Flow prediction by three back propagation techniques using *k*-fold partitioning of neural network training data

H. Kerem Cigizoglu[1] and Özgür Kişi

Istanbul Technical University, Civil Engineering Faculty, Division of Hydraulics, Maslak 34469 Istanbul, Turkey.
[1]Corresponding author. E-mail: *cigiz@itu.edu.tr*

Received 23 June 2003; accepted in revised form 1 December 2003

Abstract Flow forecasting performance by artificial neural networks (ANNs) is generally considered to be dependent on the data length. In this study *k*-fold partitioning, a statistical method, was employed in the ANN training stage. The method was found useful in the case of using the conventional feed-forward back propagation algorithm. It was shown that with a data period much shorter than the whole training duration similar flow prediction performance could be obtained. Prediction performance and convergence velocity comparison between three different back propagation algorithms, Levenberg–Marquardt, conjugate gradient and gradient descent was the next concern of the study and the Levenberg–Marquardt technique was found advantageous thanks to its shorter training duration and more satisfactory performance criteria.

Keywords Gradient descent; *k*-fold partitioning; Levenberg–Marquardt technique

Nordic Hydrology **Vol 36 No 1 pp 49–64** © IWA Publishing 2005

Introduction

Many of the activities associated with the planning and operation of the components of a water resource system require forecasts of future events. For the hydrologic component, there is a need for both short term and long term forecasts of streamflow events in order to optimize the system or to plan for future expansion or reduction.

Neural networks (NN) have been successfully applied in a number of diverse fields including water resources. In ANN models, the statistical distribution of the data need not be known and non-stationarities in the data, such as trends and seasonal variations, are implicitly accounted for by the internal structure of the ANNs (Maier and Dandy 1996). ANNs differ from the traditional approaches in synthetic hydrology in the sense that they belong to a class of data-driven approaches. Data-driven approaches are suited to complex problems. They have the ability to determine which model inputs are critical, so that there is no need for prior knowledge about the relationships amongst the variables being modelled.

In the hydrological forecasting context, recent experiments have reported that ANNs may offer a promising alternative for rainfall–runoff modelling (Zhu and Fujita 1994; Smith and Eli 1995; Hsu *et al.* 1995; Minns and Hall 1996; Shamseldin 1997; Sajikumar and Thandaveswara 1999; Tokar and Johnson 1999), streamflow prediction (Kang *et al.* 1993; Karunanithi *et al.* 1994; Thirumalaiah and Deo 1998; Clair and Ehrman 1998; Zealand *et al.* 1999; Campolo *et al.* 1999; Cigizoglu 2003a, b, c, 2004*a*; Kisi 2004), reservoir inflow forecasting (Saad *et al.* 1996; Coulibaly *et al.* 1998; Jain *et al.* 1999) and suspended sediment estimation (Jain 2001; Cigizoglu (2004*a*, *b*); Cigizoglu and Alp 2003). Recently, Coulibaly *et al.* (1998) reviewed the ANN-based modelling in hydrology over the last few years and reported that about 90% of the experiments extensively make use of the multi-layer feed-forward neural networks (FNN) trained by the standard back propagation (BP) algorithm (Rumelhart *et al.* 1986).

In the majority of the back-propagation applications the gradient descent technique is employed. However, it is seen that this technique has some drawbacks such as a long training

duration with a high number of iterations. That is why methods such as the conjugate gradient method have been discussed by several authors in the literature (Adeli and Hung 1995). Another backpropagation technique, the Levenberg–Marquardt (LM) weight optimization approach seems to be a solution in tackling this problem. The first goal of this study is to compare the performance of LM approach with gradient descent and conjugate gradient methods in the flow prediction problem.

The second issue investigated in the study is the data scarcity frequently faced by the water resources engineers in different projects. In works like water reservoir planning the length of the observed flow record might be quite short, making the flow forecasting studies difficult. Therefore methods helping to obtain more information from the available limited data carries significance. In a recent study it was shown that extending the ANN training sets with the synthetically generated flow data increased the flow prediction performance noticeably by ANNs (Cigizoglu 2003b). Another statistical method to handle the available observed record is the *k*-fold partitioning (Ali and Pazzani 1996). Here the record is divided into smaller data sets and handled separately. In other words statistical work is carried out for each sub-group independently and the sub-group which provides more information (even more than the whole data set) is selected. The other goal of this study is to incorporate the *k*-fold partioning into the ANN prediction works and analyze the performance differences.

In the first part of the work the training data set was divided into *k* data sets and prediction performances following a neural network training stage using either a small sub data set or complete data set was analyzed. The training data were split into seven different sets and the effect of each sub-training data to forecasting efficiency was investigated. In the second part of the study three different feed forward back propagation (FFBP) algorithms, gradient descent (standard back propagation), conjugate gradient and Levenberg–Marquardt (Hagan and Menhaj 1994) – were employed in daily mean stream flow forecasting. The results of both models were compared with each other and with a conventional stochastic model. The final aspect of the study was the investigation of prediction performance of the best propagation technique in 2-days and 3-days ahead forecasting using the previously selected training sub-group.

Neural networks

Back-propagation

Back-propagation is, by far, the most commonly used method for training multilayer feedforward networks. The term "back-propagation" refers to two different things. First, it describes a method to calculate the derivatives of the network training error with respect to the weights by application of the derivative chain-rule. Second, it describes a training algorithm, equivalent to gradient descent optimization, for using derivatives to adjust the weights to minimize the error.

The algorithm was popularized by Rumelhart *et al.* (1986), although earlier work had been done by Parker (1985) and Le Chun (1985) (summarized by Le Chun (1986)). Together with the Hopfield network, it was responsible for much of the interest in neural networks in the mid-1980 s. Before back-propagation, most networks used non-differentiable hard-limiting binary nonlinearities such as step functions and there were no well-known general methods for training multilayer networks (Reed and Marks 1998). The breakthrough was perhaps not so much the application of the chain-rule, but the demonstration that layered networks of *differentiable* nonlinearities could perform useful non-trivial calculations and that they offer (in some implementations) attractive features such as fast response, the ability to "learn" from examples and some ability to generalize beyond the training data.

The gradient-descent (standard back-propagation) method

There are N data input patterns, each having a set of input values at the input nodes with output values at the output nodes. The input values are multiplied by the first interconnection weights at the hidden nodes and the products are summed and become the inputs of hidden layers. Each hidden node is transformed through a selected function to produce a hidden node output. The output serves as the input to the succeeding layer and its procedure is continued until the output layer is reached. This is referred to as forward activation flow.

These input values are processed through the selected function defined earlier to give the neural network output values. The subsequent weight adoption or learning process is accomplished by the back-propagation learning algorithm. For all input patterns the average system error or mean square error (MSE) is computed. The aim of the back-propagation algorithm is to minimize iteratively the averaged square error. This is accomplished by first computing the gradient for each node on the output layer. The error gradient is then recursively determined for the hidden layers by computing the weighted sum of the errors at the previous layer. The error gradients are then used to update the network weights. More information for the gradient-descent method can be found in standard text books (Rumelhart *et al.* 1986).

Conjugate gradient method

The basic back propagation algorithm adjusts the weights in the steepest descent direction (the negative of the gradient). This is the direction in which the performance function is decreasing most rapidly. It turns out that, although the function decreases most rapidly along the negative of the gradient, this does not necessarily produce the fastest convergence. In the conjugate gradient algorithms a search is performed along conjugate directions, which produces generally faster convergence than the steepest descent directions (Adeli and Hung 1995).

In most of the conjugate gradient algorithms the step size is adjusted at each iteration. A search is made along the conjugate gradient direction to determine the step size, which will minimize the performance function along that line.

All of the conjugate gradient algorithms start out by searching in the steepest descent direction (negative of the gradient) on the first iteration:

$$p_0 = -g_0 \tag{1}$$

A line search is then performed to determine the optimal distance to move along the current search direction:

$$x_{k+1} = x_k + \alpha_k g_k \tag{2}$$

where x_k is a vector of current weights and biases, g_k is the current gradient and α_k is the learning rate. Then the next search direction is determined so that it is conjugate to the previous search directions. The general procedure for determining the new search direction is to combine the new steepest descent direction with the previous search direction:

$$p_k = -g_k + \beta_k p_{k-1} \tag{3}$$

The various versions of the conjugate gradient are distinguished by the manner in which the constant β_k is computed. For the Fletcher–Reeves update the procedure is

$$\beta_k = \frac{g_k^T g_k}{g_{k-1}^T g_{k-1}} \tag{4}$$

This is the ratio of the norm squared of the current gradient to the norm squared of the previous gradient.

The Levenberg–Marquardt method

While the back propagation with gradient descent technique is a steepest descent algorithm, the Marquardt–Levenberg algorithm is an approximation to Newton's method (Marquardt 1963). If we have a function $V(\mathbf{x})$ which we want to minimize with respect to the parameter vector $\mathbf{x}$, then Newton's method would be

$$\Delta\mathbf{x} = -\left[\nabla^2 V(\mathbf{x})\right]^{-1}\nabla V(\mathbf{x}) \tag{5}$$

where $\nabla^2 V(\mathbf{x})$ is the Hessian matrix and $\nabla V(\mathbf{x})$ is the gradient. If we assume that $V(\mathbf{x})$ is a sum of squares function

$$V(\mathbf{x}) = \sum_{i=1}^{N} e_i^2(x) \tag{6}$$

then it can be shown that

$$\nabla V(\mathbf{x}) = J^T(\mathbf{x})\mathbf{e}(\mathbf{x}) \tag{7}$$

$$\nabla^2 V(\mathbf{x}) = J^T(\mathbf{x})J(\mathbf{x}) + S(\mathbf{x}) \tag{8}$$

where $J(\mathbf{x})$ is the Jacobian matrix and

$$S(\mathbf{x}) = \sum_{i=1}^{N} e_i \nabla^2 e_i(\mathbf{x}) \tag{9}$$

For the Gauss–Newton method it is assumed that $S(\mathbf{x}) \approx 0$, and the update (Eq. (11)) becomes

$$\Delta\mathbf{x}_{-} = [J^T(\mathbf{x})J(\mathbf{x})]^{-1}J^T(\mathbf{x})\mathbf{e}(\mathbf{x}) \tag{10}$$

The Marquardt–Levenberg modification to the Gauss–Newton method is

$$\Delta\mathbf{x} = [J^T(\mathbf{x})J(\mathbf{x}) + \mu I]^{-1}J^T(\mathbf{x})\mathbf{e}(\mathbf{x}) \tag{11}$$

The parameter μ is multiplied by some factor (β) whenever a step would result in an increased $V(\mathbf{x})$. When a step reduces $V(\mathbf{x})$, μ is divided by β. When μ is large the algorithm becomes steepest descent (with step $1/\mu$), while for small μ the algorithm becomes Gauss–Newton. The Marquardt–Levenberg algorithm can be considered a trust-region modification to Gauss–Newton. The key step in this algorithm is the computation of the Jacobian matrix. For the neural network-mapping problem the terms in the Jacobian matrix can be computed by a simple modification to the back propagation algorithm (Hagan and Menhaj 1994).

Description of data

In this study the daily mean flow data belonging to Derecikviran station (station no. 1335) on the Filyos River in the Western Black Sea region of Turkey has been employed. The station has a drainage area of 13 300 km^2. The Filyos River carries floods occasionally and they cause significant damage in the region. The data has a length of 38 years covering a time period between 1964 and 2001. The data was found to be homogeneous for this period. The plot of the whole record and the map of the region are illustrated in Fig. 1. The daily flow statistics of the whole data set are presented in Table 1. The observed daily flow series show high positive skewness ($c_{sx} = 4.09$). The autocorrelations for the first lags are quite high as a signal of high persistence ($r_1 = 0.92$, $r_2 = 0.82$, $r_3 = 0.74$). This shows that the flows of the antecedent days with lags higher than 1 should have a considerable effect for the forecasting work as well as the lag-1 flow. The ratio between the maximum and the mean of the whole record is nearly 22 (Table 1).

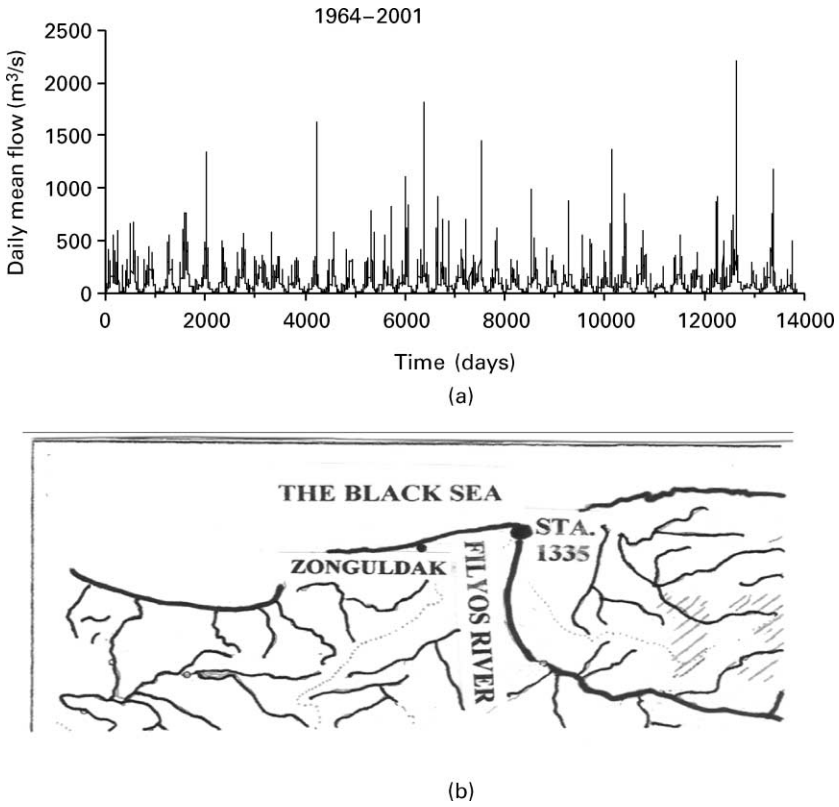

Figure 1 Plot of the whole record (a) and the map of the region (b)

Table 1 The daily statistical parameters of each data set (x_{mean}, s_x, c_{sx}, x_{min}, x_{max}, r_1, r_2, r_3 denote the overall mean, standard deviation, skewness, lag-1, lag-2 and lag-3 autocorrelation coefficients, respectively)

Data set	Data period	x_{mean} (m^3/s)	s_x (m^3/s)	c_{sx} (m^3/s)	x_{min} (m^3/s)	x_{max} (m^3/s)	r_1	r_2	r_3
Sub-group 1	01.10.1963–30.09.1969	113.3	118.1	1.77	13.4	763	0.94	0.86	0.81
Sub-group 2	01.10.1968–30.09.1973	104.1	100.9	3.08	4.50	1340	0.93	0.82	0.75
Sub-group 3	01.10.1973–30.09.1978	83.70	102.8	4.95	0.10	1618	0.88	0.75	0.68
Sub-group 4	01.10.1978–30.09.1983	112.8	130.6	4.53	11.0	1816	0.90	0.76	0.66
Sub-group 5	01.10.1983–30.09.1988	99.20	104.2	3.38	8.54	1453	0.92	0.83	0.77
Sub-group 6	01.10.1988–30.09.1993	100.4	108.1	3.31	9.28	1373	0.92	0.81	0.73
Sub-group 7	01.10.1993–30.09.1998	107.6	145.3	5.30	5.00	2204	0.95	0.86	0.78
All training data	01.10.1963–30.09.1998	103.0	117.1	4.11	0.10	2204	0.92	0.81	0.74
Test data	01.10.1998–30.09.2001	83.31	107.4	3.92	9.80	1181	0.93	0.82	0.75
Entire data	01.10.1963–30.09.2001	101.5	116.5	4.09	0.10	2204	0.92	0.82	0.74

Application

Employment of *k*-fold partitioning in neural network training data

In some of the water resources modelling studies the limited size of the data length is a significant issue decreasing the model performance. The general belief in forecasting studies is the success of the prediction model increases in parallel to an increase in the data length employed in model calibration. This is an intuitive approach that stems from the conventional wisdom: "the more data the better". The effect of extending the neural network

training data set by synthetic data series was discussed by Cigizoglu (2003b). Accordingly, extending the available record could be benefical under the condition that the original statistics such as mean, standard deviation and skewness are preserved. Therefore methods which help to obtain as much information from the available data set as possible carry significance. Neural networks are black box models which necessitate the use of a certain amount of data during the training of the selected neural network structure in order to obtain the weight set providing the best performance criterion. This weight set is then employed in the testing stage for a data set not used during training. The general tendency is to allocate the significant part of the data set for training and only a limited part for testing. In this study the emphasis was given to investigate the neural network prediction performance in the case of dividing the training data set into small sub-groups. This approach, referred to as k-fold partitioning, is employed in statistics for different purposes (Ali and Pazzani 1996; Ting and Low 1996). The term k represents the number of sub-training sets. In our study, arbitrarily k was taken equal to 7. Hence the training data of neural networks were divided into seven parts of equal length. The period for each data set is shown in Table 1. As seen from the table, each sub-training data set has a length of five years whereas the testing set has a three years' long record. The daily flow statistics for each sub-training set, whole training set and testing set are presented in Table 1. For the sub-training sets, the skewness varies within a range between 1.77 and 5.30 and the sub-group $x_{\min}$ and $x_{\max}$ values fall in the ranges 0.10–13.4 m^3/s and 763–2204 m^3/s, respectively. The ratio between the maximum and the mean of the sub-training sets changes between 6.7 and 20.5. The testing data set extreme ($x_{\min} = 9.80\ m^3/s$, $x_{\max} = 1181\ m^3/s$) falls within the sub-training group limits except in sub-group 1 where $x_{\max}$ (763 m^3/s) is less than the corresponding testing set value. The sub-group no. 1 has a skewness ($c_{sx} = 1.77$), quite a bit lower than the testing set ($c_{sx} = 3.92$). Compared with other sub-data sets and all the training data set, sub-group 5 seems to have the closest statistics to the testing data set. The ANN simulations will show whether the agreement in statistics will be reflected in performance criteria.

The ANN simulations were carried out using the FFBP algorithm with Levenberg–Marquardt technique (FFBP-LM) and the ANN structures were calibrated using each training data set shown in Table 1. For each data set 6 different input layer combinations were employed (Table 2). Accordingly an ANN structure like ANN(4,6,1) consists of 4 inputs, 6 hidden and one output nodes. In this case the input layer covers the daily mean flows of the preceding 4 days (days t-1, t-2, t-3 and t-4) and the output layer consists of the unique flow at day t. The hidden layer node numbers are found simply by trial and error. Throughout all ANN simulations adaptive learning and the momentum rate values were employed.

The mean square errors (MSE) for each FFBP-LM model in the testing period are presented in Table 3. In this table FFBP-LM 1 shows the model calibrated by using the first sub-training data set and vice versa. It can be seen that FFBP-LM 5 models provided the lowest MSEs for various input combinations. For the ANN(3,5,1) configuration (3 input and

Table 2 The input layer combinations used for each sub-training data set

The input layer variables	The ANN structure
Q_{t-1}	ANN(1,4,1)
Q_{t-1} and Q_{t-2}	ANN(2,3,1)
Q_{t-1}, Q_{t-2} and Q_{t-3}	ANN(3,5,1)
Q_{t-1}, Q_{t-2}, Q_{t-3} and Q_{t-4}	ANN(4,6,1)
Q_{t-1}, Q_{t-2}, Q_{t-3}, Q_{t-4} and Q_{t-5}	ANN(5,8,1)
Q_{t-1}, Q_{t-2}, Q_{t-3}, Q_{t-4}, Q_{t-5} and Q_{t-6}	ANN(6,9,1)

Table 3 The mean square errors (MSE m^6/s^2) for each FFBP-LM model – testing period

ANN models	1-input ANN(1,4,1)	2-input ANN(2,3,1)	3-input ANN(3,5,1)	4-input ANN(4,6,1)	5-input ANN(5,8,1)	6-input ANN(6,9,1)
FFBP-LM 1	1328	1195	1188	1180	1268	1239
FFBP-LM 2	1168	942	879	890	931	923
FFBP-LM 3	1171	1020	1019	1070	1003	1027
FFBP-LM 4	1187	952	916	967	957	944
FFBP-LM 5	1176	899	**862**	871	878	880
FFBP-LM 6	1195	950	891	891	945	957
FFBP-LM 7	1277	1080	1045	940	944	960
FFBP-LM	1177	942	917	908	904	893

5 hidden nodes) FFBP-LM 5 provided lowest MSE (862 m^6/s^2). For this case the input layer consisted of the three previous daily flows (for times t-3, t-2 and t-1). Again for this configuration the performance of FFBP-LM 5 is superior to FFBP-LM (917 m^6/s^2). In general the prediction performance obtained using all sub-groups except sub-group 1 is comparable and, in some cases, even superior to the training using whole data. The comparably weaker performance using sub-group 1 can be explained with the narrow variation range of this data set (13.4–763 m^3/s). As discussed before sub-group no. 1 skewness is quite a bit lower than the testing set. Since the testing data has lower and higher x_{min} and x_{max} values, respectively, the trained neural networks face difficulties in making extrapolations. The forecasting performances of FFBP-LM 5 and FFBP-LM for 3 and 5 input nodes are compared in Figs 2, 3, 4 and 5 in the form of hydrograph and scatter plots. These

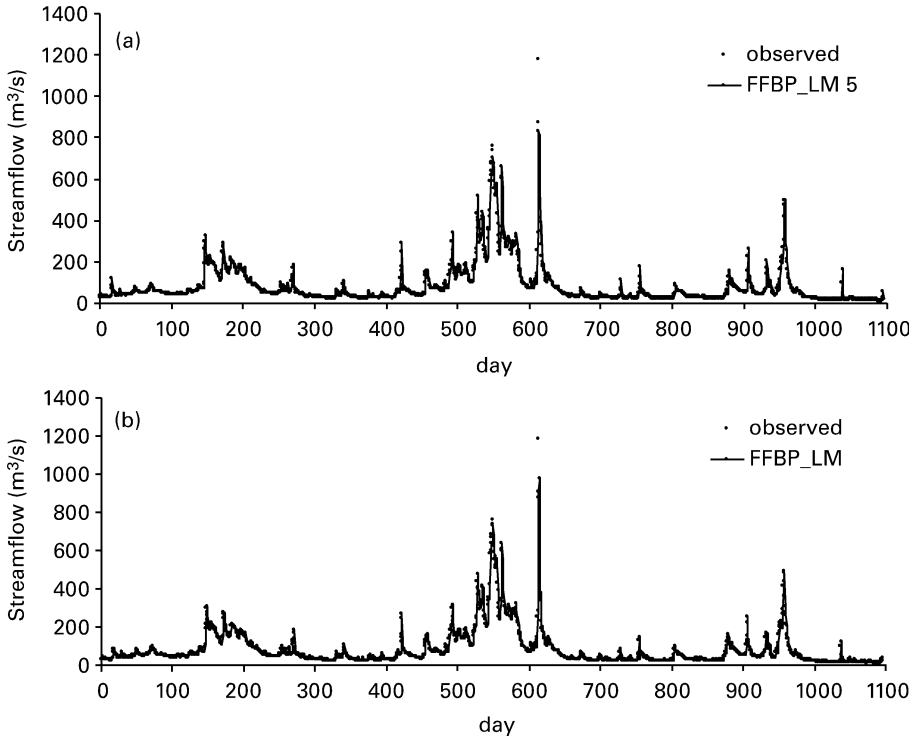

Figure 2 Plotting of forecasting performances for the testing period using (a) FFBP-LM 5 and (b) FFBP-LM for an input layer with three nodes (ANN(3,5,1))

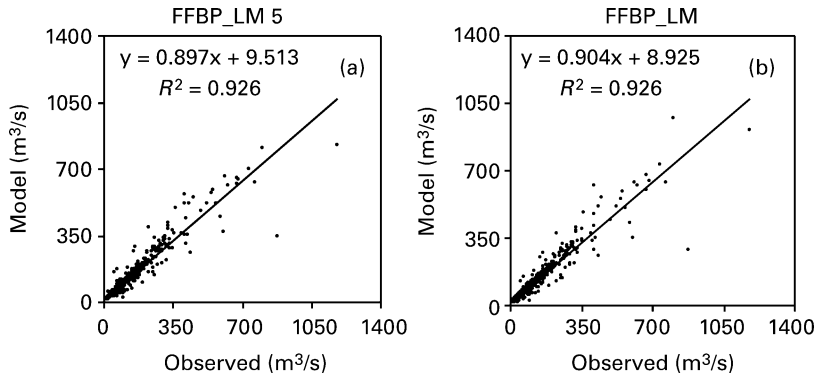

Figure 3 Scatter plots for (a) FFBP-LM 5 and (b) FFBP-LM for an input layer with three nodes (ANN(3,5,1)) for the testing period

plots show that the forecasted hydrographs are close to the observed ones for both FFBP-LM5 and FFBP-LM cases with nearly identical determination coefficients. The lower MSE obtained by FFBP-LM5 can be explained by a closer approximation of some medium and high flows.

Comparison of GD, CG and LM performances in one-day ahead flow prediction

In the majority of back-propagation applications to water resources the data gradient descent method was employed (Raman and Sunilkumar 1995; Minns and Hall 1996). However, this technique has some drawbacks such as a long training duration due to the necessity of a high iteration number for the desired error convergence. For this reason another back propagation method, the conjugate gradient algorithm, has been proposed in the literature (Adeli and

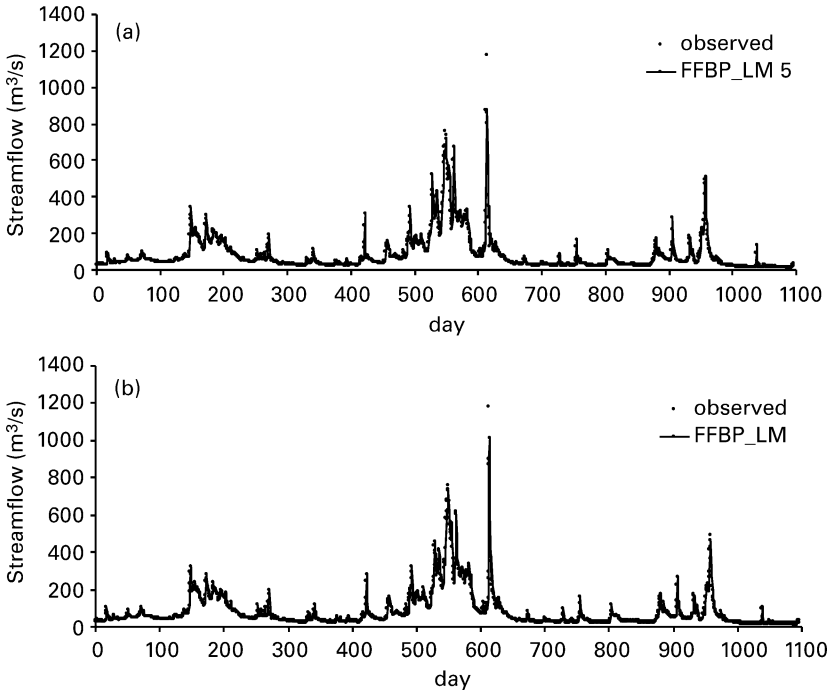

Figure 4 Plotting of forecasting performances for the testing period using (a) FFBP-LM 5 and (b) FFBP-LM for an input layer with five nodes (ANN(5,8,1))

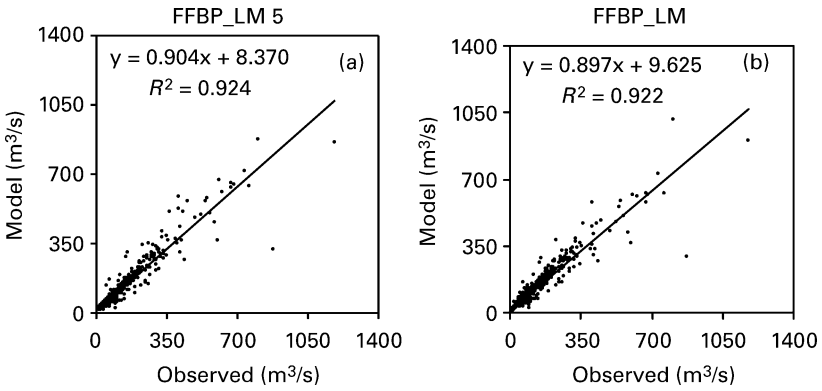

Figure 5 Scatter plots for (a) FFBP-LM 5 and (b) FFBP-LM for an input layer with five nodes (ANN(5,8,1)) for the testing period

Hung 1995). Another back-propagation algorithm which has not been employed in water resources applications is the Levenberg–Marquardt technique (Hagan and Menhaj 1994). In this part of the study three back propagation techniques were applied to the selected 7 sub-training data sets and the full training data set. The ANN structure was taken as ANN(3,5,1) since the best forecasting performance was obtained for this configuration in the previous section. An initial test was carried out to compare the CPU-times of three techniques for the training data taking a certain MSE value as stopping criteria. For each sub-group different limiting MSE values were assigned considering the training process results obtained in the previous section (Table 4). It was seen that training with FFBP-LM provided the shortest ANN training duration for all sub-groups. FFBP-GD training processes were quite slow and therefore these simulations were interrupted after a certain iteration number (100 000 iterations, Table 4). Though FFBP-CG simulations provided significant time savings compared with FFBP-GD, their training durations were still longer than FFBP-LM.

The testing stage results for all three back-propagation techniques are presented in Table 5. The MSE values presented here show the superiority of FFBP-LM performances over the FFBP-GD and FFBP-CG applications clearly. Figs 6 and 7 show the forecasting performances of FFBP-GD 7 and FFBP-GD for 3 input nodes in the form of hydrograph and scatter plots. The FFBP_GD 7 gave an R^2 value equal to 0.844, which was higher than the FFBP_GD value ($R^2 = 0.760$). The MSE for the FFBP_GD 7 model was found as 1825 m^6/s^2, being lower than the corresponding FFBP-GD value (2786 m^6/s^2, Table 5). For the purpose of comparison the ANN results were compared with a conventional ARMA (p,q) model, AR(3). A third-order AR model was selected in order to obtain a fair comparison with an ANN structure having 3 inputs. The AR(3) model was calibrated using 7 various training sub-groups and the whole training data. The AR(3) application performances for testing data were presented in the last column of Table 5. It can be seen that all AR(3) results are weaker compared with FFBP-LM performances except for sub-group 1. However, AR(3) was found superior to FFBP-GD for all sub-groups and to FFBP-CG for some sub-groups (Table 5). FFBP-GD simulations can give close or superior testing stage performances than AR(3) in the case of quite high training iteration numbers but this would be a very time consuming process. The best sub-group performances of all three ANN techniques are illustrated in Fig. 8 (FFBP-GD7, FFBP-CG5 and FFBP-LM5). It is obvious that FFBP-LM5 provides the closest forecasts for the observed hydrograph for the testing stage resulting in the highest R^2 (0.926). In order to see the ANN technique performances more clearly a sub-interval is selected for the testing stage (500–650th days, Figs 9 and 10).

Table 4 The mean square errors (MSE) and CPU-times for different ANN techniques (training period)

		FFBP-GD (i)			FFBP-CG (i)			FFBP-LM (i)		
Data set sub-groub number (i)	Stopping MSE (m^6/s^2)	Iteration	Time (s.)	MSE (m^6/s^2)	Iteration	Time (s.)	MSE (m^6/s^2)	Iteration	Time (s)	MSE (m^6/s^2)
1	1550	100 000	2904	1988	192	8.1	1537	46	3.2	1549
2	1200	100 000	2919	3120	98	6.3	1195	65	4.7	1199
3	2200	100 000	2682	3408	127	7.7	2185	51	3.1	2199
4	2900	100 000	2728	4704	244	14.6	2874	42	2.9	2900
5	1200	100 000	2728	4483	186	11.2	1198	37	2.9	1189
6	1350	100 000	3381	2164	54	3.6	1340	31	2.4	1331
7	1500	100 000	2874	3493	286	17.4	1465	75	6.4	1499
All data	1950	100 000	13902	9753	63	13.2	1929	38	20.4	1935

Table 5 The mean square errors (MSE) for different FFBP-GD, FFBP-CG and BP-LM applications

	3 inputs			
Data set sub-group number (*i*)	**FFBP-GD (i)**	**FFBP-CG (i)**	**FFBP-LM (i)**	**AR 3**
1	2607	1243	1199	1072
2	2308	1031	892	945
3	2575	1135	1022	1155
4	1846	1035	938	1003
5	3362	926	**862**	1056
6	2228	1096	959	1046
7	1825	1016	935	882
All data	2786	995	917	1003

In these figures it is more obvious that the deviations from the observed record are lowest for FFBP-LM5. In particular some of the peak flows are more closely estimated by FFBP-LM5 (Figs 9 and 10).

2-day and 3-day ahead forecasting using FFBP_LM 5

In this part of the study the FFBP_LM 5 model with a configuration ANN(3,5,1) was employed for 2-days and 3-days ahead daily streamflow forecasting. The testing stage MSEs were 2626 (m^6/s^2) and 4051 (m^6/s^2) for 2-days and 3-days ahead prediction, respectively. The performances were plotted in Figs 11 and 12 in the form of hydrograph and scatter plots. The FFBP_LM 5 gave R^2 values of 0.776 and 0.654 for the 2 and 3-days ahead forecasting, respectively. This was expected since forecasting performance generally deteriorates with increasing lead times. The corresponding FFBP-LM performance was weaker (MSE = 2760 m^6/s^2 and MSE = 4074 m^6/s^2 for 2 and 3-days ahead forecasting,

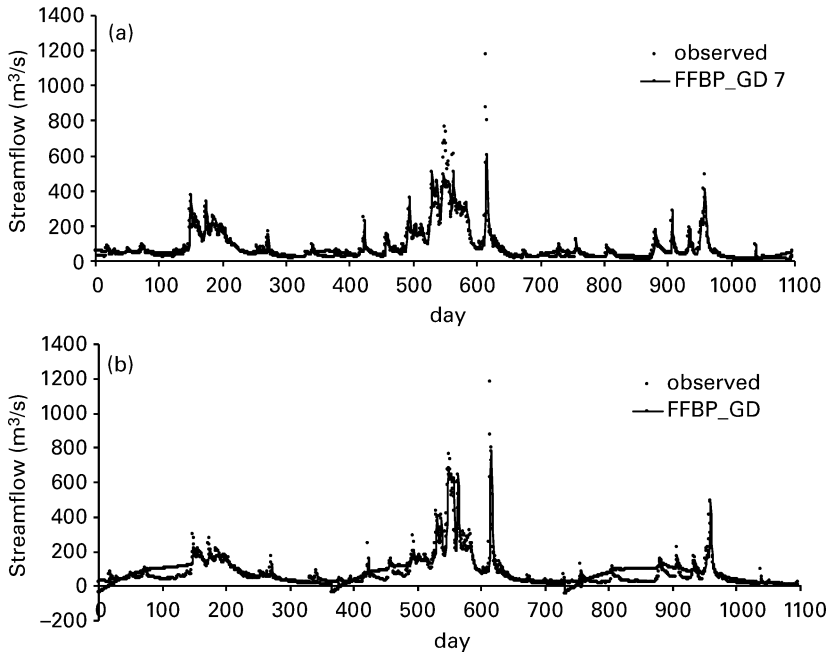

Figure 6 Plotting of forecasting performances for the testing period using (a) FFBP-GD 7 and (b) FFBP-GD for an input layer with three nodes (ANN(3,5,1))

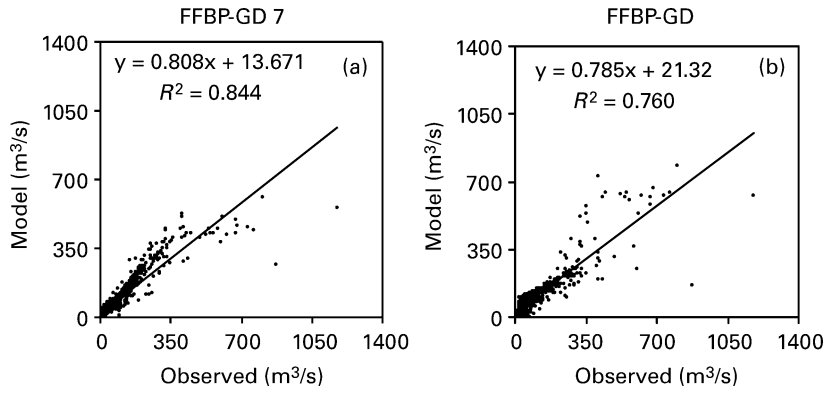

Figure 7 Scatter plots for (a) FFBP-GD 7 and (b) FFBP-GD for an input layer with three nodes (ANN(3,5,1)) for the testing period

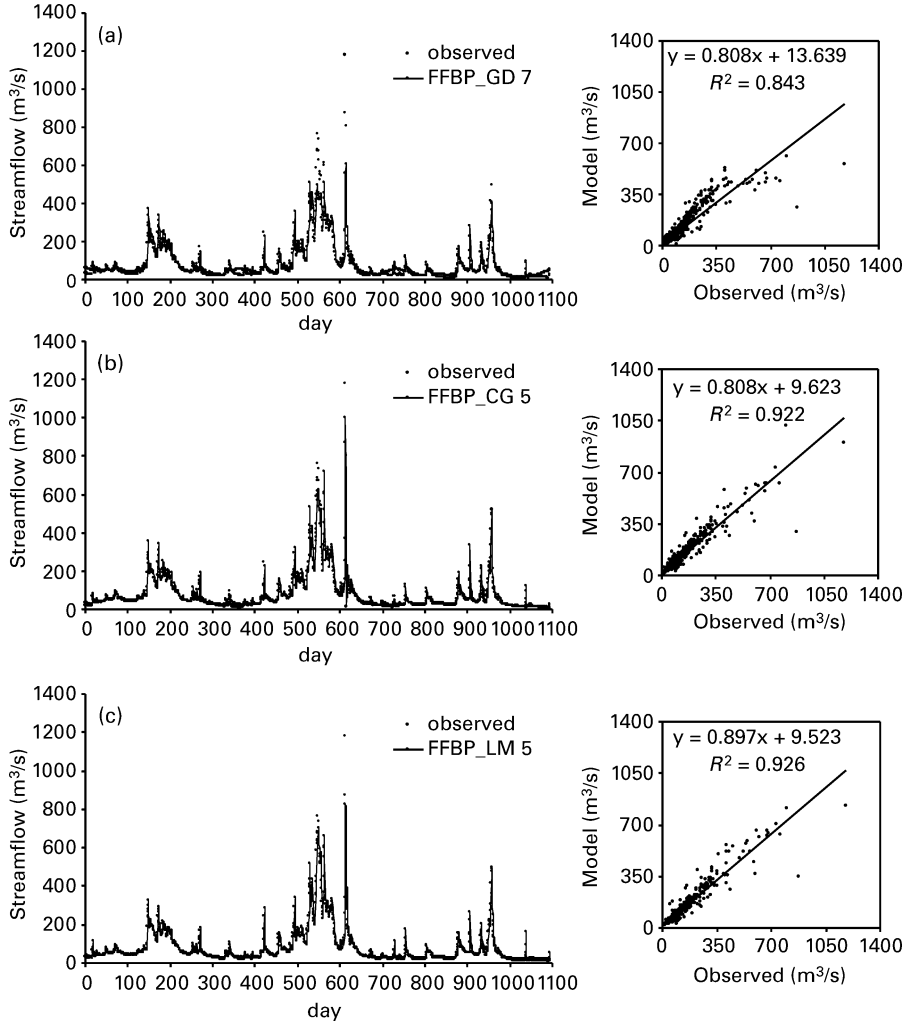

Figure 8 Plotting of forecasting performances for the testing period using (a) FFBP-GD 7, (b) FFBP-CG 5 and (c) FFBP-LM 5 for an input layer with three nodes (ANN(3,5,1))

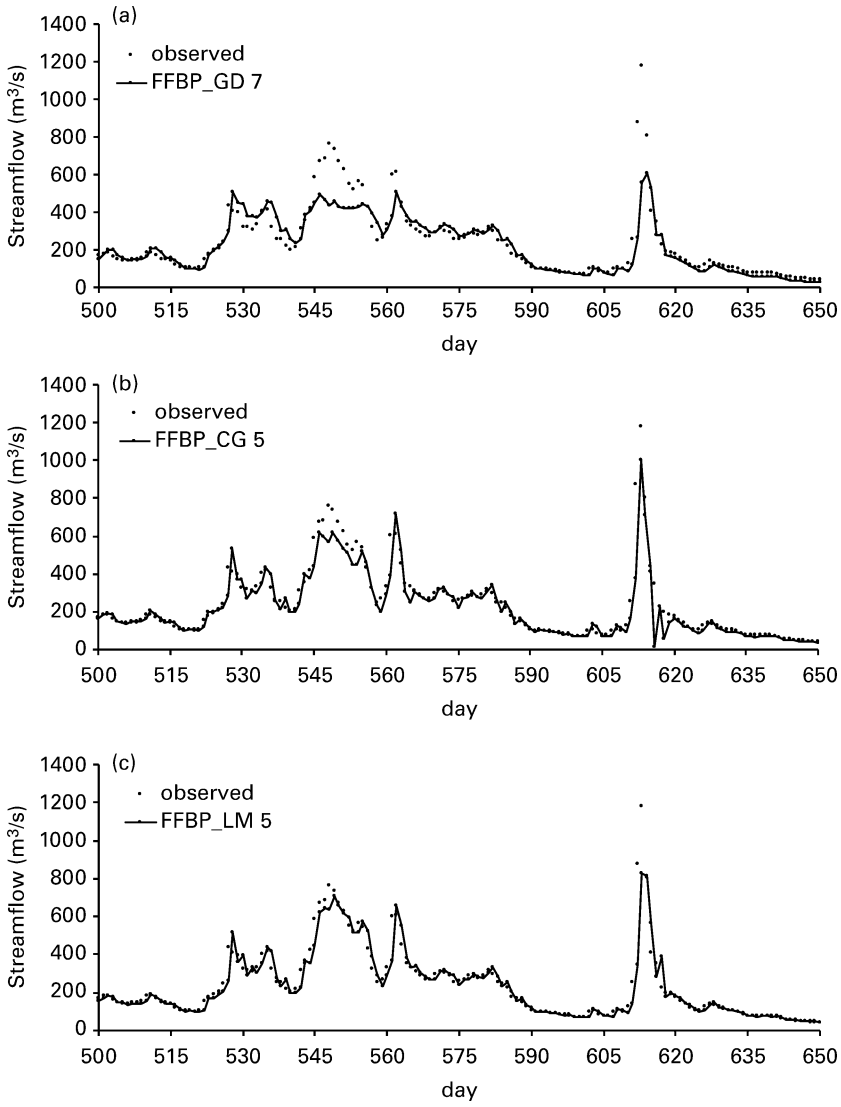

Figure 9 Plotting of forecasting performances for a short interval (500–650th days) in the testing period using (a) FFBP-GD 7, (b) FFBP-CG 5 and (c) FFBP-LM 5 for an input layer with three nodes (ANN(3,5,1))

respectively). However, it can be said that the Levenberg–Marquardt technique predictions approximated the low and medium flow regions of the observed hydrograph quite closely whereas some peak flows were underestimated (Figs 11 and 12).

Conclusions

In this study it was shown that the employment of the *k*-fold partitioning method in feed forward back propagation (FFBP) training processes provided fruitful results. It was seen that with a data period much shorter than the whole training duration similar flow prediction performance could be obtained. However, the selected sub-group statistics should be close to those of the testing values. The FFBP method with Levenberg–Marquardt technique (FFBP-LM) provided some advantages compared with FFBP with gradient descent technique (FFBP-GD) and conjugate gradient (FFBP-CG), such as a shorter training duration and more satisfactory performance criteria. The FFBP-LM applications to 2-days and 3-days ahead

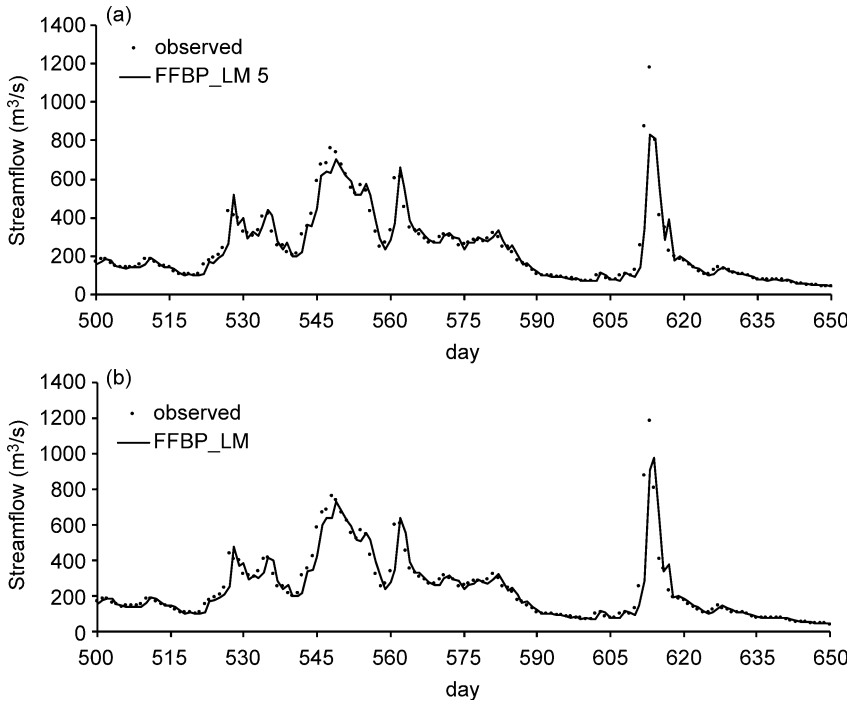

Figure 10 Plotting of forecasting performances for a short interval (500–650th days) in the testing period using (a) FFBP-LM5 and (b) FFBP-LM for an input layer with three nodes (ANN(3,5,1))

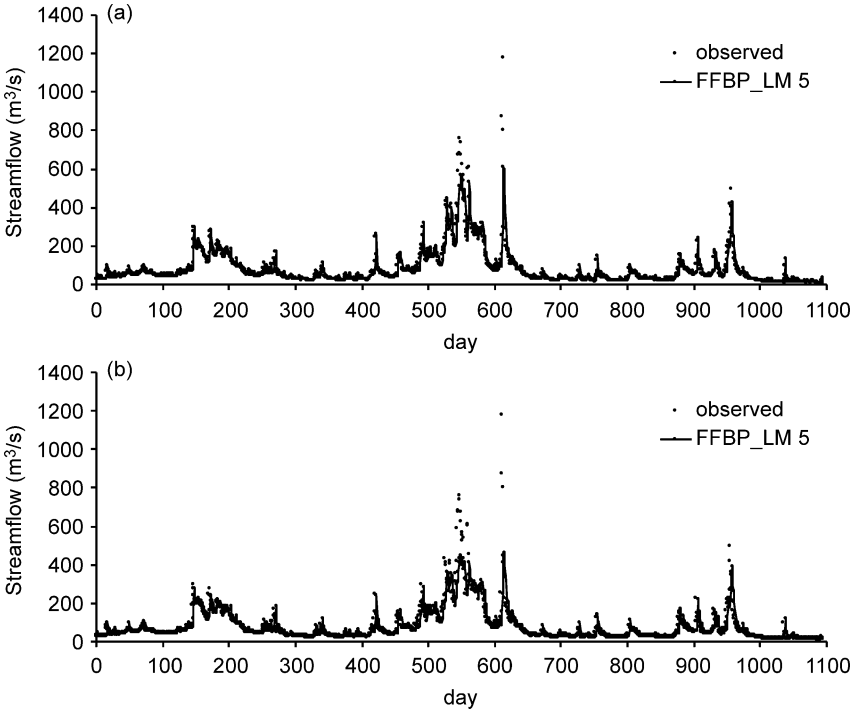

Figure 11 Plotting of forecasting performances for the testing period using FFBP-LM5 in case (a) 2 days ahead and (b) 3 days ahead prediction for an input layer with three nodes (ANN(3,5,1))

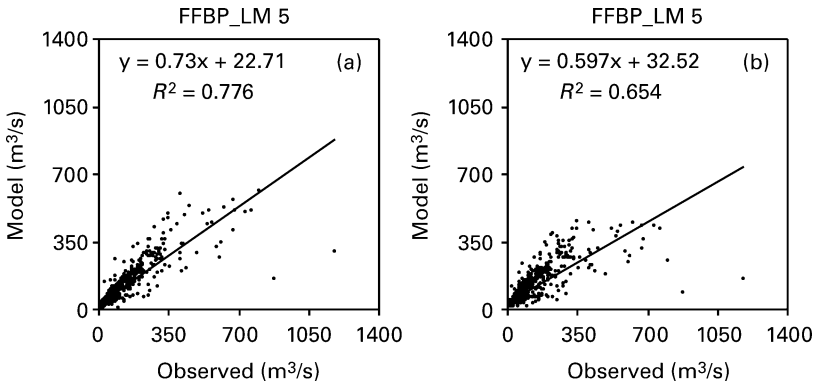

Figure 12 Scatter plots for FFBP-LM 5 in case (a) 2 days ahead and (b) 3 days ahead prediction for an input layer with three nodes (ANN(3,5,1))

forecasting were found successful for low and medium height flows. For these cases, however, deviations were noticed for some peak flows. The effect of the *k*-fold method should also be tested for other ANN training algorithms but FFBP and these results should then be compared with those of the FFBP.

References

Adeli, H. and Hung, S.L. (1995). *Machine Learning Neural Networks, Genetic Algorithms, and Fuzzy Systems*. John Wiley & Sons, New York.

Ali, K. and Pazzani, M. (1996). Error reduction through learning multiple descriptions. *Machine Learning*, **24**(3), 173–206.

Campolo, M., Andreussi, P. and Soldati, A. (1999). River flood forecasting with a neural network model. *Wat. Res. Res.*, **35**(4), 1191–1197.

Cigizoglu, H.K. (2003a). Estimation, forecasting and extrapolation of flow data by artificial neural networks. *Hydrol. Sci. J.*, **48**(3), 349–361.

Cigizoglu, H.K. (2003b). Incorporation of ARMA models into flow forecasting by artificial neural networks. *Environmetrics*, **14**(4), 417–427.

Cigizoglu, H.K (2003c). Generalized regression neural networks in daily river flow forecasting. *Proceedings of the World Water and Environmental Resources Congress*, 23–26 June, Philadelphia, PA.

Cigizoglu, H.K. (2004a). Discussion of "Performance of neural networks in daily streamflow forecasting" by S. Brikundavyi, R. Labib, H.T. Trung and J. Rousselle. *ASCE J. Hydrol. Engng*, **9**(6), 556–557.

Cigizoglu, H.K. (2004b). Estimation and forecasting of daily suspended sediment data by multi layer perceptrons. *Adv. Wat. Res.*, **27**, 185–195.

Cigizoglu, H.K. and Alp, M. (2003). Suspended sediment forecasting by artificial neural networks using hydrometeorologic data. *Proceedings of the World Water and Environmental Resources Congress*, 23–26 June, Philadelphia, PA.

Clair, T.A. and Ehrman, J.M. (1998). Using neural networks to assess the influence of changing seasonal climates in modifying discharge, dissolved organic carbon, and nitrogen export in eastern Canadian rivers. *Wat. Res. Res.*, **34**(3), 447–455.

Coulibaly, P., Anctil, F., and Bobeé, B (1998). Real time neural network based forecasting system for hydropower reservoirs. In: *Proceedings of the First International Conference on New Information Technologies for Decision Making in Civil Engineering*, 10–13 October, University of Quebec, Montreal (ed. E.T. Miresco). pp 1001–1011.

Hagan, M.T. and Menhaj, M.B. (1994). Training feedforward techniques with the Marquardt Algorithm. *IEEE Trans. Neural Networks*, **5**(6), 989–993.

Hsu, K.L., Gupta, H.V. and Sorooshian, S. (1995). Artificial neural network modeling of the rainfall-rainoff process. *Wat. Res. Res.*, **31**(10), 2517–2530.

Jain, S.K., Das, D. and Srivastava, D.K. (1999). Application of ANN for reservoir inflow prediction and operation. *J. Wat. Res. Planning Mngmnt ASCE*, **125**(5), pp. 263–271.

Jain, S.K. (2001). Development of integrated sediment rating curves using ANNs. *J. Hydraul. Engng., ASCE*, **127**(1), 30–37.

Kang, K.W., Park, C.Y. and Kim, J.H. (1993). Neural network and its application to rainfall-runoff forecasting. *Korean J. Hydrosci.*, **4**, 1–9.

Karunanithi, N., Grenney, W.J., Whitley, D. and Bovee, K. (1994). Neural networks for river flow prediction. *J. Comp. Civil Engng, ASCE*, **8**(2), 201–220.

Kisi, O. (2004). River flow modeling using artificial neural networks, ASCE. *J. Hydrol. Engng*, **9**(1), 60–63.

Le Chun, Y. (1985). Une procedure d'apprentissage pour reseau a seuil assymetrique. In: *COGNITIVA 85: A la Frontiere de l'Intelligence Artificielle des Science de la Connaissance des Neurosciences (Paris)* pp. 599–604.

Le Chun, Y. (1986). Learning process in an asymmetric threshold network. In I.E. Bienenstock, F. Fogelman Soulie and G. Weishbuch (Eds), *Disordered Systems and Biological Organization*, Springer-Verlag, New York, pp. 233–240.

Maier, H.R. and Dandy, G.C. (1996). Use of artificial neural networks for prediction of water quality parameters. *Wat. Res. Res.*, **32**(4), 1013–1022.

Marquardt, D. (1963). An algorithm for least squares estimation of non-linear parameters. *J. Soc. Ind. Appl. Math.*, 431–441.

Minns, A.W. and Hall, M.J. (1996). Artificial neural networks as rainfall-runoff models. *Hydrol. Sci. J.*, **41**(3), 399–417.

Parker, D.B. (1985). Learning logic. *Technical Report* TR-47. Center for Computational Research in Economics and Management Science, Massachusetts Institute of Technology, Cambridge, MA.

Raman, H. and Sunilkumar, N. (1995). Multivariate modelling of water resources time series using artificial neural networks. *Hydrol. Sci. J.*, **40**, 145–163.

Reed, R.D. and Marks, R.J. (1998). *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*, MIT Press, Cambridge, MA.

Rumelhart, D.E., Hinton, G.E. and Williams, R.J. (1986). Learning internal representation by error propagation. In D.E. Rumelhart and J.L. McClelland (Eds), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1, MIT Press, Cambridge, MA, pp. 318–362.

Saad, M., Bigras, P., Turgeon, A. and Duquette, R. (1996). Fuzzy learning decomposition for the scheduling of hydroelectric power systems. *Wat. Res. Res.*, **32**(1), 79–186.

Sajikumar, N. and Thandaveswara, B.S. (1999). A non-linear rainfall-runoff model using an artificial neural network. *J. Hydrol.*, **216**, 32–55.

Shamseldin, A.Y. (1997). Application of a neural network technique to rainfall-runoff modeling. *J. Hydrol.*, **199**, 272–294.

Smith, J. and Eli, R.N. (1995). Neural-network models of rainfall-runoff process. *J. Wat. Res. Plan. Mngmnt, ASCE*, **121**(6), 499–508.

Thirumalaiah, K. and Deo, M.C. (1998). Real-time flood forecasting using neural networks. *Computer-Aided Civil Infrastruct. Engng.*, **13**(2), 101–111.

Ting, K.M. and Low, B.T. (1996). Theory combination: an alternative to data combination. *Working Paper* 96/19, Department of Computer Science, University of Waikato.

Tokar, A.S. and Johnson, P.A. (1999). Rainfall-runoff modeling using artificial neural networks. *J.Hydrol. Engng., ASCE*, **4**(3), 232–239.

Zealand, C.M., Burn, D.H. and Simonovic, S.P. (1999). Short term streamflow forecasting using artificial neural networks. *J. Hydrol.*, **214**, 32–48.

Zhu, M.L. and Fujita, M. (1994). Comparisons between fuzzy reasoning and neural network methods to forecast runoff discharge. *J. Hydrosci. Hydraul. Engng.*, **12**(2), 131–141.