44

It was inconclusive whether a 5 hidden node ANN or a 6 hidden node ANN was better for modelling data set III; therefore, the performances of both models, which were trained with the SCE-UA algorithm, were evaluated. Scatter plots of the 5 hidden node ANN model predictions versus the "measured" and "true" data are shown in Figure 3.27, with the corresponding model performance results presented in Table 3.30. It can be seen that a good fit to the data was obtained with relatively small error values given the values of the data, which are significantly higher than those for data sets I and II. Scatter plots of the 6 hidden node ANN model predictions versus the "measured" and "true" data are shown in Figure 3.28, with the corresponding model performance results presented in Table 3.31. In comparison to the results presented in Table 3.30 for the 5 hidden node ANN, it can be seen that a slightly better fit to both the "measured" and "true" data was obtained with the 6 hidden node model.

A plot of the 5 hidden node ANN model predictions against the "measured" and "true" data for the entire recombined training, testing and validation data set is shown in Fig-

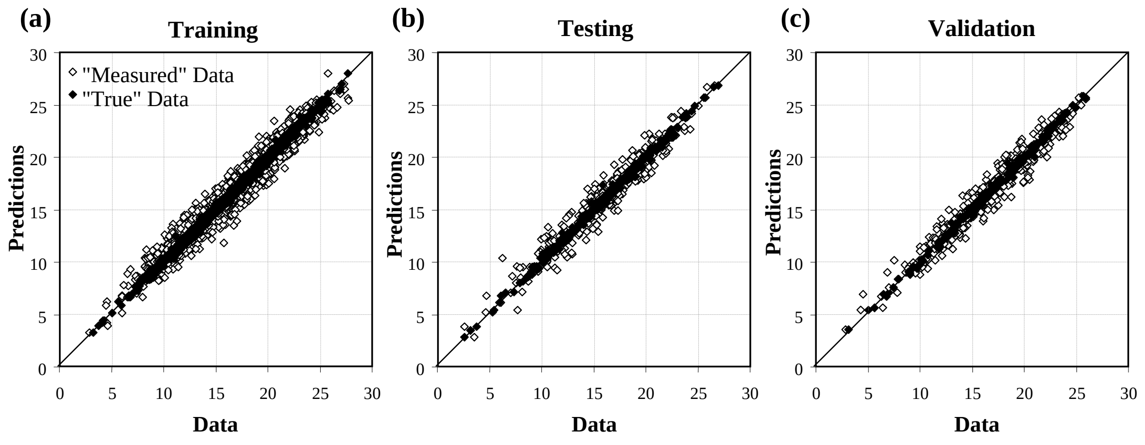

Figure 3.27 Scatter plots of the 5 hidden node ANN model predictions versus “measured” and “true” (a) training, (b) testing and (c) validation data for data set III.

Table 3.30 Performance of 5 hidden node ANN developed for modelling data set III.

Performance	“Measured” Data			“True” Data		
Measure	Train	Test	Validation	Train	Test	Validation
$\hat{\sigma}_{\mathbf{y}}^2$	1.067	1.170	0.924	0.058	0.073	0.068
MAE	0.815	0.856	0.768	0.187	0.205	0.201
RMSE	1.033	1.082	0.961	0.240	0.270	0.262
CE	0.942	0.939	0.947	0.997	0.996	0.996
Actual $\sigma_{\mathbf{y}}^2$	*1.048*	*1.096*	*0.869*	*0.000*	*0.000*	*0.000*
Actual MAE	*0.816*	*0.832*	*0.752*	*0.000*	*0.000*	*0.000*

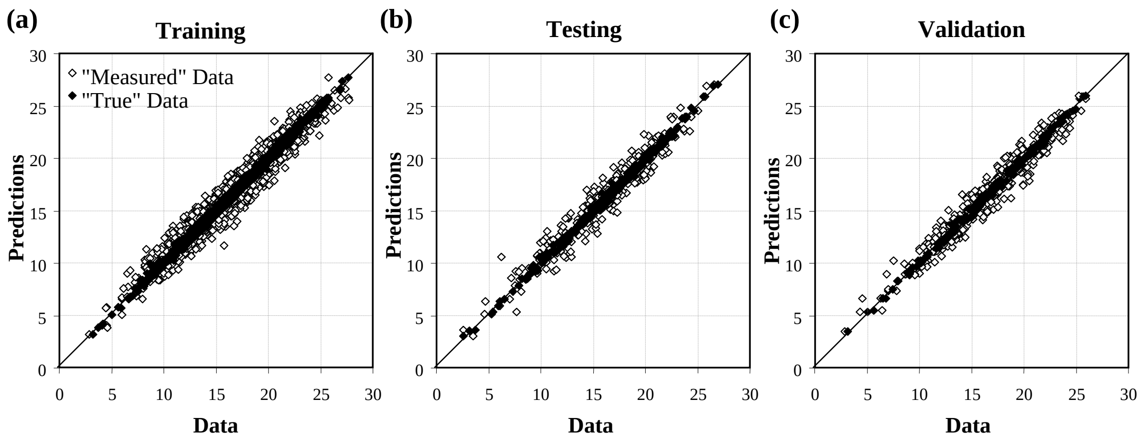

Figure 3.28 Scatter plots of the 6 hidden node ANN model predictions versus “measured” and “true” (a) training, (b) testing and (c) validation data for data set III.

Table 3.31 Performance of 6 hidden node ANN developed for modelling data set III.

Performance	"Measured" Data			"True" Data		
Measure	Train	Test	Validation	Train	Test	Validation
$\hat{\sigma}^2_{\mathbf{y}}$	1.050	1.149	0.909	0.051	0.058	0.059
MAE	0.811	0.856	0.769	0.175	0.184	0.185
RMSE	1.025	1.072	0.953	0.225	0.240	0.242
CE	0.942	0.940	0.948	0.997	0.997	0.997
Actual $\sigma^2_{\mathbf{y}}$	*1.048*	*1.096*	*0.869*	*0.000*	*0.000*	*0.000*
Actual MAE	*0.816*	*0.832*	*0.752*	*0.000*	*0.000*	*0.000*

ure 3.29. As can be seen, the model appears to have fit the data well; however, a number of the smaller data values were overpredicted. The output plot of the 6 hidden node ANN model's predictions was found to be almost identical to that shown in Figure 3.29 and is therefore not shown. The 6 hidden node ANN was also found to have overpredicted a number of the smaller data values.

The estimated RI values for the inputs of the 5 and 6 hidden node ANNs are given in Table 3.32 in comparison to the PMI-based estimates. It can be seen that the RI values of inputs x_1, x_2 and x_3 are quite different between the two models and in comparison to the PMI-based estimates. It is, however, difficult to determine which of the modelled relationships is more correct.

Table 3.32 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 - modified connection weight	12.25	13.27	23.98	32.97	17.53
6 hidden nodes - modified connection weight	23.81	19.93	0.09	36.20	19.98
PMI-based	20.07	22.65	10.65	30.80	15.83

3.4.7 Conclusions

3.4.7.1 Comparison of Training Algorithms

The results of the training algorithm comparison demonstrated that the SCE-UA algorithm is the most suitable training method for consistently obtaining good solutions for ANNs, given a range of different model specification conditions (e.g. underparameterised, overparameterised), initial weights, and data sets with different nonlinearity and noise characteristics. However, a shortcoming of this training method is the time required for

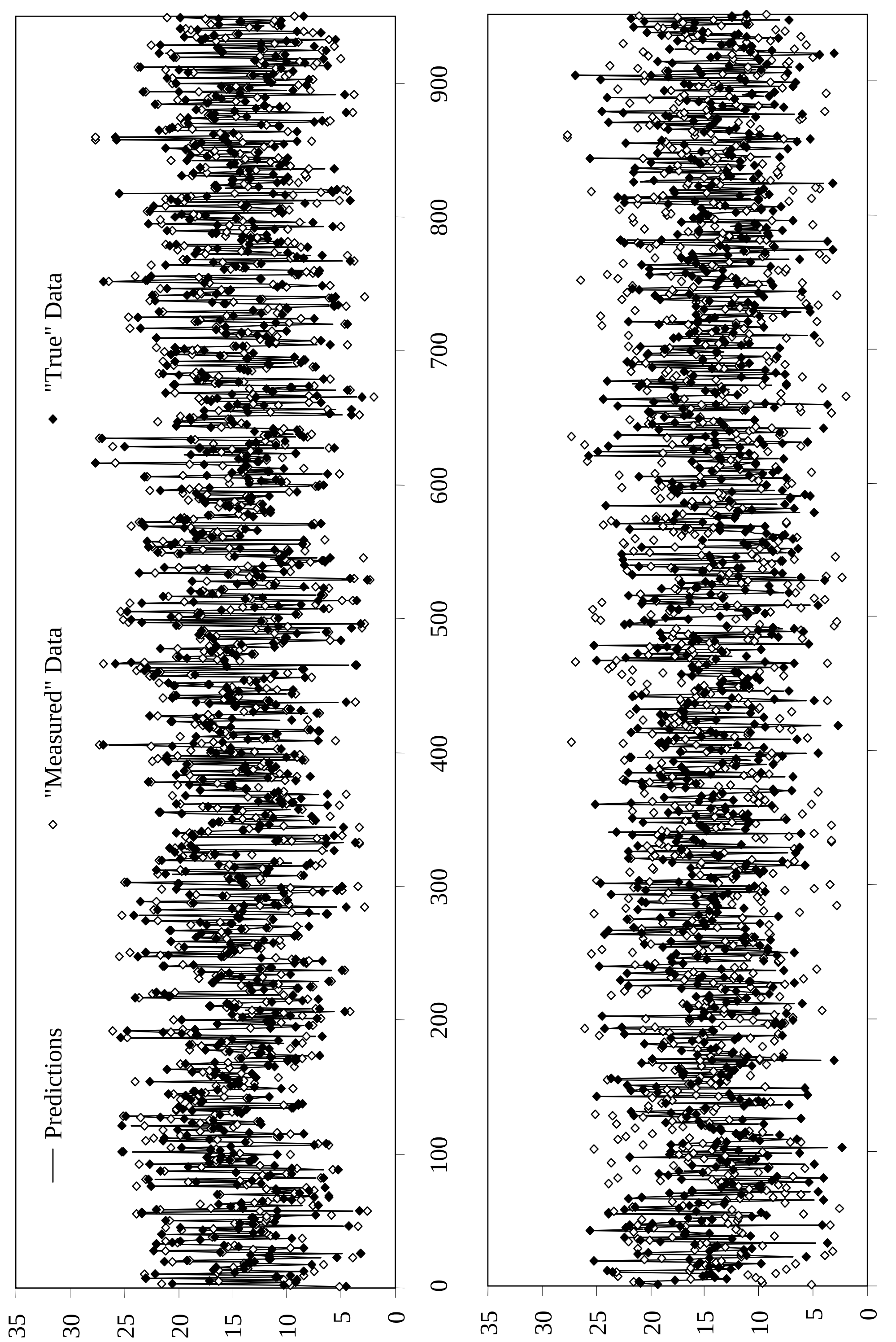

Figure 3.29 Output plot of the 5 hidden node ANN predictions against the combined training/testing/validation “measured” and “true” data for data set III.

training, particularly for larger sized networks. For high dimensional ANNs with a known and differentiable error function, it is unlikely that a randomised search method like the SCE-UA algorithm would beat the efficiency of a gradient-based search method such as backpropagation, even if the local search method were initialised several times to improve its robustness. Nevertheless, as the main aim of developing a state-of-the-art deterministic ANN development approach in this research was to provide the best comparison to the Bayesian ANN development approach introduced in the following chapter, the SCE-UA algorithm will be used to train the deterministic models developed in this research. To prevent excessively long training times for large networks, it is recommended that modifications to, or upper limits for, the algorithm parameters be considered when training ANNs containing more than approximately 40 weights (i.e. $d > 40$), which roughly corresponds to an 8 hidden node ANN for data set I, a 7 hidden node ANN for data set II and a 6 hidden node ANN for data set III, all of which took over an hour to train in comparison to approximately 10 minutes required by the BP algorithm and the GA.

3.4.7.2 Assessment of Model Section Criteria

It was found that both the in-sample BIC and the out-of-sample AIC criteria were the most suitable for selecting the appropriate number of hidden nodes in an ANN, with both criteria correctly selecting the optimal network size for modelling data sets I, II and III. It was also seen that whether training was stopped early or allowed to converge had little impact on the generalisability of the models selected using the in-sample BIC, suggesting that a test data set would not be required when this criterion is used to select the appropriate ANN configuration. However, it is considered that this may not always be the case in complex real-world problems and is an issue that requires further investigation. A limitation of both the in-sample BIC and the out-of-sample AIC is that they are deterministic and their results can vary depending on the solution obtained during training. In this investigation, only the best models developed for each network size using each training algorithm were considered and there was still some variation in both the in-sample BIC and out-of-sample AIC results. Therefore, if the optimum weights are not obtained for any given network size in a trial-and-error model selection process, such as the one carried out in this investigation, the use of deterministic BIC and AIC values may lead to the incorrect selection of the appropriate network size.

3.4.7.3 Assessment of Input Importance Measures

Overall, it was found that the modified Connection Weight Approach developed as part of this research was the most accurate method for quantifying the relative importance of ANN inputs of the methods investigated. It was also found that, on average, both of the modified methods considered (i.e. modified Connection Weight Approach and modified Garson's measure) were an improvement on the original input importance measures. However, similar to the model selection criteria, these methods were only applied to the best models developed for modelling the synthetic data sets and the resulting RI values still exhibited (sometimes substantial) variation. Although the modified Connection Weight Approach was found to be relatively robust to the different weights obtained for various network sizes using different training algorithms, there is still some danger in using such deterministic methods to quantify the relative importance of ANN inputs.