HYDROLOGICAL PROCESSES
Hydrol. Process. **16**, 2577–2588 (2002)
Published online 18 April 2002 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/hyp.1015

Real-time recurrent learning neural network for stream-flow forecasting

Fi-John Chang,* Li-Chiu Chang and Hau-Lung Huang
Department of Bioenvironmental Systems Engineering, National Taiwan University, Taipei, Taiwan, ROC

Abstract:

Various types of neural networks have been proposed in previous papers for applications in hydrological events. However, most of these applied neural networks are classified as static neural networks, which are based on batch processes that update action only after the whole training data set has been presented. The time variate characteristics in hydrological processes have not been modelled well. In this paper, we present an alternative approach using an artificial neural network, termed real-time recurrent learning (RTRL) for stream-flow forecasting. To define the properties of the RTRL algorithm, we first compare the predictive ability of RTRL with least-square estimated autoregressive integrated moving average models on several synthetic time-series. Our results demonstrate that the RTRL network has a learning capacity with high efficiency and is an adequate model for time-series prediction. We also investigated the RTRL network by using the rainfall–runoff data of the Da-Chia River in Taiwan. The results show that RTRL can be applied with high accuracy to the study of real-time stream-flow forecasting networks. Copyright © 2002 John Wiley & Sons, Ltd.

KEY WORDS recurrent neural networks; stream-flow forecasting; rainfall–runoff modelling

INTRODUCTION

Building a real-time stream-flow forecasting model has always been one of the most challenging and important tasks for hydrologists in Taiwan, which has the attributes of subtropical climate and high mountains with steep slopes all over the island. In addition, typhoons hit Taiwan around four times a year, bringing heavy rainfalls, which can flood downstream cities within a few hours. Owing to these particular watershed–rainfall characteristics, accurate site-specific predictions of the real-time stream-flow remain a difficult task. Various types of deterministic and stochastic models, as described by Sherman (1932), the Hydrological Engineering Center (1990) and Salas *et al*. (1985) have been used to construct the rainfall–runoff processes in Taiwan. However, some unrealistic assumptions and the lack of verified data/parameters in these models have limited their application and practicality.

Capable of modelling non-linear and complex systems, artificial neural networks (ANNs) provide an alternative approach for accurate stream-flow forecasting. Generally speaking, neural networks are information processing systems devised via imitating brain activity. After McCulloch and Pitts (1943) established the first neural network, many ANNs, such as the back-propagation neural network (Rumelhart *et al*., 1986), the Hopfield neural network (Hopfield, 1984) and the fuzzy neural network (Nie and Linkens, 1994). were developed to solve different problems. More recently, they have been used to deal with stream-flow prediction (Hsu *et al*., 1995; Shamseldin, 1997; Chang and Hwang, 1999; Sajikumar and Thandareswara 1999; Chang *et al*., 2001; Chang and Chen, 2001). Although satisfactory results have been reported, the applied neural networks for hydrological processes, to the best of our knowledge, are all based on batch processes, by

* Correspondence to: Professor F.-J. Chang, Department of Bioenvironmental Systems Engineering and Hydrotech Research Institute, National Taiwan University, Taipei, Taiwan, ROC. E-mail: changfj@ccms.ntu.edu.tw

Received 8 May 2001
Accepted 3 September 2001

which the update action takes place only after the whole training data set has been presented. These neural networks are classified as static neural networks, which can only simulate the short-term memory structures within processes. The extraordinary time variate characteristics of hydrological time series, especially in our case, could not be modelled well. Therefore, we present an alternative approach of the ANN for streamflow forecasting—the real-time recurrent learning (RTRL) algorithm (Williams and Zipser, 1989), which provides a representation of dynamic internal feedback loops to store information for later use and to enhance the efficiency of learning. The application of feedback enables recurrent networks to acquire state representation, which make them suitable devices for diverse applications, such as non-linear prediction, speech processing and plant control (Haykin, 1999).

We began with a series of simulation experiments to investigate the properties of the RTRL algorithm, and then implemented it to model a watershed rainfall–runoff process for real-time stream-flow forecasting.

REAL-TIME RECURRENT LEARNING (RTRL) ALGORITHM

Figure 1 shows the architecture of a multiplayer perceptron neural network, which includes a concatenated input–output layer, a processing layer and an output layer. There are M external inputs and K outputs. Let $\boldsymbol{x}(t)$ denote the $M \times 1$ input vector to the network at discrete time t, $\boldsymbol{z}(t+1)$ denote the corresponding $K \times 1$ output vector and $\boldsymbol{y}(t+1)$ denote the corresponding $N \times 1$ vector one step later at time $t+1$ in the processing layer. The input $\boldsymbol{x}(t)$ and one-step delayed output vector in the processing layer $\boldsymbol{y}(t)$ are concatenated to form the $(M+N) \times 1$ vector $\boldsymbol{\mu}(t)$, in which the ith element is denoted by $\boldsymbol{\mu}_i(t)$. Let A denote the set of indices i for which $\boldsymbol{x}_i(t)$ is an external input, and B denote the set of indices i for which $\boldsymbol{y}_i(t)$ is the output of a unit in the network. We thus have

$$\boldsymbol{\mu}_i(t) = \begin{cases} \boldsymbol{x}_i(t) & \text{if } i \in A \\ \boldsymbol{y}_i(t) & \text{if } i \in B \end{cases}$$

The network is fully interconnected. There are $M \times N$ forward connections and $N \times N$ feedback connections. Let $\mathbf{W}$ denote $N \times (M+N)$, the recurrent weight matrix of the network. To allow each unit a bias weight, we simply include, among the M input lines, one input with a value that is always 1. $\mathbf{W} \leftrightarrow w_{ji}$ and $\mathbf{V} \leftrightarrow v_{kj}$ are the matrix form.

The processing and output layers are also fully connected. Let $\mathbf{V}$ denote the $N \times K$ weight matrix.

The net activity of neuron j at time t, for $j \in B$, is computed by

$$\text{net}_j(t) = \sum_{i \in A \cup B} w_{ji}(t-1)\boldsymbol{\mu}_i(t-1)$$

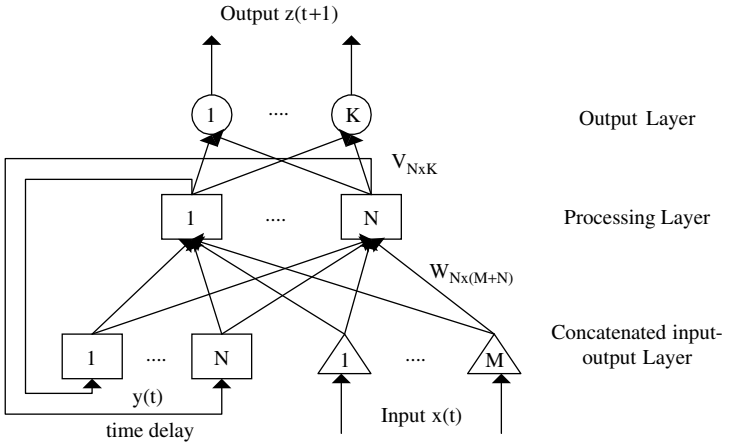

Figure 1. Architecture of RTRL network

The output of neuron j is given by passing $\text{net}_j(t)$ through the non-linearity $f(.)$, yielding

$$\boldsymbol{y}_j(t) = f(\text{net}_j(t))$$

The net output of neuron k at time t is computed by

$$\text{net}_k(t) = \sum v_{kj}(t)\boldsymbol{y}_j(t)$$
$$\boldsymbol{z}_k(t) = f(\text{net}_k(t))$$

Notice that the external input at time t does not influence the output of any neuron until time $t+1$. The above system of equations constitutes the entire dynamics of the network.

Let $d_k(t)$ denote the target value of neuron k at time t. Then we define a time-varying $K \times 1$ error vector $\boldsymbol{e}(t)$, whose kth element is

$$\boldsymbol{e}_k(t) = d_k(t) - \boldsymbol{z}_k(t)$$

Define the instantaneous overall network error at time t as

$$E(t) = \frac{1}{2}\sum_{k=1}^{K} \boldsymbol{e}_k^2(t)$$

The cost function is obtained by summing $E(t)$ over all time T

$$E_{\text{total}} = \sum_{t=1}^{T} E(t)$$

To minimize the cost function, the gradient descent method is applied to adjust the weights ($\mathbf{V}$ and $\mathbf{W}$) along the negative of ∇E_{total}. Because the total error is the sum of the errors at the individual time steps, one way to compute this gradient is by accumulating the value of ∇E for each time step along the trajectory. The weight change for any particular weight v_{kj} can thus be written as

$$\Delta \boldsymbol{V}_{kj} = \sum_{t=t_0+1}^{t_1} \Delta v_{kj}(t)$$
$$\Delta v_{kj}(t) = -\eta_1 \frac{\partial E(t)}{\partial v_{kj}(t)}$$

where η_1 is the learning-rate parameter. Now

$$\frac{\partial E(t)}{\partial v_{kj}(t)} = -\boldsymbol{e}_k(t) f'(\text{net}_k(t))\boldsymbol{y}_j(t)$$

The same method is also implemented for weight w_{mn}, where

$$\Delta w_{mn}(t-1) = -\eta_2 \frac{\partial E(t)}{\partial w_{mn}(t-1)}$$

and η_2 is the learning-rate parameter. The partial derivative $\dfrac{\partial E(t)}{\partial w_{mn}(t-1)}$ can be obtained by the chain rule for differentiation as follows

$$\frac{\partial E(t)}{\partial w_{mn}(t-1)} = \left[\sum_{k=1}^{K} -\boldsymbol{e}_k(t) f'(\text{net}_k(t)) v_{kj}(t)\right] \frac{\partial \boldsymbol{y}_j(t)}{\partial w_{mn}(t-1)}$$

$$\Rightarrow \frac{\partial \boldsymbol{y}_j(t)}{\partial w_{mn}(t-1)} = f'(\text{net}_j(t)) \frac{\partial \text{net}_j(t)}{\partial w_{mn}(t-1)}$$

$$\Rightarrow \frac{\partial \text{net}_j(t)}{\partial w_{mn}(t-1)} = \sum_{i \in A \cup B} \frac{\partial (w_{ji}(t-1)\boldsymbol{\mu}_i(t-1))}{\partial w_{mn}(t-1)}$$

$$\Rightarrow \frac{\partial \text{net}_j(t)}{\partial w_{mn}(t-1)} = \sum_{i \in A \cup B} \left[w_{ji}(t-1) \frac{\partial \boldsymbol{\mu}_i(t-1)}{\partial w_{mn}(t-1)} + \frac{\partial w_{ji}(t-1)}{\partial w_{mn}(t-1)} \boldsymbol{\mu}_i(t-1) \right]$$

The derivative $\dfrac{\partial w_{ji}(t-1)}{\partial w_{mn}(t-1)}$ is equal to 1 only when $j = m$ and $i = n$; otherwise, it is zero. We may therefore rewrite the above equations as

$$\frac{\partial \text{net}_j(t)}{\partial w_{mn}(t-1)} = \sum_{i \in A \cup B} w_{ji}(t-1) \frac{\partial \boldsymbol{\mu}_i(t-1)}{\partial w_{mn}(t-1)} + \delta_{mj} \boldsymbol{\mu}_n(t-1)$$

where δ_{mj} is the Kronecker delta with value 1 if and only if $j = m$; otherwise zero.

From the definition of $\boldsymbol{\mu}_i(t)$, we also note that

$$\frac{\partial \boldsymbol{\mu}_i(t-1)}{\partial w_{mn}(t-1)} = \begin{cases} 0 & \text{if } i \in A \\ \dfrac{\partial \boldsymbol{y}_i(t-1)}{\partial w_{mn}(t-1)} & \text{if } i \in B \end{cases}$$

We assume that the initial state of the network at time $t = 0$ has no functional dependence on the synaptic weights, so that

$$\frac{\partial \boldsymbol{y}_j(0)}{\partial w_{mn}(0)} = 0$$

$$\frac{\partial \boldsymbol{y}_j(t)}{\partial w_{mn}(t-1)} = f'(\text{net}_j(t)) \left[\sum_{i \in B} w_{ji}(t-1) \frac{\partial y_i(t-1)}{\partial w_{mn}(t-1)} + \delta_{mj} \boldsymbol{\mu}_n(t-1) \right]$$

Let $\dfrac{\partial \boldsymbol{y}_j(t)}{\partial w_{mn}(t)} \approx \dfrac{\partial \boldsymbol{y}_j(t)}{\partial w_{mn}(t-1)}$ and define a dynamic system described by a triple indexed set of variables $\{\pi^j_{mn}\}$, where $\pi^j_{mn}(t) = \dfrac{\partial \boldsymbol{y}_j(t)}{\partial w_{mn}(t)}$ for all $j \in B$, $m \in B$ and $n \in A \cup B$.

For each time step t and all appropriate m, n and j, the dynamics of the system are governed by

$$\pi^j_{mn}(t) = f'(\text{net}_j) \left[\sum_{i \in B} w_{ji}(t-1) \pi^i_{mn}(t-1) + \delta_{mj} \boldsymbol{u}_n(t-1) \right]$$

with initial condition $\pi^j_{mn}(0) = 0$. Then the weight changes can be computed as

$$\Delta w_{mn}(t-1) = \eta_2 \left[\sum \boldsymbol{e}_k(t) f'(\text{net}_k(t)) v_{kj}(t) \right] \pi^j_{mn}(t)$$

$$\Delta v_{kj}(t) = \eta_1 \boldsymbol{e}_k(t) f'(\text{net}_k(t)) \boldsymbol{y}_j(t)$$

SIMULATING SYNTHETIC TIME SERIES

Statistical approach to the hydrological time-series forecasting involves constructing stochastic models to predict the value based on previous observations. This often is accomplished by using linear stochastic differential equation models with random inputs. By far the most profound models are the autoregressive integrated moving average (ARIMA) models. Detailed discussions and applications of these models may be found in Box and Jenkins (1976) and Salas *et al*. (1985). To show the robustness of RTRL, we compare the predictive ability of the least-squares estimated ARIMA predictors and RTRL on five different time-series synthetic data. The five time-series models are shown as:

1. AR(1) $(x(t) - 0{\cdot}4) = 0{\cdot}75(x(t-1) - 0{\cdot}4) + \boldsymbol{e}_t$, with $\boldsymbol{e}_t^{\text{i.i.d}} \approx N(0, 1{\cdot}0)$;
2. AR(2) $(x(t) - 0{\cdot}4) = 0{\cdot}6(x(t-1) - 0{\cdot}4) + 0{\cdot}2(x(t-2) - 0{\cdot}4) + \boldsymbol{e}_t$, with $\boldsymbol{e}_t^{\text{i.i.d}} \approx N(0, 1{\cdot}0)$;
3. IMA(1,0) $x(t) = x(t-1) + \boldsymbol{e}_t$, with $\boldsymbol{e}_t^{\text{i.i.d}} \approx N(0, 1{\cdot}0)$;
4. IMA(1,1) $(1 - B)x(t) = (1 - 0{\cdot}6B)\boldsymbol{e}_t$, with $\boldsymbol{e}_t^{\text{i.i.d}} N \approx (0, 1{\cdot}0)$;
5. ARIMA(1,1,1) $(1 - B)x(t) = 0{\cdot}4(1 - B)x(t-1) + \boldsymbol{e}_t - 0{\cdot}3\boldsymbol{e}_{t-1}$, with $\boldsymbol{e}_t^{\text{i.i.d}} \approx N(0, 1{\cdot}0)$

The $\boldsymbol{e}_t$ values are chosen from an independent and identically normal distribution (i.i.d) with mean zero and variance one. Each series is generated by MATLAB the Maths Works, Inc. After the first 50 generated data are skipped, the initial 100 data are used to construct the RTRL network and to estimate the optimal parameters of the ARIMA models. For simplicity, without model identification, we use the given types of the ARIMA models, and only their parameters are estimated. The least-squares estimator in MATLAB is used to estimate the ARIMA models' parameters. The architecture of the RTRL network for one-step ahead prediction is illustrated in Figure 1. There is only one single input, the current value $\boldsymbol{x}(t)$, and the processing layer has been investigated through the training data set and identified to have five to seven nodes for different times series. Both learning rates (η_1 and η_2) are 100·0. Because we only perform one-step ahead prediction, $\hat{\boldsymbol{x}}(t+1)$, the output node is, then, set as 1.

The one-step ahead prediction is performed based on (i) the constructed RTRL networks and (ii) the fitted ARIMA models for each case, respectively. As the prediction focuses on optimum prediction by the minimum mean squared error, the criteria of mean square error (MSE) and mean absolute error (MAE) are used for the purpose of comparison.

To learn the models' performance at different stages, the models' predictions in (i) training sets (100 data) and (ii) testing sets (200 data) at three different periods are presented, respectively. The results are summarized in Tables I–V. To illustrate the models' performances in general, the series of ARIMA(1,1,1) and its corresponding prediction errors by (i) the fitted ARIMA model and (ii) the constructed RTRL networks are shown in Figure 2. Apparently, the results show that the RTRL networks have very similar performances as those of the fitted ARIMA models and the networks could persist with the main statistical properties, in terms of reasonable MAE and MSE values, of the generated ARIMA series. The results of RTRL in the

Table I. The performance of fitted AR(1) and RTRL at different stages

Set	Time-step	AR(1)		RTRL	
		MAE	MSE	MAE	MSE
Training	1–30	1·016	1·569	1·640	4·906
	1–100	0·836	1·107	1·053	2·059
Testing	101–130	0·676	0·829	0·821	1·152
	151–180	0·741	0·848	0·852	1·197
	271–300	0·614	0·657	0·667	0·789
	101–300	0·722	0·876	0·809	1·114

Table II. The performance of fitted AR(2) and RTRL at different stages

Set	Time-step	AR(2)		RTRL	
		MAE	MSE	MAE	MSE
Training	1–30	0·879	1·155	1·115	2·994
	1–100	0·853	1·087	1·015	1·626
Testing	101–130	0·698	0·733	0·736	0·777
	151–180	0·709	0·807	0·787	0·955
	271–300	0·620	0·700	0·782	0·809
	101–300	0·745	0·880	0·846	1·114

Table III. The performance of fitted IMA(1,0) and RTRL at different stages

Set	Time-step	IMA(1,0)		RTRL	
		MAE	MSE	MAE	MSE
Training	1–30	0·721	0·737	1·112	3·379
	1–100	0·654	0·672	0·998	2·770
Testing	101–130	0·891	1·1819	0·976	1·594
	151–180	1·023	1·6276	1·033	1·588
	271–300	0·825	1·0487	0·809	1·011
	101–300	0·891	1·2053	0·923	1·271

Table IV. The performance of fitted IMA(1,1) and RTRL at different stages

Set	Time-step	IMA(1,1)		RTRL	
		MAE	MSE	MAE	MSE
Training	1–30	0·878	1·097	1·627	11·735
	1–100	0·828	1·060	0·988	1·596
Testing	101–130	0·964	1·452	0·929	1·455
	151–180	0·951	1·278	0·931	1·280
	271–300	0·777	0·837	0·788	0·830
	101–300	0·821	1·096	0·853	1·170

Table V. The performance of fitted ARIMA(1,1,1) and RTRL at different stages

Set	Time-step	ARIMA(1,1,1)		RTRL	
		MAE	MSE	MAE	MSE
Training	1–30	0·604	0·596	1·686	12·828
	1–100	0·790	0·966	1·243	3·257
Testing	101–130	0·772	0·880	0·952	1·545
	151–180	0·764	0·898	0·742	0·871
	271–300	0·797	0·909	0·774	0·916
	101–300	0·781	0·954	0·808	1·059

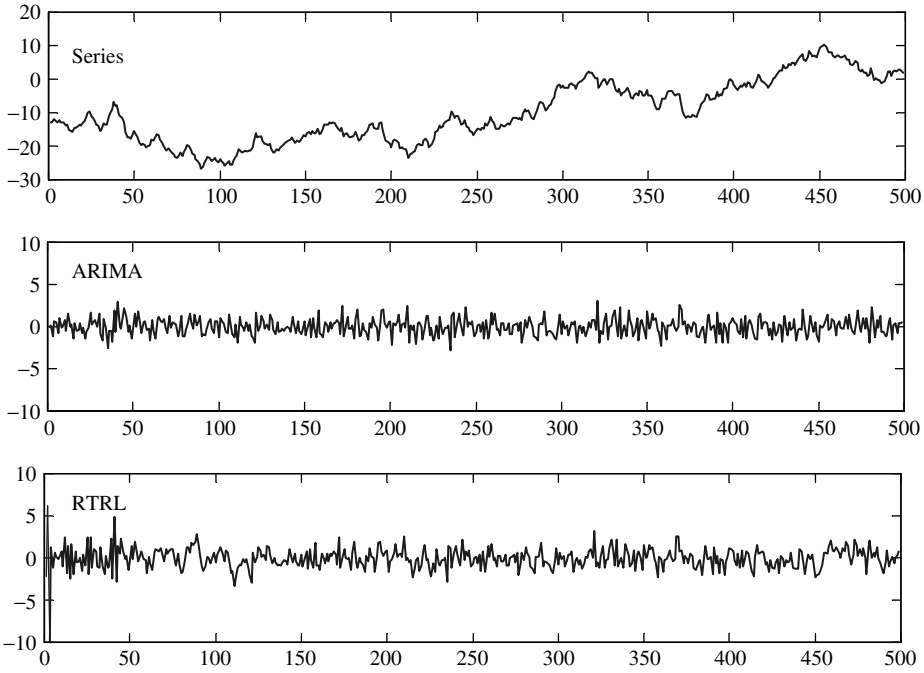

Figure 2. The series of ARIMA(1,1,1) and its corresponding errors using a fitted ARIMA model and a constructed RTRL network

training set indicate that after about 30 steps on-line input training, RTRL can grasp the major trends in all cases. All of them indicate that RTRL has an efficient ability to learn and a high accuracy.

APPLICATION

The above method is applied to the upstream of the Da-Chia River for predicting real-time stream flow. The Da-Chia River is located in central Taiwan and has a total catchment size of 1236 km^2. Being the steepest channel in Taiwan, the Da-Chia River has a length of 140 km and an average channel slope of 1/39. A series of hydraulic structures were constructed for power generation. Locations of the basin studied (area = 514 km^2) and gauge stations used are shown in Figure 3. A station for the stream-flow data is denoted by a triangle and the precipitation stations are shown as circles. Son-Mou gauge station was established to measure the inflow to the De-Chi Reservoir, the upmost and pivotal reservoir in the Da-Chia River. Accurate stream-flow forecasting is extremely important for the operation of the De-Chi Reservoir. The stream flow (m^3/s) and precipitation (mm/h) data used here are gathered by the Taiwan Power Company.

There are four rainfall gauges above the Son-Mou stream-flow gauge. To construct a one-step ahead stream-flow forecasting, we investigate the RTRL and ARMAX (Autoregressive moving average with exogenous inputs) models' performance based on current rainfall at four gauges and stream-flow data. The performances of these two methods are presented based on the criteria of annual peak-flow estimation, mean absolute error (MAE) and relative mean absolute error (RMAE) as shown below

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} (\hat{Q}_i - Q_i)}{n}$$

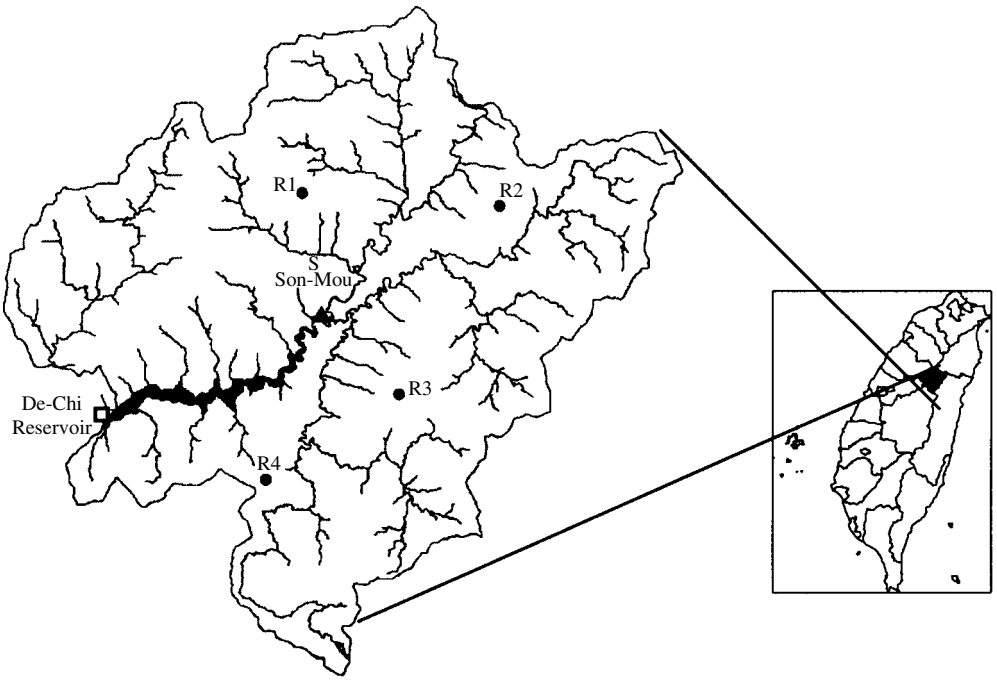

Figure 3. Location of gauge stations in the upstream part of the Da-Chia River

Table VI. Table of model results for different years: Q_o is annual mean of observed stream-flow (cms), Q_p is observed annual peak flow (cms) and $\hat{Q}_p$ is estimated annual peak flow (cms)

Year	Q_o	Q_p	ARMAX(1,1,3)			RTRL		
			MAE	RMAE	$\hat{Q}_p$	MAE	RMAE	$\hat{Q}_p$
1992	45·87	699	0·792	0·0173	695	0·648	0·0141	703
1993	15·85	214	0·215	0·0136	194	0·190	0·0120	199
1994	34·66	1120	0·930	0·0268	1075	0·741	0·0214	1190
1995	19·01	105	0·239	0·0126	102	0·223	0·0118	102
1996	24·04	1090	0·476	0·0198	1020	0·415	0·0172	1062

$$\mathrm{RMAE} = \frac{\mathrm{MAE}}{\overline{Q}}$$

The hourly data of 1992 are used to construct the ARMAX and the RTRL network. The ARMAX(1,1,3) model is shown by the following

$$S(t) = a_1 S(t-1) + \sum_{i=1}^{3} b_{1i} R1(t-i) + \sum_{i=1}^{3} b_{2i} R2(t-i) + \sum_{i=1}^{3} b_{3i} R3(t-i) + \sum_{i=1}^{3} b_{4i} R4(t-i) + \boldsymbol{e}(t) + c_1 \boldsymbol{e}(t-1)$$

where $S(t)$ is stream flow of the Son-Mou stream-flow gauge at time t; $R1(t)$, $R2(t)$, $R3(t)$ and $R4(t)$ are precipitation of four rainfall gauges at time t; $\boldsymbol{e}(t)$ is the model error at time t; and a_1, b_{ij}, c_1 are the model's parameters to be estimated. The model and its parameters can be obtained by using MATLAB.

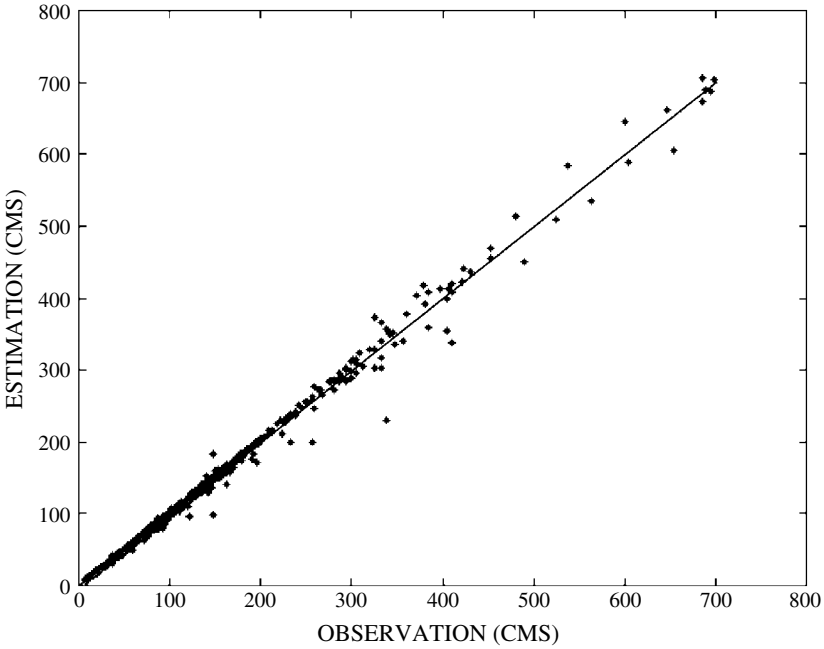

Figure 4. Comparison of observed and forecast stream flow at the Son-Mou gauge in 1992

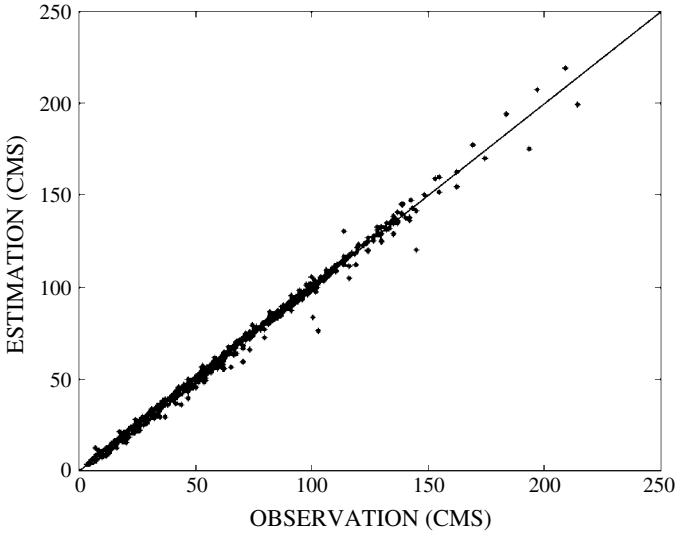

Figure 5. Comparison of observed and forecast stream flow at Son-Mou gauge in 1993

The processing layer of RTRL is constructed to have five nodes. Both learning rates (η_1 and η_2) are 30·0. We found that after 30 steps on-line input training, RTRL can appropriately predict the one-hour ahead stream flow; that is, the network tends to be stable in its forecasting ability. The structures of ARMAX and RTRL are then applied to four different years without any further modifications. Summarized results of both methods are presented in Table VI. Apparently RTRL has a better performance than ARMAX. Figures 4 through to 8 show the observed and forecast streamflow by RTRL in 1992 to 1996, respectively. According to these

results the network provides stable and precise predictions. The mean absolute error (MAE) is smaller than 0·8 cms and the RMAE is less than 2·5%. The result of one-step ahead prediction by RTRL in the case of typhoon Herb, which hit Taiwan in 1996 and was the largest typhoon in last 20 years, is presented in Figure 9. Although it has a slight time shift problem, the forecast, in general, is quite adequate.

CONCLUSIONS

The RTRL is an algorithm for training a completely recurrent, continually updated network to learn tasks. This feature is especially important for the extraordinary time variate characteristics of hydrological time-series. We

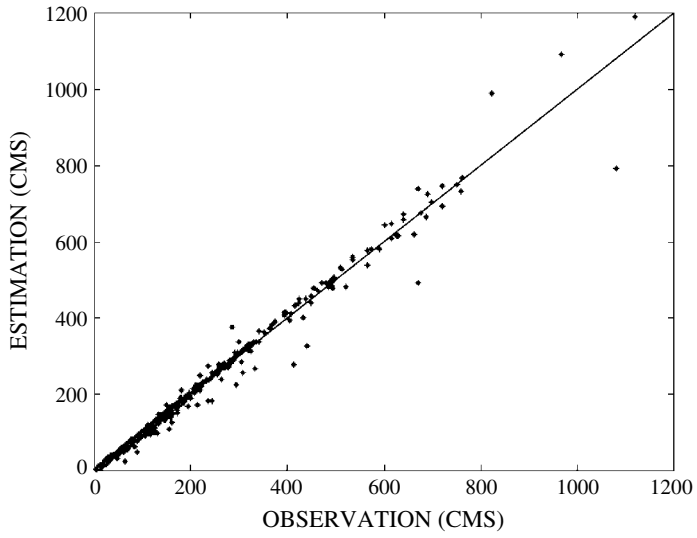

Figure 6. Comparison of observed and forecast stream flow at Son-Mou gauge in 1994

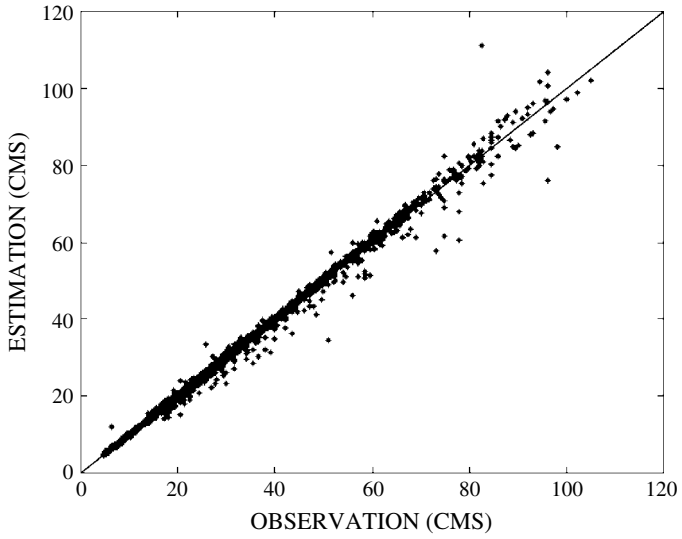

Figure 7. Comparison of observed and forecast stream flow at Son-Mou gauge in 1995

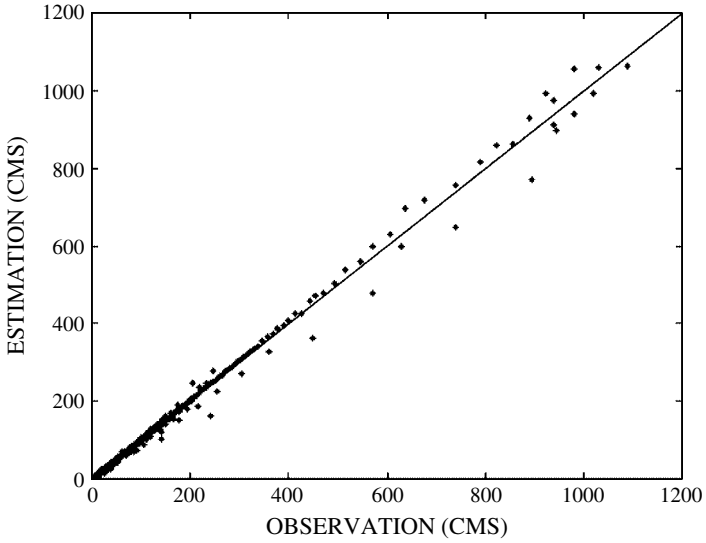

Figure 8. Comparison of observed and forecast stream flow in Son-Mou gauge in 1996

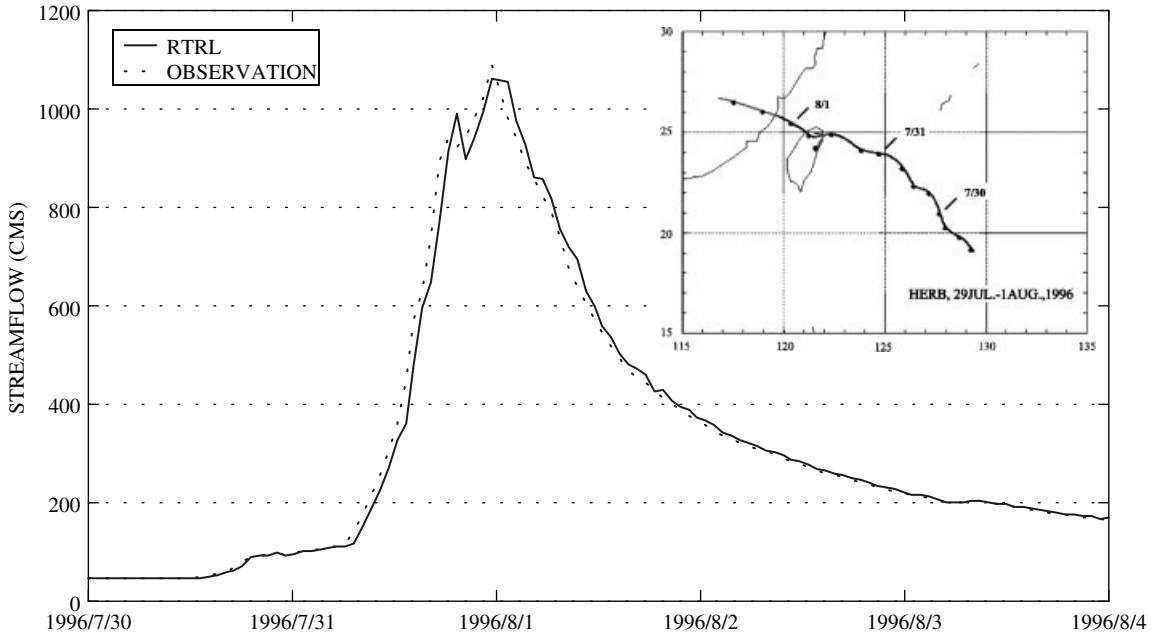

Figure 9. The observed and forecast stream flow at Son-Mou gauge during passage of Typhoon Herb in 1996

first use five different ARIMA time-series of simulation experiments to investigate the power and properties of this algorithm. The results indicate that the RTRL networks have similar performance to those of the fitted ARIMA models for one-step ahead prediction, and the networks could persist with the main statistical properties of the generated ARIMA series. This reveals the excellent learning ability and good performance of RTRL. Overall, RTRL is a well-suited model for time series that possess autoregressive moving average components.

The rainfall and runoff data of Da-Chia river in Taiwan are used to demonstrate the practicability and applicability of RTRL for real-time stream-flow forecasting. A comparison of RTRL and ARMAX is also performed. The results show that (i) RTRL has better performance than ARMAX, and (ii) RTRL can be applied successfully to building a real-time stream-flow forecasting network with high accuracy. To deal with the one-step-ahead forecasting $\boldsymbol{x}(t+1)$, both on synthetic time-series and rainfall–runoff series, only current information $\boldsymbol{x}(t)$ is used as input. This provides evidence that the use of a dynamically driven feedback algorithm has the potential of reducing the memory requirement significantly.

ACKNOWLEDGEMENT

This paper is based on partial work supported by National Science Council, R.O.C. (Grant No. NSC89-2313-B-002-041).

REFERENCES

Box GEP, Jenkins GM. 1976. *Time Series Analysis Forecasting and Control*, 2nd edn. Holden-Day: San Francisco.

Chang FJ, Chen YC. 2001. A counterpropagation fuzzy-neural network modeling approach to real-time stream-flow prediction. *Journal of Hydrology* **245**: 153–164.

Chang FJ, Hwang YY. 1999. A Self-organization algorithm for real-time flood forecast. *Hydrological Processes* **13**(2): 123–138.

Chang FJ, Hu HF, Chen YC. 2001. Counterpropagation fuzzy neural network for stream-flow reconstruction. *Hydrological Processes* **15**(2): 219–232.

Haykin S. 1999. *Neural Networks: a Comprehensive Foundation*, 2nd edn. Prentice Hall: Upper Saddle River, NJ.

Hopfield JJ. 1984. Neuron with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences, USA* **81**: 3088–3092.

Hsu K-L, Gupta HV, Sorooshian S. 1995. Artificial neural network modeling of the rainfall–runoff process. *Water Resources Research* **31**(10): 2517–2530.

Hydrological Engineering Center. 1990. *HEC-1 Flood Hydrograph Package. Program Users Manual*. U.S. Army Corps of Engineers: Davis, CA.

McCulloch WS, Pitts W. 1943. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics* **5**: 115–133.

Nie J, Linkens DA. 1994. Fast self-learning multivariable fuzzy controllers constructed from a modified CPN network. *International Journal of Control* **60**(3): 369–393.

Rumelhart DE, Hinton GE, Williams RJ. 1986. Learning internal representation by error propagation. *Parallel Distributed Processing* **1**: 318–362.

Sajikumar N, Thandaveswara BS. 1999. A non-linear rainfall–runoff model using an artificial neural network. *Journal of Hydrology* **216**: 32–55.

Salas JD, Delleur JW, Yevjevich V, Lane WL. 1985. *Applied Modeling of Hydrologic Time Series*. Water Resources Publications: Littleton, Colorado.

Shamseldin AY. 1997. Application of a neural network technique to rainfall–runoff modelling. *Journal of Hydrology* **199**: 272–294.

Sherman LK. 1932. Stream-Flow from rainfall by the unit-graph method. *Engineering News Record* **108**: 501–505.

Williams R, Zipser D. 1989. A learning algorithm for continually running fully recurrent neural network. *Neural Computation* **1**: 270–280.

