International Journal of Civil Engineering (IJCE)
ISSN: 2278-9987
Vol.1, Issue 1 Aug 2012 58-69


International Academy of Science, Engineering and Technology
Connecting Researchers; Nurturing Innovations

# PREDICTION OF AVERAGE ANNUAL DISCHARGE IN A RIVER: A NEURAL NETWORK APPROACH

**SABITA MADHAVI SINGH & P. R. MAITI**

Department of Civil Engineering, Institute of Technology, Banaras Hindu University, Varanasi, India

## ABSTRACT

This paper evaluates the feasibility of using artificial neural network (ANN) technique for predicting the average discharge using available rainfall data. Input data for both models include the current and preceding records of rainfall gathered at the 56 observation stations in the Godavari basin. Here radial basis function neural network (RBFNN) and back propagation neural network (BPNN) are employed to develop model for discharge forecasting using Matlab6.5 Neural Network Toolbox. Six different types of network architectures and training algorithms are investigated and compared in terms of model prediction efficiency and accuracy. The proposed ANN technique using minimum number of hidden nodes along with minimum number of variables consistently produced the best performing network based simulation models. Results obtained indicate that RBFNN show better predicting capacity than BPNN and can provide an alternate and practical tool for discharge forecast, which is particularly useful for assisting small urban watersheds to issue timely and early flood warnings.

**KEYWORDS**: Annual discharge, Rainfall, Artificial neural network, Radial basis function neural network

## INTRODUCTION

The rainfall-runoff transformation is among one of the most complex hydrological phenomena to comprehend, as it usually involves a number of interconnected elements, such as evapotranspiration, infiltration, surface and subsurface runoff generation and routing, etc. The understanding of these hydrological processes is further complicated with the consideration of heterogeneity of the watershed geo-morphological characteristics (such as soil type, vegetation cover, etc.) and the spatial and temporal variations of model inputs (such as rainfall patterns). For many years, hydrologists have endeavored to better understand the rainfall-runoff transformation process in an effort to develop conceptual models with improved performance. However, the quest for this understanding is not straightforward. Watershed analysis and modeling require extensive time series of stream flow and rainfall data, not frequently available. Many analytical methods have been developed to estimate stream flow from rainfall measurements over the watershed. Runoff estimation is of high importance for many practical engineering applications and in watershed management activities. Earlier hydrological models are based on the mathematical representation of watershed processes which significantly requires effort of data

collection of rainfall, stream-flow and watershed characteristics. Such models also require assessment of model calibration and verification of performance of the model (Bertoni et al. (1992) and Zealand et al. (1997)).Increasing applications of ANN in various aspects of hydrology have been found in ASCE, (2000) and many studies have shown the potential of ANN in modeling the rainfall–runoff relationship by Tokar and Johnson(1999) and Coulibaly et al.,( 2000).

Monthly runoff simulation using ANN have been studied by many researchers and reached encouraging results for the examined basins found in Lorrai and Sechi (1995); Anmala et al.(2000). Xu et al. (1996) has developed and applied a set of conceptual water balance models to estimate monthly river flow in eleven catchments in central Sweden, based on monthly temperature and precipitation. In the most of these studies, the feed-forward method (FF) was employed to train the neural networks. The performance of the FFBP was found to be superior to conventional statistical and stochastic methods in continuous flow series forecasting (Brikundavyi et al., (2002); Cigizoglu, (2003)). Maier and Dandy (2000) summarized the methods used in the literature to overcome this problem of training a number of networks starting with different initial weights, the on-line training mode used to help the network to escape local minima, the inclusion of the addition of random noise, and the employment of second order or global methods. Thirumalaiah and Deo (1998); Thirumalaiah and Deo (2000) used conjugate gradient and cascade correlation algorithms together for different hydrological applications.

The objectives of this study is to examine the capability of back propagation neural network (BPNN) and radial basis function neural network(RBFNN) for the prediction of average annual discharge of 30 stations during the year 1993-94 in Godavari basin. It can provide a viable alternative of the hydrologic application requires that an accurate prediction of discharge behavior be provided using only the available time series and rainfall data, and with relatively moderate conceptual understanding of the hydrologic dynamics of the particular basin under investigation.

## STUDY AREA AND THE DATA USED

The river Godavari rises in the Nashik district of Maharastra, about 80 Km from the Arabian Sea, at an elevation of 1067 m . The total length of the river from the source to its outfall into the sea (Bay of Bengal) is about 1465 Km of which 694 Km is in Maharastra. And remaining 771 Km is in Andhra Pradesh. It is the largest of the peninsular river and the third largest in India. The Godavari Basin extends over an area of 312812 Km which is nearly 10% of the total geographical area of the country. The Basin lies in the Deccan Plateau and is situated latitude 16 $^{o}$16´N and 23$^{o}$16´N and longitude 73$^{o}$26´E and 83$^{o}$07´E. Central Water Commission (CWC) has been conducting hydrological observations in Godavari basin since mid 1960s. Hydrological observation stations are located on the main river and as well as on all tributaries. There are 56 hydrological observation stations under operation in 1994-95 (Fig. 1). Out of these 8 stations are on the Godavari and the remaining are on the tributaries. The data used in this paper is taken from report “Statistical Profile of Godavari Basin” published by Information System Directorate Performance Overview & Management Improvement Organization Water Planning & Project Wing, Central Water Commission on January, 1999.

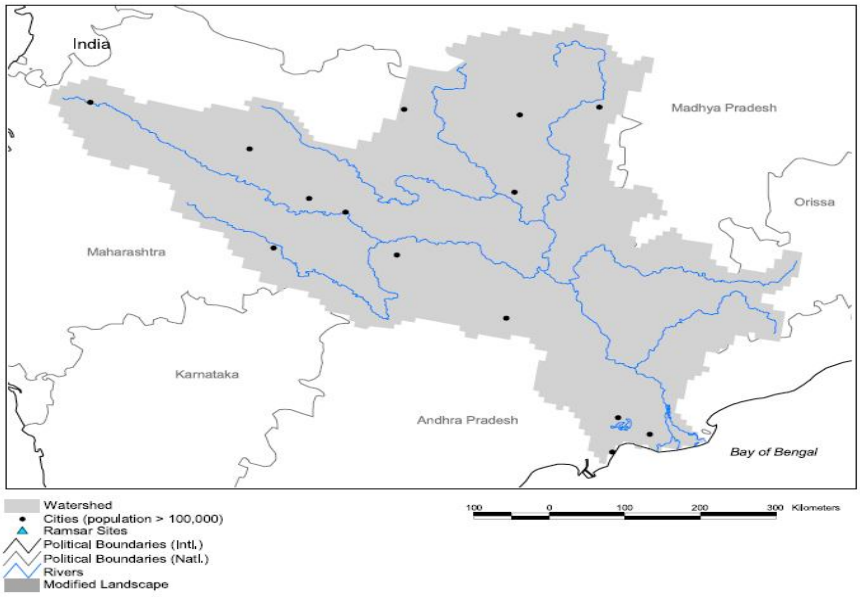


**Fig.1. Godavari Basin**

## ARTIFICIAL NEURAL NETWORKS

An artificial neural network is a highly interconnected network of many simple processing units called neurons, which are analogous to the biological neurons in the human brain. Neurons having similar characteristics in an ANN are arranged in groups called layers. The neurons in one layer are connected to those in the adjacent layers, but not to those in the same layer. The strength of connection between the two neurons in adjacent layers is represented by what is known as a 'connection strength' or 'weight'. An ANN normally consists of three layers, an input layer, a hidden layer, and an output layer. There are primarily two types of training mechanisms, supervised and unsupervised. A supervised training algorithm requires an external teacher to guide the training process. This typically involves a large number of examples (or patterns) of inputs and outputs for training. The inputs in an ANN are the cause variables and outputs are the effect variables of the physical system being modeled. The primary goal of training is to minimize the error function by searching for a set of connection strengths that cause the ANN to produce outputs that are equal to or closer to the targets. A supervised training mechanism called back-propagation training algorithm is normally adopted in most of the engineering applications.

### Back Propagation Neural Network (BPNN)

In the back-propagation training mechanism, the input data are presented at the input layer, the information is processed in the forward direction, and the output is calculated at the output layer. The target values are known at the output layer, so that the error can be estimated. The tan-sigmoid function is the transfer function employed in the hidden layers and has the following form:

$$f(x) = \tan\left(\frac{1}{1+e^{-x}}\right) \tag{1}$$

where *x* is the input. This function of equation (1) is bounded between –1 and 1 so input data are usually normalized within the same range. Neurons in the output layer receive weighted input and generate the final output using a linear transfer function.

The total error at the output layer is distributed back to the ANN and the connection weights are adjusted. The error function is given by equation(2):

$$E = \frac{1}{2}\sum_{k=1}^{K} e_k^2 = \frac{1}{2}\sum_{k=1}^{K}(t_k - z_k)^2 \tag{2}$$

where, $t_k$ is a predefined network output (or desired output or target value) and $e_k$ is the error in each output node. The goal is to minimize E so that the weight in each link is accordingly adjusted and the final output can match the desired output. To get the weight adjustment, the gradient descent strategy is employed. In the link between hidden and output layers, computing the partial derivative of E with respect to the weight $w_{jk}$ produces

$$\begin{aligned}\frac{\partial E}{\partial w_{jk}} &= \frac{\partial E}{\partial z_k}\frac{\partial z_k}{\partial Y_k}\frac{\partial Y_k}{\partial w_{jk}} = -e_k\frac{\partial f(Y_k)}{\partial Y_k}y_j \\ &= -e_k f'(Y_k)y_j = -\delta_k y_j\end{aligned} \tag{3}$$

where,

$$\delta_k = e_k f'(Y_k) = (t_k - z_k) f'(Y_k)$$

The weight adjustment in the link between the hidden and output layers is computed by

$$\Delta w_{jk} = \alpha \times y_j \times \delta_k \tag{4}$$

where α is the learning rate, a positive constant between 0 and 1. The new weight herein can be updated by the following

$$w_{jk}(n+1) = w_{jk}(n) + \Delta w_{jk}(n) \tag{5}$$

where n is the number of iterations.

Similarly, the error gradient in links between input and hidden layers can be obtained by taking the partial derivative with respect to $w_{ij}$

$$\frac{\partial E}{\partial w_{ij}} = \left[\sum_{k=1}^{K}\frac{\partial E}{\partial z_k}\frac{\partial z_k}{\partial Y_k}\frac{\partial Y_k}{\partial y_j}\right].\frac{\partial y_j}{\partial X_j}.\frac{\partial X_j}{\partial w_{ij}} = -\Delta_j x_i \tag{6}$$

where the $\Delta_j$ can be derived as follows:

$$\Delta_j = f'(X_j)\sum_{k=1}^{K}\delta_k w_{jk} \tag{7}$$

The new weight in the hidden-input links can be now corrected as:

$$\Delta w_{ij} = \alpha \times x_i \times \Delta_j \tag{8}$$

and

$$w_{jk}(n+1) = w_{jk}(n) + \Delta w_{jk}(n) \tag{9}$$

Training the BP-networks with many samples is sometimes a time-consuming task. The learning speed can be improved by introducing the momentum term $\eta$. Usually, $\eta$ falls in the range [0, 1]. For the iteration n, the weight change $\Delta$ can be expressed as

$$\Delta w(n+1) = \eta \times \Delta w(n) + \alpha \times \frac{\partial E}{\partial w(n)} \tag{10}$$

The back-propagation learning algorithm used in artificial neural networks is shown in many textbooks (Schalkoff, (1997); Luger, (2002); Negnevitsky,(2002)).

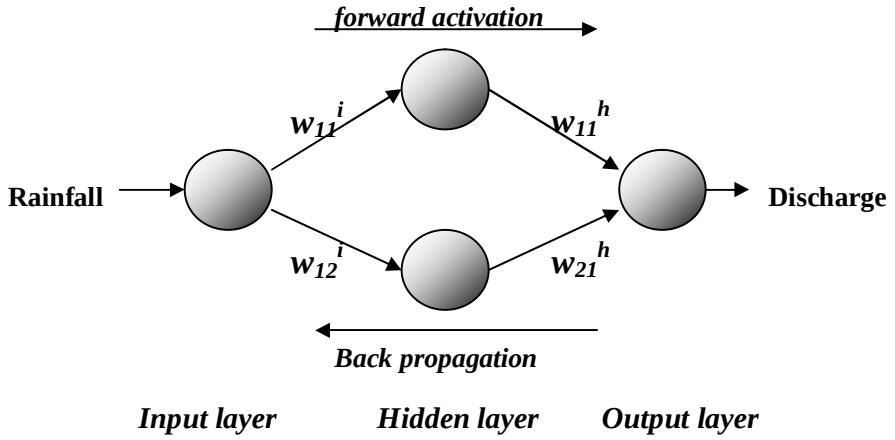


**Fig. 2** Architecture of BPNN

Among the many BPNN training methods available in the MATLAB Toolbox, the Levenberg–Marquardt training algorithm is selected because of its fast convergence rate (Demuth and Beale, (2003). During the backpropagation training process, the suitability of the simulation is validated by estimating the mean square error (MSE) between the observed and ANN-simulated data. In our study, the maximum number of epochs, target error goal MSE, and minimum performance gradient are set to 600, $10^{-5}$ and $10^{-4}$, respectively. Training stops when the maximum number of epochs is reached or when either the MSE or performance gradient is minimized to arrive at the pre-determined goal. Through a trial-and-error method, the optimal network structure for the current BPNN simulation is determined to include 1 hidden layer, with 1 hidden neuron1. The learning rate and momentum term is set to 0.4 and 0.56 respectively. Fig 2 gives the architecture of BPNN employed in this study.

**Radial Basis Function Neural Network (RBFNN)**

RBFNN is another type of neural network. Such networks have 3 layers, i.e., the input layer, the nonlinear hidden layer, and the linear output layer. RBFNN can overcome some of the limitations of

BPNN by using a rapid training phase, having a simple architecture, and maintaining complicated mapping abilities. In most cases, the RBFNN simulates phenomena of interest by using Gaussian basis functions in the hidden layer and linear transfer functions in the output layer. The major difference in operation between RBFNN and BPNN specifically exists in the hidden layer. Instead of the weighted sum of the input vector used in BPNN, the distance between the input and center, explained below, is employed in the RBFNN learning process. The first layer of the RBFNN collects the input data. Its training process determines the number of hidden neurons, *m*, which can be larger than that of BPNN to achieve a certain accuracy of prediction (Demuth and Beale (2003). For each neuron in the hidden layer, the distance between the input data and the center is activated by a nonlinear radial basis function, as shown in the following equation:

$$R_i = \exp\left[-\left(\|x_i - c_i\| b_1\right)^2\right] \tag{11}$$

Where, *x* is the input vector, and $b_1$ and $c_i$ are parameters that represent the bias in the hidden layer and center vector, respectively. Each neuron in the hidden layer will produce a value between 0 and 1, according to how close the input is to the center location. Therefore, neurons with centers closer to inputs will have more contributions to outputs; on the other hand, if neurons have centers away from inputs, then their outputs are nullified and so vanished. Later, the output layer neurons receive the weighted inputs and produce results by using a linear combination, which is of a similar form to that of the BPNN:

$$\hat{y} = \sum_{i-1}^{m} w_i R_i(x) + b_2 \tag{12}$$

where, $\hat{y}$ is the RBFNN simulation result (output), $w_i$ is the optimized connection weight determined through the training process and $b_2$ is the bias in the output layer.

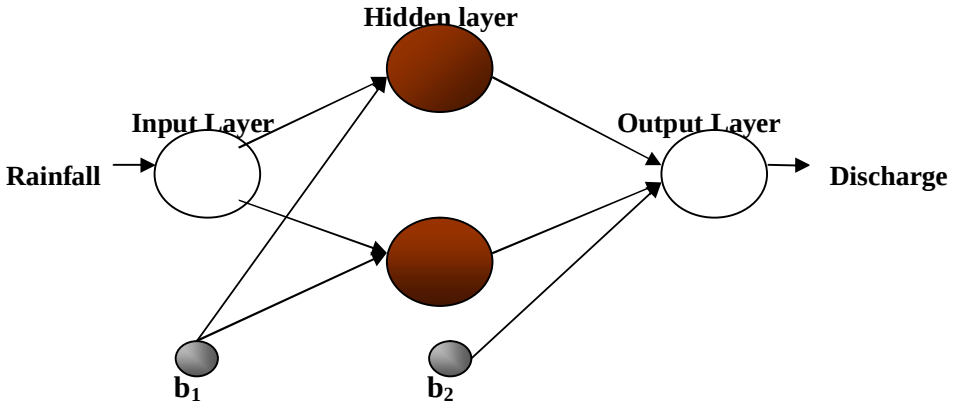


**Fig.3. Architecture of RBFNN**

Fig. 3 gives an insight into the structure and working procedure of the RBFNN. The open circles are bias neurons that adjust the sensitivity of the network. The biases are controlled by a specific value, called

"spread" number, and each bias is set to 0.8326/spread. The selected "spread" value should be large enough for neurons in the hidden layer to include the whole range of input data.$b_1$ and $b_2$ are the biases to the input and the output layer.

In RBFNN simulations, the proper initial choice of centers and weights should be regarded as key issues. Using the Neural Network Toolbox, we choose the weight vectors as the center parameter. Instead of being randomly generated, the initial value of the weights is set as the transpose of the input vector and newrb function is used to create radial basis network. Once the centers are developed, the weights linking the hidden and output layers should be updated during the training procedure. Training is an optimization procedure in which the network weights are adjusted in order to minimize the selected error value. The training procedure of the RBFNN (unlike the BPNN) determines the number of hidden neurons required for the simulation.

In this study, the root mean square error (RMSE) is the function used to estimate the performance of neural network:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{13}$$

Where, y is the target output value, $\hat{y}$ is the neural network output and n is the number of data patterns used. The training of RBFNN is initiated by a single neuron in the hidden layer, followed by continuously adding neurons to the hidden layer one at a time. Before training starts, the key training parameters should be assigned. In our study, the target error goal and the maximum number of neurons are set to 0.1 and 5, respectively. Training is stopped if the MSE is minimized to less than the pre-determined goal or if the maximum number of neurons is reached. Because the spread value affects the quality of prediction, a trial-and error method is employed, wherein several spread numbers are tested and the best one is accepted. By doing so, we find that the optimal spread number is 8.

## RESULTS AND DISCUSSIONS

All tests and results derived through programming in Matlab 6.5. By means of trial and error, an optimum network and parameter configuration for all networks was derived. The mean annual rainfall data for 1987-94 was used as input to the network and annual average discharge as the network output of 56 observation stations. Out of these, 408 datasets, 56 sets belong to year 1994 are chosen for network validation/testing and remaining for network calibration/training. Among 56 datasets, 30 are selected for the network validation based on the performance. Approximately 7% of data is used for testing, and the remaining 93% is used for network training. As all empirical models are only able to interpolate between the application boundary values, the training data should be representative of the entire range of conditions. Therefore, extreme values of the hydrological data need to be included as part of the training set. Prediction accuracy was evaluated using the coefficient of determination ($R^2$), the root mean squared

error (RMSE). $R^2$ measures the degree to which two variables are linearly related. RMSE provides information about the predictive capabilities of the model.

The performance efficiencies of the BPNN and RBFNN with one hidden layer (an architecture 1-1-1 represents 1 neuron each in input, hidden and output layer) is given in Table. 1 in terms of $R^2$ and RMSE. It is easily noticed that the both RBFNN and BPNN with network architecture 1-2-1 outperformed the other network architectures. However the RBFNN has the lowest RMSE and the highest $R^2$. It can be also seen that the training phase performance is better than those of testing phase performance

**Table .1 Performance Capabilities of Different Network Architectures**

| NN Type | NN Architecture | Prediction Performance of NN model | | | |
|---|---|---|---|---|---|
| | | Training | | Validation | |
| | | $R^2$ | RMSE | $R^2$ | RMSE |
| BPNN | 1/1/2001 | 0.9532 | 0.0753 | 0.9293 | 0.3574 |
| BPNN | 1/2/2001 | 0.9753 | 0.0182 | 0.9695 | 0.2647 |
| BPNN | 1/3/2001 | 0.9284 | 0.1753 | 0.8932 | 0.5392 |
| RBFNN | 1/1/2001 | 0.9482 | 0.0643 | 0.9176 | 0.2593 |
| RBFNN | 1/2/2001 | 0.9976 | 0.0146 | 0.9755 | 0.0989 |
| RBFNN | 1/3/2001 | 0.9342 | 0.2967 | 0.9031 | 0.1984 |

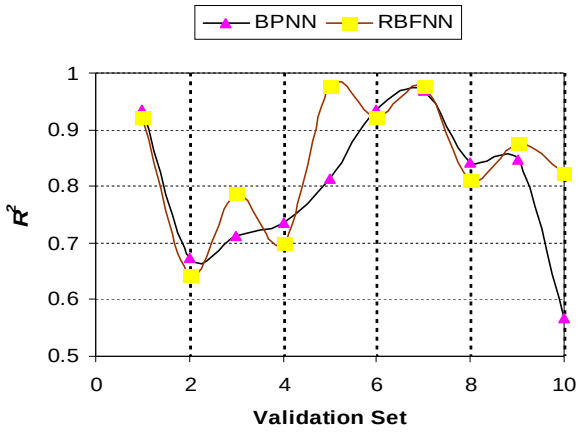


**Fig.4 Performance of Different Validation Sets**

Fig.4 shows the performance of the different validation data sets each consisting of 30 out of 56. There are 10 sets randomly selected from the 56 sets belonging to the year 1994. In most of the datasets the $R^2$ for RBFNN is higher than the BPNN except for dataset 8. However some of the datasets like 1, 6 and 7 show nearly equal predicting capacity. But dataset 7 has highest $R^2$ for the both the networks. So, it has been selected as the set for testing the set.

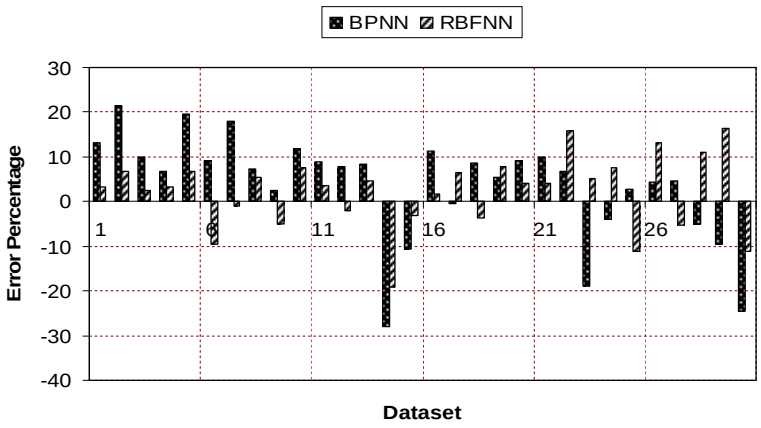


**Fig.5 Error Percentage Occurring in Each Observation Stations**

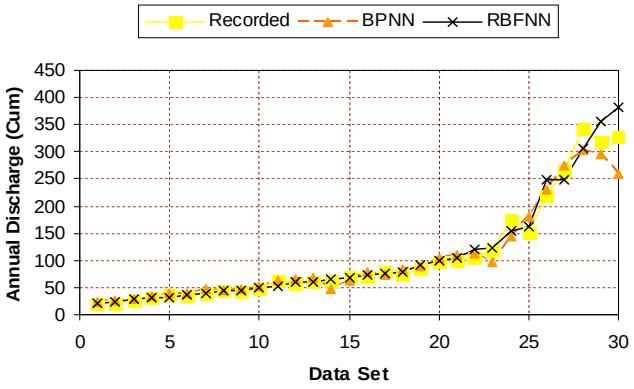


**Fig.6. Comparison of Recorded and Predicted Discharges for 30 Observation Stations**

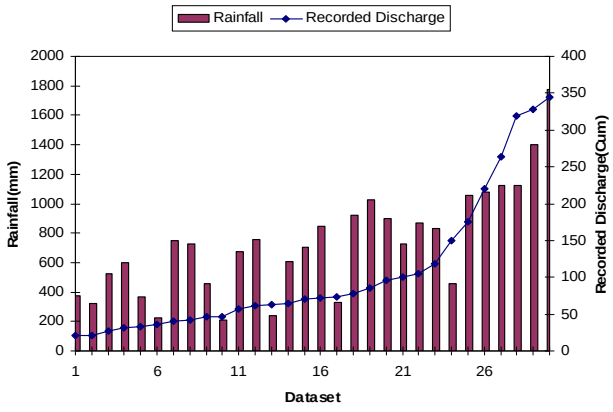


**Fig.7. Comparison of Rainfall and Recorded Discharges for 30 Observation Stations**

The error percentage occurring in each observation stations is shown in fig. 5. in 14$^{th}$ station the percentage of error is highest for both the networks near about 29% for BPNN and 18% for RBFNN. It is

due to the fact the discharge is unexceptionally higher than the usual ones in the surrounding stations due very less amount of soil absorption. Fig.6 gives an idea by how much the predicted and recorded discharges are varying. One can easily see that for lower discharges the predicted values are very close to each other but in case of higher discharges there is a greater deviation. Figure.7, 8 and 9 gives the rainfall and annual average discharges.

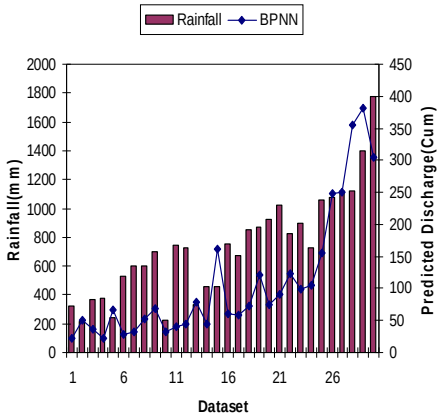


**Fig.8. Comparison of Rainfall and Predicted Discharges for 30 Observation Stations**

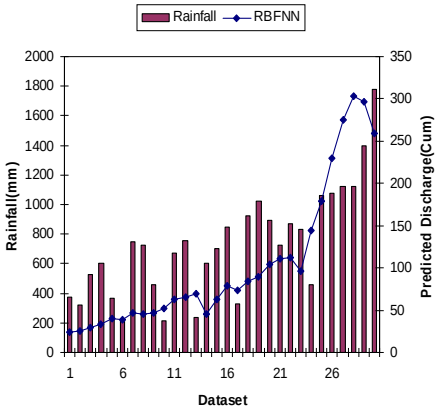


**Fig.9. Comparison of Rainfall and Predicted Discharges for 30 Observation Stations**

Fig 10 and 11 gives the scatter plot of predicted and recorded discharge, $R^2$ and the best fit linear plot. $R^2$ is 0.9695 and 0.9755 for BPNN and RBFNN respectively.

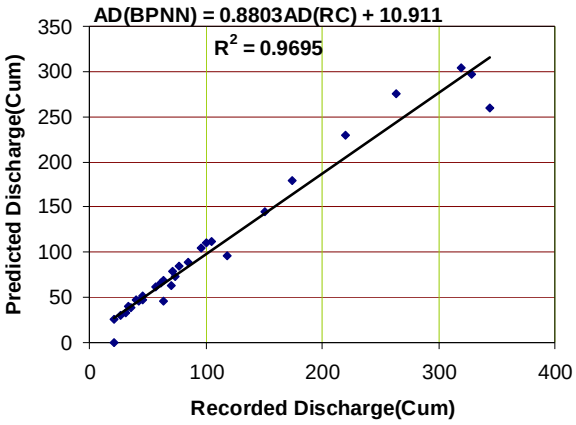


**Fig.10 Scatter Plot of Predicted Discharge by BPNN and Recorded Discharge**

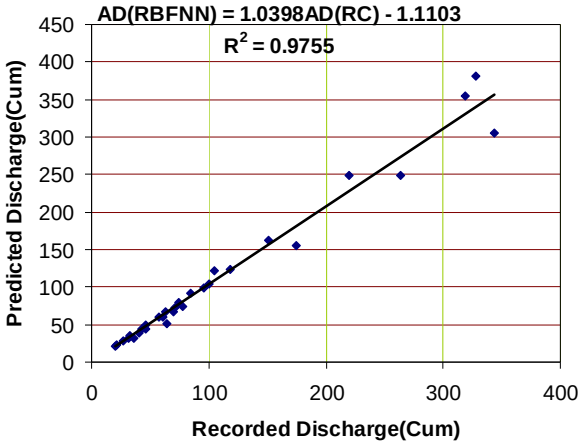


**Fig.11 Scatter Plot of Predicted Discharge by RBFNN and Recorded Discharge**

## CONCLUSIONS

By comparing the simulations of the RBFNN model and the BPNN model with the discharge records, it is found that the performance of the RBFNN model is better than that of the BPNN model with the same input data for all studied sites. The meteorological variables have non-linear relationships with rainfall and discharge. Data selected for model training must be representative of the watershed or flow conditions and the training process should be repeated periodically to reflect the changing hydrological characteristics of the area under investigation. ANN in hydrological applications can handle processes poorly define or not well understood; previous knowledge of processes in the training not needed; no specific solution are available; weights are readily adjusted as new data are available; small errors are not amplified since the processing is distributed; relevant information is stored so no need to keep previous processed data records; and no need for additional input and output data except those specified in the training. The ANN can be effectively applied to model nonlinear systems such as the transformation of rainfall into runoff without explicitly specifying hydrological process of the watershed, but it is not a substitute for conceptual models, based on physical process. It can be a feasible alternative for hydrological forecasting when the runoff is predicted from input and output time series. Most importantly, this paper presents indications that neural networks can also be applied in cases where the datasets manifest trends and shifts and the desired output has lies outside of the range of previously introduced input, as shown by the results. Further study needs to take in more factors which may not be statistically associated with dust storm occurrence (because of the absence of a linear relationship), but have obvious physical meaning to improve the prediction accuracy.

## REFERENCES

1. Anmala, J., Zhang, B., Govindaruja, R.S., (2000). Comparison of ANNs and empirical approaches for predicting watershed runoff. J. Water Resour. Planning and Management, 126 (3), 156–166.

2. ASCE Task Commitee,(2000). Artificial neural networks in hydrology. II: hydrologic applications. J. Hydrol. Engng ASCE , 5(2**)** , 124–137.

3. Bertoni, J. C., Tucci, C. E., and Clarke, R. T. (1992). "Rainfall-based real-time flood forecasting." J. Hydrol., 131, 313–339.

4. Brikundavyi, S., Labib, R., Trung, H.T., Rousselle, J., (2002).Performance of neural networks in daily streamflow forecasting, J. Hydrol. Engng ASCE ,**7 (5)**, 392–398.

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

6. Coulibaly, P., Anctil, F., Bobee, B., (2000). Daily reservoir inflow forecasting using artificial neural networks with stopped training approach. J. Hydrol, 3(4), 244–257.

7. H. Demuth and M. Beale, (2003) ,Neural Network Toolbox: For Use with Matlab. Mathworks, Inc., Natick, MA.

8. Lorrai, M., Sechi, G.M., (1995). Neural nets for modeling rainfall runoff transformations. Water Resour. Management, 9, 299–313.

9. Luger, George F., (2002). Artificial Intelligence–Structures and Strategies for Complex Problem Solving. 4th Ed. AddisonWesley, USA.

10. Maier, H.R., Dandy, G.C.,( 2000). Neural network for the prediction and forecasting of water resources variables: a review of modeling issues and applications. Environ Modeling and Software, 15, 101–124.

11. Negnevitsky, Michael,( 2002). Artificial Intelligence: A Guide to Intelligent Systems. Addison Wesley, Harlow, England.

12. Schalkoff, Robert J., (1997). Artificial Neural Networks. International Editions. McGraw-Hill.

13. Thirumalaiah, K., Deo, M.C., (1998). River stage forecasting using artificial neural networks. J. Hydrol. Engng ASCE ,3(1), 26–32.

14. Thirumalaiah, K., Deo, M.C.,(2000). Hydrological forecasting using artificial neural networks. J. Hydrol. Engng ASCE ,**5(2)**, 180–189.

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

16. Xu, C.-Y., Seibert, J., Halldin, S., (1996), Regional water balance modeling in the NOPEX area: development and application of monthly water balance model. J. Hydrol. 180, 211–236.

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