International Journal of Technical Innovation in Modern Engineering & Science (IJTIMES)
Impact Factor: 3.45 (SJIF-2015), e-ISSN: 2455-2585
Volume 4, Issue 01, January-2018

ANN model development for Rainfall-Runoff Modelling.

Lateef Ahmad Dar

National Institute of Technology, Srinagar, India

ABSTRACT :- The Artificial Neural Network (ANN) approach has been successfully used in many hydrological studies especially the rainfall-runoff modelling. This study deals with a detailed description of development of Artificial Neural Network models. The various aspects of the model development and the processes undergoing to develop those models are discussed in this study.

KEYWORDS: Artificial Neural Networks (ANNs); Rainfall-Runoff Process

Introduction

Rainfall-Runoff process is a very complex phenomenon. The calculation of Runoff generated over a catchment in response to Rainfall is a very hard task and there are various parameters involved. Over the years researchers have developed many models to simulate this process. Based on the problem statement and on the complexities involved, these models are categorized as empirical, black-box, conceptual or physically-based distributed models. Physically based distributed models are very complex and required too many data and tedious for the application purpose. The conceptual models attempt to represent the known physical process occurring in the rainfall-runoff transformation in a simplified manner by way of linear or nonlinear mathematical formulations but their implementation and calibration is complicated and time consuming. While black-box models, which establish a relationship between input and the output functions without considering the complex physical laws governing the natural process such as rainfall-runoff transformation. ANN is one such black box models and is able to model and simulate the hydrological processed with impressive performance. This study describes the various aspects of ANN model development right from data preparation, data optimization to model optimization.

Study Area and Data Used

The study area spatially lies between 33° 21′ 54″ N to 34° 27′ 52″ N latitude and 74° 24′ 08″ E to 75° 35′ 36″ E longitude with a total area of 8600.78 sq.kms (Fig.1). It covers almost all the physiographic divisions of the Kashmir Valley and is drained by the most important tributaries of river Jhelum. Srinagar city which is the largest urban centre in the valley is settled on both the sides of Jhelum River and is experiencing a fast spatial growth. Physical features of contrasting nature can be observed in the study area that ranges from fertile valley floor to snow-clad mountains and from glacial barren lands to lush green forests.

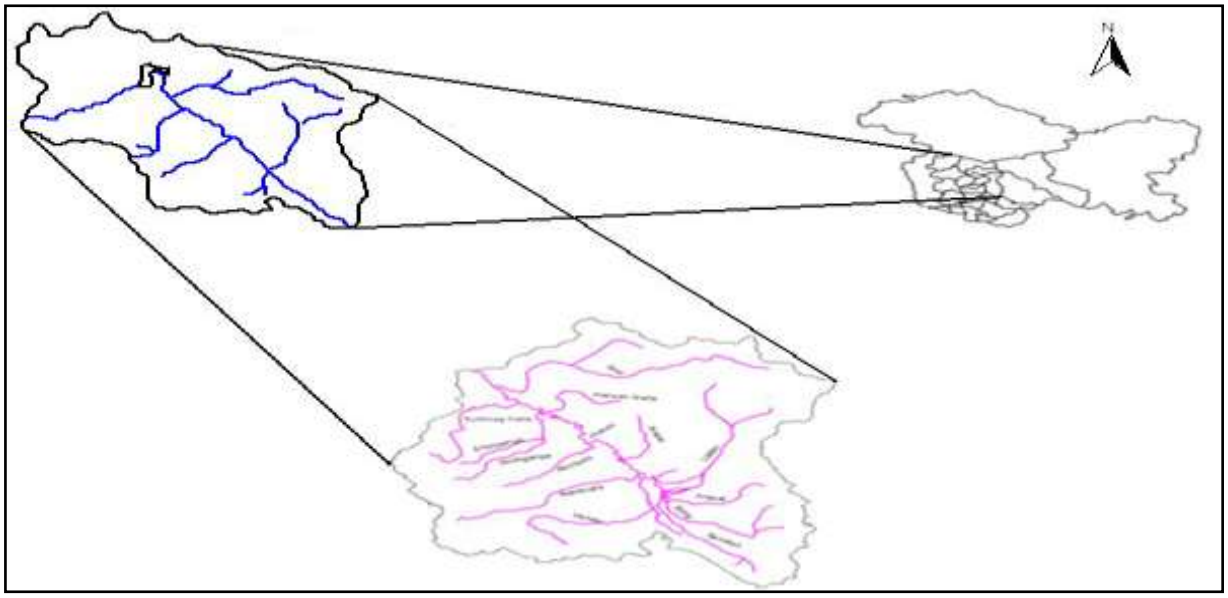


Fig.1 Location Map of the Study Area *(Source: Generated from SOI toposheets, 1961)*

Input Data Preparation

The rainfall data was available at three stations in the catchment. There is always spatial variation in precipitation hence the rainfall at these three stations needs to be normalized over the whole catchment so it can be fed to the neural network. A true areal representation of the rainfall is determined by drawing Thessian polygons and calculating the average rainfall over the whole catchment.

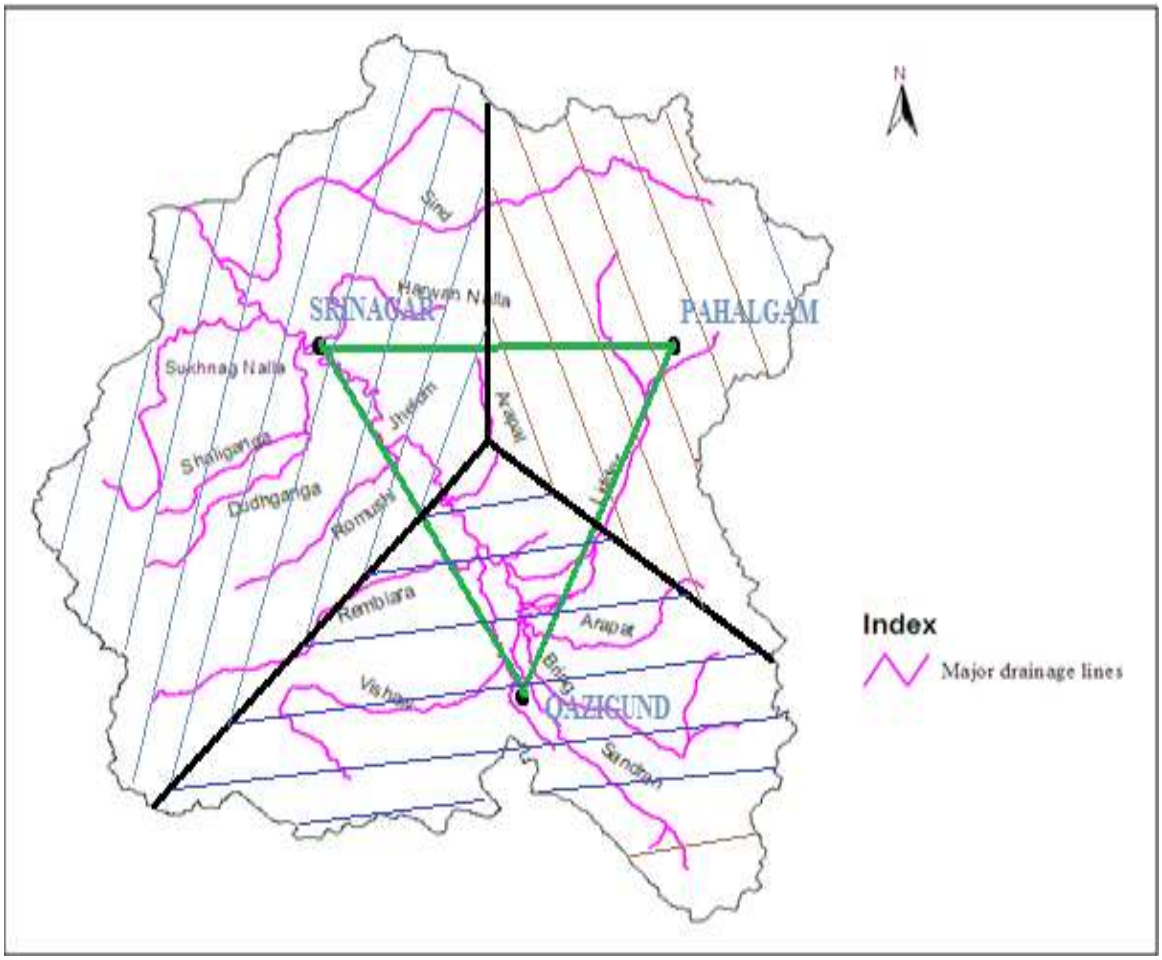


Fig.2 Thessian polygons for various stations

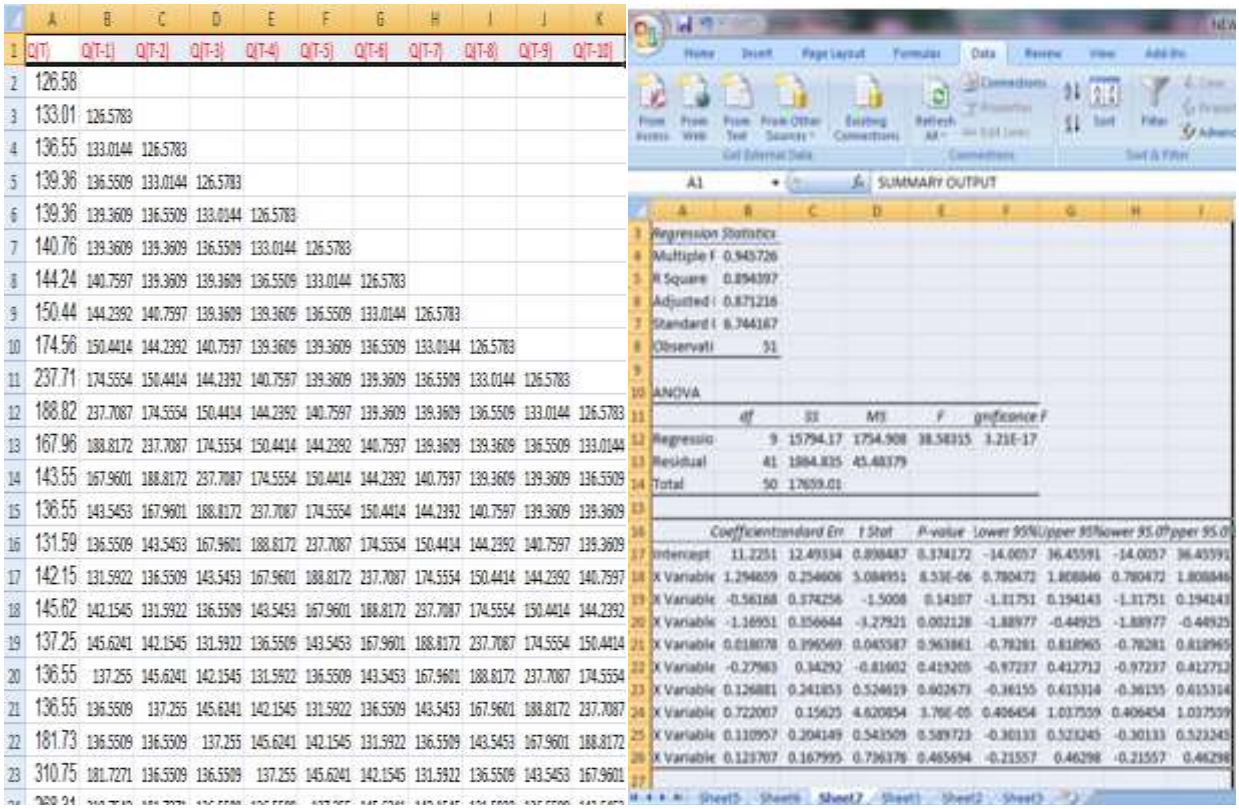
Table 1: Calculated Thessian weights or various stations selected in the catchment.

STATION NAME	THESSIAN WEIGHT (%)
PAHALGAM	23.60
SRINAGAR	41.90
QAZIGUND	34.50

The influencing stations and their corresponding Thessian weights are presented in the table 1. Using the above weights, the corresponding daily rainfall data of the stations, the average areal precipitation of the catchment has been estimated.

Identification Of The Input Vector

Identification of the no. of flow series was carried by the predicative analysis. The selection of the predictors was carried out on the basis of p-test.



Q(T)	Q(T-1)	Q(T-2)	Q(T-3)	Q(T-4)	Q(T-5)	Q(T-6)	Q(T-7)	Q(T-8)	Q(T-9)	Q(T-10)
126.58										
133.01	126.5783									
136.55	133.0144	126.5783								
139.36	136.5509	133.0144	126.5783							
139.36	139.3609	136.5509	133.0144	126.5783						
140.76	139.3609	139.3609	136.5509	133.0144	126.5783					
144.24	140.7597	139.3609	139.3609	136.5509	133.0144	126.5783				
150.44	144.2392	140.7597	139.3609	139.3609	136.5509	133.0144	126.5783			
174.56	150.4414	144.2392	140.7597	139.3609	139.3609	136.5509	133.0144	126.5783		
237.71	174.5554	150.4414	144.2392	140.7597	139.3609	139.3609	136.5509	133.0144	126.5783	
188.82	237.7087	174.5554	150.4414	144.2392	140.7597	139.3609	139.3609	136.5509	133.0144	126.5783
167.96	188.8172	237.7087	174.5554	150.4414	144.2392	140.7597	139.3609	139.3609	136.5509	133.0144
143.55	167.9601	188.8172	237.7087	174.5554	150.4414	144.2392	140.7597	139.3609	139.3609	136.5509
136.55	143.5453	167.9601	188.8172	237.7087	174.5554	150.4414	144.2392	140.7597	139.3609	139.3609
131.59	136.5509	143.5453	167.9601	188.8172	237.7087	174.5554	150.4414	144.2392	140.7597	139.3609
142.15	131.5922	136.5509	143.5453	167.9601	188.8172	237.7087	174.5554	150.4414	144.2392	140.7597
145.62	142.1545	131.5922	136.5509	143.5453	167.9601	188.8172	237.7087	174.5554	150.4414	144.2392
137.25	145.6241	142.1545	131.5922	136.5509	143.5453	167.9601	188.8172	237.7087	174.5554	150.4414
136.55	137.255	145.6241	142.1545	131.5922	136.5509	143.5453	167.9601	188.8172	237.7087	174.5554
136.55	136.5509	137.255	145.6241	142.1545	131.5922	136.5509	143.5453	167.9601	188.8172	237.7087
181.73	136.5509	136.5509	137.255	145.6241	142.1545	131.5922	136.5509	143.5453	167.9601	188.8172
310.75	181.7271	136.5509	136.5509	137.255	145.6241	142.1545	131.5922	136.5509	143.5453	167.9601

SUMMARY OUTPUT

Regression Statistics	
Multiple R	0.945726
R Square	0.894397
Adjusted R	0.871216
Standard E	6.744167
Observati	51

ANOVA

	df	SS	MS	F	gnificance F
Regressio	9	15794.17	1754.908	38.58315	3.21E-17
Residual	41	1864.835	45.48379		
Total	50	17659.01			

	Coefficients	tandard Er	t Stat	P-value	Lower 95%	pper 95%	ower 95.0%	pper 95.0
Intercept	11.2251	12.49334	0.898487	0.374172	-14.0057	36.45591	-14.0057	36.45591
X Variable	1.294659	0.254606	5.084951	8.53E-06	0.780472	1.808846	0.780472	1.808846
X Variable	-0.56168	0.374256	-1.5008	0.14107	-1.31751	0.194143	-1.31751	0.194143
X Variable	-1.16951	0.356644	-3.27921	0.002128	-1.88977	-0.44925	-1.88977	-0.44925
X Variable	0.018078	0.396569	0.045587	0.963861	-0.78281	0.818965	-0.78281	0.818965
X Variable	-0.27983	0.34292	-0.81602	0.419203	-0.97237	0.412712	-0.97237	0.412712
X Variable	0.126881	0.241853	0.524619	0.602673	-0.36155	0.615314	-0.36155	0.615314
X Variable	0.722007	0.15625	4.620854	3.76E-05	0.406454	1.037559	0.406454	1.037559
X Variable	0.110957	0.204149	0.543509	0.589723	-0.30133	0.523245	-0.30133	0.523245
X Variable	0.123707	0.167995	0.736376	0.465694	-0.21557	0.46298	-0.21557	0.46298

Fig.3: Predicative analysis for the number of input flow series.

The predicative analysis suggests incorporating flow values with three days lag in the input vector to the network.

Identifying the Number Of Rainfall Patterns In The Input Vector

The number of previous day's rainfall which influences the flow rate to be predicted was determined in the trial and error manner. The procedure that was used to identify the number of rainfall patterns as input to the network is summarized below.

A sample model is selected by representing stream flow at the present time't' as a function of precipitation at (t-1) and stream flow at t-1,t-2,t-3 .The model can then be represented as

Q(t) =f [P(t-1),Q(t-1),Q(t-2),Q(t-3)]

Various ANN configurations were trained and tested using the model the no of neurons in the hidden layer in BPN were varied from 1 to 20 during training. Among the models the models are selected which give better results. The precipitation at time (t-2) is added as an additional input parameter to the above model and the network is trained again, hence the model becomes

Q(t) =f [P(t-1),P(t-2),Q(t-1),Q(t-2),Q(t-3)]

If the goodness of fit statistics for the present model are significantly different from the previous model, then the precipitation at time (t-3) is added as an as another input parameter and we go on adding the input precipitation parameters till time (t-5).

Model Development

In the present study two types of ANN's viz. back propagation network (BPN) and radial based function network (RBF) were developed. These models were developed in MATLAB environment using ANN tool-pack.

Back Propagation Network

Back propagation is the most widely used of the neural networks and has been applied successfully in the application studies in a wide range of areas.

The back propagation algorithm involves a forward propagating step followed by a back propagating step. Both the forward and back propagation steps are done for each pattern presentation during training. The forward propagation step begins with the presentation of an input pattern to the input layer of the network, and continues as activation level calculation (activation level parameter associated with each processing unit is its output value) propagate forward through the hidden layers in each successive layer, every processing unit sums its inputs and then applies a transfer function to compute its output. The output layer of units then produces the output of the network. The back-propagation step begins with the comparison of the network's output pattern to the target vector, when the difference of error is calculated. The back propagation step then calculates error values for hidden units and changes for their incoming weights , starting with the output layer and moving backward through the successive hidden layers. In this back

propagation step, the network corrects its weight in such a way as to decrease the observed error. A general structure of the BPN is depicted in the figure. For getting the best network architecture we have to optimise the neural network i.e., we have to optimize the neural network for the number of hidden layers and the number of neurons in the hidden layers.

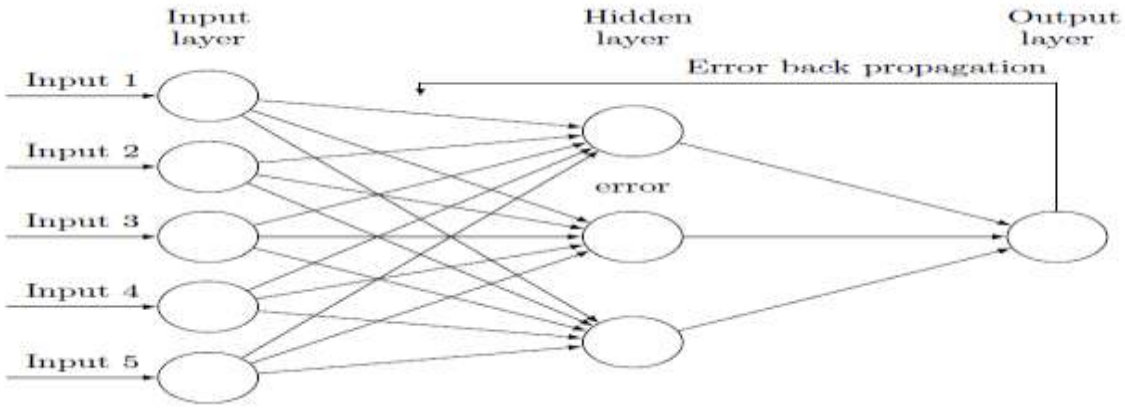


Fig.4: Network architecture of a BPN network.

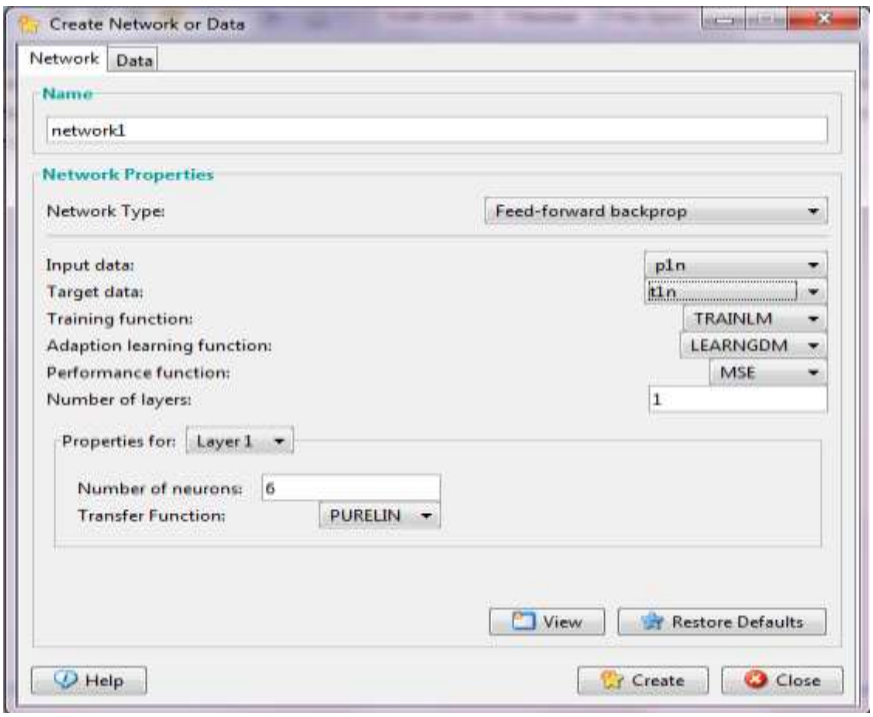


Fig.5: View of dialogue box for designing the network.

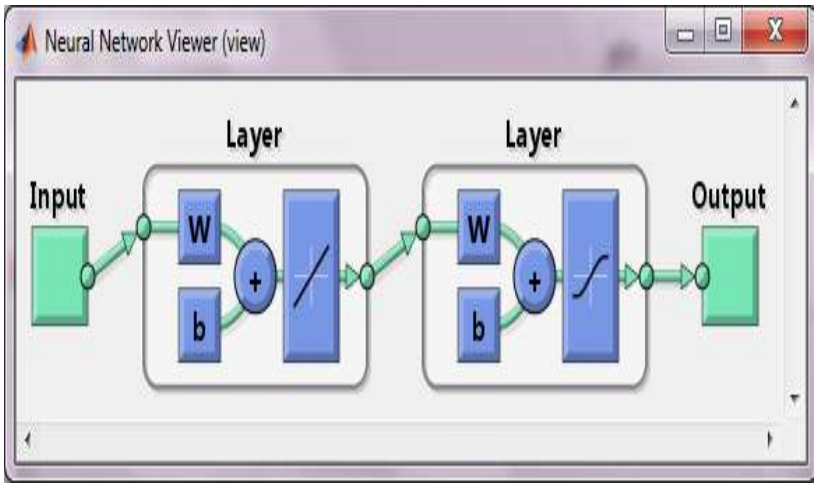


Fig.6: View of the network generated

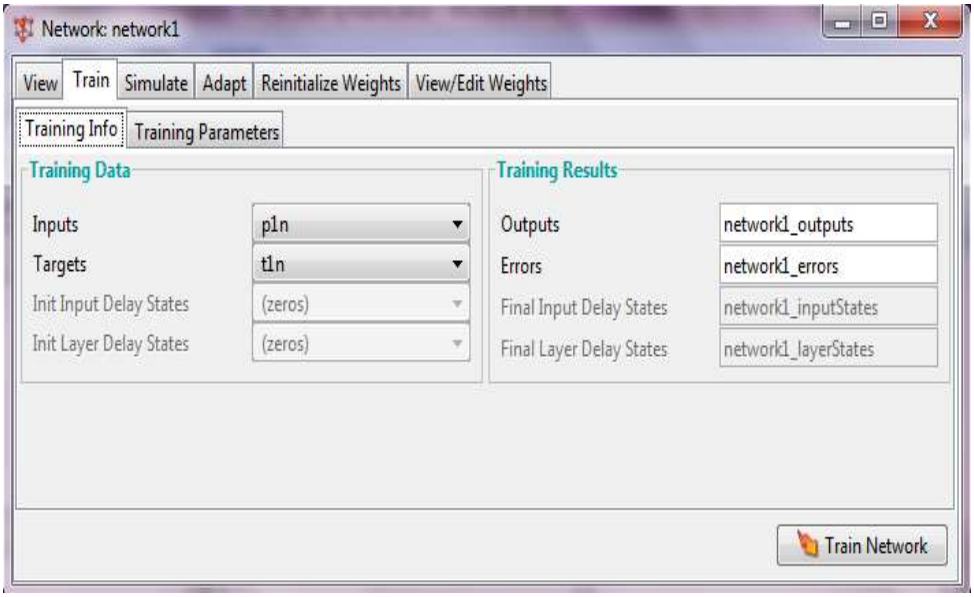


Fig.7: Dialogue box generated for training the network.

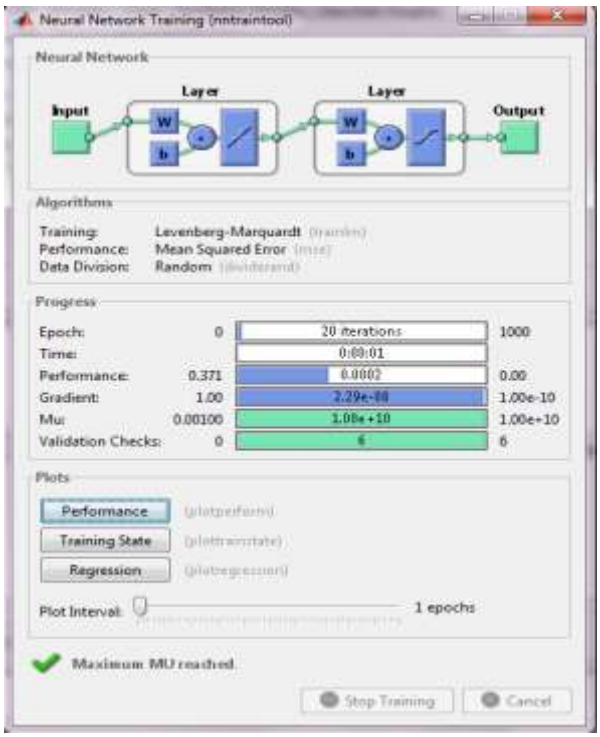


Fig.8: Dialogue box determining the performance during training.

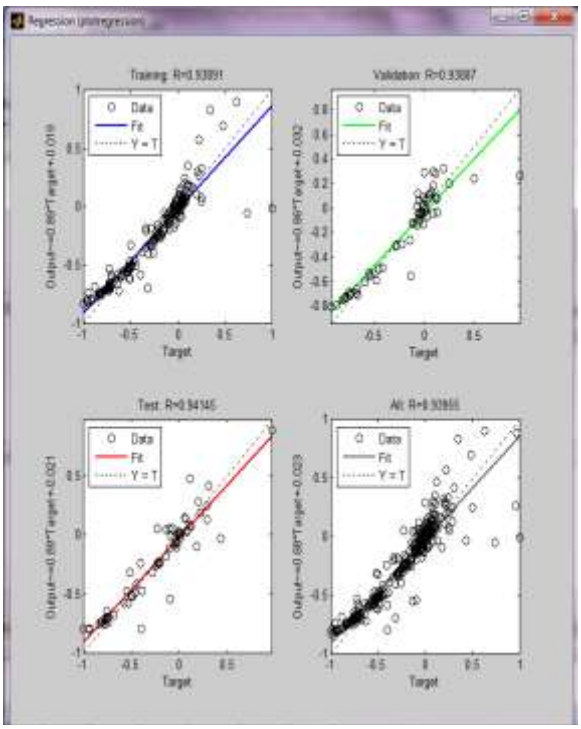


Fig.9:Dialogue box showing performance during training

Radial Basis Function (RBF) network

The Radial Basis Function (RBF) network is a variant of the standard feed-forward network. It can be considered as a two-layer feed-forward network in which the hidden layer performs a fixed non-linear transformation with no adjustable internal parameters. The output layer, which contains the only adjustable weights in the network, then linearly combines the outputs of the hidden neurons [after Chen et al., 1991]. The RBF network is trained by determining the connection weights between the hidden and output layer through a performance training algorithm. The hidden layer consists of a number of neurons and internal parameter vectors called 'centers', which can be considered the weight vectors of the hidden neurons. A neuron (and thus a centre) is added to the network for each training sample presented to the network.

The input for each neuron in this layer is equal to the Euclidean distance between an input vector and its weight vector (centre), multiplied by the neuron bias. The transfer function of the radial basis neurons typically has a Gaussian shape.

This means that if the vector distance between input and centre decreases, the neuron's output increases (with a maximum of 1). In contrast, radial basis neurons with weight vectors that are quite different from the input vector have outputs near zero. These small outputs only have a negligible effect on the linear output neurons.

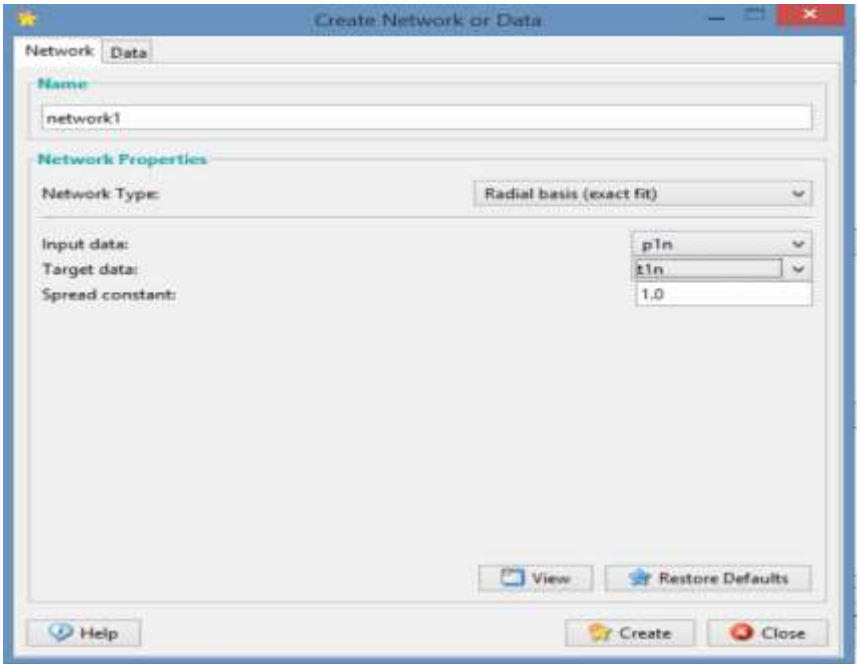


Fig.10:View of dialogue box for designing the RBF network .

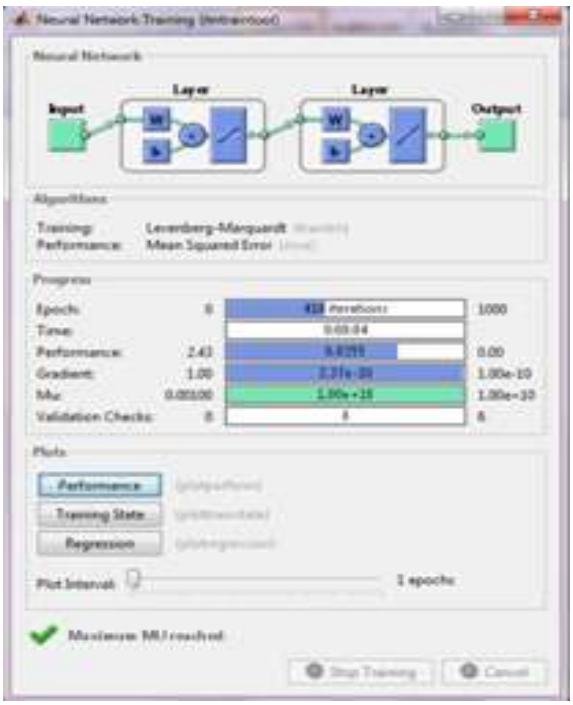


Fig.11: Dialogue box determining the performance during training.

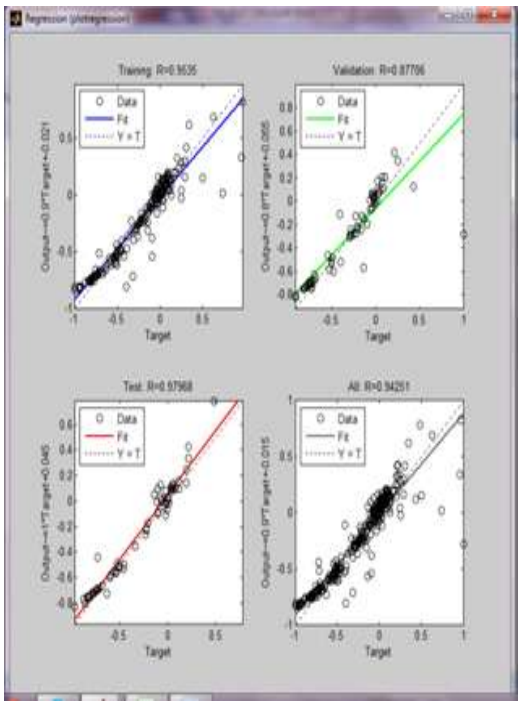


Fig.12: Dialogue box showing the training performance of RBF.

If a neuron has an output of 1 the weight values between the hidden and output layer are passed to the linear output neurons. In fact, if only one radial basis neuron had an output of 1, and all others had outputs of 0's (or very close to 0), the output of the linear output layer would be the weights between the active neuron and the output layer. This would, however, be an extreme case. Typically several neurons are always firing, to varying degrees.

Summarizing, a RBF network determines the likeness between an input vector and the network's centers. It consequently produces an output based on a combination of activated neurons (i.e. centers that show a likeness) and the weights between these hidden neurons and the output layer. The primary difference between the RBF network and back-propagation lies in the nature of the nonlinearities associated with hidden neurons. The nonlinearity in back-propagation

is implemented by a fixed function such as a sigmoid. The RBF method, on the other hand, bases its nonlinearities on the data in the training set [after Govindaraju, 2000]. The original RBF method requires that there be as many RBF centers (neurons) as training data points, which is rarely practical, since the number of data points is usually very large [after Chen et al., 1991]. A solution to this problem is to monitor the total network error while presenting training data (adding neurons), and to stop this procedure when the error does no longer significantly decrease. RBF networks are generally capable of reaching the same performance as feed-forward networks while learning faster. On the downside, more data is required to reach the same accuracy as feed-forward networks. According to Chen, Cowan and Grant [1991], RBF network performance critically depends on the centres that result from the inputted training data. In practice, these training data are often chosen to be a subset of the total data, which suitably samples the input domain.

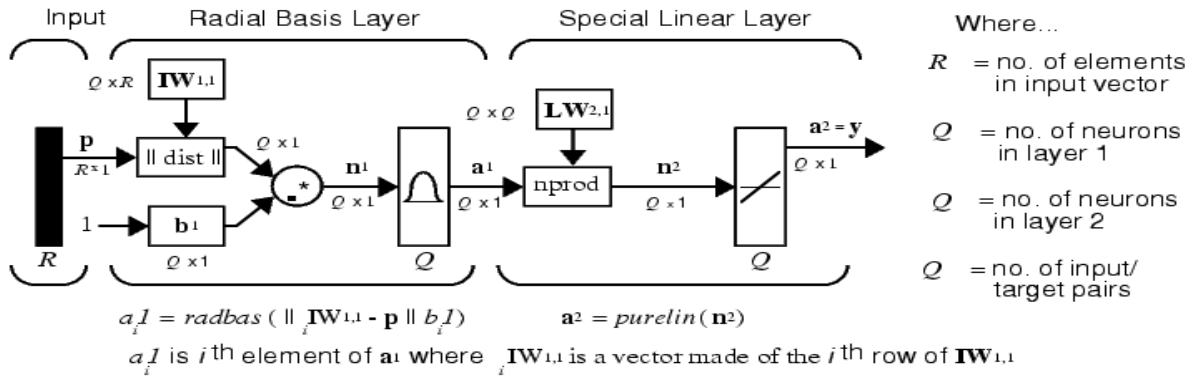


Fig.13: Network architecture of an RBF network.

Post Training Analysis

The performance of a trained network can be measured to some extent by the errors on the training, validation and test sets, but it is often useful to investigate the network response in more detail. One option is to perform a regression analysis between the network response and the corresponding targets. The "postreg" is designed to perform this analysis.

The following commands were used to perform a regression analysis on the network that was previously trained:

a = sim(net,p);

[m,b,r] = postreg(a,t)

Here the network output and the corresponding targets are passed to postreg. It returns three parameters. The first two, m and b, correspond to the slope and the y-intercept of the best linear regression relating targets to network outputs. For a perfect fit (outputs exactly equal to targets), the slope would be 1, and the y-intercept would be 0. The third variable returned by postreg is the correlation coefficient (R-value) between the outputs and targets. It is a measure of how well the variation in the output is explained by the targets. If this number is equal to 1, then there is perfect correlation between targets and outputs.

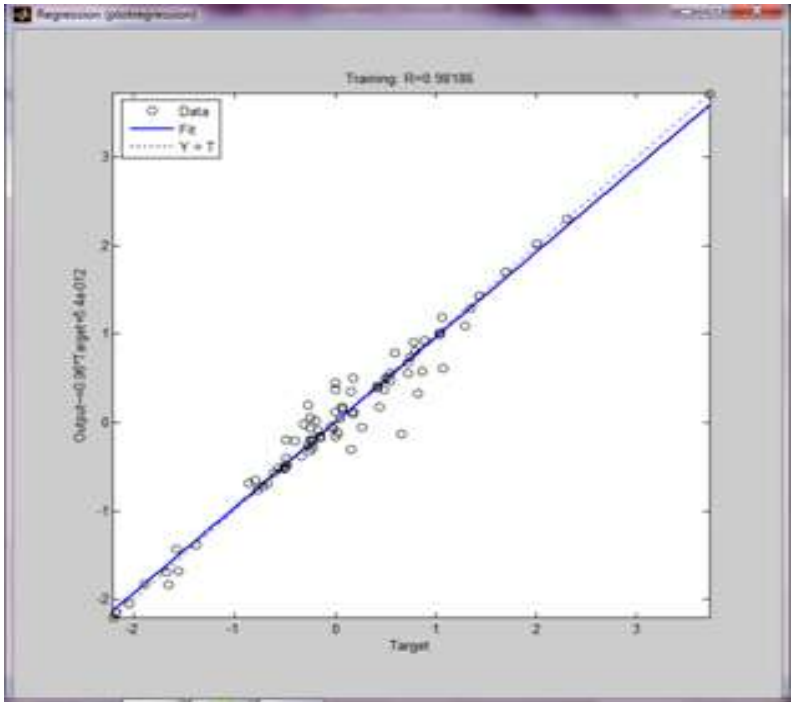


Fig.14: Dialogue box showing the training performance of RBF.

In order to convert the normalized outputs back into the same units that were used for the original targets, postmnmx was used. The following commands were used to simulate the network that was trained in the previous code, and then convert the network output back into the original units.

```
on = sim(net,pn);
o = postmnmx(on,mint,maxt);
```

The network output of ANN will correspond to the normalized targets tn. The un-normalized network output 'o' is in the same units as the original targets t. The unormalised out-put 'o' can then be compared with the observed output by various statistical indices.

Conclusion.

This paper presents a detailed explanation of various aspects of ANN model development rainfall-runoff process. These models are very easy and less time consuming as compared to physical or conceptual models and hence are more likely to be used in future and one needs to optimize the models for better performance by repeatedly changing the network structure till the model yields maximum accuracy.

References

1. ASCE Task Committee on Application of Artificial Neural Networks in Hydrology (2000a) Artificial neural networks in hydrology. I: preliminary concepts. *J. Hydrol. Eng. ASCE* **5(2),** 115-123.
2. ASCE Task Committee on Application of Artificial Neural Networks in Hydrology (2000b) Artificial neural networks in hydrology. II: hydrologic applications. *J. Hydrol. Eng. ASCE* **5(2),** 124-137.
3. Dawson, C.W. & Wilby, R.L. (1998) An artificial neural network approach to rainfall-runoff modeling. *Hydrol. Sci. J.* **43(1),** 47-65.
4. Dawson, C.W. & Wilby, R.L. (2001) Hydrological modelling using artificial neural networks. *Prog. Phys. Geog.* **25(1),** 80-108.
5. Garson, G.D. (1991) Interpreting neuralnetwork connection weights. *Artificial Intell. Expert.* **6**, 47-51.
6. Giustolisi, O. & Laucelli, D. (2005) Improving generalization of artificial neural networks in rainfall-runoff modelling. *Hydrol. Sci. J.* **50(3),** 439-457.
7. Hsu, K.L., Gupta, H.V. & Sorooshian, S. (1995) Artificial neural network modeling of the rainfall-runoff process. *Wat. Resour. Res.* **31(10),** 2517-2530.
8. Jain, A. & Srinivasulu, S. (2006) Integrated approach to model decomposed flow hydrograph using artificial neural network and conceptual techniques. *J. Hydrol.* **317**, 291-306.