Available online at www.sciencedirect.com

SCIENCE DIRECT®

ELSEVIER

Journal of Hydrology 301 (2005) 75–92

Journal of Hydrology

www.elsevier.com/locate/jhydrol

Input determination for neural network models in water resources applications. Part 1—background and methodology

Gavin J. Bowden[a], Graeme C. Dandy[b], Holger R. Maier[b,*]

[a]Division of Engineering and Applied Sciences, Harvard University, 29 Oxford Street, Pierce Halt 110J, Cambridge, MA 02138, USA
[b]Centre for Applied Modelling in Water Engineering, School of Civil and Environmental Engineering, The University of Adelaide, Adelaide 5005, Australia

Received 29 April 2003; revised 28 May 2004; accepted 15 June 2004

Abstract

The use of artificial neural network (ANN) models in water resources applications has grown considerably over the last decade. However, an important step in the ANN modelling methodology that has received little attention is the selection of appropriate model inputs. This article is the first in a two-part series published in this issue and addresses the lack of a suitable input determination methodology for ANN models in water resources applications. The current state of input determination is reviewed and two input determination methodologies are presented. The first method is a *model-free* approach, which utilises a measure of the mutual information criterion to characterise the dependence between a potential model input and the output variable. To facilitate the calculation of dependence in the case of multiple inputs, a partial measure of the mutual information criterion is used. In the second method, a self-organizing map (SOM) is used to reduce the dimensionality of the input space and obtain independent inputs. To determine which inputs have a significant relationship with the output (dependent) variable, a hybrid genetic algorithm and general regression neural network (GAGRNN) is used. Both input determination techniques are tested on a number of synthetic data sets, where the dependence attributes were known a priori. In the second paper of the series, the input determination methodology is applied to a real-world case study in order to determine suitable model inputs for forecasting salinity in the River Murray, South Australia, 14 days in advance.
© 2004 Elsevier B.V. All rights reserved.

Keywords: Artificial neural networks; Input determination; Self-organizing map; Genetic algorithm; Mutual information; General regression neural network

1. Introduction

Artificial neural networks (ANNs) have been used in a wide variety of hydrologic contexts (ASCE Task Committee on Application of Artificial Neural Networks in Hydrology, 2000b; Dawson and Wilby, 2001; Maier and Dandy, 2000), ranging from rainfall-runoff models (e.g. Minns and Hall, 1996; Tokar and Johnson, 1999) to the development of ANNs for temporal rainfall disaggregation (Burian et al., 2001). One of the most important steps in the ANN development process is the determination of significant input variables. Usually, not all of the potential

* Corresponding author.
E-mail addresses: gbowden@deas.harvard.edu (G.J. Bowden), gdandy@civeng.adelaide.edu.au (G.C. Dandy), hmaier@civeng.adelaide.edu.au (H.R. Maier).

0022-1694/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.jhydrol.2004.06.021

input variables will be equally informative since some may be correlated, noisy or have no significant relationship with the output variable being modelled. Despite this, in most water resources ANN applications, very little attention is given to the task of selecting appropriate model inputs (Maier and Dandy, 2000). This is primarily because ANNs belong to the class of data driven approaches, whereas conventional statistical methods are model driven (Chakraborty et al., 1992). In the latter, the model's structure is determined a priori by using empirical or analytical approaches, before estimating the unknown model parameters, whereas data driven approaches are usually assumed to be able to determine which model inputs are critical. This has meant that ANN practitioners often present a large number of inputs to the ANN and rely on the network to identify the critical model inputs. There are a number of shortcomings associated with this approach, including (Back and Trappenberg, 1999; Maier and Dandy, 1997; Zheng and Billings, 1996):

- As input dimensionality increases, the computational complexity and memory requirements of the model increase.
- Learning becomes more difficult with irrelevant inputs.
- Misconvergence and poor model accuracy may result from the inclusion of irrelevant inputs due to an increase in the number of local minima present in the error surface.
- Understanding complex models is more difficult than understanding simple models that give comparable results.
- Due to the curse of dimensionality, some types of ANN models with many irrelevant inputs behave poorly since the network uses almost all its resources to represent irrelevant portions of the input-output mapping. Other types of networks that can efficiently concentrate on important regions of the input space require more data to efficiently estimate the connection weights when irrelevant inputs are included.

Consequently, there are obvious advantages in using analytical procedures to select an appropriate set of inputs for ANN models. The challenge of input determination is to select a subset of inputs from all potential inputs that will lead to a superior model as measured by some optimality criterion. For d potential inputs, there are 2^d-1 input subsets, hence, it is possible to test all subset combinations for small values of d, but for large values of d, as is often the case for complex problems, efficient algorithms are required. The problem is further exacerbated in time series studies, where appropriate lags must also be chosen. The difficulty lies in determining how many lagged values to include from each input time series. In general, if there are n input time series ($x_{j,t-1}$, $x_{j,t-2},\ldots,x_{j,t-N}$, $j=1,2,\ldots,n$), then the problem is finding the maximum lag for each input time series (k_j: $k_{j,\max}<N, j=1,2,\ldots,n$) beyond which, values of the input time series have no significant effect on the output time series. The maximum lag is also called the memory length. As the memory length increases, so too does the number of inputs and the complexity of the ANN model.

The objective of this article is to present a methodology that can be used for selecting the significant inputs to an ANN model. In developing this methodology, a review of the current input determination techniques used in hydrologic applications of ANN models has been conducted. The advantages and limitations of the techniques used in the past are addressed and used to formulate two input determination methods applicable to all water resources case studies. The efficacy of the proposed methods is determined by applying them to a number of test problems. In the second paper of this two-part series, the proposed methods are applied to a hydrologic case study involving forecasting salinity in a river environment.

2. Review of input determination in water resources ANN applications

Maier and Dandy (2000) reviewed 43 journal papers (published up until the end of 1998) on the application of ANNs for modelling water resources variables and found that, in many cases, the lack of a methodology for determining input variables raised doubt about the optimality of the inputs obtained. In some instances, inputs were chosen arbitrarily. In other cases, a priori knowledge was used for input selection and, when different approaches such as trial-and-error were employed, often the validation data were used as part of the training process. More recently, the importance of input determination has begun to be recognised in the water resources literature, however, there are still only a small number of papers that treat it as an important step in the modelling process. The main approaches that have been employed for input determination in the water resources ANN modelling literature can be broadly classified into five methods. These include:

1. *Methods that rely upon the use of a priori knowledge of the system being modelled.* Usually some degree of a priori knowledge is used to specify the initial set of candidate inputs. This has been acknowledged by ASCE Task Committee on Application of Artificial Neural Networks in Hydrology (2000a), who pointed out, that in hydrological applications of ANNs, a good level of understanding of the hydrologic system being modelled is very important in order to select appropriate model inputs. If important candidate inputs are not included, then some information about the system may be lost. On the other hand, if spurious inputs are included, it may tend to confuse the training process. Many of the papers reviewed relied solely on the use of a priori knowledge to select the appropriate model inputs (e.g. Campolo et al., 1999; Jayawardena et al., 1997; Thirumalaiah and Deo, 2000). Although a priori identification is widely used in many ANN applications and is necessary to define a candidate set of inputs, it is dependent on an expert's knowledge, and hence, is very subjective and case dependent. Intuitively, the preferred approach for determining appropriate inputs and lags of inputs, involves a combination of a priori knowledge and analytical approaches (Fernando and Jayawardena, 1998; Maier and Dandy, 1997; Maier et al., 1998).

2. *Methods based on linear cross-correlation.* When the relationship to be modelled is not well understood, then an analytical technique, such as cross-correlation, is often employed (e.g. Golob et al., 1998; Huang and Foo, 2002; Luk et al., 2000; Sajikumar and Thandaveswara, 1999; Silverman and Dracup, 2000). Cross-correlation methods represent the most popular analytical techniques for selecting appropriate inputs. For example, Imrie et al. (2000) developed flow forecasting ANNs for two catchments in the UK and used cross-correlation analysis to identify suitable lags of time series from upstream gauging stations as inputs to the ANN model. Sajikumar and Thandaveswara (1999) used the cross-correlogram of rainfall and runoff to determine the linear component of the memory length of rainfall required by their flow forecasting model. Maier and Dandy (1997) used the method of Haugh and Box (1977), which uses cross-correlation analysis to determine the strength of the relationship between the input time series and the output time series at various lags. Other papers reporting the use of cross-correlation to select ANN inputs include: Huang and Foo (2002), Golob et al. (1998), Brion et al. (2001), Lekkas et al. (2001), and Coulibaly et al. (2000). The major disadvantage associated with using cross-correlation is that it is only able to detect linear dependence between two variables. Therefore, cross-correlation is unable to capture any nonlinear dependence that may exist between the inputs and the output, and may possibly result in the omission of important inputs that are related to the output in a nonlinear fashion.

3. *Methods that utilise a heuristic approach.* Heuristic methods are also commonly used in the literature. In this approach, various ANN models are trained using different subsets of inputs. For example, Jain et al. (1999) developed a univariate ANN for reservoir inflow prediction. To select a suitable number of lags, ANNs consisting of different numbers of lags were trialled until the best model was identified. In a similar approach, Jain and Chalisgaonkar (2000) tried different combinations of stage and discharge inputs in order to model river-rating curves.

Another commonly used heuristic approach is stepwise selection of inputs. This method is often employed to avoid total enumeration i.e. the consideration of all subsets. The two standard stepwise approaches are forward selection and backward selection. Forward selection is the more commonly used approach and begins by finding the best single input and selecting it for the final model. In each subsequent step, given a set of selected inputs, the input that improves the model's performance most is added to the final model. Backward elimination starts with a set of all inputs, and sequentially deletes the input that reduces performance the least.

Maier et al., (1998) used a forward stepwise selection method to determine the important input variables for forecasting the concentration of the cyanobacteria *Anabaena* spp. in the River Murray, Australia. Tokar and Johnson (1999) also used a forward stepwise method to determine the inputs for ANN models to forecast daily runoff as a function of daily precipitation, temperature, and snowmelt for the Little Patuxent River watershed in Maryland, USA. The main disadvantage of these approaches is that they are based on trial-and-error, and as such, there is no guarantee that they will find the globally best subsets. Another disadvantage of stepwise approaches is that they are computationally intensive.

4. *Methods that extract knowledge contained within trained ANNs*. Sensitivity analyses are the most commonly used method of extracting information from a trained ANN (e.g. Liong et al., 2000; Maier and Dandy, 1996, 1997; Maier et al., 1998; Schleiter, 1999). Plots of the sensitivities for each of the inputs are usually inspected and the significant inputs are chosen using judgement. However, the difficulty with this approach is choosing a reasonable value to perturb the input by and selecting the appropriate cut-off point for input significance.

Abrahart et al. (2001) used a different approach known as saliency analysis to disaggregate a neural network solution in terms of its forecasting inputs. In this approach, the effect that each input has on the error function is examined by removing one input at a time from the trained ANN model. The ANN model investigated was a one-step-ahead flow forecasting model for the Upper River Wye in Central Wales. The saliency analysis was achieved by setting one input data stream at a time to zero, performing the forecasting using the trained ANN, replacing the input data stream after the computation and repeating the process on the next input and so on. In this way, the relative importance of each input was determined by examining the change in forecasting error and plots of the flood hydrograph. Abrahart et al. (2001) stated that this approach differs to sensitivity analysis in that it does not investigate the rate of change of one factor with respect to the change in another, but rather, disaggregates the solution in a piecemeal fashion to determine the most influential input variable and the relevant importance of each input. However, the main disadvantage of their approach is that they did not retrain the ANN after removing each input. It is possible that setting the input to zero may produce nonsensical outputs if zero is not a reasonable value for that input in the input-output pattern under investigation. As pointed out by Sarle (1997), this is particularly the case if the inputs are statistically dependent, because the effects of different inputs cannot generally be separated.

5. *Methods that use various combinations of the above four approaches*. Many papers report the combined use of some of the above approaches. For example, Schleiter (1999) used a number of input determination methods including Pearson correlation, stepwise forward regression analysis and sensitivity analysis to select appropriate inputs for the modelling of water quality, bioindication and population dynamics in lotic streams. Silverman and Dracup (2000) used a priori knowledge of the system under investigation and combined this with a trial-and-error approach to determine a suitable set of inputs for long-range forecasts of precipitation in California using ANNs.

Some alternative methods for ANN input determination that could not be classified into the above five categories have also been used in the water resources literature. For example, Braddock et al. (1998) used the input identification methodology introduced by Lachtermacher and Fuller (1994) and extended this methodology to the multivariate case. In this method, a transfer function, as described by Box and Jenkins (1976), is fitted to the data to determine if a Box-Cox transformation of the data is required and to determine the number of lagged dependent and independent variables that should be used. However, this has the major limitation of only capturing the inputs that are linearly correlated with the output. In another approach, Abrahart et al. (1999) used a genetic algorithm to optimise the inputs to an ANN model used to forecast runoff from a small catchment.

3. Proposed methods

3.1. Introduction

Input determination can be divided into two broad stages. In the first stage, the objective is to reduce the dimensionality of the original set of inputs, resulting in a set of independent inputs, which are not necessarily related to the model output. This subset of inputs can then be used in the second stage to determine which of these inputs are related in some way to the output. The type of methods commonly used in the first stage of input preprocessing can be categorised as *model-free*, meaning that they do not depend in any way on a pre-existing model.

In the second stage, where the aim is to find those inputs exhibiting a significant relationship with the output, the methods commonly used include both *model-free* and *model-based* approaches. In model-based approaches, the input selection problem can be formulated as having a set of input variables to a model, and an output value, which can be used to evaluate the fitness or merit of the model using those variables. For example, for the problem of forecasting river streamflow, the inputs to a model may be variables such as precipitation, air temperature, snowmelt equivalent, and streamflow itself. Since a river is a dynamic environment, lagged inputs of each of these variables should also be included in the model. From these inputs, a subset of inputs must be selected that yield the model of highest fitness. This subset of inputs forms a n-dimensional input vector $\mathbf{X}^n$, which can be used to forecast the streamflow Y. The aim of the model is to produce a relationship of the form:

$$Y = f(\mathbf{X}^n) + e \tag{1}$$

where e is the model error term to be minimised. The model's fitness can be determined using an appropriate measure of the model error e.g. smallest root mean square prediction error (RMSE). Therefore, by using an efficient search procedure, the most suitable input subset $\mathbf{X}^n$ can be identified as the input set that produces the model with the smallest RMSE.

In *model-free* approaches, instead of using a model to characterise the relationship between input and output, some statistical measure of dependence is used (e.g. correlation). If the dependence between input and output is found to be significant, the input is retained in the final subset, otherwise it is discarded as it is unlikely to improve predictive ability when used in a model.

In the present study, the following two input determination methods are proposed for finding an appropriate set of inputs to an ANN model:

1. The stepwise PMI algorithm, which is a model-free approach. This method utilises a stepwise procedure based on the concept of partial mutual information (PMI). It does not require the variables to be preprocessed (i.e. to ensure independence), as it is capable of dealing with correlated input variables. Unlike correlation, this method can account for both linear and nonlinear dependence between variables.
2. The second method consists of two steps. In the first step, an unsupervised technique known as the self-organizing map (SOM) is used to reduce the dimensionality of the input space and ensure that the remaining inputs are independent. The SOM input reduction technique was devised in this study, as it is capable of dealing with both linear and nonlinear dependence between variables. Once redundant inputs have been removed, the second step is employed. In the second step, the input subset obtained from the SOM method is further refined using a model-based input selection method. The model-based method used in this study is a hybrid genetic algorithm and general regression neural network (GAGRNN).

3.2. Background

3.2.1. Partial mutual information algorithm

Nonlinear relationships are commonly encountered in water resources modelling and cannot be suitably detected or quantified by using methods based on the linear correlation between two variables (Harrold et al., 2001). The use of linear correlation methods in hydrological modelling can lead to fitted models that do not adequately represent the system they are attempting to model. Recently, Sharma (2000) proposed an input determination method based on the partial mutual information (PMI) criterion to overcome the limitation of the correlation coefficient in selecting appropriate model inputs. The PMI criterion is able to adequately capture all dependence (linear and/or nonlinear) between two variables and avoids the need for making any major assumptions about the underlying model structure and as such, can be considered a *model-free* approach. The PMI criterion is an extension of the concept of mutual information.

Mutual information (MI) measures the dependence between two random variables (Yang et al., 2000). The mutual information function between variables X and Y is given by

$$\mathrm{MI} = \int\int f_{X,Y}(x,y)\ln\left[\frac{f_{X,Y}(x,y)}{f_X(x)f_Y(y)}\right]\mathrm{d}x\,\mathrm{d}y \tag{2}$$

where $f_X(x)$ and $f_Y(y)$ are the marginal PDFs of X and Y, respectively, and $f_{X,Y}(x,y)$ is the joint (bivariate) PDF of X and Y. Mutual information measures the reduction in uncertainty of Y due to knowledge of the variable X (Cover and Thomas, 1991). If there is no dependence between X and Y, then the two random variables are statistically independent and by definition the joint probability density $f_{X,Y}(x,y)$ would equal the product of the marginal densities ($f_X(x)f_Y(y)$). In this case, the MI score in Eq. (2) would equal a value of 0, since the ratio of the joint and marginal densities would equal unity and hence, the logarithm would equal 0. Conversely, if the random variables were strongly related, the MI score would take on a high value.

Given any bivariate sample, the discrete version of Eq. (2) can be used to estimate the MI score, and is given by

$$\mathrm{MI} = \frac{1}{N}\sum_{i=1}^{N}\ln\left[\frac{P_{X,Y}(x_i,y_i)}{P_X(x_i)P_Y(y_i)}\right] \tag{3}$$

where x_i and y_i are the ith bivariate sample pair in a sample of size N, and $P_X(x_i)$, $P_Y(y_i)$ and $P_{X,Y}(x_i,y_i)$ are the respective univariate and joint probability densities estimated at the sample data points. Using Eq. (3), the three probability distributions $P_X(x)$, $P_Y(y)$ and $P_{X,Y}(x,y)$ are used to describe, in probabilistic terms, the relationship between an input variable X and an output Y. However, the key to successful computation of the MI score in Eq. (3) is the accurate estimation of these probability distributions from a finite set of examples. The original MI algorithms utilised crude measures to approximate the probability densities, such as a histogram with fixed bin width (e.g. Fraser and Swinney, 1986; Zheng and Billings, 1996). More recently, kernel density estimation techniques have been used because of their stability, efficiency and robustness (Sharma, 2000). For example, the Gaussian kernel function (Scott, 1992) can be used, as given by

$$\hat{f}_X(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{(2\pi)^{d/2}\lambda^d\det(S)^{1/2}} \times\exp\left(-\frac{(\mathbf{x}-\mathbf{x}_i)^T\mathbf{S}^{-1}(\mathbf{x}-\mathbf{x}_i)}{2\lambda^2}\right) \tag{4}$$

where $\hat{f}_X(\mathbf{x})$ is the multivariate kernel density estimate of the d-dimensional variable set $\mathbf{X}$ at coordinate location $\mathbf{x}$; $\mathbf{x}_\mathrm{i}$ the ith multivariate data point, for a sample of size N; $\mathbf{S}$ is the sample covariance matrix of the variable set $\mathbf{X}$; $\det(\cdot)$ represents the determinant operation; and λ is the smoothing parameter, known as the 'bandwidth' of the kernel density estimate. In order to produce an accurate density estimate, λ must be chosen very carefully. Small values of λ tend to lead to density estimates that give too much emphasis to individual points. In this case, spurious fine structure may become present with bumps in the tails of the probability density. Larger values of λ tend to over-smooth the probability density with all detail, spurious or otherwise, becoming obscured. Only for intermediate values of λ will a good density estimate be obtained. Various rules-of-thumb are available in the literature to help choose an optimal value of the bandwidth λ. Sharma (2000) used the Gaussian reference bandwidth (Scott, 1992) because it is relatively simple and computationally efficient

$$\lambda_{\mathrm{ref}} = \left(\frac{4}{d+2}\right)^{1/(d+4)} N^{(-1/(d+4))} \tag{5}$$

where N and d are the sample size and dimension of the multivariate variable set, respectively. The Gaussian reference bandwidth has also been used in this study.

The MI criterion can be used to identify appropriate model inputs but is not directly able to deal with the issue of redundant inputs. For example, if an input x has a strong dependence relationship with the output variable y as indicated by a high MI score, then a second input $z=2x$ will also have a high MI score. Simply using the MI score, the new variable z would be considered as a significant input, however, z is a redundant input because it is already described by the pre-existing input x. To avoid this problem, Sharma (2000) introduced the concept of partial mutual information (PMI), which provides a measure of the partial or additional dependence the new input can add to the existing prediction model. The PMI between the output (dependent variable) y and the input (independent variable) x, for a set of pre-existing inputs z, can be given by

$$\mathrm{PMI} = \int\int f_{X',Y'}(x',y')\ln\left[\frac{f_{X',Y'}(x',y')}{f_{X'}(x')f_{Y'}(y')}\right]\mathrm{d}x'\,\mathrm{d}y' \qquad (6)$$

where

$$x' = x - E[x|z] \quad y' = y - E[y|z] \qquad (7)$$

where $E[\cdot]$ denotes the expectation operation. By using the conditional expectations in Eq. (7), the variables x' and y' only contain the residual information after the effect of the pre-existing set of inputs z has been taken into consideration.

The discrete version of Eq. (6) can be used to provide a sample estimate of the PMI score, and is given by

$$\mathrm{PMI} = \frac{1}{N}\sum_{i=1}^{N}\ln\left[\frac{P_{X',Y'}(x'_i,y'_i)}{P_{X'}(x'_i)P_{Y'}(y'_i)}\right] \qquad (8)$$

where x'_i and y'_i are the ith residuals in a sample of size N, and $P_{X'}(x'_i)$, $P_{Y'}(y'_i)$ and $P_{X',Y'}(x'_i,y'_i)$ are the respective marginal and joint probability densities.

3.2.2. *Self-organizing map*

The self-organizing map (SOM) (Kohonen, 1982) is a powerful technique capable of ordering multivariate data by similarity, while preserving the topological structure of the data. The underlying basis behind the development of the SOM was that topologically correct maps could be formed in an n-dimensional array of processing elements that did not have this initial ordering to begin with. In this way, input stimuli, which may have many dimensions, can come to be represented by a one- or two-dimensional vector, which preserves the order of the higher dimensional data. The SOM employs a type of learning commonly referred to as competitive, unsupervised or self-organizing, in which adjacent cells within the network are able to interact and develop adaptively into detectors of a specific input pattern (Kohonen, 1990).

In competitive learning, neurons in the network adapt gradually to become sensitive to different input categories. The SOM network generally consists of two layers, an input layer and a Kohonen layer (Fig. 1). The input layer is fully connected to the Kohonen layer, which in most common applications is two-dimensional. None of the processing elements (PEs) in the Kohonen layer are connected to each other. The PEs in the Kohonen layer measure the distance of their weights to the input pattern.

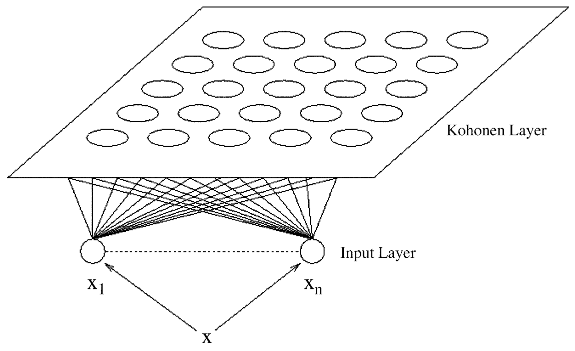

Fig. 1. A typical SOM consisting of n inputs in the input layer and a 5 by 5 Kohonen layer.

The procedure for determining the winning PE is as follows:

The first step is to determine the extent to which the weights of each PE match the corresponding input pattern. If the input data have n inputs and are denoted by

$$X = \{x_i;\ i = 1,\ldots,n\} \qquad (9)$$

then each of the M PEs in the Kohonen layer will also have n weight values and can be denoted by

$$W_{ji} = \{w_{ji}; j = 1,\ldots,M;\ i = 1,\ldots,n\} \qquad (10)$$

For each of the M Kohonen PEs, the distance, such as the Euclidean distance, is calculated using

$$D_j = ||X - W_j|| = \left[\sum_{i=1}^{n}(x_i - w_{ji})^2\right]^{\frac{1}{2}}, \quad j = 1,\ldots,M. \qquad (11)$$

The PE with the lowest value of D_j is the winner during recall. To ensure biological plausibility, lateral interaction with neighbouring PEs is enforced by applying arbitrary network structures called neighbourhood sets, N_c. Throughout the process, all PEs within the winner's neighbourhood set will have their weights updated, whilst PEs outside of this set are left intact. The width or radius of N_c can be time variable.

The updating process to implement this procedure is given by

$$W_j(t+1) = \begin{cases} W_j(t) + \alpha(t)(X(t) - W_j(t)) & \text{if } j \in N_c(t) \\ W_j(t) & \text{if } j \notin N_c(t) \end{cases} \tag{12}$$

where α is a scalar valued adaptation gain $0<\alpha(t)<1$ and N_c is the neighbourhood set. After the weights have been updated, the next input is presented to the network and the process continues until convergence has been reached. After successively presenting different inputs to the SOM, the net effect is that the weights reflect the topological relationship that exists within the input data (Islam and Kothari, 2000).

3.2.3. Genetic algorithm

A genetic algorithm (GA) is a powerful optimisation technique that is based upon the underlying principles of natural evolution and selection (Goldberg, 1989). GAs have already been used successfully as an optimisation tool in a number of water resources case studies (e.g. Dandy et al., 1996; Liong et al., 2001). GAs are well suited to the task of selecting an appropriate combination of inputs to a model as they have the ability to search through large numbers of combinations, where interdependencies between variables may exist.

The main steps in the GA optimisation process are shown in Fig. 2. A GA differs from many other optimisation methods as a population, or collection of possible solutions, is used rather than a single solution. The three genetic operators: selection, crossover (mating) and mutation are used to determine which subset of all possible combinations of decision variables to evaluate during the search procedure. The possible solutions are generally coded as binary strings and these strings are equivalent to biological chromosomes. Other non-binary codings have proven to be useful in some applications. Each bit of the binary string (chromosome) is referred to as a gene. The search is initialised with a random population of chromosomes, each representing a possible solution. Next, each chromosome in the population is decoded into a solution and its fitness is evaluated using an objective function.

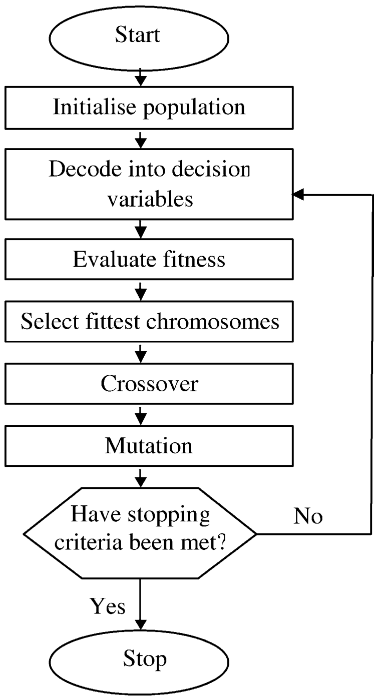
Start
Initialise population
Decode into decision variables
Evaluate fitness
Select fittest chromosomes
Crossover
Mutation
Have stopping criteria been met?
No
Yes
Stop

Fig. 2. Steps in a genetic algorithm.

Based on the fitness of each chromosome, a selection procedure is then used to determine which chromosomes will have the opportunity to mate during the process of crossover. Various selection schemes exist, however, a commonly used approach is tournament selection, where chromosomes are paired randomly and the chromosome with the higher fitness in the pair progresses to the next generation (Goldberg and Deb, 1991). This scheme is referred to as binary or two-member tournament. Since only half of the chromosomes progress, another tournament is held with another set of random pairs and the winners make up the other half of the crossover pool. Two copies of the best chromosome progress and no copies of the worst chromosome are replicated.

Once the chromosomes have been selected, a crossover procedure is used to partially exchange genetic information between two parent chromosomes. Chromosomes from the parent pool are randomly paired up and are tested to determine if an exchange will take place based on a crossover probability. If an exchange is to take place, a crossover site is selected at random for the two chromosomes and the genetic material (genes) after the crossover site is then exchanged between the two parent strings. In so doing, two child chromosomes are produced, which form the members of a new population. If an exchange is not to take place (i.e. the crossover probability is less than the crossover probability parameter), then the two parents enter the new population unchanged. Other types of crossover have also been used in many applications, including two-point crossover and uniform crossover.

The final genetic operator is mutation, which has the purpose of keeping the population diverse and preventing the GA from prematurely converging onto a local minimum. Each chromosome is tested on a probability basis to determine whether or not it will be mutated. In the most commonly used form of mutation, the probability that each bit in the chromosome will be mutated is determined by the mutation probability parameter. If a bit is to be mutated, then this occurs by flipping its value (i.e. a 0 will become a 1 and vice versa). The application of the mutation operator marks the end of one GA cycle (or generation). The GA is usually allowed to run for a prespecified number of generations, or until some stopping criterion is met, such as convergence of the population to a single solution. A comprehensive description of GAs can be found in Goldberg (1989).

3.2.4. General regression neural network

The general regression neural network (GRNN) is a type of supervised feedforward ANN and has been shown to be a universal approximator for smooth functions (Sarle, 1997). Developed by Specht (1991), the GRNN does not require any particular form for the regression function to be assumed but rather, utilises a nonparametric estimate of the probability density function of the observed data. In so doing, GRNNs are very similar in their underlying philosophy to statistical models such as Nadaraya–Watson kernel regression (Sarle, 1994). The advantages of using a GRNN include:

- Nonlinear relationships between the inputs and output can be modelled.
- The network architecture is fixed and does not need to be determined.
- Training times are much quicker than other ANNs, such as multilayer perceptrons (MLPs) trained using the backpropagation algorithm.

The GRNN paradigm is briefly outlined below. Further information can be found in Specht (1991).

Assume a vector **x** of n independent, random variables is used to predict a dependent scalar random variable y. Let **X** be a particular measured value of the random variable **x**. If the joint density $f(\mathbf{X},y)$ is known, then it is possible to compute the conditional mean of y given **X** (or regression of y on **X**) by using Eq. (13)

$$E[y|X] = \frac{\int_{-\infty}^{\infty} y.f(\mathbf{X},y)\mathrm{d}y}{\int_{-\infty}^{\infty} f(\mathbf{X},y)\mathrm{d}y} \tag{13}$$

If, however, the joint density $f(\mathbf{X},y)$ is not known, then an estimate $\hat{f}(\mathbf{X},y)$ based on a sample of observations of **x** and y must be used. The GRNN utilises a class of consistent nonparametric estimators known as Parzen window estimators. Using Parzen window density estimation, Specht (1991) has shown that an estimate of the conditional mean, designated $\hat{Y}(\mathbf{X})$, can be written as

$$\hat{Y}(\mathbf{X}) = \frac{\sum_{i=1}^{N} Y^i \cdot \exp\left(-\frac{D_i^2}{2\sigma^2}\right)}{\sum_{i=1}^{N} \exp\left(-\frac{D_i^2}{2\sigma^2}\right)} \tag{14}$$

where σ is the smoothing parameter (sigma weight), N is the sample size and $D_i^2 = (\mathbf{X}-\mathbf{X}^i)^T(\mathbf{X}-\mathbf{X}^i)$. The regression in Eq. (14) is directly applicable to numerical data and can be easily implemented via four layers of parallel neural network architecture, as shown in Fig. 3. To implement Eq. (14), the A summation layer processing elements (PEs) in Fig. 3 have their weights set to the actual output values i.e. $A^i = Y^i$; $i = 1,\ldots N$ and the B summation layer PEs in Fig. 3 have their weights set to unity i.e. $B^i = 1$; $i = 1,\ldots N$. For the network to generalize well, the optimal sigma weight σ must be found empirically. The most common methods for determining a suitable value for the sigma weight are based on trial-and-error (e.g. Caudill, 1993). The curve of mean squared error (MSE) versus σ typically is very flat near the minimum, and hence, it is not difficult to choose a good value for σ (Specht, 1991). The speed at which a GRNN can be trained is a very

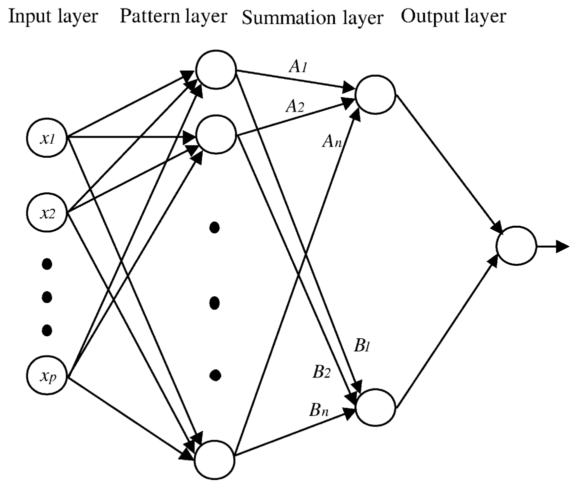

Fig. 3. The GRNN architecture.

important consideration because typically, many thousands of networks, each consisting of different subsets of inputs, need to be trained in the GA optimisation process.

3.3. Method 1: PMI algorithm for ANN input selection

An adaptation of the input determination approach used by Sharma (2000) is proposed as a means for determining inputs to ANNs. This approach uses the PMI criterion and proceeds as follows:

1. Identify the set of variables that could be useful inputs for the system being modelled. Denote this input set as z_{in}. Denote the vector that will store the final inputs of the system as z. This is a null vector at the start of the algorithm.
2. Estimate the PMI between the output variable y and each of the potential new inputs in z_{in}, conditional to the pre-existing input set z by using Eq. (8). To estimate the marginal and joint probability densities, the city block distance kernel was used, and is given by

$$\hat{f}_X(\mathbf{x}) = \frac{1}{N(2\lambda)^d}\sum_{i=1}^{N}\prod_{j=1}^{d} e^{-|\mathbf{x}_j-\mathbf{x}_{ij}|/\lambda} = \frac{1}{N(2\lambda)^d}\sum_{i=1}^{N}\exp\left[-\frac{1}{\lambda}\sum_{j=1}^{d}|\mathbf{x}_j-\mathbf{x}_{ij}|\right] \qquad (15)$$

where $\hat{f}_X(\mathbf{x})$ is the multivariate kernel density estimate of the d-dimensional variable set $\mathbf{X}$ at coordinate location $\mathbf{x}$, $\mathbf{x}_i$ the ith multivariate data point, for a sample of size N, and λ is the smoothing parameter, known as the bandwidth of the kernel density estimate. The univariate version of this kernel was first proposed by Parzen (1962). However, Theorem 4.1 in Cacoullous (1966) shows that Parzen's results can be extended to the multivariate case when the multivariate kernel is a product of a number of univariate kernels as is the case in Eq. (15). The use of the city block distance kernel differs from previous studies (e.g. Sharma, 2000), which have used the Gaussian kernel function (Eq. (4)). The main advantage of Eq. (15) is its computational simplicity, resulting in a reduction in the amount of processing time required.

To compute the conditional expectations $E[x|z]$ in Eq. (7) (which are then used in Eq. (8)), a GRNN is used. This is considered an improvement over previous approaches (e.g. Sharma, 2000), as the GRNN also uses the city block distance kernel function and therefore, also has the advantage of computational simplicity.

3. Identify the input in z_{in} having the highest PMI score in step 2.
4. Estimate the 95th percentile randomised sample PMI score for the input identified in step 3. This significance measure is used to indicate whether the selected input is a significant predictor of the system being modelled and was used in the original implementation by Sharma (2000). Independence is forced between the input under consideration and the output by using 100 randomly shuffled sequences (surrogates) of the input. These are created by rearranging or bootstrapping the input variable. By construction, these surrogates satisfy the null hypothesis of independence between the input and output and can be used to calculate the randomised sample PMI scores. The decision rule for selecting an input is that the sample PMI score for that input be greater than the estimated 95th percentile randomised sample PMI score. In other words, the null hypothesis of independence must be rejected for the input to be considered significant, and if this is the case, it is reasonable to expect that there is less

than 5% chance that the two variables are truly independent.

5. If the PMI score for the identified input is higher than the 95th percentile randomised sample PMI score of step 4, include the input in the input set z, and remove it from z_{in}. If the dependence is not significant, go to step 7.
6. Repeat steps 2–5 as many times as are needed.
7. Stop once all significant inputs have been selected.

3.4. Method 2: SOM–GAGRNN

3.4.1. Step 1: SOM

In the implementation used in this study, the SOM is used to cluster the input variables into groups of similar inputs. By then sampling one input from each cluster, it is possible to remove highly correlated, redundant variables from the original data set. There is no theoretical principle for determining the optimum size of the Kohonen layer (Cai et al., 1994), hence, the Kohonen layer should be kept large enough to ensure that a suitable number of clusters are formed from the training data. For each Kohonen layer PE containing a cluster, only one input is sampled. Once one input from each of these clusters is sampled, this set of selected inputs is then used in the GAGRNN input procedure. The criterion for choosing the representative input from each cluster is to choose the input with the smallest Euclidean distance to the cluster's weights.

The SOM input procedure is conducted for each variable, in order to find a subset of lags with limited cross-dependence. Next, the lags that have been identified for each variable are combined into one data set. The SOM input determination procedure is then conducted again, this time to ensure that there is limited cross-dependence between the variables selected.

3.4.2. Step 2: Hybrid GAGRNN

Substantial variation amongst the input variables does not necessarily imply any relationship with the output variable. Therefore, after the input dimensionality has been reduced using the SOM, a model-based technique (e.g. GAGRNN) can be used to ultimately decide upon which of the independent variables have the most significant relationship with the output.

For the problem of determining inputs to a GRNN, a population of randomly selected GRNNs is generated, each with a different subset of input variables as depicted by a binary string. Hence, for n variables the string will have length n. If the jth value of the string is equal to one, then the jth input is included, otherwise, if it equals zero, it is not included. The models are trained and the output of each GRNN is used to determine the predictive error, or fitness of the solution. Based on the fitness of each member in the population, a selection scheme can then be employed to create a new population for the next generation. In this way, fit solutions are allowed to live and subsequently breed in the process of crossover. In the crossover process, two parent strings are cut and part of the strings exchanged to produce two new individuals. To maintain genetic diversity and ensure that no important genetic material is overlooked, a mutation operation introduces small random changes. The process is continued in an iterative manner until the error from the GRNN model converges to an acceptable value or a prespecified maximum number of generations is completed. Due to the selective pressure applied over the generations, the overall trend is the evolution of chromosomes with higher fitness, representing improved input subsets.

4. Testing of proposed input determination approaches

4.1. City block distance kernel

A study was conducted to verify that the city block distance kernel used in the PMI algorithm was able to approximate a joint density to a suitable level of accuracy. To achieve this, the city block distance kernel was tested on a data set consisting of a pair of 500 random deviates from a standard bivariate normal distribution with zero correlation between the two variables. A bivariate probability density function $f(x,y)$, written for two normally distributed parameters x and y, is given by

$$f(x,y) = \frac{e^{-Q(1-\rho^2)/2}}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} \tag{16}$$

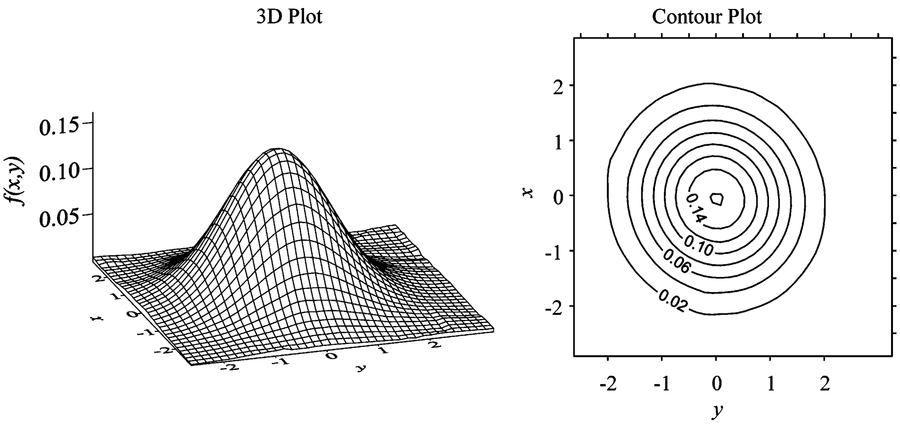

Fig. 4. Actual bivariate probability density between x and y.

where the variable Q is equal to

$$Q = \frac{(x-\mu_x)^2}{\sigma_x^2} + \frac{(y-\mu_y)^2}{\sigma_y^2} - 2\rho\frac{(x-\mu_x)(y-\mu_y)}{\sigma_x\sigma_y} \tag{17}$$

and σ_x is the standard deviation in parameter x, σ_y is the standard deviation in parameter y, and ρ is the correlation coefficient defined as

$$\rho = \frac{\sigma_{xy}}{\sigma_x\sigma_y} \tag{18}$$

where σ_{xy} is the covariance between parameters x and y. Using the 500 values of x and y, the actual bivariate probability density $f(x,y)$ was calculated using Eq. (16). Two types of plots of the probability density function are useful, a 3-dimensional plot and a contour plot (Fig. 4).

As is expected, Fig. 4 shows there is no correlation between the variables x and y as indicated by the perfectly concentric rings on the bivariate probability density contour plot. This data set formed the reference set with which to compare the estimated bivariate probability density $\hat{f}(x,y)$ from the Gaussian and city block distance kernels. For both kernels, the bandwidth was estimated using the Gaussian reference bandwidth (Eq. (5)). Fig. 5 shows the estimated bivariate probability density produced by the Gaussian kernel. From the 3-dimensional plot it can be seen that the overall density is reasonably well approximated. The contour plot shows that there is

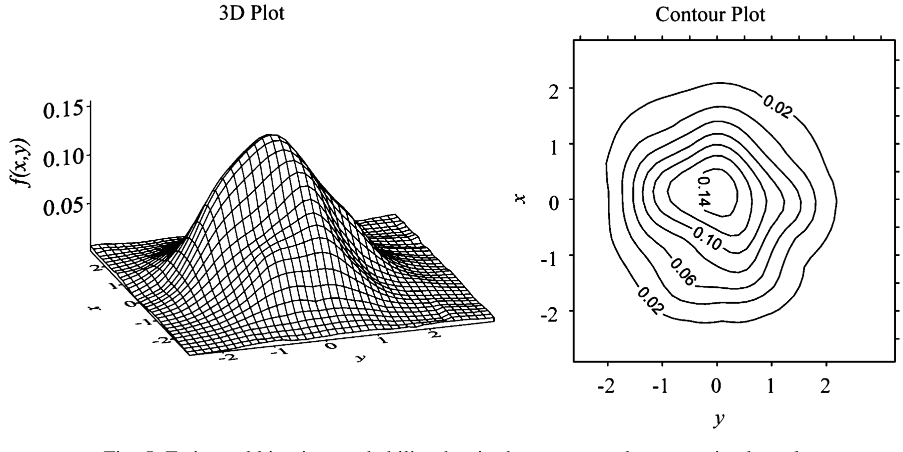

Fig. 5. Estimated bivariate probability density between x and y—gaussian kernel.

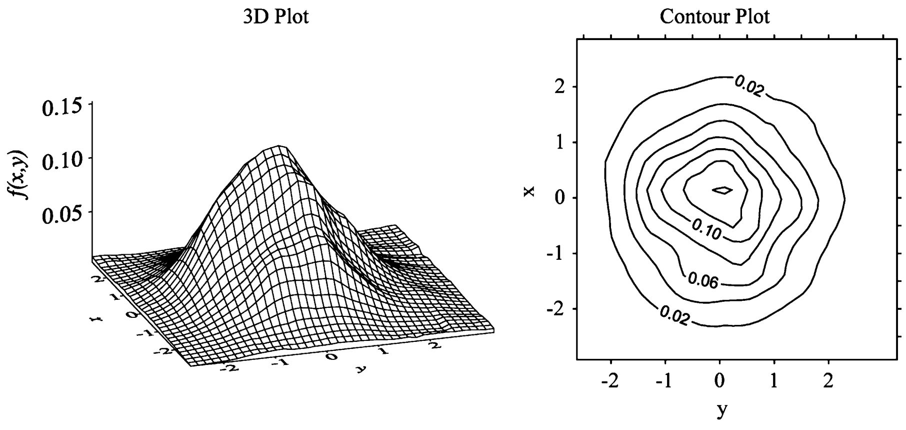

Fig. 6. Estimated bivariate probability density between x and y—city block distance kernel.

very little correlation between x and y, however, the rings are not perfectly concentric because it is only an estimate of the true probability density based on a sample of 500 values. The correlation coefficient between the actual bivariate normal probability density $f(x,y)$ and the estimated bivariate normal probability density $\hat{f}(x,y)$ is equal to 0.98, indicating a good approximation of the actual density.

Fig. 6 shows the estimated bivariate probability density produced by the city block distance kernel. From the 3-dimensional plot it can be seen that the overall density is also reasonably well approximated but is not as smooth as the estimate produced by the Gaussian kernel. The contour plot shows that there is very little correlation between x and y, but once again because it is only an estimate, the rings are not perfectly concentric. The estimate produced by the city block distance kernel is very similar to the estimate produced by the Gaussian kernel and the correlation coefficient between the actual bivariate normal probability density $f(x,y)$ and the estimated bivariate normal probability density $\hat{f}(x,y)$ is also equal to 0.98, indicating a good approximation of the actual density. Based on the similar approximation abilities of the Gaussian and city block distance kernels, it was decided to proceed with the city block distance kernel because of its computational simplicity and the consequent reduction in processing time required. For the bivariate normal test data, the city block distance kernel resulted in a 12.3% reduction in computational time when compared to the Gaussian kernel.

4.2. Input determination methods

4.2.1. Test problems

To test the applicability of the PMI algorithm and the SOM–GAGRNN, both methods were tested on four data sets with known dependence attributes. It was considered necessary to compare both of the input determination methods on synthetic data, before applying these methods to the real-world case study. The first three models used to generate the synthetic data are time series models used by Sharma (2000). The fourth model is a nonlinear system. The models are (a) AR1:

$$x_t = 0.9x_{t-1} + 0.866e_t \tag{19}$$

where e_t is a Gaussian random deviate with a zero mean and unit standard deviation.
(b) AR9

$$x_t = 0.3x_{t-1} - 0.6x_{t-4} - 0.5x_{t-9} + e_t \tag{20}$$

(c) TAR2—Threshold Autoregressive order 2

$$x_t = \begin{cases} -0.5x_{t-6} + 0.5x_{t-10} + 0.1e_t, & \text{if } x_{t-6} \le 0 \\ 0.8x_{t-10} + 0.1e_t, & \text{if } x_{t-6} > 0 \end{cases} \tag{21}$$

(d) Nonlinear system

$$y = (x_2)^3 + \cos(x_6) + 0.3\sin(x_9) \tag{22}$$

4.2.2. Data

For the first three time series models, 520 data points from each of the above synthetic models were generated with the first 20 points being discarded to reduce the effect of an arbitrary initialisation. The first 15 lags were chosen as potential model inputs. From these potential inputs, the final input subset was selected using each input determination technique. For the nonlinear system, 15 standard gaussian random variables were generated, each consisting of 500 data points. These 15 variables were chosen as the potential model inputs and the final input subset was selected using each input determination technique.

4.2.3. Implementation

In this study, the SOM was implemented using the *NCS NeuFrame* software. To cluster the data, the input variables were presented to the network as the SOM's inputs. The data were standardised to put each variable on an equal basis. The output of the SOM was obtained using a Dynamic Patterns grid, which showed a dynamic representation of the nodes that were winning each pattern. Each individual cell in the grid represents a node in the Kohonen layer. By then selecting one input from each cell (i.e. the input closest to the cluster's centre), redundant, highly correlated inputs were excluded.

To implement the PMI algorithm, the required software was developed using Fortran 77. To implement the GAGRNN, the commercially available software package, *NeuroGenetic Optimizer* (*NGO*) (BioComp Systems, 1998) was used in this research. The RMSE between the actual and predicted values was used to determine the fitness of each GRNN model.

4.2.4. Results and discussion

The results of the PMI algorithm applied to the four test problems are shown in Table 1. The bold numbers in the table represent the higher of the PMI and the PMI value of the 95th percentile randomised sample. The last row in each model's results represents the cutoff point, i.e. the input that was not included because the PMI value of the 95th percentile randomised sample was greater than the corresponding PMI score. From Table 1 it can be seen that the PMI input selection algorithm was able to select the correct inputs for each of the four models. This is an improvement over the original algorithm used by Sharma (2000), which selected one input in addition to what is necessary in the case of the two variable threshold autoregressive model (TAR2). Therefore, the PMI input selection algorithm proposed in this study can be considered a suitable means of correctly identifying the inputs to both linear as well as more complex nonlinear models.

Table 1
PMI input selection algorithm results on the synthetic data test problems

Model	Predictors	PMI	95th percentile randomised sample PMI
AR1	x_{t-1}	**0.604**	0.080
	x_{t-9}	0.029	**0.033**
AR9	x_{t-4}	**0.354**	0.080
	x_{t-9}	**0.204**	0.059
	x_{t-1}	**0.104**	0.044
	x_{t-3}	0.026	**0.034**
TAR2	x_{t-10}	**0.363**	0.098
	x_{t-6}	**0.077**	0.064
	x_{t-9}	0.053	**0.060**
Nonlinear	x_2	**0.558**	0.097
	x_9	**0.082**	0.031
	x_6	**0.072**	0.026
	x_{10}	0.009	**0.013**

An advantage of the PMI algorithm is that additional information about the system is also obtained, including the order of significance of the inputs and their relative significance. For example, for the AR9 model in Eq. (20), the autoregressive coefficients for lags 1, 4 and 9, are 0.3, −0.6 and −0.5, respectively. The PMI algorithm correctly selected the inputs in order of greatest significance, i.e. lag 4 first, lag 9 second and then lag 1 with PMI scores of 0.354, 0.204 and 0.104, respectively. As a result, important information about the system under investigation can be obtained when using the PMI algorithm.

Table 2 shows the subsets of inputs selected by the SOM–GAGRNN input determination procedures for each test problem, after (i) the SOM was applied, and (ii) the GAGRNN was applied. In this experiment, the GA had a population size of 50 and was run for 100 generations. It is important to note that the SOM was not used for the nonlinear system test problem since all 15-candidate inputs were already independent by construction.

Table 2 shows that the SOM was able to reduce the original dimensionality of the time series problems by clustering similar inputs together. However, since it is

Table 2
Input subsets selected using the SOM–GAGRNN method for the synthetic data test problems

Test problem	Input determination method	After unsupervised technique	After supervised technique
AR1	SOM–GAGRNN	Lags 1, 4, 6, 9, 12, 14	Lag 1
AR9	SOM–GAGRNN	Lags 1, 3, 5, 6, 7, 8, 9, 11	Lags 1, 3, 5, 7, 8, 9
TAR2	SOM–GAGRNN	Lags 1, 3, 4, 5, 6, 9	Lags 3, 4, 5, 6
Nonlinear	GAGRNN	Not used	Inputs 2, 14

Note: No unsupervised technique (i.e SOM) was used for the nonlinear system test problem since all inputs are independent by construction.

an unsupervised technique, there is no guarantee that the actual inputs to the test problem are included in the resulting subset after this technique has been applied. Only the input closest to the centre of the cluster is selected. Therefore, if the actual test problem input clusters with other inputs, it is possible that one of the other inputs may be selected instead of the actual test problem input. If these variables are closely related, then the effect of selecting either one should make little difference to the final model from a predictive sense. For the AR1 test problem, the actual input (i.e. lag 1) was included in the subset obtained after applying the SOM technique. The GAGRNN was able to successfully determine the actual input for this problem. For the AR9 test problem, only two of the three actual test problem inputs (i.e. lags 1 and 9) were included in the subset obtained after applying the SOM technique. The other remaining actual input (i.e. lag 4) clustered with lag 3, and since lag 3 was closest to the centre of the cluster, this lag was selected in preference to lag 4. This is because lag 3 had a smaller Euclidean distance to the cluster's weights than lag 4. After applying the GAGRNN to the subset of inputs obtained after the SOM technique, lags 1, 3, and 9 were included in the final subset of inputs, however, lags 5, 7, and 8 were also included. For the TAR2 test problem, only one of the two actual test problem inputs (i.e. lag 6) was included in the subset obtained after applying the SOM technique. The other actual input (i.e. lag 10) clustered with lag 9, and since lag 9 was closest to the centre of the cluster, this lag was selected in preference to lag 10. The GAGRNN was then applied to the subset of inputs obtained after the SOM technique and lags 3, 4, 5 and 6 were included in the final subset of inputs, however, lag 9 was not included. Finally, for the nonlinear system test problem, the GAGRNN only found one of the actual inputs (i.e. input 2) and also included input 14 in the final subset. This is most likely due to the fact that the nonlinear system in Eq. (22) is comprised of three nonlinear terms added together. Input 2 appears in the first term, and this term has a much larger magnitude than both the second and third terms. Since the first term dominates the nonlinear system equation, the input from this term was selected by the GAGRNN.

The PMI algorithm was successful in identifying all of the actual inputs for each test problem, whereas the SOM–GAGRNN technique failed to identify all of the actual inputs for the AR9, TAR2 and nonlinear system test problems, and often included additional inputs. The inability of the GAGRNN to select the actual model inputs is most likely due to the fact that the GRNN is only able to provide an approximation of the true underlying function. This approximation may be adequate from a predictive sense, but is less useful when knowledge of the system is required since the actual model inputs could not be identified. An additional reason that may have contributed to the inability of the GAGRNN in selecting the actual model inputs may have resulted from the objective function used during the GA optimisation. The test set RMSE was used and it is possible that this is not the most suitable means for evaluating the fitness of different input subsets. The use of an alternative objective function may give different results. This is symptomatic of all model-based approaches, which are prone to the difficulties involved in selecting an appropriate objective function. Model-free approaches, such as the PMI algorithm, have the advantage that they avoid the need to select an objective function.

From the point of view of gaining insight into the underlying processes, it is important that the input determination technique is able to identify the actual inputs. However, from the point of view of predictive performance, identifying the actual inputs may be less important as a closely related input may suffice. Consequently, to compare both of the input determination procedures using predictive performance, GRNN models were developed for each input subset. Since GRNN models had already been developed as part of the SOM–GAGRNN input determination technique, the only additional models that needed to be developed were those using the inputs identified by the PMI algorithm. For each synthetic model, the 500 data points were divided into statistically similar training and testing sets. Eighty percent of the data (400 data points) were used for the training set and 20% (100 data points) were kept for testing purposes. For a strict comparison of each model's true generalisation ability, an independent validation set should be used, however, since the aim of this experiment was to study how model performance varied with different input subsets, it was not deemed necessary to use a validation set.

The results of the comparison are shown in Table 3. Both methods provided suitable means of determining the appropriate inputs from a predictive sense, as indicated by the similar RMSEs obtained from both input determination procedures for each test problem. For the AR1 test problem the results from the two methods were identical, since both methods identified the actual model input (i.e. lag 1). For the AR9 and TAR2 test problems, the PMI algorithm performed marginally better when measured using the testing set RMSE. For the nonlinear system test problem, the GAGRNN performed slightly better on the test set. Based on the results obtained from this study, if predictive importance is the primary objective, then either the SOM–GAGRNN approach or the PMI algorithm could be used. However, if an understanding of the system under investigation is also an important objective, then it is recommended that the PMI algorithm be used as it was able to successfully identify the true model inputs for each of the test problems considered.

Table 3
RMSEs of the GRNN models produced from each input determination method for the synthetic data test problems

Test problem	Input determination method	Training set RMSE	Testing set RMSE
AR1	SOM–GAGRNN	0.443	0.485
	PMI Algorithm	0.443	0.485
AR9	SOM–GAGRNN	0.333	0.673
	PMI Algorithm	0.369	0.586
TAR2	SOM–GAGRNN	0.548	0.754
	PMI Algorithm	0.626	0.731
Nonlinear	GAGRNN	0.164	0.219
	PMI Algorithm	0.064	0.223

5. Summary and conclusions

This article presented a review of the current state of input determination methods in water resources applications of ANN modelling. Previous methods that have been widely used include those based on:

- A priori information, however this requires extensive knowledge regarding the system under investigation.
- Linear cross-correlation, where only linear dependence can be detected between variables.
- Heuristic approaches, which do not guarantee that globally best subsets of inputs are found and are computationally intensive.
- Extraction of knowledge contained within the trained ANN, such as sensitivity analyses, which require a great deal of judgement in determining an appropriate cut-off point for input significance.

Many other papers reviewed failed to describe the input determination methodology used, and consequently, raised doubts about the optimality of the inputs obtained. Presenting an ANN with a large number of inputs has a number of detrimental impacts on ANN learning and tends to increase the training times required.

To address the lack of a robust procedure for selecting important inputs, two input determination methodologies were presented in this paper. The first technique investigated was the PMI algorithm. This is a *model-free* approach, as it does not require the construction of a model of the relationship between the two variables. In addition, it is able to capture nonlinear dependencies between two variables and can provide useful insights into the relative significance of each of the inputs identified. The computationally efficient city block distance kernel was used as the density estimation technique and general regression neural networks were used to compute conditional expectations in this implementation. GRNNs do not impose any restricting assumptions on the form of the underlying regression model and are a flexible and easy to implement method for estimating the residuals. When tested on synthetic data sets with known dependence attributes, the PMI algorithm correctly selected all significant inputs and represented an improvement over the implementation proposed by Sharma (2000) as it was able to exclude all insignificant inputs.

In the second input determination approach investigated in this study, an unsupervised technique (i.e SOM) was used to reduce the dimensionality of the input space and a GAGRNN was used to select the final subset of important model inputs. The SOM–GAGRNN was also applied to the synthetic test problems. This approach has the advantage that unlike the PMI algorithm, a significance level does not need to be selected. However, it has the disadvantage that the SOM technique does not guarantee the actual model inputs will be selected, since inputs that are highly correlated to the actual inputs may be selected if they are closest to the cluster centre. In addition, the GA introduces additional parameters (i.e. number of generations and population size) and an appropriate objective function also needs to be selected. The subsets selected by the SOM–GAGRNN included some unimportant inputs. Despite this, when GRNN models were developed using this subset, the predictive results were comparable with those obtained using the inputs selected by the PMI algorithm. Consequently, both approaches are recommended when predictive performance is the primary aim. However, the PMI algorithm should be used when insight into the underlying process is of utmost importance, as this method has demonstrated that it is able to identify the actual model inputs to a number of test problems. One limitation that the PMI algorithm may experience is the selection of an appropriate bandwidth when the data are highly non-Gaussian. In such a case, the Gaussian reference bandwidth may be unsuitable, and the impact that this may have on the selected inputs needs to be investigated in future studies.

References

Abrahart, R.J., See, L., Kneale, P.E., 1999. Using pruning algorithms and genetic algorithms to optimise network architectures and forecasting inputs in a neural network rainfall-runoff model. Journal of Hydroinformatics 1 (2), 103–114.

Abrahart, R.J., See, L., Kneale, P.E., 2001. Investigating the role of saliency analysis with a neural network rainfall-runoff model. Computers and Geosciences 27, 921–928.

ASCE Task Committee on Application of Artificial Neural Networks in Hydrology, 2000a. Artificial neural networks in hydrology. I: Preliminary concepts. Journal of Hydrologic Engineering, ASCE 5 (2), 115–123.

ASCE Task Committee on Application of Artificial Neural Networks in Hydrology, 2000b. Artificial neural networks in hydrology. II: Hydrologic applications. Journal of Hydrologic Engineering, ASCE 5 (2), 124–137.

Back, A.D., Trappenberg, T.P., 1999. Input Variable Selection using Independent Component Analysis International Joint Conference on Neural Networks, Washington 1999.

BioComp Systems, I., 1998. NeuroGenetic Optimizer (NGO), Redmond, WA.

Box, G.E.P., Jenkins, G.M., 1976. Time Series Analysis, Forecasting and Control. Holden-Day Inc., San Francisco. 553 pp..

Braddock, R.D., Kremmer, M.L., Sanzogni, L., 1998. Feed-forward artificial neural network model for forecasting rainfall run-off. Environmetrics 9, 419–432.

Brion, G.M., Neelakantan, T.R., Lingireddy, S., 2001. Using neural networks to predict peak cryptosporidium concentrations. Journal of the American Water Resources Association 37 (1), 99–105.

Burian, S.J., Durrans, S.R., Nix, S.J., 2001. Training artificial neural networks to perform rainfall disaggregation. Journal of Hydrologic Engineering 6 (1), 43–51.

Cacoullous, T., 1966. Estimation of a multivariate density. Annals of the Institute of Statistical Mathematics (Tokyo) 18 (2), 179–189.

Cai, S., Toral, H., Qiu, J., Archer, J.S., 1994. Neural network based objective flow regime identification in air-water two phase flow. The Canadian Journal of Chemical Engineering 72, 440–445.

Campolo, M., Soldati, A., Andreussi, P., 1999. Forecasting river flow rate during low-flow periods using neural networks. Water Resources Research 35 (11), 3547–3552.

Caudill, M., 1993. GRNN and bear it. AI Expert 8 (5), 28–33.

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

Coulibaly, P., Anctil, F., Bobee, B., 2000. Daily reservoir inflow forecasting using artificial neural networks with stopped training approach. Journal of Hydrology 230, 244–257.

Cover, T.M., Thomas, J.A., 1991. Elements of Information Theory. Wiley, New York. 542 pp..

Dandy, G.C., Simpson, A.R., Murphy, L.J., 1996. An improved genetic algorithm for pipe network optimisation. Water Resources Research 32 (2), 449–458.

Dawson, C.W., Wilby, R.L., 2001. Hydrological modelling using artificial neural networks. Progress in Physical Geography 25 (1), 80–108.

Fernando, D.A.K., Jayawardena, A.W., 1998. Runoff forecasting using RBF networks with OLS algorithm. Journal of Hydrologic Engineering 3 (3), 203–209.

Fraser, A.M., Swinney, H.L., 1986. Independent coordinates for strange attractors from mutual information. Physical Review A 33 (2), 1134–1140.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading, MA. 412 pp.

Goldberg, D.E., Deb, K., 1991. A Comparative Analysis of Selection Schemes Used in Genetic Algorithms, in: Rawlins, J.E. (Ed.), Foundations of Genetic Algorithms. Morgan Kaufmann, San Mateo, CA, pp. 69–93.

Golob, R., Stokelj, Grgic, D., 1998. Neural-network-based water inflow forecasting. Control Engineering Practice 6, 593–600.

Harrold, T.I., Sharma, A., Sheather, S., 2001. Selection of a kernel bandwidth for measuring dependence in hydrologic time series using the mutual information criterion. Stochastic Environmental Research and Risk Assessment 15, 310–324.

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.

Huang, W., Foo, S., 2002. Neural network modeling of salinity variation in Apalachicola River. Water Research 36, 356–362.

Imrie, C.E., Durucan, S., Korre, A., 2000. River flow prediction using artificial neural networks: generalisation beyond the calibration range. Journal of Hydrology 233, 138–153.

Islam, S., Kothari, R., 2000. Artificial neural networks in remote sensing of hydrologic processes. Journal of Hydrologic Engineering 5 (2), 138–144.

Jain, S.K., Chalisgaonkar, D., 2000. Setting uo stage-discharge relations using ANN. Journal of Hydrologic Engineering 5 (4), 428–433.

Jain, S.K., Das, A., Srivastava, D.K., 1999. Application of ANN for reservoir inflow prediction and operation. Journal of Water Resources Planning and Management 125 (5), 263–271.

Jayawardena, A.W., Fernando, D.A.K., Zhou, M.C., 1997. Comparison of multilayer perceptron and radial basis function networks as tools for flood forecasting IAHS Publication (International Association of Hydrological Sciences) (239) 1997.

Kohonen, T., 1982. Self-organized formation of topologically correct feature maps. Biological Cybernetics 43, 59–69.

Kohonen, T., 1990. The Self-Organizing Map. Proceedings of IEEE 78 (9), 1464–1480.

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 Publishers, Dordrecht, pp. 229–242.

Lekkas, D.F., Imrie, C.E., Lees, M.J., 2001. Improved non-linear transfer function and neural network methods of flow routing for real-time forecasting. Journal of Hydroinformatics 3 (3), 153–164.

Liong, S.-Y., Lim, W.-H., Paudyal, G.N., 2000. River stage forecasting in Bangladesh: Neural network approach. Journal of Computing in Civil Engineering 14 (1), 1–8.

Liong, S.-Y., Khu, S.-T., Chan, W.-T., 2001. Derivation of pareto front with genetic algorithm and neural network. Journal of Hydrologic Engineering 6 (1), 52–61.

Luk, K.C., Ball, J.E., Sharma, A., 2000. A study of optimal model lag and spatial inputs to artificial neural network for rainfall forecasting. Journal of Hydrology 227, 56–65.

Maier, H.R., Dandy, G.C., 1996. 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., 1997. Determining inputs for neural network models of multivariate time series. Microcomputers in Civil Engineering 12 (5), 353–368.

Maier, H.R., Dandy, G.C., 2000. Neural networks for the prediction and forecasting of water resources variables: a review of modelling issues and applications. Environmental Modelling and Software 15, 101–124.

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, 257–272.

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

Parzen, E., 1962. On estimation of probability density function and mode. Annals of Mathematical Statistics 33, 1065–1076.

Sajikumar, N., Thandaveswara, B.S., 1999. A non-linear rainfall-runoff model using an artificial neural network. Journal of Hydrology 216, 32–35.

Sarle, W.S., 1994. Neural networks and statistical models Proceedings of the Nineteenth Annual SAS Users Group International Conference. SAS Institute pp. 1538–1550.

Sarle, W.S., 1997. Neural Network FAQ, periodic posting to the Usenet newsgroup www.comp.ai.neural-nets

Schleiter, I.M., et al., 1999. Modelling water quality, bioindication and population dynamics in lotic ecosystems using neural networks. Ecological Modelling 120 (2–3), 271–286.

Scott, D.W., 1992. Multivariate Density Estimation: Theory, Practice and Visualization. Probability and Mathematical Statisics. Wiley, New York. 317 pp..

Sharma, A., 2000. Seasonal to interannual rainfall probabilistic forecasts for improved water supply management: Part 1—a strategy for system predictor identification. Journal of Hydrology 239, 232–239.

Silverman, D., Dracup, J.A., 2000. Artificial neural networks and long-range precipitation in California. Journal of Applied Meteorology 31 (1), 57–66.

Specht, D.F., 1991. A general regression neural network. IEEE Transactions on Neural Networks 2 (6), 568–576.

Thirumalaiah, K., Deo, M.C., 2000. Hydrological forecasting using neural networks. Journal of Hydrologic Engineering 5 (2), 180–189.

Tokar, A.S., Johnson, P.A., 1999. Rainfall-runoff modeling using artificial neural networks. Journal of Hydrologic Engineering 4 (3), 232–239.

Yang, H.H., Van Vuuren, S., Sharma, S., Hermansky, H., 2000. Relevance of time-frequency features for phonetic and speaker-channel classification. Speech Communication 31, 35–50.

Zheng, G.L., Billings, S.A., 1996. Radial Basis Function Network Configuration Using Mutual Information and the Orthogonal Least Squares Algorithm. Neural Networks 9 (9), 1619–1637.