ELSEVIER

Environmental Modelling & Software 15 (2000) 101–124

Environmental Modelling & Software

www.elsevier.com/locate/envsoft

Neural networks for the prediction and forecasting of water resources variables: a review of modelling issues and applications

Holger R. Maier *, Graeme C. Dandy

Department of Civil and Environmental Engineering, The University of Adelaide, Adelaide, S.A. 5005 Australia

Received 4 November 1998; received in revised form 16 February 1999; accepted 5 March 1999

Abstract

Artificial Neural Networks (ANNs) are being used increasingly to predict and forecast water resources variables. In this paper, the steps that should be followed in the development of such models are outlined. These include the choice of performance criteria, the division and pre-processing of the available data, the determination of appropriate model inputs and network architecture, optimisation of the connection weights (training) and model validation. The options available to modellers at each of these steps are discussed and the issues that should be considered are highlighted. A review of 43 papers dealing with the use of neural network models for the prediction and forecasting of water resources variables is undertaken in terms of the modelling process adopted. In all but two of the papers reviewed, feedforward networks are used. The vast majority of these networks are trained using the backpropagation algorithm. Issues in relation to the optimal division of the available data, data pre-processing and the choice of appropriate model inputs are seldom considered. In addition, the process of choosing appropriate stopping criteria and optimising network geometry and internal network parameters is generally described poorly or carried out inadequately. All of the above factors can result in non-optimal model performance and an inability to draw meaningful comparisons between different models. Future research efforts should be directed towards the development of guidelines which assist with the development of ANN models and the choice of when ANNs should be used in preference to alternative approaches, the assessment of methods for extracting the knowledge that is contained in the connection weights of trained ANNs and the incorporation of uncertainty into ANN models.

Keywords: Artificial neural networks; Water resources; Forecasting; Prediction; Modelling process; Model development; Review

1. Introduction

In recent years, Artificial Neural Networks (ANNs) have become extremely popular for prediction and forecasting in a number of areas, including finance, power generation, medicine, water resources and environmental science. Although the concept of artificial neurons was first introduced in 1943 (McCulloch and Pitts, 1943), research into applications of ANNs has blossomed since the introduction of the backpropagation training algorithm for feedforward ANNs in 1986 (Rumelhart et al., 1986a). ANNs may thus be considered a fairly new tool in the field of prediction and forecasting. The rules governing traditional statistical models are seldom considered in the ANN model building process and "there is a tendency among users to throw a problem blindly at a neural network in the hope that it will formulate an acceptable solution…" (Flood and Kartam, 1994). In many applications, the model building process is described poorly, making it difficult to assess the optimality of the results obtained.

Recent studies indicate that consideration of statistical principles in the ANN model building process may improve model performance (e.g. Cheng and Titterington, 1994; Ripley, 1994; Sarle, 1994). Consequently, it is vital to adopt a systematic approach in the development of ANN models, taking into account factors such as data pre-processing, the determination of adequate model inputs and a suitable network architecture, parameter estimation (optimisation) and model validation (Maier and Dandy, 1999b). In addition, careful selection of a number of internal model parameters is required.

In Section 2, the similarities and differences between ANN and traditional statistical models are discussed.

* Corresponding author. E-mail: hmaier@civeng.adelaide.edu.au

1364-8152/00/$ - see front matter
PII: S1364-8152(99)00007-9

Next the steps that should be followed in the development of ANN prediction/forecasting models are outlined, the issues involved at each stage of the model development process are discussed and the available options are reviewed (Sections 3–9). In Section 10, 43 papers dealing with the prediction and forecasting of water resources variables are reviewed *in terms of the modelling process adopted*. Finally, conclusions and directions for further research are given (Section 11). Throughout the paper, in-depth descriptions of methodologies are not given, as they are available elsewhere. It is also assumed that readers are familiar with the basic concepts of neural networks (see Flood and Kartam, 1994; Hassoun, 1995; Maren et al., 1990; Masters, 1993; Rojas, 1996). The material covered is restricted primarily to feedforward networks with sigmoidal-type transfer functions, as these have been used almost exclusively for the prediction and forecasting of water resources variables (see Section 10). However, feedforward networks with radial basis transfer functions and recurrent networks have been proposed recently as possible alternatives, and will be discussed briefly.

2. ANNs and statistics

ANN modelling approaches have been embraced enthusiastically by practitioners in water resources, as they are perceived to overcome some of the difficulties associated with traditional statistical approaches. In the words of Sarle (1994), users of ANNs "…want their networks to be black boxes requiring no human intervention—data in, predictions out". More recently, researchers have examined ANN models from a statistical perspective (e.g. Cheng and Titterington, 1994; Hill et al., 1994; Ripley, 1994; Sarle, 1994; Warner and Misra, 1996; White, 1989). Such studies indicate that certain models obtained when ANN geometry, connectivity and parameters are changed are either equivalent, or very close to, existing statistical models. Consequently, some neural network models "…are not really new inasmuch as they represent variations on common statistical themes" (Bienenstock and Geman, 1994). Statistical models that can be expressed in neural network form include a number of regression, discriminant, density estimation and graphical interaction models such as simple linear regression, projection pursuit regression, polynomial regression, non-parametric regression, logistic regression, linear discriminant functions, classification trees, finite mixture models, kernel regression and smoothing splines (Cheng and Titterington, 1994; Sarle, 1994). In addition, there are neural-network approaches for computing quadratic discriminant rules, for calculating principal components and for approximating Bayesian probabilities (Cheng and Titterington, 1994). Sarle (1994) discusses how particular statistical models can be obtained by varying the transfer function and the number of hidden nodes in a neural network model. These relationships are summarised in Tables 1 and 2, respectively. It has also been suggested that certain ANN models are equivalent to time series models of the ARMA type (Hill et al., 1994). Connor et al. (1994) have shown that feedforward neural networks are a special case of non-linear autoregressive (NAR) models and that recurrent neural networks are a special case of non-linear autoregressive moving average (NARMA) models. Chon and Cohen (1997) have demonstrated the equivalence of ARMA and NARMA models with feedforward ANNs utilising polynomial transfer functions.

Table 1
Equivalent statistical models for neural network models with different transfer functions (after Sarle, 1994)

Activation function	Equivalent statistical model
Threshold	Linear discriminant function
Linear	Linear regression
Logistic	Logistic regression

Although ANN models are not significantly different from a number of standard statistical *models*, they are extremely valuable as they provide a flexible way of *implementing* them. Model complexity can be varied simply by altering the transfer function or network architecture. In addition, unlike some statistical models, ANN models can be extended easily from univariate to multivariate cases. However, as the number of different models that can be generated using ANNs is so vast, many have not yet been examined from a statistical perspective. As pointed out by White (1989), "…the field of statistics has much to gain from the connectionist literature. Analysing neural learning procedures poses a host of interesting theoretical and practical challenges for statistical method; all is not cut and dried".

Until recently, there has been little interaction between the neural network and statistical communities and ANN and statistical models have developed virtually independently. This has led to a number of differences in the *modelling approach*, even though the underlying *models* are similar. ANNs have their roots in artificial intelligence (AI) and were developed by engineers and com-

Table 2
Equivalent statistical models for neural network models with different numbers of hidden nodes (after Sarle, 1994)

No. of hidden nodes	Model type	Equivalent statistical model
Small	Parametric	Polynomial regression
Moderate	Quasi-parametric	Projection pursuit
Increase with sample size	Non-parametric sieve	Kernel regression Smoothing splines

puter scientists. This has resulted in a difference in the terminology used by statistical and ANN modellers, which can cause some confusion. A glossary of commonly used ANN terminologies and their statistical equivalents is given by Sarle (1994) and is summarised in Table 3. One of the major misunderstandings arising from the difference in terminology is that many researchers who apply ANNs to water resources problems claim that ANNs can 'learn from examples', and that this is one of the major advantages ANNs have over other methods. However, the 'learning' or 'training' phase in ANNs is no different than the parameter estimation phase in conventional statistical models. The reason this terminology is used in AI circles is to distinguish rule-based approaches, such as expert systems, from those that 'learn' from empirical examples.

Breiman (1994); Tibshirani (1994) discuss the differences between the modelling objectives and approach taken by statisticians and neural network practitioners. The major differences arise from the fact that the statisticians' primary objectives are to develop universal methodology and to achieve statistical optimality. In contrast, neural network practitioners are primarily concerned about prediction accuracy and finding methods that work. As a result, neural network practitioners generally tackle more complex problems, the dimensionality of the models tends to be much higher, and methodologies are hand tailored to particular applications.

The differences in the modelling approach between ANN and traditional statistical models, coupled with a lack of strict rules governing the development of the former, are probably the major reasons for the increased popularity of neural networks. Even though statistical models that are similar or equivalent to ANN models have been available for many years, practitioners in water resources have primarily used simple linear regression or time series models to obtain approximations of the relationships between variables. One reason for this is that the rules governing sophisticated statistical models have generally been considered to be too restrictive and to make it too difficult to utilise them for real-life applications. ANNs have placed such sophisticated models within the reach of practitioners. This does not mean that sound statistical principles should not be considered in the ANN model building process. In the contrary, they have a vital role to play in improving the performance of ANN models and in defining their areas of applicability. However, as pointed out by Tibshirani (1994), "…it is often better to get an approximate solution to a real problem than an exact solution to an oversimplified one".

Table 3
Glossary of ANN and statistical terminology (after Sarle, 1994)

ANN terminology	Statistical terminology
Input	Independent variable
Ouput	Predicted value
Training values	Dependent variables
Errors	Residuals
Training or learning	Estimation
Error function or cost function	Estimation criterion
Patterns or training pairs	Observations
Weights	Parameter estimates
Generalisation	Interpolation and extrapolation

3. Choice of performance criteria

At the beginning of the model building process, it is important to clearly define the criteria by which the performance of the model will be judged, as they can have a significant impact on the model architecture and weight optimisation techniques chosen. In most applications, performance criteria include one or more of the following: prediction accuracy, training speed and the time delay between the presentation of inputs and the reception of outputs for a trained network. The time delay between the presentation of network inputs and the reception of the corresponding outputs is a function of processing speed. For a particular computational platform, this is a function of the number of connection weights and the type of connection between them. In order to maximise processing speed, it is desirable to keep the number of connection weights as small as possible and the connections between them as simple as possible (e.g. processing speed in feedforward networks is greater than that in recurrent networks of equivalent size). The time taken to train a network is highly problem dependent. However, for a particular case study, training speed is a function of a number of factors. The optimisation method and its associated parameters have a major influence on convergence speed, as discussed in Section 8. The size of the training set can also play a significant role. Redundancy in the training data is undesirable as it slows down training, particularly when the entire training set is presented to the network between weight updates (see Section 8.1) (Rojas, 1996). Another factor that affects training speed is the size of the network. Larger networks generally require fewer weight updates to find an acceptable solution. However, the time taken to perform one weight update is increased (see Section 7.2).

Prediction accuracy is affected by the optimisation algorithm used (see Section 8). The method's ability to escape local minima in the error surface is of particular importance. A number of measures of prediction accuracy have been proposed in the literature (see Hecht-Nielsen, 1990; Lachtermacher and Fuller, 1994; Maier and Dandy, 1996b; Masters, 1993; Shukla et al., 1996; Tawfik et al., 1997; Xiao and Chandrasekar, 1997), which are generally calculated using data that have not

been utilised in the training process so that the model's generalisation ability can be assessed. Generalisation ability is defined as a model's ability to perform well on data that were not used to calibrate it (Cheng and Titterington, 1994) and is a function of the ratio of the number of training samples to the number of connection weights. If this ratio is too small, continued training can result in overfitting of the training data (Section 7.2). This problem is exacerbated by the presence of noise in the data. To minimise the overfitting problem, various techniques have been proposed for determining the smallest number of connection weights that will adequately represent the desired relationship (Section 7.2). Measures of prediction accuracy which take parsimony considerations into account include the Schwarz information criterion (SIC) (Schwarz, 1978), the network information criterion (NIC) (Murata et al., 1994) and the Vapnik-Chervonenkis (VC) dimension (Abu-Mostafa, 1989). Examples of the application of the above measures to neural networks are given by Gençay and Liu (1997); Doering et al. (1997); Faraway and Chatfield (1998). Generalisation ability is also affected by the degree with which the training and validation sets represent the population to be modelled and by the stopping criterion used (see Section 8.4).

4. Division of data

It is common practice to split the available data into two sub-sets; a training set and an independent validation set. Typically, ANNs are unable to extrapolate beyond the range of the data used for training (Flood and Kartam, 1994; Minns and Hall, 1996). Consequently, poor forecasts/predictions can be expected when the validation data contain values outside of the range of those used for training. It is also imperative that the training and validation sets are representative of the same population. When limited data are available, it might be difficult to assemble a representative validation set. One method which maximises utilisation of the available data is the holdout method (Masters, 1993). The basic idea is to withhold a small subset of the data for validation and to train the network on the remaining data. Once the generalisation ability of the trained network has been obtained with the aid of the validation set, a different subset of the data is withheld and the above process is repeated. Different subsets are withheld in turn, until the generalisation ability has been determined for all of the available data. Other methods for maximising the availability of data for training have also been proposed. Lachtermacher and Fuller (1994) generated a synthetic test set that possessed the same statistical properties as the training data. Maier (1995); Maier and Dandy (1998a) suggest using a subset of the data as a testing set in a trial phase to determine how long training should be carried out so that acceptable generalisation ability is achieved. The subset used for testing is then added to the remaining training data, and the whole data set is used to train the network for a fixed number of epochs, based on the results from the trial phase. A similar procedure was used by Khotanzad et al. (1997).

Cross-validation (Stone, 1974) is a technique that is used frequently in ANN modelling and has a significant impact on the way the available data are divided. It can be used to determine when to terminate training and to compare the generalisation ability of different models (Burden et al., 1997) (Sections 7.2.3 and 8.4). In cross-validation, an independent test set is used to assess the performance of the model at various stages of learning. As the validation set must not be used as part of the training process, a different, independent testing set is needed for the purposes of cross-validation. This means that the available data need to be divided into three subsets; a training set, a testing set and a validation set, which is very data intensive. The same applies to cases where network geometry or internal parameters are optimised by trial and error.

5. Data pre-processing

In any model development process, familiarity with the available data is of the utmost importance. ANN models are no exception (Kaastra and Boyd, 1995), and data pre-processing can have a significant effect on model performance. It is important to note that the available data need to be divided into their respective subsets (e.g. training, testing and validation) before any data pre-processing is carried out (Burden et al., 1997). Generally, different variables span different ranges. In order to ensure that all variables receive equal attention during the training process, they should be standardised. In addition, the variables have to be scaled in such a way as to be commensurate with the limits of the activation functions used in the output layer (see Minns and Hall, 1996). For example, as the outputs of the logistic transfer function are between 0 and 1, the data are generally scaled in the range 0.1–0.9 or 0.2–0.8. If the values are scaled to the extreme limits of the transfer function, the size of the weight updates is extremely small and flat-spots in training are likely to occur. It should be noted that when the transfer functions in the output layer are unbounded (e.g. linear), scaling is not strictly required (see Karunanithi et al., 1994). However, scaling to uniform ranges is still recommended (Masters, 1993).

In most traditional statistical models, the data have to be normally distributed before the model coefficients can be estimated efficiently. If this is not the case, suitable transformations to normality have to be found. It has been suggested in the literature that ANNs overcome this problem, as the probability distribution of the input data

does not have to be known (e.g. Burke and Ignizio, 1992). More recently, it has been pointed out that as the mean squared error function is generally used to optimise the connection weights in ANN models, the data need to be normally distributed in order to obtain optimal results (Fortin et al., 1997). However, this has not been confirmed by empirical trials, where the model fits were the same regardless of whether raw or transformed data were used (Faraway and Chatfield, 1998). Clearly, this issue requires further investigation.

Until recently, the issue of stationarity has been rarely considered in the development of ANN models. However, there are good reasons why the removal of deterministic components in the data (i.e. trends, variance, seasonal and cyclic components) should be considered (Masters, 1993). As discussed in Section 4, it is generally accepted that ANNs cannot extrapolate beyond the range of the training data. Consequently, it is unlikely that ANNs can account for trends and heteroscedasticity in the data. One way to deal with this problem is to remove any deterministic components using methods commonly used in time series modelling such as classical decomposition (Chatfield, 1975) or differencing (Box and Jenkins, 1976). Differencing has already been applied to neural network modelling of non-stationary time series (e.g. Chng et al., 1996). However, use of the classical decomposition model may be preferable, as differenced time series can possess infinite variance (Irvine and Eberhardt, 1992). Another way of dealing with trends in the data is to use an adaptive weight update strategy during on-line operation (Khotanzad et al., 1997). Forecasts of values beyond the range of the training data may also be obtained by increasing the maximum value in the data set by a factor in excess of one (e.g. 1.5 or 2) for scaling purposes (Lachtermacher and Fuller, 1994) or by using a clipped linear transfer function in the output layer (Karunanithi et al., 1994).

It has been suggested that the ability of ANNs to find non-linear patterns in data makes them well suited to dealing with time series with non-regular cyclic variation (Hansen and Nelson, 1997). Maier and Dandy (1996a) investigated the effect of input data with and without seasonal variation on the performance of ANN models. Their results indicate that ANNs have the ability to cater to irregular seasonal variation in the data with the aid of their hidden layer nodes. There is a rapidly increasing body of literature discussing the use of ANNs for time series analysis and associated issues. A detailed review of this literature is beyond the scope of this paper. Interested readers are referred to Chatfield (1993); Refenes et al. (1997); Masters (1993); Hill et al. (1994); Hansen and Nelson (1997); Faraway and Chatfield (1998); Gershenfeld and Weigend (1994).

6. Determination of model inputs

As in any prediction/forecasting model, the selection of appropriate model inputs is extremely important (Faraway and Chatfield, 1998; Kaastra and Boyd, 1995). However, in most ANN applications, little attention is given to this task. The main reason for this is that ANNs belong to the class of data driven approaches, whereas conventional statistical methods are model driven (Chakraborty et al., 1992). In the latter, the structure of the model has to be determined first, which is done with the aid of empirical or analytical approaches, before the unknown model parameters can be estimated. Data driven approaches, on the other hand, have the ability to determine which model inputs are critical, so there is no need for "...a priori rationalisation about relationships between variables..." (Lachtermacher and Fuller, 1994). However, presenting a large number of inputs to ANN models, and relying on the network to determine the critical model inputs, usually increases network size. This has a number of disadvantages, such as decreasing processing speed and increasing the amount of data required to estimate the connection weights efficiently (Lachtermacher and Fuller, 1994) (see Section 7.2). This is particularly true for complex problems, where the number of potential inputs is large, and where no *a priori* knowledge is available to suggest which inputs to include. The problem is exacerbated in time series applications, where appropriate lags have to be chosen for each of the input variables. Consequently, there are distinct advantages in using analytical techniques to help determine the inputs for multivariate ANN models.

Lachtermacher and Fuller (1994) developed a hybrid methodology for determining appropriate input lags for simple univariate ANN models. This method involves fitting a univariate Box–Jenkins model (Box and Jenkins, 1976) to the time series, which is used as a guide for determining the inputs to the ANN model. In multivariate cases, the first step is to choose appropriate input variables. The choice of input variables is generally based on *a priori* knowledge of causal variables in conjunction with inspections of time series plots of potential inputs and outputs. If the relationship to be modelled is less well understood, analytical techniques such as cross-correlation analysis or principal component analysis can be used (see Roadknight et al., 1997). A stepwise approach can also be used in which separate networks are trained for each input variable. The network performing best is then retained and the effect of adding each of the remaining inputs in turn is assessed. This process is repeated for three, four, five, etc. input variables, until the addition of extra variables does not result in a significant improvement in model performance (see Maier et al., 1998; Masters, 1993). The disadvantages of the latter approach are that it is computationally intensive and that it is unable to capture the importance of

certain combinations of variables that might be insignificant on their own (Masters, 1993).

Having chosen appropriate input variables, the next step is the determination of appropriate lags for each of these. Maier and Dandy (1997a) evaluated the suitability of the method of Haugh and Box (1977) and a neural network based approach for determining which lags of the input variables to include in multivariate ANN models. They found that both methods were suitable, although the neural network based approach was preferred, as it was quicker and simpler to use. Refenes et al. (1997) suggest a stepwise method for determining inputs for multivariate ANN models. The suggested procedure consists of two phases. In the first phase, ARMA models (Box and Jenkins, 1976) and stepwise multiple linear regression analysis are used to determine appropriate input variables as well as their associated lags based on linear relationships. In phase two, the effect of additional variables is assessed in an attempt to capture any non-linear residual dependencies. Methods for incorporating time structure into neural network inputs are also discussed by Narendra and Parthasarathy (1990). Rather than determining input lags explicitly, time structure in the training data can also be taken into account implicitly by using recurrent networks (see Section 7.1).

7. Determination of network architecture

Network architecture determines the number of connection weights (free parameters) and the way information flows through the network. Determination of an appropriate network architecture is one of the most important, but also one of the most difficult, tasks in the model building process.

7.1. Type of connection and degree of connectivity

Traditionally, feedforward networks, where nodes in one layer are only connected to nodes in the next layer, have been used for prediction and forecasting applications. However, recently, recurrent networks, where nodes in one layer can be connected to nodes in the next layer, the previous layer, the same layer and even to themselves, have been proposed as alternatives (Warner and Misra, 1996). It follows that feedforward networks are special cases of recurrent networks. Feedforward networks require dynamic systems to be treated explicitly (Gençay and Liu, 1997; Krishnapura and Jutan, 1997). This is achieved by including lagged inputs (Section 6). In contrast, recurrent networks can model dynamical properties implicitly (Gençay and Liu, 1997; Krishnapura and Jutan, 1997). This is achieved by using the "…feedback connections to store representations of recent input events in the form of activations (short-term memory, as opposed to long-term memory embodied by slowly changing weights)" (Hochreiter and Schmidhuber, 1997). However, it has been shown that recurrent networks have difficulties in capturing long-term dependencies (i.e. when inputs at high lags have a significant effect on network outputs) (Lin et al., 1996). The inclusion of inputs at explicit time lags, resulting in NARX recurrent networks, has been found to considerably improve performance in such cases (Lin et al., 1996; Siegelmann et al., 1997).

Connor et al. (1994) have shown that recurrent networks are able to cater to moving average (MA) components, whereas feedforward networks cannot. Consequently, recurrent networks have advantages over feedforward networks similar to those that NARMA models have over NAR models. However, it should be noted that most processes can be adequately described by NAR models. The advantage of using a mixed model (i.e. one that has AR and MA components) is that the same task can be achieved more parsimoniously. Another potential advantage of recurrent networks is that they perform better when the data are noisy (see Gençay and Liu, 1997). Examples of different types of recurrent networks that have been used for time series applications include those proposed by Elman (1990); Williams and Zipser (1989); Krishnapura and Jutan (1997).

Despite the potential benefits of using recurrent networks for time series applications, the remaining discussion in this paper will be restricted to feedforward networks as (i) they have been found to perform well in comparison with recurrent networks in many practical applications (e.g. Khotanzad et al., 1997), (ii) they have been used almost exclusively for the prediction and forecasting of water resources variables (Section 10), (iii) their processing speed "…is among the fastest of all models currently in use" (Masters, 1993) and (iv) recurrent networks "…do not provide clear practical advantages over … feedforward nets with limited time windows" (Hochreiter and Schmidhuber, 1997).

The degree of connectivity (i.e. whether the nodes in a network are fully or partially connected) has an impact on the number of connection weights that need to be optimised. In most instances, fully connected networks are utilised. However, for some studies, partial connectivity has been used (e.g. Zhu et al., 1994).

7.2. Geometry

Network geometry determines the number of connection weights and how these are arranged. This is generally done by fixing the number of hidden layers and choosing the number of nodes in each of these. However, it has also been suggested that it might be best to fix the number of nodes, rather than the number of hidden layers, and to optimise the connections between the nodes, as well as the connection weights associated with each connection, during training (Kumar, 1993).

At the outset, it is worthwhile to consider some of the relative properties of smaller and larger networks. Smaller networks usually have better generalisation ability (Castellano et al., 1997), require fewer physical resources (e.g. require less storage space), have higher processing speed (during training and testing), can be implemented on hardware more easily and economically (Bebis and Georgiopoulos, 1994) and make rule extraction simpler (Towell et al., 1991). On the other hand, the error surface of smaller networks is more complicated and contains more local minima (Bebis and Georgiopoulos, 1994). The advantages of larger networks are that they tend to learn quickly (in terms of number of training cycles) (Plaut and Hinton, 1987), can form adequately complex decision regions (Lippmann, 1987) and have an increased ability to avoid local minima in the error surface (Rumelhart et al., 1986b). However, their computational costs are high and they generally require a large number of training samples to achieve good generalisation ability (Bebis and Georgiopoulos, 1994).

It has been shown that ANNs with one hidden layer can approximate any function, given that sufficient degrees of freedom (i.e. connection weights) are provided (e.g. Hornik et al., 1989). However, in practice many functions are difficult to approximate with one hidden layer, requiring a prohibitive number of hidden layer nodes (Cheng and Titterington, 1994; Flood and Kartam, 1994). The use of more than one hidden layer provides greater flexibility and enables approximation of complex functions with fewer connection weights in many situations (Flood and Kartam, 1994; Ripley, 1994; Sarle, 1994; Tamura and Tateishi, 1997). Flood and Kartam (1994) suggest using two hidden layers as a starting point. However, it must be stressed that optimal network geometry is highly problem dependent.

The number of nodes in the input layer is fixed by the number of model inputs, whereas the number of nodes in the output layer equals the number of model outputs. The critical aspect is the choice of the number of nodes in the hidden layers and hence the number of connection weights. The importance of striking a balance between having sufficient free parameters (weights) to enable representation of the function to be approximated and having too many free parameters, which can result in overtraining, is well known and has been discussed widely in the literature (e.g. Maren et al., 1990; Rojas, 1996). Although the aim of training is to reduce the error function as much as possible, when dealing with noisy data, reducing it beyond a certain point might lead to overtraining. An overtrained network has overfitted the training data, which means that it has learnt the idiosyncrasies in the training set and has lost its ability to generalise. This is characterised by a continued decrease in the training error while the error obtained from an independent test set increases (provided that both data sets are representative of the population to be modelled).

It is important to note, however, that it is the relationship between the number of training samples and the number of adjustable connection weights that is crucial. In order to ensure good generalisation ability, a number of empirical relationships between the number of training samples and the number of connection weights have been suggested in the literature. Some of these are based on the rule of thumb that the number of weights should not exceed the number of training samples (e.g. Rogers and Dowla, 1994), others are based on the rule that the ratio of the number of training samples to the number of connection weights should be 2 to 1 (Masters, 1993) or 10 to 1 (e.g. Weigend et al., 1990). Amari et al. (1997) suggest that overfitting does not occur if the number of training samples is at least 30 times the number of free parameters. However, network geometry is generally highly problem dependent and the above guidelines do not ensure optimal network geometry, where optimality is defined as the smallest network that adequately captures the relationship in the training data. In addition, there is quite a high variability in the number of nodes suggested by the various rules. Traditionally, optimal network geometries have been found by trial and error. More recently, a number of systematic approaches for determining optimal network geometry have been proposed, including pruning and constructive algorithms (see Bebis and Georgiopoulos, 1994).

7.2.1. Pruning algorithms

The basic idea of pruning algorithms is to start with a network that is 'large enough' to capture the desired input–output relationship and to subsequently remove or disable unnecessary weights and/or nodes. Initial network size can be calculated with the aid of theoretical upper bounds suggested in the literature (e.g. Hecht-Nielsen, 1987; Huang and Huang, 1991). Pruning algorithms have been proposed by many authors, including Sietsma and Dow (1991); Karnin (1990); Chung and Lee (1992); Setiono (1997). A review of pruning algorithms is given by Reed (1993). Some of the above algorithms (e.g. Setiono, 1997) rely on regularisation techniques to reduce the number of model parameters by eliminating or disabling some of the connection weights. This is achieved by adding a term to the error function which penalises large networks. However, the number of hidden nodes is not altered, which have to be specified in advance (Kwok and Yeung, 1997a). Different techniques are used to determine how to best select the units to be deleted and how to continue the training process once a node or connection has been removed so that the overall network behaviour is preserved. One of the shortcomings of the majority of pruning algorithms is that they are sensitive to internal parameters. However, recent approaches (e.g. Castellano et al., 1997; Prechelt, 1997)

overcome this problem, as they do not require user-defined parameters. Another criticism levelled at pruning algorithms is that they may have difficulties finding the 'smallest' network, as there are generally many 'medium-size' networks that perform adequately (Bebis and Georgiopoulos, 1994). In addition, they are usually restricted to networks with one hidden layer (Kumar, 1993).

7.2.2. Constructive algorithms

Constructive algorithms approach the problem of optimising the number of hidden layer nodes from the opposite direction to pruning algorithms. The smallest possible network (i.e. one without hidden layers) is used at the start of training. Hidden layer nodes and connections are then added one at a time in an attempt to improve model performance. The major issues that need to be addressed in constructive algorithms include the way additional nodes are connected to the existing network, how to determine optimal connection weights (new and existing) once a node has been added and when to stop the addition of hidden nodes (Kwok and Yeung, 1997b). Different algorithms approach the above problems in different ways. Some methods require all additional nodes to be in the same layer, whereas others construct networks with multiple hidden layers (Kwok and Yeung, 1997a). Examples of constructive algorithms are those proposed by Hirose et al. (1991); Setiono and Hui (1995); Fahlman and Lebiere (1990); Chen et al. (1997). A review of constructive algorithms is given by Kwok and Yeung (1997a).

7.2.3. Problems, improvements and alternatives

The automatic approaches discussed above are not without their problems. For example, decisions on whether the addition or removal of a node has an impact on the performance of the network is generally assessed using the training data, which does not give an accurate indication of the generalisation ability of the network (Doering et al., 1997). However, this problem can be overcome by using some of the measures of generalisation ability discussed in Section 3. In fact, cross-validation has been used widely in a 'manual' constructive approach in an attempt to determine the optimal number of hidden layer nodes (e.g. Braddock et al., 1997). Another disadvantage of pruning and constructive approaches is that networks generally have to be trained a number of times (i.e. each time a hidden node is added or deleted) (Kwok and Yeung, 1997b), although some make use of the weights that have been optimised previously (e.g. Fahlman and Lebiere, 1990). An alternative approach is to use a network that is certain to have sufficient capacity to approximate the desired relationship and to stop training early (Amari et al., 1997; Hassoun, 1995; Hecht-Nielsen, 1990). This approach falls into the category of 'non-convergent' methods (e.g. Finnhoff et al., 1993) and does not require optimisation of the network geometry. Instead, cross-validation is used to stop training before overfitting occurs (see Section 8.4).

It has also been suggested that the methods discussed in Sections 7.2.1 and 7.2.2 are susceptible to becoming trapped in structural local optima (Angeline et al., 1994). In addition, they "only investigate restricted topological subsets rather than the complete class of network architectures" (Angeline et al., 1994). A number of algorithms based on Fogel's evolutionary programming (Fogel et al., 1966) have been proposed to overcome these problems (e.g. Angeline et al., 1994; Yao and Liu, 1997). A comprehensive review of the use of evolutionary algorithms in neural networks is given by Yao (1993). Genetic algorithms provide alternatives to evolutionary algorithms and have also been used successfully to determine optimal network architectures (e.g. Miller et al., 1989). Doering et al. (1997) have proposed using the A*-Algorithm (Nilsson, 1980) for the same task.

8. Optimisation (training)

The process of optimising the connection weights is known as 'training' or 'learning'. This is equivalent to the parameter estimation phase in conventional statistical models. The aim is to find a global solution to what is typically a highly non-linear optimisation problem (White, 1989). Consequently, the theory of non-linear optimisation is applicable to the training of feedforward networks (Battiti, 1992). The suitability of a particular method is generally a compromise between computation cost and performance (Parisi et al., 1996). The error function, E, most commonly used is the mean squared error (MSE) function (Warner and Misra, 1996). In many cases there are practical difficulties in optimising E, as the error surface is complicated with many local minima (Cheng and Titterington, 1994). Optimisation can be carried out using local or global methods. Local methods fall into two major categories: first-order and second-order methods. First-order methods are based on a linear model (gradient descent) whereas second-order models are based on a quadratic model (e.g. Newton's method) (Battiti, 1992). In both cases, iterative techniques are used to minimise the error function. The weight update equation takes the following general form (Parisi et al., 1996):

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \gamma_n \mathbf{d}_n \quad (1)$$

where $\mathbf{w}_n$ is the vector of connection weights, γ_n is the step size, $\mathbf{d}_n$ is a vector defining the direction of descent and the subscript n denotes the iteration number. The essential difference between the various algorithms is the choice of $\mathbf{d}_n$, which determines the convergence rate and computational complexity.

8.1. First-order local methods

First-order local methods are based on the method of steepest (or gradient) descent, in which the descent direction, **d**, is equal to the negative of the gradient of the error. The resulting weight update equation is given by (Parisi et al., 1996):

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \gamma_n \nabla_{\mathbf{w}_n} E \quad (2)$$

Error backpropagation (e.g. Rumelhart et al., 1986a) is by far the most widely used algorithm for optimising feedforward ANNs, and is based on the method of steepest descent. Traditionally, a fixed step size, γ, is used throughout the training process, resulting in a form of stochastic approximation. In fact, "…the method of backpropagation can be viewed as an application of the Robbins and Monro (1951) stochastic approximation procedure for solving the first-order conditions for a non-linear least-squares regression problem" (White, 1989).

In order to optimise the performance of feedforward networks trained with the backpropagation algorithm, it is essential to have a good understanding of the impact that step size has on training (Dai and Macbeth, 1997; Maier and Dandy, 1998a). If the size of the steps taken in weight space is too small, training is slow. In addition, the likelihood of the network becoming trapped in a local minimum in the error surface is increased. At the same time, when the steps taken are too large, the network can fall into oscillatory traps (Maier and Dandy, 1998a; Rojas, 1996). As the absolute step sizes which result in the two types of behaviour discussed above are highly problem dependent, finding a step size that will balance high learning speeds and minimisation of the risk of divergence is difficult. Traditionally, a trial-and-error approach has been used to optimise step size. More recently, algorithms that adapt step sizes during training have been introduced (see Section 8.1.2).

It is vital to note that the size of the steps taken in weight space during training is a function of a number of internal network parameters including the learning rate, momentum value, error function, epoch size and gain of the transfer function (Maier and Dandy, 1998a). Consequently, the *same* step size can be obtained by using different combinations of the above parameters. As the choice of appropriate internal parameters has a major impact on the performance of the backpropagation algorithm (Dai and Macbeth, 1997), and the latter is used in the majority of applications of ANNs in water resources (Section 10), the impact of various model parameters is discussed below. Empirical studies of the effects of internal parameters on backpropagation training are given by Maier and Dandy (1998a); Maier and Dandy, (1998b).

8.1.1. Internal parameters of the backpropagation algorithm

8.1.1.1. Initial weights As with all local methods, the results obtained when the backpropagation algorithm is used are sensitive to the initial conditions. Generally, the weights are initialised to zero-mean random values. Care must be taken to chose adequate upper and lower bounds $\{-\alpha, \alpha\}$ for the weights. If the value of α is too small, training may be paralysed. On the other hand, if α is too large, premature saturation of the nodes may occur, which in turn will slow down training and result in the cessation of training at suboptimal levels (e.g. Lee and Shen, 1991). The results of empirical studies (e.g. Wessels and Barnard, 1992) indicate that there is no single optimum value of α, but a large range for which similar performance is obtained. Wessels and Barnard (1992) propose setting the initial weights of node i to a value of order $1/\sqrt{f_i}$, where f_i is the number of inputs for node i. However, this relationship is not universal (Rojas, 1996) and Faraway and Chatfield (1998) suggest that a number of different sets of random starting values should be used to see whether consistent results are obtained.

8.1.1.2. Transfer (activation) function The transfer functions that are most commonly used are sigmoidal-type functions such as the logistic and hyperbolic tangent functions. However, other transfer functions may be used as long as they are differentiable. In an empirical study, Moody and Yarvin (1992) compared the performance of logistic, polynomial, rational function (ratios of polynomials) and Fourier series (sums of cosines) transfer functions on datasets containing varying degrees of noise and non-linearities. They found that the non-sigmoidal transfer functions performed best when the data were noiseless and contained highly non-linear relationships. When the data were noisy and contained mildly non-linear relationships, the performance of the polynomial transfer function was inferior while the performance of the other transfer functions was comparable. Kalman and Kwasny (1992) argue that the hyperbolic tangent transfer function should be used, which is in agreement with the empirical results obtained by Maier and Dandy (1998a). Another option is to use a radial basis transfer function. However, radial basis function networks operate quite differently from feedforward networks with polynomial or sigmoidal-type transfer functions. The use of radial basis function networks is a research area in itself and is beyond the scope of this paper. Interested readers are referred to papers by Broomhead and Lowe (1988); Chng et al. (1996); Heiss and Kampl (1996).

Generally, the same transfer function is used in all layers. However, as discussed in Section 5, using sigmoidal-type transfer functions in the hidden layers and linear transfer functions in the output layer can be an advantage when it is necessary to extrapolate beyond the

range of the training data (see Kaastra and Boyd, 1995; Karunanithi et al., 1994). It should also be noted that the type of transfer function used affects the size of the steps taken in weight space, as weight updates are proportional to the derivative of the transfer function (see Maier and Dandy, 1998a).

8.1.1.3. Epoch size The epoch size is equal to the number of training samples presented to the network between weight updates. If the epoch size is equal to 1, the network is said to operate in on-line mode. If the epoch size is equal to the size of the training set, the network is said to operate in batch mode. In many applications, batch mode is the preferred option, as it forces the search to move in the direction of the true gradient at each weight update. However, several researchers (e.g. Breiman, 1994; Hassoun, 1995) suggest using the on-line mode, as it requires less storage and "…makes the search path in the weight space stochastic… which allows for a wider exploration of the search space and, potentially, leads to better quality solutions" (Hassoun, 1995).

8.1.1.4. Error function The error function is the function that is minimised during training. The mean squared error (MSE) function is most commonly used, but other error functions have also been proposed (e.g. Solla et al., 1988). The advantages of using the MSE include that it is calculated easily, that it penalises large errors, that its partial derivative with respect to the weights can be calculated easily and that it lies close to the heart of the normal distribution (Masters, 1993). However, in order to obtain optimal results, the errors should be independently and normally distributed, which is not the case when the training data contain outliers. Liano (1996) introduced the least mean log squares (LMLS) method to overcome this problem.

8.1.1.5. Learning rate The learning rate is directly proportional to the size of the steps taken in weight space. However, it is worthwhile re-iterating that learning rate is only one of the parameters that affect the size of the steps taken in weight space. Traditionally, learning rates remain fixed during training (Warner and Misra, 1996) and optimal learning rates are determined by trial and error. Guidelines for appropriate learning rates have been proposed for single-layer networks (e.g. Bingham, 1988), but it is difficult to extend these guidelines to the multilayer case (Battiti, 1992). Many heuristics have been proposed which adapt the learning rate, and hence the size of the steps taken in weight space, as training progresses based on the shape of the error surface (see Section 8.1.2).

8.1.1.6. Momentum The momentum term may be considered to increase the effective step size in shallow regions of the error surface (Hassoun, 1995) and can speed up the training process by several orders of magnitude (Masters, 1993). It should be noted that the momentum term must be less than 1.0 for convergence (Dai and Macbeth, 1997).

8.1.2. Modifications to the backpropagation algorithm

Although backpropagation provides a computationally efficient method for changing the weights in a feedforward network and networks trained with this algorithm have been applied successfully to solve some difficult and diverse problems (Hassoun, 1995), there are certain problems associated with it. Some of these, and possible ways of overcoming them, are discussed below. Like all local methods, networks trained with the backpropagation algorithm are sensitive to initial conditions and susceptible to local minima in the error surface. One way to increase the likelihood of obtaining near-optimum local minima is to train a number of networks, starting with different initial weights. However, this is very time consuming. Alternatively, the on-line training mode can be used to help the network to escape local minima. Other methods include the addition of random noise to the connection weights (Von Lehman et al., 1988) or the addition of noise to the model inputs (Sietsma and Dow, 1988).

Another problem is that nodes with sigmoidal-type transfer functions may suffer from premature convergence to flat spots (Vitela and Reifman, 1997). If node activations are large, nodal outputs will be very close to their extreme values. This is termed saturation and slows down training as the size of the steps taken in weight space during training is very small. The mechanisms that cause premature saturation are discussed in detail by Vitela and Reifman (1997). A number of ways of overcoming the flat spot problem, and hence increasing training speed, have been proposed in the literature, including adjustment of the transfer function so that it never drops below a pre-defined level (Rojas, 1996), addition of a small, constant value to the derivative of the transfer function (Rojas, 1996), use of modified transfer functions (e.g. Yang and Yu, 1993), dynamic adjustment of the slope of the transfer function for individual nodes (Vitela and Reifman, 1993) and temporary modification of the momentum value (Vitela and Reifman, 1997).

The fact that optimal network parameters have to be found by trial and error is also a major concern (Dai and Macbeth, 1997; Sarle, 1994). A number of heuristics have been proposed which dynamically adapt the learning rate, and in some cases the momentum value, as training progresses. The majority of these are based on the principle of increasing the size of the steps taken in weight space when successive weight updates result in a reduction of the error or are in the same direction (i.e. when the algorithm continues to move down the error surface) and decreasing it when an increase in the error

occurs or when the steps are in opposing directions over consecutive iterations (i.e. when a valley in the error function is being jumped over). The methods proposed by Lapedes and Farber (1986); Battiti (1989); Jacobs (1988) fall in the above category. The major differences between the algorithms include the information used to modify the step size, the parameters that are modified (i.e. learning rate, momentum) and whether the parameters are modified globally or individually for each node. With some methods (e.g. Jacobs, 1988), parameters have to be selected that determine the rate at which learning rate adjustments are made. Empirical studies show that optimisation of these parameters can be highly problem dependent (Maier and Dandy, 1998a). Masters (1993) comments that the above approaches "…heap more empirically derived tweaking on top of an already largely empirical algorithm", which "…may lead to overadjustment of the weights, resulting in dramatic divergence" (Yu and Chen, 1997). Other approaches include the so-called search-then-converge schedules (e.g. Darken and Moody, 1990), algorithms that make use of second-order information (e.g. Fahlman, 1988) and a novel approach that dynamically optimises learning rate and momentum (Yu and Chen, 1997). However, despite the increased training speed the above methods offer, their rate of convergence is still slow, being linear at best (Parisi et al., 1996). In order to achieve superlinear convergence speed, second-order methods can be used. Reviews of second-order algorithms are given by Battiti (1992); Golden (1996).

8.2. Second-order local methods

The weight update equation for the classical Newton algorithm is given by (Parisi et al., 1996):

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \mathbf{H}^{-1}\nabla_{\mathbf{w}_n} E \tag{3}$$

where $\mathbf{H}^{-1}$ is the inverse of the Hessian matrix. The fact that the inverse of the Hessian has to be calculated at each iteration step requires space of order k^2 and time of order k^3, where k is the number of free parameters (Kwok and Yeung, 1997a). In contrast, the memory/computational requirement of first-order methods is $O(k)$. This places serious restrictions on the use of Newton's method, as it is only applicable to networks with a limited number of weights (< 100) (Battiti, 1992). Another problem with the classical Newton method is that the inverse of the Hessian is not guaranteed to be positive definite, which means that the algorithm may move 'uphill' (Golden, 1996).

The Levenberg–Marquardt (LM) modification of the classical Newton algorithm (see Golden, 1996) overcomes one of the problems of the classical Newton algorithm, as it guarantees that the Hessian is positive definite (Parisi et al., 1996). However, the computation/memory requirements are still $O(k^2)$. The LM algorithm may be considered to be a hybrid between the classical Newton and steepest descent algorithms. When far away from a local minimum, the algorithm's behaviour is similar to that of gradient descent methods. However, in the vicinity of a local minimum, it has a convergence rate of order two (Golden, 1996). This enables it to escape local minima in the error surface. The quasi-Newton approach proposed by Shanno (1978) overcomes both problems associated with the classical Newton algorithm while maintaining a convergence rate of order two. The computation/memory requirements are reduced to $O(k)$ by using an approximation of the inverse of the Hessian. This approximation also has the property of positive definiteness, avoiding the problem of 'uphill' movement (Golden, 1996). One potential problem with the Shanno algorithm is that it cannot escape local minima in the error surface, and may thus converge to a sub-optimal solution. Conjugate gradient methods (e.g. Møller, 1993) may be viewed as approximations to the Shanno algorithm (Shanno, 1978). However, "…the Shanno algorithm is expected to converge more rapidly to a nearby strict local minimum, take fewer uphill steps, and have greater numerical robustness than the generic conjugate gradient algorithm" (Golden, 1996).

A number of other algorithms have been proposed, which are roughly based on second-order methods (Yu and Chen, 1997). These include the extended Kalman learning algorithm (Singhal and Wu, 1989) and the nonlinear recursive least-squares learning algorithm and its variations (Kollias and Anastrassiou, 1989). Osowski et al. (1996) applied the efficient, limited memory BFGS quasi-Newton method (Gill et al., 1981) to feedforward networks. At present, the search for faster, more efficient training algorithms continues and new approaches are being suggested on an ongoing basis (e.g. Ma et al., 1997; Parisi et al., 1996; Verma, 1997; Wang and Chen, 1996).

8.3. Global methods

Global methods have the ability to escape local minima in the error surface and are thus able to find optimal or near optimal weight configurations. In stochastic gradient algorithms (e.g. Heskes and Kappen, 1993), the error function is perturbed to help the network escape local minima in the error surface. As training progresses, these perturbations are gradually removed (Hassoun, 1995). Simulated annealing (Kirkpatrick et al., 1983) is another optimisation method that enables networks to escape local minima in the error surface. The analogy is between the energy state of the network and the energy of a physical solid as it is slowly being cooled. At high temperatures, the network behaves like a random model and is able to jump freely from one local minimum to another. As the temperature is reduced slowly, the

energy of the system is decreased and the system can still jump out of 'shallow' local minima, but not the 'deeper' ones. At these intermediate temperatures, the model has a stochastic component, the magnitude of which depends on the temperature. At low temperatures, the network behaves more like a deterministic model.

When genetic algorithms (GAs) (Goldberg, 1989) are used for training, connection weights are chosen at random at the beginning of the training process, and are encoded in binary strings. The cost function is then calculated and the next set of connection weights is obtained with the aid of the genetic operators of selection, crossover and mutation, which favour solutions of higher 'fitness'. The above process is applied repeatedly until convergence to a near-global solution is achieved. The genetic algorithm is similar to simulated annealing in the sense that the two methods employ random search strategies. However, GAs have the ability to search effectively for an optimum in several directions simultaneously. Another advantage of GAs is that they can be parallelised easily (Rojas, 1996). It has also been shown that the asymptotic convergence of a GA with appropriate mutation parameters can be faster than that of simulated annealing (Davis and Principe, 1993). However, in general, the convergence speed of GAs is slow compared with that of second-order methods.

8.4. Stopping criteria

The criteria used to decide when to stop the training process are vitally important, as they determine whether the model has been optimally or sub-optimally trained. Examples of the latter include stopping training too early or once overfitting of the training data has occurred (see Maier and Dandy, 1998a). At this stage in the discussion, it is important to recall that overfitting is intricately linked to the ratio of the number of training samples to the number of connection weights. Amari et al. (1997) show that overfitting does not occur if the above ratio exceeds 30. In such cases, training can be stopped when the training error has reached a sufficiently small value or when changes in the training error remain small. When the above condition is not met, there are clear benefits in using cross-validation (Amari et al., 1997). In practical terms, however, this is not a straightforward task, and there has been much discussion about the relative merits of the use of cross-validation as a stopping criterion. For example, Hecht-Nielsen (1990); Hassoun (1995) discuss the benefits and effectiveness of using cross-validation methods. However, some researchers (e.g. Masters, 1993; Ripley, 1994) suggest that it is impossible to determine the optimal stopping time and that there is a danger that training is stopped prematurely (i.e. even though the error obtained using the test set might increase at some stage during training, there is no guarantee that it will not reach lower levels at a later stage if training were continued). Consequently, when cross-validation is used, it is vital to continue training for some time after the error in the test set first starts to rise.

9. Validation

Once the training (optimisation) phase has been completed, the performance of the trained network has to be validated on an independent data set using the criteria chosen (Section 3). It is important to note that it is vital that the validation data should not have been used as part of the training process *in any capacity*. If the difference in the error obtained using the validation set is markedly different than that obtained using the training data, it is likely that the two data sets are not representative of the same population or that the model has been overfitted (Masters, 1993). Poor validation can also be due to network architecture, a lack of, or inadequate, data pre-processing and normalisation of training/validation data. Diagnostic tools used to check the adequacy of statistical models can also be applied to neural networks. For example, Hsu et al. (1995); Lek et al. (1996) studied the normality and correlation of the residuals in order to assess the adequacy of the chosen model. In addition, error bounds can be used to assess the prediction accuracy of ANN models (e.g. Hwang and Ding, 1997; Whitley and Hromadka, 1999).

10. Applications in water resources

In this section, 43 papers in which ANNs have been used for the prediction or forecasting of *water resources variables* are reviewed *in terms of the modelling process adopted*. The reviews are restricted to papers which have been published in international journals until the end of 1998. However, due to the large number of such papers, and the rapid increase in, and widespread nature of, journals in which they appear, it is unlikely that complete coverage has been achieved. The modelling steps investigated include selection of performance criteria, data division and pre-processing, determination of model inputs, selection of network architecture, optimisation of connection weights and validation (see Sections 3–9). The major features of the models investigated are summarised in Tables 4 and 5, including background information (variable modelled, location of application, model time step, forecast length), information about the data used (data type (i.e. real or synthetic), normalisation range, number of training samples, number of testing samples), information about network architecture (connection type, method used to obtain optimal network geometry, number of nodes per layer), information about the optimisation algorithm used (optimisation method,

Table 4
Details of the paper reviewed (background information and data)

Background information				Data					
Ref.	Author(s) and year	Variable	Location(s)	Time step	Forecast length	Data type	Normalisation range	No. train. samples	No. test samples
1	Whitehead et al., 1997	Algal conc.	River Thames (England)	week	?	Real	?	125	31
2	Recknagel et al., 1997	Algal conc.	Lakes in Japan and Finland, River Darling (Australia)	day	+ 1	Real	?	2191–3653	730
3	Yabunaka et al., 1997	Algal conc.	Lake Kasumigaura (Japan)	day	+ 7	Real	?	365	4015
4	Recknagel, 1997	Cyanobact. conc.	Lake Kasumigaura (Japan)	day	+ 1	Real	?	2922	730
5	Maier and Dandy, 1997b	Cyanobact. conc.	River Murray (Australia)	week	+ 2	Real	?	468	52
6	Maier et al., 1998	Cyanobact. conc.	River Murray (Australia)	week	+ 4	Real	?	368	28
7	Crespo and Mora, 1993	Flow	Pisuena River (Spain)	10 days	0	Real	− 1–1	402, 201	0, 201
8	Karunanithi et al., 1994	Flow	Huron River (USA)	day	0	Synth/Real	÷ 100	4748	731
9	Hsu et al., 1995	Flow	Leaf River (USA)	day	+ 1	Real	0.1–0.9	365	1826
10	Lorrai and Sechi, 1995	Flow	Araxisi River (Italy)	month	0	Real	?	120	240
11a	Smith and Eli, 1995	Flow (single)	N/A	?	0	Synth	0.1–0.9	250	250
11b	Smith and Eli, 1995	Flow (multiple)	N/A	?	0	Synth	0.1–0.9	750	250
12	Raman and Sunilkumar, 1995	Flow	Reservoirs in India	month	+ 1	Real	0–1	10	2
13	Minns and Hall, 1996	Flow	N/A	hour	0	Synth	0–1	764	794
14	Poff et al., 1996	Flow	Little Patuxent and Independence Rivers (USA)	day	0	Real	?	1096	1096
15	Clair and Ehrman, 1996	Flow, Carb., Nit.	Canadian Rivers	year	0	Real	?	85%	15%
16	Shamseldin, 1997	Flow	Catchments in Nepal, China, Ireland, USA, Australia	day	0	Real	0.1–0.85	1461–2922	365–730
17	Tawfik et al., 1997	Flow	River Nile	day	0	Real	0.05–0.95	70–144	70–144
18	Muttiah et al., 1997	Flow	US River Basins	N/A	N/A	Real	?	75–1000	47–559
19	Sureerattanan and Phien, 1997	Flow	Mae Klong River (Thailand)	day	+ 1	Real	0.05–0.95	1095–2190	365–1460
20a	Dawson and Wilby, 1998	Flow	River Mole (UK)	15 min	+ 24	Real	0–1	9600	9600
20b	Dawson and Wilby, 1998	Flow	River Amber (UK)	15 min	+ 24	Real	0–1	2761	1321
21a	Fernando and Jayawardena, 1998	Flow	Kaminonsha (Japan)	hour	+ 1	Real	?	146	588
21b	Fernando and Jayawardena, 1998	Flow	Kaminonsha (Japan)	hour	+ 1	Real	?	146	588
22a	Jayawardena and Fernando, 1998	Flow	Kaminonsha (Japan)	hour	+ 1	Real	?	146	588
22b	Jayawardena and Fernando, 1998	Flow	Kaminonsha (Japan)	hour	+ 1	Real	?	146	588
23a	Thirumalaiah and Deo, 1998a	Flow	Bhasta River (India)	hour	+ 3	Real	?	560	162
23b	Thirumalaiah and Deo, 1998a	Flow	Bhasta River (India)	hour	+ 3	Real	?	560	162
23c	Thirumalaiah and Deo, 1998a	Flow	Bhasta River (India)	hour	+ 3	Real	?	560	162
24	Golob et al., 1998	Flow	Soca River (Slovenia)	hour	+ 2 – + 6	Real	?	3287	1700

Continued.

Table 4
Continued

Background information				Data					
Ref.	Author(s) and year	Variable	Location(s)	Time step	Forecast length	Data type	Normalisation range	No. train. samples	No. test samples
25	French et al., 1992	Rainfall	N/A	hour	+ 1	Synth	?	1000	500
26	Allen and le Marshall, 1994	Rainfall	Melbourne (Australia)	day	+ 1/2	Real	0–1	1997	665
27	Goswami and Srividya, 1996	Rainfall	India	year	+ 2– + 15	Real	?	50, 100	15
28	Hsu et al., 1997	Rainfall	Japan, Florida (USA)	hour	0	Real	0–1	?	?
29	Miller, 1997	Rainfall	Western Pacific	?	0	Real	?	300	33000
30	Venkatesan et al., 1997	Rainfall	India	year	0	Real	0–1	49	7
31	Xiao and Chandrasekar, 1997	Rainfall	Central Florida (USA)	min	0	Real	0–1	?	?
32	Tsintikidis et al., 1997	Rainfall	Western Pacific	N/A	0	Synth/Real	0.1–0.9	~ 733	~ 240
33	Chow and Cho, 1997	Rainfall	Hong Kong	hour	+ 1/2	Real	?	96	144
34a	Loke et al., 1997	Rainfall	Denmark	min	N/A	Real	?	1032	4560
34b	Loke et al., 1997	Runoff coeff.	Catchments in Europe and America	N/A	N/A	Real	?	35	7
35	Kuligowski and Barros, 1998a	Rainfall	Mount Carmel (USA)	6 hour	+ 1	Real	0–1	11344	873
36	Kuligowski and Barros, 1998b	Rainfall	Youghiogheny River and Swatare Creek basins (USA)	6 hour	+ 1– + 4	Real	?	649	162
37	Kuligowski and Barros, 1998c	Rainfall	Mid-Atlantic Region (USA)	6 hour	N/A	Real	?	3688	922
38a	DeSilets et al., 1992	Salinity (bottom)	Chesapeake Bay (USA)	N/A	N/A	Real	0–1	395–2712	78–523
38b	DeSilets et al., 1992	Salinity (total)	Chesapeake Bay (USA)	N/A	N/A	Real	0–1	3924–36258	1171–7133
39	Maier and Dandy, 1996b	Salinity	River Murray (Australia)	day	+ 14	Real	?	1461	365
40a	Bastarache et al., 1997	Salinity	Moose Pit Brook (Canada)	day	0	Real	?	285	32
40b	Bastarache et al., 1997	Salinity	Pine Martin Brook (Canada)	day	0	Real	?	356	39
40c	Bastarache et al., 1997	pH	Moose Pit Brook (Canada)	day	0	Real	?	285	32
40d	Bastarache et al., 1997	pH	Pine Martin Brook (Canada)	day	0	Real	?	356	39
41	Yang et al., 1996	Water table level	N/A	day	+ 1	Synth	?	2392	2392
42	Shukla et al., 1996	Water table level	N/A	N/A	N/A	Synth	?	26140	26140
43a	Thirumalaiah and Deo, 1998b	Water level	River Godavari (India)	day	+ 1, + 2	Real	− 0.5–0.5	800	295
43b	Thirumalaiah and Deo, 1998b		River Godavari (India)	day	+ 1, + 2	Real	− 0.5–0.5	800	295
43c	Thirumalaiah and Deo, 1998b		River Godavari (India)	day	+ 1, + 2	Real	− 0.5–0.5	800	295

?, not specified; N/A, not applicable.

Table 5
Details of papers reviewed (network architecture and optimisation)

	Network architecture			Optimisation algorithm							
Ref.[a]	Connect. type	Geometry method	Optimum I-H1-H2-O	Optim. method	Param. method	Transfer function	Learn. rate	Momentum	Epoch size	Initial weights	Stopping criterion
1	FF	?	?	BP	?	?	?	?	?	?	?
2	FF	T&E	?	BP	T&E	HT	0.1–0.9	0.05–0.6	?	?	FI
3	FF	T&E	10-50-0-5	BP	T&E	?	0.15–0.4	?	?	?	CV
4	FF	?	?	BP	?	HT	?	?	?	?	?
5	FF	F	102-120-40-1	BP	?	HT	?	?	?	?	CV
6	FF	T&E	20-17-0-1	BP	T&E	HT	0.004	0.6	16	?	FI
7	FF	F	3-3-2-1	BP	?	HT	?	?	?	?	?
8	FF	CC	15-0.34-0-1	QP	N/A	Log	N/A	N/A	?	− 1, 1	TE
9	FF	LLSSIM	9-3-0-1	LLSSIM	N/A	Log	N/A	N/A	?	?	?
10	FF	F	17-17-17-1	BP	F	Log	0.5	0.9	?	− 0.5, 0.5	FI
11a	FF	T&E	25-15-0-1	BP	F	Log	0.5	0.9	TS	− 0.5, 0.5	TE or FI
11b	FF	T&E	25-50-0-21	BP	F	Log	0.1	0.9	TS	− 0.5, 0.5	TE or FI
12	FF	T&E	4-?-?-2	BP	F	Log	0.5	0.9	?	?	CV
13	FF	F	18-10-0-1	BP	?	Log	?	?	?	?	FI
14	FF	?	?	BP	?	HT	?	?	?	?	
15	FF	?	?	BP	?	?	?	?	?	?	TE
16	FF	F	(1, 3, 5)-2-0-1	CG	N/A	Log	N/A	N/A	?	?	TE
17	FF	T&E	2-2-0-1	BP	F	Lin	0.95	0.1	TS	?	TE
18	FF	CC	?	QP	N/A	Log	N/A	N/A	?	?	TE
19	FF	F	5-2-0-1	BP	F	Log	0.01	0.5	TS	?	TE or FI
20a	FF	T&E	7-20-0-1	BP	F	Log	0.1	?	TS	− 0.29–0.29	FI
20b	FF	T&E	15-10-0-1	BP	F	Log	0.1	?	TS	− 0.13–0.13	FI
21a	FF	OLS	5-14-0-1	OLS	N/A	RBF	N/A	N/A	?	?	TE
21b	FF	T&E	5-6-0-1	BP	F	?	0.05	?	?	?	FI
22a	FF	T&E	6-11-0-1	OLS	N/A	RBF	N/A	N/A	?	?	TE or FI
22b	FF	T&E	6-6-0-1	BP	?	?	?	?	?	?	TE or FI
23a	FF	T&E	5-8-0-1	BP	?	?	?	?	?	?	TE
23b	FF	T&E	5-8-0-1	CG	N/A	?	N/A	N/A	?	− 0.5, 0.5	TE
23c	FF	CC	5-13-0-1	QP	?	?	N/A	N/A	?	?	TE
24	FF	T&E	14-40-0-3	BP	V	Log	V	?	TS	?	CV
25	FF	T&E	625-100-0-625	BP	F	Log	V	?	TS	− 1, 1	FI
26	FF	T&E	6-4-0-1	BP	?	Log	?	?	?	?	FI
27	FF	F	5-5-0-1	BP	?	HT	?	?	?	?	?
28	HYB	F	6-225-225-1	MCP	?	N/A	N/A	N/A	1, TS	?	?
29	FF	?	?	BP	?	?	?	?	?	?	?
30	FF	T&E	2-6-0-1	BP	?	Log	?	?	TS	− 1, 1	FI
31	FF	F	39-47-21-1	RLS	N/A	TL	N/A	N/A	?	?	CV
32	FF	T&E	4-16-0-1	BP	?	Log	V	V	?	− 1, 1	?
33	RC	F	31-21-3	Mod BP	F	HT	0.1	0.01	?	− 0.5, 0.5	FI
34a	FF	T&E	50-10-0-1	Mod BP	T&E	?	?	?	?	?	?
34b	FF	T&E	3-25-0-1	Mod BP	T&E	?	?	?	?	?	?
35	FF	T&E	20-10-0-1	BP	F	HT	0.01	0.001	TS	− 0.1, 0.1	CV
36	FF	T&E	25-11-0-1	BP	F	HT	0.05	0.005	TS	?	CV
37	FF	T&E	5-?-?-1	BP	F	HT	0.01	0.001	TS	?	CV
38a	FF	T&E	6-1-0-1 to	BP	T&E	Log	0.2–0.8	0.1–0.7	1	?	TE or FI
38b	FF	T&E	6-3-0-1	BP	T&E	Log	0.2–0.8	0.1–0.8	1	?	TE or FI
39	FF	T&E	51-45-0-1	BP	T&E	HT	0.02	0.6	?	− 0.1, 0.1	FI
40a	FF	T&E	3-20-0-1	LM	N/A	HT	N/A	N/A	?	?	?
40b	FF	T&E	3-15-0-1	LM	N/A	HT	N/A	N/A	?	?	?
40c	FF	T&E	3-10-0-1	LM	N/A	HT	N/A	N/A	?	?	?

Continued.

internal network parameters (method used to optimise internal network parameters, hidden layer transfer function, learning rate, momentum value, epoch size, initial weight distribution range)) and the stopping criterion adopted. It should be noted that the model parameters shown in Tables 4 and 5 pertain to the models that performed best out of the models investigated in each of the case studies. Not all of the values shown in Tables

Table 5
Continued

	Network architecture			Optimisation algorithm							
Ref.[a]	Connect. type	Geometry method	Optimum I-H1-H2-O	Optim. method	Param. method	Transfer function	Learn. rate	Momentum	Epoch size	Initial weights	Stopping criterion
40d	FF	T&E	3-20-0-1	LM	N/A	HT	N/A	N/A	?	?	?
41	FF	T&E	9-6-0-1	BP	?	HT	?	?	16	?	FI
42	FF	T&E	5-6-6-2	BP	F	Log	?	?	?	?	FI
43a	FF	T&E	1-3-0-2	BP	T&E	Log	0.1	0.2	TS	− 0.5, 0.5	TE
43b	FF	?	1-3-0-2	CG	N/A	?	N/A	N/A	TS	− 0.5, 0.5	TE
43c	FF	CC	1-2-0-2	QP	?	?	N/A	N/A	TS	?	TE

[a]Refer to Table 4 for details of reference.
?, not specified; BP, backpropagation; CG, conjugate gradient; CV, cross-validation; CC, cascade correlation; F, fixed; FI, fixed number of iterations; FF, feedforward; HT, hyperbolic tangent; H1, number of nodes in hidden layer 1; H2, number of nodes in hidden layer 2; HYB, hybrid; I, number of nodes in input layer; Lin, linear; LLSSIM, linear least-squares simplex; LM, Levenberg–Marquardt; Log, logistic; MCP, modified counterpropagation; Mod BP, modified backpropagation; Mod Log, modified logistic; N/A, not applicable; O, number of nodes in output layer; OLS, orthogonal least squares; QP, quickprop; RBF, radial basis function; RC, recurrent; RLS, recursive least-squares; T&E, trial and error; TE, training error; TS, training set; TL, threshold logic; V, variable.

4 and 5 are given explicitly in the papers and some have been inferred from the information provided. The papers are grouped according to variable modelled and in ascending order of publication year within each of these groups.

A distribution of papers by year of publication is shown in Table 6. It can be seen that the first papers were published in 1992 and that there has been a marked increase in the number of publications in the last few years. Forecasting of flow (including rainfall–runoff, streamflow and reservoir inflows) was the focus of 18 papers while rainfall forecasting was the subject of a further 13 papers. In one of the latter papers (Loke et al., 1997), runoff coefficients were estimated in addition to rainfall and in one of the former papers (Clair and Ehrman, 1996), carbon and nitrogen levels, as well as streamflow, were predicted. There are nine papers that deal with the forecasting of water quality variables such as algal concentrations (3), cyanobacterial concentrations (3) and salinity levels (3). In one of the latter papers (Bastarache et al., 1997), pH was predicted in addition to salinity. The remaining three papers are concerned with the prediction of water levels (including water table level and river stage).

Table 6
Distribution of papers reviewed by publication date

Year of Publication	Number of Papers
1998	10
1997	17
1996	7
1995	4
1994	2
1993	1
1992	2

The modelling time steps used range from one minute to one year. Prediction (i.e. forecast length of 0) and forecasting (i.e. forecast length > 0) were the focus of an approximately equal number of papers. The forecast length varied from 1 to 24 time steps. In one paper (Whitehead et al., 1997) the forecast length was not disclosed.

10.1. Performance criteria

The criteria which were used to assess model performance were generally stated clearly. In all but two papers, prediction accuracy was the main performance criterion and was measured using a variety of metrics. Shukla et al. (1996); Yang et al. (1996) used processing speed during recall as the main performance criterion, as ANN models were used as alternatives to time consuming procedural models for real-time applications. Short training times were also mentioned as being desirable in a number of papers, but were only assessed explicitly in papers in which the performance of a number of training algorithms was compared (Fernando and Jayawardena, 1998; Jayawardena and Fernando, 1998; Thirumalaiah and Deo, 1998a, b). The interrelationships between the various performance criteria and network architecture and the optimisation algorithm used, which are discussed in Section 3, were generally not considered.

10.2. Data

Real data were used in 35 papers, while data were generated synthetically in the remaining eight papers (Table 4). It should be noted that in two papers belonging to the latter group (Karunanithi et al., 1994; Tsintikidis et al., 1997), real data were used in some of the validation sets. Data division was carried out incorrectly

in most papers. Although at least two data sets were used in all but one paper (Crespo and Mora, 1993), the validation data were used as part of the training process in many instances either by using a trial-and-error procedure to optimise model inputs, network geometry and internal parameters, or by using cross-validation as the stopping criterion. In only two cases was an independent test set used in addition to the training and validation sets (Golob et al., 1998; Raman and Sunilkumar, 1995). The proportion of data used for training and validation varied greatly (Table 4). Generally, the division of data was carried out on an arbitrary basis and the statistical properties of the respective data sets were seldom considered. However, the training/validation data split can have a significant impact on the results obtained (Dawson and Wilby, 1998; Maier and Dandy, 1996b). The holdout method, which overcomes the above problem, was only used on a few occasions (e.g. Kuligowski and Barros, 1998a, c; Lorrai and Sechi, 1995; Maier and Dandy, 1997b).

Data pre-processing was described poorly in most papers. Scaling of the data to a range that is commensurate with the transfer function in the output layer was only discussed in 18 of the 43 papers. In the majority of these, the data were scaled to the extreme ranges of the transfer function (Table 4), which can lead to flatspots during training (see Section 5 and Section 8.1.2). The distribution of the input data was not considered in any of the papers. However, as discussed in Section 5, it is unclear whether there is a need to address this point. Stationarity is another issue that was largely ignored. It should be noted that this does not apply to all papers, as some did not consider time series data. As discussed in Section 5, the main concern is with the ability of trained models to deal with data that fall outside of the range used for training (i.e. if the data contain trends). Attempts to deal with this issue were made in some of the papers by using a linear transfer function in the output layer (e.g. Bastarache et al., 1997; Crespo and Mora, 1993; Karunanithi et al., 1994) or by developing separate models for different months of the year (Raman and Sunilkumar, 1995).

10.3. Model inputs

The way the model inputs were determined was generally described well. However, some of the methods used raise considerable doubt about the optimality of the model inputs obtained. In the majority of papers, the input variables were chosen using *a priori* knowledge. However, in some instances, the number of input variables was optimised using stepwise discriminant analysis (e.g. Allen and le Marshall, 1994), principal component analysis (e.g. Allen and le Marshall, 1994), correlation analysis (e.g. Dawson and Wilby, 1998; Kuligowski and Barros, 1998a, b), polynomial networks (Bastarache et al., 1997), sensitivity analysis (Yabunaka et al., 1997), a stepwise model building approach (e.g. Maier et al., 1998) or a trial-and-error procedure (e.g. Thirumalaiah and Deo, 1998b; Tsintikidis et al., 1997; Venkatesan et al., 1997; Xiao and Chandrasekar, 1997; Yang et al., 1996). However, as pointed out in Section 10.2, by using a trial-and-error procedure, the validation data are used as part of the training process.

There were some forecasting applications (i.e. forecast length > 0) in which lagged inputs were not considered explicitly (e.g. Chow and Cho, 1997; Dawson and Wilby, 1998; Goswami and Srividya, 1996). However, Chow and Cho (1997) used a recurrent network to take dynamic input information into account and Dawson and Wilby (1998) incorporated some 'memory' into the input data by using moving averages. In some papers in which past inputs were considered, appropriate lags were chosen with the aid of *a priori* information (e.g. Crespo and Mora, 1993; Maier and Dandy, 1996b; Minns and Hall, 1996; Sureerattanan and Phien, 1997) or by using a trial-and-error approach (e.g. Hsu et al., 1995; Lorrai and Sechi, 1995; Thirumalaiah and Deo, 1998b; Yang et al., 1996). However, as pointed out in Section 10.2, the latter method uses the validation data during training. In other papers, the number of lags was chosen arbitrarily (e.g. Karunanithi et al., 1994; Raman and Sunilkumar, 1995; Recknagel, 1997). Consequently, it could not be determined whether the inputs resulting in optimum model performance were included. As discussed in Section 6, analytical approaches supported by *a priori* knowledge are the preferred option for determining appropriate input lags. However, these were only used in three papers (Fernando and Jayawardena, 1998; Maier and Dandy, 1997b; Maier et al., 1998).

10.4. Network architecture

Feedforward networks were used in 41 papers. The exceptions are papers by Chow and Cho (1997); Hsu et al. (1997), in which recurrent and hybrid architectures were used, respectively. In the papers in which feedforward networks were used to obtain forecasts (i.e. forecast length > 0), the possibility of using recurrent networks was generally not considered. However, as discussed in Section 10.3, in the majority of these papers, dynamic information was incorporated into the model explicitly by using lagged inputs. Feedforward networks were also used in the two papers in which processing speed was of importance (Shukla et al., 1996; Yang et al., 1996). Network connectivity was not discussed by many authors, although it is probably safe to assume that the majority of networks were fully connected.

Details of the optimum network geometry were given in 34 papers. Partial information was given in some of the remaining 9 papers, while no information at all was provided in others (Table 5). Networks with one hidden layer were used in the majority of case studies, although

networks with two and three hidden layers were also used. In most instances, no reason for the number of hidden layers used was given. However, as discussed in Section 7.2, there is some debate as to whether it is better to use one or multiple hidden layers. In the vast majority of papers, the optimal number of hidden layer nodes was obtained using trial and error (Table 5). However, as discussed in Section 10.2, this resulted in the use of the validation data during the training process. In addition, the number of hidden layer nodes obtained was greater than the theoretical upper bounds suggested in the literature in some cases, raising some doubt about the way training was carried out (e.g. Bastarache et al., 1997; Golob et al., 1998; Thirumalaiah and Deo, 1998a; Tsintikidis et al., 1997; Yabunaka et al., 1997). In ten papers, fixed network geometries were chosen (see Table 5). However, in most of these, no justification for the choice was given. The relationship between the number of training samples and the number of connection weights was mostly ignored, giving rise to the potential of overtraining. In two of these papers (Maier and Dandy, 1997b; Xiao and Chandrasekar, 1997), the risk of overtraining was eliminated by using cross-validation as the stopping criterion. Algorithms which optimise network geometry automatically were only used in six papers. Karunanithi et al. (1994); Muttiah et al. (1997); Thirumalaiah and Deo (1998a, b) used the Cascade-Correlation algorithm (Fahlman and Lebiere, 1990), Hsu et al. (1995) used the LLSSIM algorithm and Fernando and Jayawardena (1998) used the OLS algorithm (Table 4). However, the results obtained when such algorithms are used can be a function of a number of user-defined parameters.

10.5. Optimisation

The standard backpropagation algorithm was used to optimise the connection weights in 36 of the 43 papers reviewed (Table 5). In four of these, the performance of the backpropagation algorithm was compared with that of alternative training methods. Fernando and Jayawardena (1998); Jayawardena and Fernando (1998) compared the performance of backpropagation and radial basis function networks trained with the OLS algorithm. Thirumalaiah and Deo (1998a, b) compared the performance of networks trained with the backpropagation, conjugate gradient and Quickprop (Fahlman, 1988) algorithms. The latter combines dynamic learning rate adjustment with second-order information. In all four papers, the forecasts obtained using the various methods were comparable. The Quickprop algorithm was also used by Karunanithi et al. (1994); Muttiah et al. (1997). Hsu et al. (1995) used the LLSSIM algorithm, which combines a linear least-squares procedure for faster learning with the multistart simplex algorithm, allowing near-global optima to be found. The recurrent Sigma-Pi network used by Chow and Cho (1997) was trained with the aid of a simple method based on the steepest descent algorithm, whereas the modified counter propagation network in Hsu et al. (1997) used an unsupervised learning rule to train a Self Organising Feature Map (SOFM) layer and a supervised method based on linear least-squares estimation to optimise the weights in a modified Grossberg linear layer. Xiao and Chandrasekar (1997) used a recursive least-squares algorithm, and Shamseldin (1997) used a conjugate gradient method, to speed up the training process. Global optimisation algorithms (see Section 8.3) were not used in any of the papers.

In the papers in which the backpropagation algorithm was used, reasons for this choice were generally not given. However, given that prediction accuracy was the adopted performance criterion in most papers, use of the backpropagation algorithm appears justified, as its main drawback is its slow rate of convergence (Breiman, 1994). On the other hand, full details of the internal parameters controlling the backpropagation algorithm were rarely provided (Table 5). This makes it difficult to confirm the optimality of the results presented. In seven of the 36 papers in which the backpropagation algorithm was used, optimum internal parameters were obtained by trial and error. In the majority of these, a fixed number of learning iterations were performed or the stopping criteria were not disclosed (Table 5). Consequently, the optimality of different internal parameters could not be assessed adequately, as it is likely that networks with different parameters were at different stages of learning after a fixed number of iterations (see Section 8.1). A further complicating factor is the potential for oscillatory behaviour of the backpropagation algorithm in the vicinity of local minima (see Maier and Dandy, 1998a, b), which can mask any minor differences in performance resulting from different internal parameters. In 13 papers, fixed values of the internal parameters were chosen at the beginning of training. However, in most of these, reasons for the choices made were not given. In the remaining 16 papers, no details of how the internal parameters were selected were provided. Algorithms which automatically adjust some of the internal parameters as training progresses were only applied in the papers in which the Quickprop algorithm was used. However, as discussed in Section 8.1.2, many of these algorithms require the selection of empirical parameters which can have a significant effect on the results obtained.

In 13 papers, training was stopped once a fixed number of training iterations had been performed (Table 5). As discussed above, this could mean that training was stopped before a local minimum in the error surface was reached or after overtraining has occurred. On only two occasions (Maier and Dandy, 1996b; Maier et al., 1998) were preliminary trials carried out on independent data sets to determine how many training samples should be presented to the networks to ensure that they were in the vicinity of a local minimum at the completion of train-

ing. In 11 instances, training was stopped once the error of the training set had reached a pre-specified level (see Table 5). However, in most instances the ratio of the number of training samples to the number of free parameters was not taken into account. In four of the above cases, training was terminated if the desired error level was not reached after a certain number of iterations (Table 5). Cross-validation was used in eight instances and no information on which stopping criteria were used was given in the remaining 11 papers.

10.6. Validation

Independent validation sets were not used in one paper (Crespo and Mora, 1993). In addition, the independent data set was used to optimise network geometry and/or the internal network parameters or to decide when to stop training in the majority of papers. This means that the independent data set was used as part of the training process and that a third data set should have been used for validation. However, this was not the case. This issue needs to be addressed in order to ensure that the results obtained are valid.

11. Conclusions and recommendations

ANNs are being used increasingly for the prediction and forecasting of a number of water resources variables, including rainfall, flow, water level and various water quality parameters. In most papers, a good description of basic ANN theory, the case study considered and the results obtained is given. However, the modelling process is generally described poorly. This does not necessarily mean that the modelling process is carried out incorrectly. It could be argued that such information is not given explicitly as the focus of the papers is on the use of ANNs for the prediction and forecasting of water resources variables, and not on ANN modelling procedure. However, in the absence of complete information about the modelling process, the optimality of the results cannot be assessed, and it is difficult to draw meaningful comparisons between the performance of different models.

In a number of the papers reviewed, the modelling process is carried out incorrectly. The main areas of concern include (i) the utilisation of the validation data during the training process, either by using them to optimise the network inputs, geometry and parameters as part of a trial-and-error procedure, or by using them to decide when to stop training, (ii) the use of a fixed number of iterations as the stopping criterion, especially when the results obtained are used to compare the performance of models with different inputs, architectures and internal parameters, (iii) the arbitrary choice of model inputs, network architecture and internal model parameters and (iv) the scaling of the input data to the extreme ranges of the transfer function in the output layer. More attention needs to be paid to the effects of data pre-processing, model inputs, network architecture, optimisation algorithms and stopping criteria on the results obtained. There is also a need to consider the interdependencies between the various steps in the modelling process. Examples are (i) the impact of the optimisation algorithm on training speed and prediction accuracy, (ii) the influence of network architecture on processing speed and the amount of data needed to estimate the connection weights efficiently and (iii) the effect of the stopping criterion and the method used for determining the optimum number of hidden layer nodes on the way the data set should be divided.

Based on the results obtained thus far, there is little doubt that ANNs have the potential to be a useful tool for the prediction and forecasting of water resources variables. However, in order to achieve significant advances in the field, there needs to be a change in the mind-set of users, from the application of basic ANN models to an ever-increasing number of case studies to the development of guidelines for ANN modellers. At present, there is a tendency among researchers to apply ANNs to problems for which other methods have been unsuccessful. ANNs need to be viewed as *alternatives* to more traditional approaches in certain situations and not as "…a remedy for all existing computational failings" (Flood and Kartam, 1994). As pointed out by Chatfield (1993) when commenting on the suitability of ANNs for time series analysis and forecasting, "when the dust has settled, it is usually found that the new technique is neither a miraculous cure-all nor a complete disaster, but rather an addition to the analyst's toolkit which works well in some situations and not in others". Research efforts should be directed towards the identification of applications and circumstances in which particular ANN approaches do *not* perform well in order to define their boundaries of applicability. In situations where ANN modelling approaches *are* adopted, modellers are faced with a number of alternatives at each stage of the model development process. Consequently, there is a need to develop guidelines which identify the circumstances under which particular approaches should be adopted and how to optimise the parameters that control them. There have already been some empirical comparisons of various optimisation algorithms (e.g. Fernando and Jayawardena, 1998; Maier and Dandy, 1999a; Thirumalaiah and Deo, 1998b), but these should only be viewed as the first step in a continued research effort. One complicating factor is the fact that ANN modelling is a very active research area, and new network architectures and optimisation algorithms are being proposed on an ongoing basis. The dissemination of the vast amount of information that is being generated into a form that is useful for practitioners is one of the greatest chal-

lenges facing researchers in the field of ANN modelling. It is also important to ensure that guidelines assist, and do not unnecessarily restrict, ANN modellers. 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.

Future research efforts should also be directed towards the extraction of the knowledge that is contained in the connection weights of trained ANN models. Sensitivity analysis has already been used successfully for this purpose (e.g. Maier et al., 1998). However, the potential of rule extraction algorithms should also be investigated (see Andrews et al., 1995). Another possibility is the use of neurofuzzy approaches, which have already been used for atmospheric pollution prediction (Nunnari et al., 1998). 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 (Vicens et al., 1975). Consequently, techniques for dealing with uncertainty (e.g. Monte Carlo Simulation, First-Order Analysis) should be considered in the development of ANN models.

Acknowledgements

The authors would like to thank Dr. Barbara Lence from the University of British Columbia and Dr. Andrews Takyi from TetrES Consultants Inc. for their thoughtful and insightful comments on draft versions of this paper and Dr. Anthony Minns from IHE Delft for his thoughts on the use of ANNs in hydrology.

References

Abu-Mostafa, Y.S., 1989. The Vapnik-Chervonenkis dimension: Information versus complexity in learning. Neural Computation 1 (3), 312–317.

Allen, G., Le Marshall, J.F., 1994. An evaluation of neural networks and discriminant analysis methods for application in operational rain forecasting. Australian Meteorological Magazine 43 (1), 17–28.

Amari, S.-i., Murata, N., Müller, K.-R., Finke, M., Yang, H.H., 1997. Asymptotic statistical theory of overtraining and cross-validation. IEEE Transactions on Neural Networks 8 (5), 985–996.

Andrews, R., Diederich, J., Tickle, A.B., 1995. A survey and critique of techniques for extracting rules from trained artificial neural networks. Knowledge Based Systems 8, 373–389.

Angeline, P.J., Sauders, G.M., Pollack, J.B., 1994. An evolutionary algorithm that constructs recurrent neural networks. IEEE Transactions on Neural Networks 5, 54–65.

Bastarache, D., El-Jabi, N., Turkham, N., Clair, T.A., 1997. Predicting conductivity and acidity for small streams using neural networks. Canadian Journal of Civil Engineering 24 (6), 1030–1039.

Battiti, R., 1989. Accelerated back-propagation learning: Two optimization methods. Complex Systems 3, 331–342.

Battiti, R., 1992. First- and second-order methods for learning: Between steepest descent and Newton's method. Neural Computation 4, 141–166.

Bebis, G., Georgiopoulos, M., 1994. Feed-forward neural networks: Why network size is so important. IEEE Potentials October/November, 27–31.

Bienenstock, E., Geman, S., 1994. Comment on 'Neural networks: A review from a statistical perspective' by B. Cheng and D.M. Titterington. Statistical Science 9 (1), 36–38.

Bingham, J.A.C., 1988. The Theory and Practice of Modem Design. Wiley, New York.

Box, G.E.P., Jenkins, G.M., 1976. Time Series Analysis, Forecasting and Control. Holden-Day Inc., San Francisco, CA.

Braddock, R.D., Kremmer, M.L., Sanzogni, L., 1997. Feed-forward artificial neural network model for forecasting rainfall run-off. Proceedings of the International Congress on Modelling and Simulation (Modsim 97), The Modelling and Simulation Society of Australia Inc., Hobart, Australia, pp. 1653–1658.

Breiman, L., 1994. Comment on 'Neural networks: A review from a statistical perspective' by B. Cheng and D.M. Titterington. Statistical Science 9 (1), 38–42.

Broomhead, D.S., Lowe, D., 1988. Multivariate functional interpolation and adaptive networks. Complex Systems 2, 321–355.

Burden, F.R., Brereton, R.G., Walsh, P.T., 1997. Cross-validatory selection of test and validation sets in multivariate calibration and neural networks as applied to spectroscopy. Analyst 122 (10), 1015–1022.

Burke, L.I., Ignizio, J.P., 1992. Neural networks and operations research: an overview. Computer and Operations Research 19 (3/4), 179–189.

Castellano, G., Fanelli, A.M., Pelillo, M., 1997. An iterative pruning algorithm for feedforward neural networks. IEEE Transactions on Neural Networks 8 (3), 519–531.

Chakraborty, K., Mehrotra, K., Mohan, C.K., Ranka, S., 1992. Forecasting the behaviour of multivariate time series using neural networks. Neural Networks 5, 961–970.

Chatfield, C., 1975. The Analysis of Time Series: Theory and Practice. Chapman and Hall, London.

Chatfield, C., 1993. Neural networks: Forecasting breakthrough or just a passing fad? International Journal of Forecasting 9, 1–3.

Chen, K., Yang, L.P., Yu, X., Chi, H.S., 1997. A self-generating modular neural network architecture for supervised learning. Neurocomputing 16 (1), 33–48.

Cheng, B., Titterington, D.M., 1994. Neural networks: A review from a statistical perspective. Statistical Science 9 (1), 2–54.

Chng, E.S., Chen, S., Mulgrew, B., 1996. Gradient radial basis function networks for nonlinear and nonstationary time series prediction. IEEE Transactions on Neural Networks 7 (1), 191–194.

Chon, K.H., Cohen, R.J., 1997. Linear and nonlinear ARMA model parameter estimation using an artificial neural network. IEEE Transactions on Biomedical Engineering 44 (3), 168–174.

Chow, T.W.S., Cho, S.Y., 1997. Development of a recurrent sigma-pi neural network rainfall forecasting system in Hong Kong. Neural Computing and Applications 5 (2), 66–75.

Chung, F.L., Lee, T., 1992. A node pruning algorithm for backpropagation networks. International Journal of Neural Systems 3 (3), 301–314.

Clair, T.A., Ehrman, J.M., 1996. Variations in discharge and dissolved organic carbon and nitrogen export from terrestrial basins with changes in climate: a neural network approach. Limnology and Oceanography 41 (5), 921–927.

Connor, J.T., Martin, R.D., Atlas, L.E., 1994. Recurrent neural networks and robust time series prediction. IEEE Transactions on Neural Networks 5 (2), 240–254.

Crespo, J.L., Mora, E., 1993. Drought estimation with neural networks. Advances in Engineering Software 18 (3), 167–170.
Dai, H.C., Macbeth, C., 1997. Effects of learning parameters on learning procedure and performance of a BPNN. Neural Networks 10 (8), 1505–1521.
Darken, C., Moody, J., 1990. Note on learning rate schedules for stochastic optimization. In: Lippmann, R.P., Moody, J.E., Touretzky, D.S. (Eds.), Advances in Neural Information Processing Systems 3. Morgan Kaufmann, San Mateo, CA.
Davis, T.E., Principe, J.C., 1993. A Markov chain framework for the simple genetic algorithm. Evolutionary Computation 1 (3), 269–288.
Dawson, C.W., Wilby, R., 1998. An artificial neural network approach to rainfall–runoff modelling. Hydrological Sciences Journal 43 (1), 47–66.
DeSilets, L., Golden, B., Wang, Q., Kumar, R., 1992. Predicting salinity in the Chesapeake Bay using backpropagation. Computer and Operations Research 19 (3/4), 227–285.
Doering, A., Galicki, M., Witte, H., 1997. Structure optimization of neural networks with the A*-algorithm. IEEE Transactions on Neural Networks 8 (6), 1434–1445.
Elman, J.L., 1990. Finding structure in time. Cognitive Science 14, 179–211.
Fahlman, S.E., 1988. Faster-learning variations on back-propagation: An empirical study. 1988 Connectionist Models Summer School.
Fahlman, S.E., Lebiere, C., 1990. The cascade-correlation learning architecture. In: Touretzky, D.S. (Ed.), Advances in Neural Information Processing Systems 2. Morgan Kaufmann, San Mateo, CA.
Faraway, J., Chatfield, C., 1998. Time series forecasting with neural networks: a comparative study using the airline data. Applied Statistics 47 (2), 231–250.
Fernando, D.A.K., Jayawardena, A.W., 1998. Runoff forecasting using RBF networks with OLS algorithm. Journal of Hydrologic Engineering 3 (3), 203–209.
Finnhoff, W., Hergert, F., Zimmermann, H.G., 1993. Improving model selection by nonconvergent methods. Neural Networks 6, 771–783.
Flood, I., Kartam, N., 1994. Neural networks in civil engineering. I: Principles and understanding. Journal of Computing in Civil Engineering 8 (2), 131–148.
Fogel, L.J., Owens, A.J., Walsh, M.J., 1966. Artificial Intelligence Through Simulated Evolution. Wiley, New York.
Fortin, V., Ouarda, T.B.M.J., Bobée, B., 1997. Comment on 'The use of artificial neural networks for the prediction of water quality parameters' by H.R. Maier and G.C. Dandy. Water Resources Research 33 (10), 2423–22424.
French, M.N., Krajewski, W.F., Cuykendall, R.R., 1992. Rainfall forecasting in space and time using a neural network. Journal of Hydrology 137, 1–31.
Gençay, R., Liu, T., 1997. Nonlinear modelling and prediction with feedforward and recurrent networks. Physica D 108 (1–2), 119–134.
Gershenfeld, N.A., Weigend, A.S., 1994. The future of time series: Learning and understanding. In: Weigend, A.S., Gershenfeld, N.A. (Eds.), Time Series Prediction: Forecasting the Future and Understanding the Past. Addison-Wesley, Reading, MA.
Gill, P., Murray, W., Wright, M., 1981. Practical Optimization. Academic Press, New York.
Goldberg, D., 1989. Genetic Algorithms. Addison-Wesley, Reading, MA.
Golden, R.M., 1996. Mathematical Methods for Neural Network Analysis and Design. MIT Press, Cambridge.
Golob, R., Stokelj, T., Grgic, D., 1998. Neural-network-based water inflow forecasting. Control Engineering Practice 6 (5), 593–600.
Goswami, P., Srividya, 1996. A novel neural network design for long range prediction of rainfall pattern. Current Science 70 (6), 447–457.
Hansen, J.V., Nelson, R.D., 1997. Neural networks and traditional time series methods: A synergistic combination in state economic forecasts. IEEE Transactions on Neural Networks 8 (4), 863–873.
Hassoun, M.H., 1995. Fundamentals of Artificial Neural Networks. MIT Press, Cambridge.
Haugh, L.D., Box, G.E.P., 1977. Identification of dynamic regression (distributed lag) models connecting two time series. Journal of the American Statistical Association 72 (397), 121–130.
Hecht-Nielsen, R., 1987. Kolmogorov's mapping neural network existence theorem. Proceedings of the First IEEE International Joint Conference on Neural Networks, San Diego, California, pp. 11–14, IEEE, New York.
Hecht-Nielsen, R., 1990. Neurocomputing. Addison-Wesley, Reading, MA.
Heiss, M., Kampl, S., 1996. Multiplication-free radial basis function network. IEEE Transactions on Neural Networks 7 (6), 1461–1464.
Heskes, T.M., Kappen, B., 1993. On-line learning processes in artificial neural networks. In: Taylor, J.G. (Ed.), Mathematical Approaches to Neural Networks. Elsevier Science Publishers, Amsterdam.
Hill, T., Marquez, L., O'Connor, M., Remus, W., 1994. Artificial neural network models for forecasting and decision making. International Journal of Forecasting 10, 5–15.
Hirose, Y., Yamashita, K., Hijiya, S., 1991. Back-propagation algorithm which varies the number of hidden units. Neural Networks 4 (1), 61–66.
Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Computation 9, 1735–1780.
Hornik, K., Stinchcombe, M., White, H., 1989. Multilayer feedforward networks are universal approximators. Neural Networks 2, 359–366.
Hsu, K.L., Gao, X.G., Sorooshian, S., Gupta, H.V., 1997. Precipitation estimation from remotely sensed information using artificial neural networks. Journal of Applied Meteorology 36 (9), 1176–1190.
Hsu, K.-L., Gupta, H.V., Sorooshian, S., 1995. Artificial neural network modeling of the rainfall–runoff process. Water Resources Research 31 (10), 2517–2530.
Huang, S.C., Huang, Y.F., 1991. Bounds on the number of hidden neurons in multilayer perceptrons. IEEE Transactions on Neural Networks 2, 47–55.
Hwang, J.T.G., Ding, A.A., 1997. Prediction intervals for artificial neural networks. Journal of the American Statistical Association 92 (438), 748–757.
Irvine, K.N., Eberhardt, A.J., 1992. Multiplicative, seasonal ARIMA models for Lake Erie and Lake Ontario water levels. Water Resources Bulletin 28 (2), 385–396.
Jacobs, R.A., 1988. Increased rates of convergence through learning rate adaptation. Neural Networks 1, 295–307.
Jayawardena, A.W., Fernando, D.A.K., 1998. Use of radial basis function type artificial neural networks for runoff simulation. Computer-Aided Civil and Infrastructure Engineering 13 (2), 91–99.
Kaastra, I., Boyd, M.S., 1995. Forecasting futures trading volume using neural networks. The Journal of Futures Markets 15 (8), 953–970.
Kalman, B.L., Kwasny, S.C., 1992. Why Tanh? Choosing a sigmoidal function. In: Proceedings of the International Joint Conference on Neural Networks, Baltimore, MD IEEE, New York.
Karnin, E.D., 1990. A simple procedure for pruning backpropagation trained neural networks. IEEE Transactions on Neural Networks 1, 239–242.
Karunanithi, N., Grenney, W.J., Whitley, D., Bovee, K., 1994. Neural networks for river flow prediction. Journal of Computing in Civil Engineeirng 8 (2), 201–220.
Khotanzad, A., Afkhami-Rohani, R., Lu, T.-L., Abaye, A., Davis, M., Maratukulam, D.J., 1997. ANNSTLF—a neural-network-based electric load forecasting system. IEEE Transactions on Neural Networks 8 (4), 835–846.

Kirkpatrick, S., Gilatt, C.D., Vecchi, M.P., 1983. Optimization by simulated annealing. Science 220, 671–680.

Kollias, S., Anastrassiou, D., 1989. An adaptive least square algorithm for the efficient training of artificial neural networks. IEEE Transactions on Circuits and Systems 36, 1092–1101.

Krishnapura, V.G., Jutan, A., 1997. ARMA neuron networks for modeling nonlinear dynamical systems. Canadian Journal of Chemical Engineering 75 (3), 574–582.

Kuligowski, R.J., Barros, A.P., 1998a. Experiments in short-term precipitation forecasting using artificial neural networks. Monthly Weather Review 126, 470–482.

Kuligowski, R.J., Barros, A.P., 1998b. Localized precipitation forecasts from a numerical weather prediction model using artificial neural networks. Weather and Forecasting 13, 1194–1204.

Kuligowski, R.J., Barros, A.P., 1998c. Using artificial neural networks to estimate missing rainfall data. Journal of the American Water Resources Association 34 (6), 1437–1447.

Kumar, K.K., 1993. Optimization of the neural net connectivity pattern using a backpropagation algorithm. Neurocomputing 5, 273–286.

Kwok, T.-Y., Yeung, D.-Y., 1997a. Constructive algorithms for structure learning in feedforward neural networks for regression problems. IEEE Transactions on Neural Networks 8 (3), 630–645.

Kwok, T.-Y., Yeung, D.-Y., 1997b. Objective functions for training new hidden units in constructive neural networks. IEEE Transactions on Neural Networks 8 (5), 1131–1148.

Lachtermacher, G., Fuller, J.D., 1994. Backpropagation in hydrological time series forecasting. In: Hipel, K.W., McLeod, A.I., Panu, U.S., Singh, V.P. (Eds.), Stochastic and Statistical Methods in Hydrology and Environmental Engineering. Kluwer Academic, Dordrecht.

Lapedes, A., Farber, R., 1986. A self-optimizing, nonsymmetrical neural net for content addressable memory and pattern recognition. Physica D 22, 247–259.

Lee, B.W., Shen, B.J., 1991. Hardware annealing in electronic neural networks. IEEE Transactions on Circuits and Systems 38, 134–137.

Lek, S., Delacoste, M., Baran, P., Dimopoulos, I., Lauga, J., Aulagnier, S., 1996. Application of neural networks to modelling nonlinear relationships in ecology. Ecological Modelling 90, 39–52.

Liano, K., 1996. Robust error measure for supervised neural network learning with outliers. IEEE Transactions on Neural Networks 7 (1), 246–250.

Lin, T., Horne, B.G., Tiño, P., Giles, C.L., 1996. Learning long-term dependencies in NARX recurrent neural networks. IEEE Transactions on Neural Networks 7 (6), 1329–1338.

Lippmann, R.P., 1987. An introduction to computing with neural nets. IEEE Acoustics, Speech and Signal Processing Magazine 4, 4–22.

Loke, E., Warnaars, E.A., Jacobsen, P., Nelen, F., Almeida, M.D., 1997. Artificial neural networks as a tool in urban storm drainage. Water Science and Technology 36 (8–9), 101–109.

Lorrai, M., Sechi, G.M., 1995. Neural nets for modelling rainfall–runoff transformations. Water Resources Management 9 (4), 299–313.

Ma, S., Ji, C.Y., Farmer, J., 1997. An efficient EM-based training algorithm for feedforward neural networks. Neural Networks 10 (2), 243–256.

Maier, H.R., 1995. Use of artificial neural networks for modelling multivariate water quality time series. PhD Thesis, The University of Adelaide.

Maier, H.R., Dandy, G.C., 1996a. Neural network models for forecasting univariate time series. Neural Network World 6 (5), 747–771.

Maier, H.R., Dandy, G.C., 1996b. The use of artificial neural networks for the prediction of water quality parameters. Water Resources Research 32 (4), 1013–1022.

Maier, H.R., Dandy, G.C., 1997a. Determining inputs for neural network models of multivariate time series. Microcomputers in Civil Engineering 12 (5), 353–368.

Maier, H.R., Dandy, G.C., 1997b. Modelling cyanobacteria (blue-green algae) in the River Murray using artificial neural networks. Mathematics and Computers in Simulation 43 (3–6), 377–386.

Maier, H.R., Dandy, G.C., 1998a. The effect of internal parameters and geometry on the performance of back-propagation neural networks: An empirical study. Environmental Modelling and Software 13 (2), 193–209.

Maier, H.R., Dandy, G.C., 1998b. Understanding the behaviour and optimising the performance of back-propagation neural networks: An empirical study. Environmental Modelling and Software 13 (2), 179–191.

Maier, H.R., Dandy, G.C., 1999a. Empirical comparison of various methods for training feedforward neural networks for salinity forecasting. Water Resources Research, submitted.

Maier, H.R., Dandy, G.C. 1999b. Neural network based modelling of environmental variables: A systematic approach. Mathematical and Computer Modelling, in press.

Maier, H.R., Dandy, G.C., Burch, M.D., 1998. Use of artificial neural networks for modelling cyanobacteria *Anabaena* spp. in the River Murray, South Australia. Ecological Modelling 105 (2/3), 257–272.

Maren, A., Harston, C., Pap, R., 1990. Handbook of Neural Computing Applications. Academic Press, San Diego, CA.

Masters, T., 1993. Practical Neural Network Recipes in C + +. Academic Press, San Diego, CA.

McCulloch, W.S., Pitts, W., 1943. A logical calculus of the ideas imminent in nervous activity. Bulletin and Mathematical Biophysics 5, 115–133.

Miller, G.F., Todd, P.M., Hedge, S.U., 1989. Designing neural networks using genetic algorithms. In: Proceedings of the Third International Conference on Genetic Algorithms, Arlington, pp. 379–384. Morgan Kaufman, San Meteo.

Miller, S.W., 1997. Rain rate estimation using neural networks. AI Applications 11 (1), 95–98.

Minns, A.W., Hall, M.J., 1996. Artificial neural networks as rainfall–runoff models. Hydrological Sciences Journal 41 (3), 399–417.

Møller, M.S., 1993. A scaled conjugate gradient algorithm for fast supervised learning. Neural Networks 6 (4), 525–534.

Moody, J., Yarvin, N., 1992. Networks with learned unit response functions. In: Moody, J.E., Hanson, S.J., Lippmann, R.P. (Eds.), Advances in Neural Information Processing Systems 4. Morgan Kaufmann, San Mateo, CA.

Murata, N., Yoshizawa, S., Amari, S., 1994. Network information criterion—Determining the number of hidden units for an artificial neural network model. IEEE Transactions on Neural Networks 5, 865–872.

Muttiah, R.S., Srinivasan, R., Allen, P.M., 1997. Prediction of two-year peak stream discharges using neural networks. Journal of the American Water Resources Association 33 (3), 625–630.

Narendra, K.S., Parthasarathy, K., 1990. Identification and control of dynamical systems using neural networks. IEEE Transactions on Neural Networks 1, 4–27.

Nilsson, N., 1980. Principles of Artificial Intelligence. Springer Verlag, New York.

Nunnari, G., Nucifora, A.F.M., Randieri, C., 1998. The application of neural techniques to the modelling of time-series of atmospheric pollution data. Ecological Modelling 111 (2–3), 187–205.

Osowski, S., Bojarczak, P., Stodolski, M., 1996. Fast second order learning algorithm for feedforward multilayer neural networks and its applications. Neural Networks 9 (9), 1583–1596.

Parisi, R., Di Claudio, E.D., Orlandi, G., Rao, B.D., 1996. A generalized learning paradigm exploiting the structure of feedforward neural networks. IEEE Transactions on Neural Networks 7 (6), 1451–1460.

Plaut, D.C., Hinton, G.E., 1987. Learning sets of filters using backpropagation. Comput. Speech Language 2, 35–61.

Poff, N.L., Tokar, S., Johnson, P., 1996. Stream hydrological and ecological responses to climate change assessed with an artificial neural network. Limnology and Oceanography 41 (5), 857–863.

Prechelt, L., 1997. Connection pruning with static and adaptive pruning schedules. Neurocomputing 16 (1), 49–61.
Raman, H., Sunilkumar, N., 1995. Multivariate modelling of water resources time series using artificial neural networks. Hydrological Sciences Journal 40 (2), 145–163.
Recknagel, F., 1997. ANNA—Artificial neural network model for predicting species abundance and succession of blue-green algae. Hydrobiologia 349, 47–57.
Recknagel, F., French, M., Harkonen, P., Yabunaka, K.-I., 1997. Artificial neural network approach for modelling and prediction of algal blooms. Ecological Modelling 96, 11–28.
Reed, R., 1993. Pruning algorithms—A review. IEEE Transactions on Neural Networks 4 (5), 740–747.
Refenes, A., Burgess, A.N., Bents, Y., 1997. Neural networks in financial engineering: A study in methodology. IEEE Transactions on Neural Networks 8 (6), 1223–1267.
Ripley, B.D., 1994. Neural networks and related methods of classification. Journal of the Royal Statistical Society B 56 (3), 409–456.
Roadknight, C.M., Balls, G.R., Mills, G.E., Palmer-Brown, D., 1997. Modeling complex environmental data. IEEE Transactions on Neural Networks 8 (4), 852–862.
Robbins, H., Monro, S., 1951. A stochastic approximation method. Annals of Mathematical Statistics 22, 400–407.
Rogers, L.L., Dowla, F.U., 1994. Optimization of groundwater remediation using artificial neural networks with parallel solute transport modeling. Water Resources Research 30 (2), 457–481.
Rojas, R., 1996. Neural Networks: A Systematic Introduction. Springer-Verlag, Berlin.
Rumelhart, D.E., Hinton, G.E., Williams, R.J., 1986a. Learning internal representations by error propagation. In: Rumelhart, D.E., McClelland, J.L. (Eds.), Parallel Distributed Processing. MIT Press, Cambridge.
Rumelhart, D.E., Hinton, G.E., Williams, R.J., 1986b. Learning representations by backpropagating errors. Nature 323, 533–536.
Sarle, W.S., 1994. Neural networks and statistical models. In: Proceedings of the Nineteenth Annual SAS Users Group International Conference, pp. 1538–1550. SAS Institute.
Schwarz, G., 1978. Estimating the dimension of a model. Annals of Statistics 6, 461–464.
Setiono, R., 1997. A penalty-function approach for pruning feedforward neural networks. Neural Computation 9 (1), 185–204.
Setiono, R., Hui, L.C.K., 1995. Use of a quasi-Newton method in a feedforward neural-network construction algorithm. IEEE Transactions on Neural Networks 6, 273–277.
Shamseldin, A.Y., 1997. Application of a neural network technique to rainfall–runoff modelling. Journal of Hydrology 199, 272–294.
Shanno, D.F., 1978. Conjugate gradient methods with inexact line searches. Mathematics of Operations Research 3, 244–256.
Shukla, M.B., Kok, R., Prasher, S.O., Clark, G., Lacroix, R., 1996. Use of artificial neural networks in transient drainage design. Transactions of the ASAE 39 (1), 119–124.
Siegelmann, H.T., Horne, B.G., Giles, C.L., 1997. Computational capabilities of recurrent NARX neural networks. IEEE Transactions on Systems Man and Cybernetics, Part B: Cybernetics 27 (2), 208–215.
Sietsma, J., Dow, R.J.F., 1988. Neural net pruning—Why and how. In: Proceedings of the IEEE International Conference on Neural Networks, San Diego, CA, pp. 325–333. IEEE, New York.
Sietsma, J., Dow, R.J.F., 1991. Creating artificial neural networks that generalize. Neural Networks 4, 67–79.
Singhal, S., Wu, L., 1989. Training feedforward networks with the extended Kalman algorithm. In: Proceedings of the International Conference on Acoustics, Speech and Signal Processing, Scotland, pp. 1187–1190. IEEE, New York.
Smith, J., Eli, R.N., 1995. Neural-network models of rainfall–runoff processes. Journal of Water Resources Planning and Management 121 (6), 499–508.
Solla, S.A., Levin, E., Fleisher, M., 1988. Accelerated learning in layered neural networks. Complex Systems 2, 625–639.
Stone, M., 1974. Cross-validatory choice and assessment of statistical predictions. Journal of the Royal Statistical Society B 36, 111–147.
Sureerattanan, S., Phien, H.N., 1997. Back-propagation networks for daily streamflow forecasting. Water Resources Journal December, 1–7.
Tamura, S.i., Tateishi, M., 1997. Capabilities of a four-layered feedforward neural network: Four layers versus three. IEEE Transactions on Neural Networks 8 (2), 251–255.
Tawfik, M., Ibrahim, A., Fahmy, H., 1997. Hysteresis sensitive neural network for modeling rating curves. Journal of Computing in Civil Engineering 11 (3), 206–211.
Thirumalaiah, K., Deo, M.C., 1998a. Real-time flood forecasting using neural networks. Computer-Aided Civil and Infrastructure Engineering 13 (2), 101–111.
Thirumalaiah, K., Deo, M.C., 1998b. River stage forecasting using artificial neural networks. Journal of Hydrologic Engineering 3 (1), 26–32.
Tibshirani, R., 1994. Comment on 'Neural networks: A review from a statistical perspective' by B. Cheng and D.M. Titterington. Statistical Science 9 (1), 48–49.
Towell, G.G., Craven, M.K., Shavlik, J.W., 1991. Constructive induction in knowledge-based neural networks. In; Proceedings of the 8th International Workshop on Machine Learning, pp. 213–217. Morgan Kaufman, San Mateo.
Tsintikidis, D., Haferman, J.L., Anagnostou, E.N., Krajewski, W.F., Smith, T.F., 1997. A neural network approach to estimating rainfall from spaceborne microwave data. IEEE Transactions on Geoscience and Remote Sensing 35 (5), 1079–1093.
Venkatesan, C., Raskar, S.D., Tambe, S.S., Kulkarni, B.D., Keshavamurty, R.N., 1997. Prediction of all India summer monsoon rainfall using error-back-propagation neural networks. Meteorology and Atmospheric Physics 62 (3–4), 225–240.
Verma, B., 1997. Fast training of multilayer perceptrons. IEEE Transactions on Neural Networks 8 (6), 1314–1320.
Vicens, G.J., Rodriguez-Iturbe, I., Schaake, J.C., 1975. A Bayesian framework for the use of regional information in hydrology. Water Resources Research 11 (3), 405–414.
Vitela, J.E., Reifman, J., 1993. Enhanced backpropagation training algorithm for transient event identification. Transactions of the American Nuclear Society 69, 148–149.
Vitela, J.E., Reifman, J., 1997. Premature saturation in backpropagation networks—mechanism and necessary conditions. Neural Networks 10 (4), 721–735.
Von Lehman, A., Paek, E.G., Liao, P.F., Marrakchi, A., Patel, J.S., 1988. Factors influencing learning by back-propagation. In: Proceedings of the IEEE International Conference on Neural Networks, San Diego, CA, pp. 335–341. IEEE, New York.
Wang, G.-J., Chen, C.-C., 1996. A fast multilayer neural-network training algorithm based on the layer-by-layer optimizing procedures. IEEE Transactions on Neural Networks 7 (3), 768–775.
Warner, B., Misra, M., 1996. Understanding neural networks as statistical tools. American Statistician 50 (4), 284–293.
Weigend, A.S., Rumelhart, D.E., Huberman, B.A., 1990. Predicting the future: A connectionist approach. International Journal of Neural Systems 1 (3), 193–209.
Wessels, L., Barnard, E., 1992. Avoiding false local minima by proper initialization of connections. IEEE Transactions on Neural Networks 3 (6), 899–905.
White, H., 1989. Learning in artificial neural networks: A statistical perspective. Neural Computation 1, 425–464.
Whitehead, P.G., Howard, A., Arulmani, C., 1997. Modelling algal growth and transport in rivers—a comparison of time series analysis, dynamic mass balance and neural network techniques. Hydrobiologia 349, 39–46.
Whitley, R., Hromadka, T.V., 1999. Approximate confidence intervals

for design floods for a single site using a neural network. Water Resources Research 35 (1), 203–210.

Williams, R.J., Zipser, D., 1989. A learning algorithm for continually running fully recurrent networks. Neural Computation 1, 270–280.

Xiao, R.R., Chandrasekar, V., 1997. Development of a neural network based algorithm for rainfall estimation from radar observations. IEEE Transactions on Geoscience and Remote Sensing 35 (1), 160–171.

Yabunaka, K.-i., Hosomi, M., Murakami, A., 1997. Novel application of back-propagation artificial neural network model formulated to predict algal bloom. Water Science and Technology 36 (5), 89–97.

Yang, C.C., Prasher, C.O., Lacroix, R., 1996. Applications of artificial neural networks to land drainage engineering. Transactions of the ASAE 39 (2), 525–533.

Yang, L., Yu, W., 1993. Backpropagation with homotopy. Neural Computation 5 (3), 363–366.

Yao, X., 1993. A review of evolutionary artificial neural networks. Int. J. Intell. Syst. 8 (4), 539–567.

Yao, X., Liu, Y., 1997. A new evolutionary system for evolving artificial neural networks. IEEE Transactions on Neural Networks 8 (3), 694–713.

Yu, X.H., Chen, G.A., 1997. Efficient backpropagation learning using optimal learning rate and momentum. Neural Networks 10 (3), 517–527.

Zhu, M.-L., Fujita, M., Hashimoto, N., 1994. Application of neural networks to runoff prediction. In: Hipel, K.W., McLeod, A.I., Panu, U.S., Singh, V.P. (Eds.), Stochastic and Statistical Methods in Hydrology and Environmental Engineering. Kluwer Academic, Dordrecht.