International Journal of Advanced Information Science and Technology (IJAIST) ISSN: 2319:2682
Vol.5, No.9, September 2016 DOI:10.15693/ijaist/2016.v5i9.1-5

Performance Evaluation of Multiple ANN models in Flow Simulations

Shyama Debbarma
Research Scholar/Department of CE
NIT Silchar, India

Dr. Parthajit Roy
Asstt. Professor/Department of CE
NIT Silchar, India

***Abstract*— Neural networks have been used successfully to a wide range of areas in different forms. In water resources management aspect too, neural networks have been applied extensively. Neural networks are used mainly in two different ways in solving static and dynamic problems. This paper investigates the ability of both static and dynamic ANN models in six different forms for modeling daily mean discharge of Dholai(Rukni) river, Assam, India. Results indicated that Gamma memory neural network(GMNN) model outperformed other ANN models while Tanh transfer function and Levenberg-Marquardt learning rule is employed. GMNN model also performed best in simulating major flood events of the test data series dominating other chosen ANN models. Hence, GMNN model can be used as a better alternative technique for simulating and prediction of stream flows.**

***Index terms* - ANN, stream flow, simulation, static, dynamic.**

I. INTRODUCTION

The severity of major floods contributes significantly toward the investigation of different flow modeling techniques and hence increases their prediction accuracy. Flood modeling follows largely two major modeling approaches: conceptual (phenomenological) modeling, which retains some of the physical laws in their mathematical formulation, and black-box modeling, which relies heavily on an input-output description of the conceptual models. The large amount of data required for the conceptual models, along with the costs of collecting the data made the black-box models more attractive to hydrologists. Black-box models include linear and non-linear statistical method and artificial neural networks. Artificial neural networks proved to be better than other linear and nonlinear models due to its ability to capture the temporal features of time series problem. Adoption of memory based neural networks such as gamma, time delay and laguarre memory made it more popular than the other static ANN models. In fact, these memory based neural networks have been applied successfully in various fields [1-6]. The intent here is to focus on the contribution of ANNs in water resources management aspect. Moreover, improvement in the ability of ANN modeling techniques in flood prediction could help those affected by the flood. The present study aims to analyze the performances of six different static and dynamic neural network models in simulating daily mean discharge of Dholai(Rukni) river located in Assam, India. Dholai river is tributary of Barak river basin. The six

II. METHODOLOGY

Six ANN models are chosen in the present study which includes both static and dynamic ANN models. Three different models are created from TLRN using gamma, TDNN and laguarre memory. ANN models are discussed as follows:

a. Multilayer perceptrin(MLP)

Multilayer perceptrons (MLPs) are layered feed-forward networks typically trained with static backpropagation. This network has found countless applications requiring static pattern classification. The article by [7] is probably one of the best references for the computational capabilities of MLPs. Generally speaking, for static pattern classification, the MLP with two hidden layers is a universal pattern classifier. In other words, the discriminant functions can take any shape, as required by the input data clusters. Moreover, when the weights are properly normalized and the output classes are normalized to 0/1, the MLP achieves the performance of the maximum a posteriori receiver, which is optimal from a classification point of view [8]. In terms of mapping abilities, the MLP is believed to be capable of approximating arbitrary functions. This has been important in the study of nonlinear dynamics [9], and other function mapping problems. MLPs are normally trained with the back-propagation algorithm [10]. In fact the renewed interest in ANNs was in part triggered by the existence of back-propagation. The LMS learning algorithm proposed by Widrow can't be extended to hidden PEs, since we do not know the desired signal there. The back-propagation rule propagates the errors through the network and allows adaptation of the hidden PEs. Two important characteristics of the multilayer perceptron are: its nonlinear processing elements (PEs) which have a nonlinearity that must be smooth (the logistic function and the hyperbolic tangent are the most widely used); and their massive interconnectivity (i.e. any

element of a given layer feeds all the elements of the next layer). The multilayer perceptron is trained with error correction learning, which means that the desired response for the system must be known. In pattern recognition this is normally the case, since we have our input data labeled, i.e. we know which data belongs to which experiment.
Error correction learning works in the following way: From the system response at PE i at iteration n, $\mathrm{y_i(n)}$, and the desired response $\mathrm{d_i(n)}$ for a given input pattern an instantaneous error $\mathrm{e_i(n)}$ is defined by:

$$\mathrm{e_i(n) = d_i(n) - y_i(n)} \qquad 1$$

Using the theory of gradient descent learning, each weight in the network can be adapted by correcting the present value of the weight with a term that is proportional to the present input and error at the weight, i.e.

$$w_{ij}(n+1) = w_{ij\,(n)} + \eta\delta_i(n)x_j(n) \qquad 2$$

The local error $\delta_i(n)$ can be directly computed from $\mathrm{e_i(n)}$ at the output PE or can be computed as a weighted sum of errors at the internal PEs. The constant η is called the step size. This procedure is called the back-propagation algorithm.
Their main advantages are that they are easy to use, and that they can approximate any input/output map. The key disadvantages are that they train slowly, and require lots of training data (typically three times more training samples than network weights).

b. Generalized Feed-Forward Network(GFN)

Generalized feed-forward networks (GFN) are a generalization of the MLP such that connections can jump over one or more layers. Like MLP, the numbers of layers are specified prior to further operation, and the MLP model is constructed, in which each layer feeds forward to all subsequent layers. In theory, a MLP can solve any problem that a generalized feed-forward network can solve. In practice, however, generalized feed-forward networks often solve the problem much more efficiently. A classic example of this is the two spiral problem. Without describing the problem, it suffices to say that a standard MLP requires hundreds of times more training epochs than the generalized feed-forward network containing the same number of processing elements. The advantage of the GFN is in the ability to project activities forward by bypassing layers. The result is that the training of the layers closer to the input becomes much more efficient. The same disadvantages of the MLP apply to the GFN.

c. Jordon-Elman Network (JEN)

Jordan and Elman networks extend the multilayer perceptron with context units, which are processing elements (PEs) that remember past activity. Context units provide the network with the ability to extract temporal information from the data. JEN provides four basic topologies, differing by the layers that feed the context units. The first configuration feeds the context units with the input samples, providing an integrated past of the input (memory traces). A second configuration creates memory traces from the first hidden layer (as proposed by Elman). A third possibility is to use the past of the last hidden layer activations as input to the context units. The final choice is to use the past of the output layer to create the memory traces, as proposed by Jordan.
The context unit remembers the past of its inputs using what has been called a recency gradient, i.e., the unit forgets the past with an exponential decay. This means that events that just happened are stronger than the ones that have occurred further in the past. The context unit controls the forgetting factor through the Time constant. Useful values are between 0 and 1. A value of 1 is useless in the sense that all of the past is factored in. On the other extreme, a value of zero means that only the present time is factored in (i.e., there is no self-recurrent connection). The closer the value is to 1, the longer the memory depth and the slower the forgetting factor. Context units are required when learning patterns over time (i.e., when the past value of the network influences the present processing). In the Elman network, the outputs of the hidden PEs from the previous time step are copied to the context units (Figure 2). In the Jordan network, the output of the network is copied to the context units. In addition, the context units are locally recurrent (i.e., they feedback onto themselves). The local recurrence decreases the values by a multiplicative constant t (time constant) as they are fed back. This constant determines the memory depth (i.e., how long a given value fed to the context unit will be "remembered").

One can treat the context units as input units, just as if they were obtained from an external source such as a file. Since the recurrent connections within the context units are fixed, static back-propagation is used to train these networks. Note that if the recurrent connections were adaptive, then back-propagation through time would be required. The JEN models are advantageous over the previous neural models that can only solve static problems. Temporal problems are ones where the previous value of the input affects the current output. The Jordan and Elman networks can solve temporal problems by processing information over time using recurrent connections. But, both of these nets are constrained in their ability to handle time. The time constant of the Jordan network is fixed and often difficult to set optimally for a given problem. Moreover, the past is always exponentially attenuated, which may not be very representative of the problem.

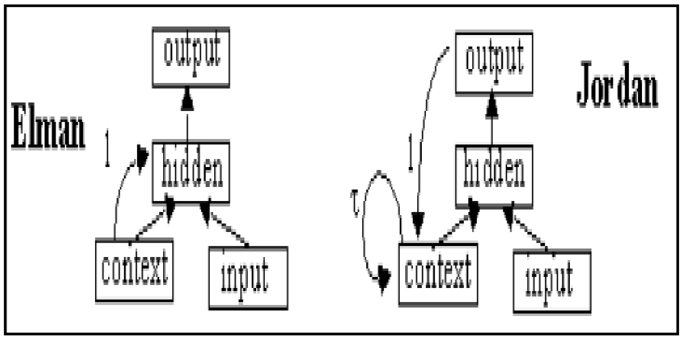

Fig. 2. Block diagrams of Jordan and Elman neural models

d. Time Lagged Recurrent Network (TLRN)

TLRNs are MLPs extended with short term memory structures that have local recurrent connections. The TLRN is a very appropriate model for processing temporal (time-varying) information. Examples of temporal problems include time series prediction, system identification and temporal pattern recognition. The training algorithm used with TLRNs is more advanced than standard back-propagation.

There are three memory structures to choose from namely, TDNN; Gamma; and Laguarre memories. The TDNN memory structure is simply a cascade of ideal delays (a delay of one sample). The gamma memory is a cascade of leaky integrators. The Laguerre memory is slightly more sophisticated than the gamma memory in that it orthogonalizes the memory space. This is useful when working with large memory kernels.
The focused topology only includes the memory kernels connected to the input layer. This way, only the past of the input is remembered. If the focused switch is not set, the hidden layers' PEs will also be equipped with memory kernels. The depth in Samples parameter (D) is used to compute the number of taps (T) contained within the memory structure(s) of the network. The number of taps within the input memory layer is dependent on the type of memory structure used. For the TDNN memory, the number of input taps T is equal to the depth D. The formula for the other two memory types is $T = 2D/3$. The number of taps for the memory structures at hidden layer n is computed (for all memory types) by the formula $Tn = T/2 * n$. This is only used as a starting point for the memory depth, since the depth will be adapted by the network.

The main advantages of TLRNs are the smaller network size required to learn temporal problems when compared to MLPs; their low sensitivity to noise; an adaptive memory depth for best duration to represent the input signal's past. TLRNs have some disadvantages too. The recurrent adaptation of the weights is nonlinear, so the training can get caught in local minima. Another disadvantage is that straight back-propagation cannot be used for training. The back-propagation through time (BPTT) algorithm is quite complex and requires a lot of memory.

Three different models were developed from TLRNs using memories such as gamma memory neural network; Time delay neural network (TDNN) and Laguarre memory neural network (LMNN). Details of these models are available in [1, 2, 11]. Figure 3-Figure 5 present block diagram of these three TLRN models.

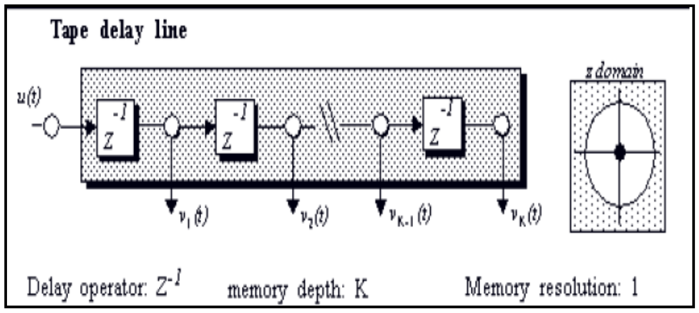

Fig. 3. Block diagrams of TLRN

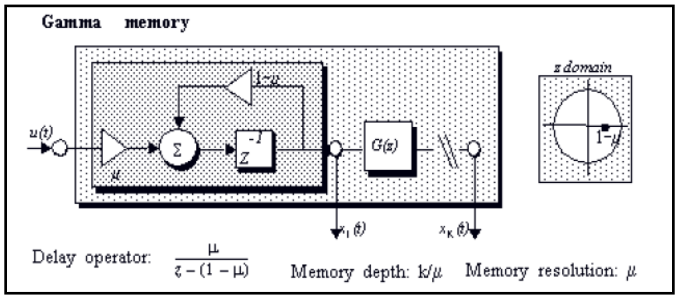

Fig. 4. Block diagrams of GMNN

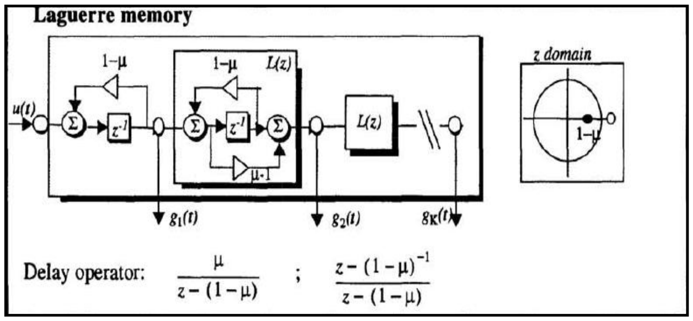
Laguerre memory
u(t)
1−μ
Σ
z^{-1}
L(z)
μ-1
$g_1(t)$
$g_2(t)$
$g_K(t)$
z domain
1−μ
Delay operator: $\frac{\mu}{z-(1-\mu)}$; $\frac{z-(1-\mu)^{-1}}{z-(1-\mu)}$

Fig.5. Block diagrams of LMNN

III. RESULTS AND DISCUSIION

Six different neural network models are developed using NeuroSolution-5 software namely, Multilayer perceptron (MLP); Generalized feed-forward network (GFN); Jordon-Elman network (JEN); Gamma memory neural network (GMNN), Time delay neural network (TDNN); and Laguarre memory neural network (LMNN). These models are trained and tested several times with various sets of network parameter such as transfer functions; learning rules; sample depth; hidden layers; and processing elements with daily observed precipitation data as input and daily observed discharge as output to the networks. The daily mean precipitation and discharge data are the data of Dholai(Rukni) river basin. The training and validation data considered here covers a period of 5 (2000-2005) years, out of which first 60% data are considered for training, next 15% data are considered for cross-validation and remaining 25% data are considered for testing the models. Upon adjustment of different parameters of each network structure during the calibration, the best performing network structures are achieved with Tanh transfer function and Levenberg-Marquardt learning rule for all the neural network models. The model performances are measured in terms of training min MSE; CV min MSE; testing MSE; NMSE; MAE; Min Abs Error; Max Abs Error; and r respectively. The training and testing results for all the ANN models are shown in Table 1. The results indicated that GMNN model dominated other models followed by JEN model. The minimum mean squared errors for training, cross-validation and testing results are 0.018, 0.011 and 551.71 respectively. Test results also indicated that GMNN model outperformed in other parameters such NMSE; MAE; Min Abs Error; Max Abs Error; RMSE and r respectively. A hydrograph of observed and best models discharge output is generated for the test series for comparing the performance of the best model (GMNN) with respect to the observed discharge (Figure 6).

Table 1. Test results of different ANN models

Model Name	Training results		Testing results						
	Training Min MSE	CV Min MSE	MSE	NMSE	MAE	Min Abs Error	Max Abs Error	r	RMSE
MLP	0.026	0.013	1216.96	0.66	25.39	0.036	151.68	0.588	34.88
GFN	0.025	0.013	1211.44	0.66	24.89	0.002	154.28	0.588	34.81
JEN	0.018	0.013	675.14	0.37	16.99	0.020	100.14	0.797	25.98
TDNN	0.011	0.014	749.41	0.41	18.31	0.003	116.16	0.777	27.37
LMNN	0.024	0.015	1041.05	0.57	17.98	0.001	182.96	0.724	32.26
GMNN	0.018	**0.011**	**551.71**	**0.30**	**14.71**	**0.050**	**113.06**	**0.848**	**23.48**

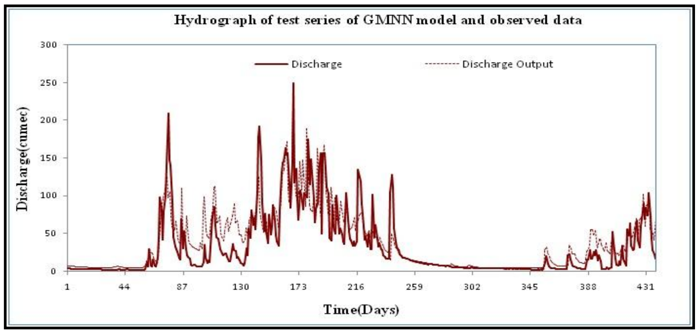

Fig. 6. Observed vs GMNN model output of test series.

To assess the accuracy of the ANN models, the model output data are analyzed in simulating major events of test series. The severe events of test data series are selected as most severe day, mean of five severe events and mean of ten severe events respectively. These flood events are analyzed in terms of percentage error with respect to the observed events. From the analysis, it is found that the minimum percentage error is achieved by GMNN model dominating other models in all the cases with minimum error as 0.57%, 13.17% and 14.42 for most severe day, mean of five severe events and mean of ten severe events respectively. The analysis results are shown in table 2. Figure 7- figure 9 are presented to show the variations in the actual mean values of the selected events of all the models with respect to the observed values. GMNN model simulated the daily mean observed discharge better than other models. Overall, GMNN model proved to be an effective tool in river flow modeling.

Table 2. Major events analysis results.

Events	Percentage errors of various events					
	MLP	GFN	JEN	TDNN	LMNN	GMNN
Most severe event error	8.46	3.24	28.99	35.28	31.39	0.57
Mean of five severe events error	49.12	51.24	39.39	36.62	37.32	13.17
Mean of ten severe events error	53.17	52.90	39.71	35.50	34.39	14.42

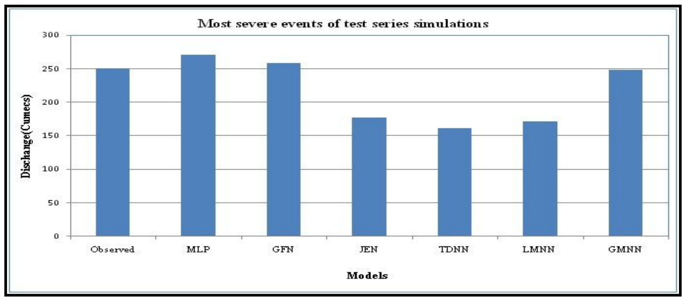

Fig. 7. Model wise simulation of most severe event.

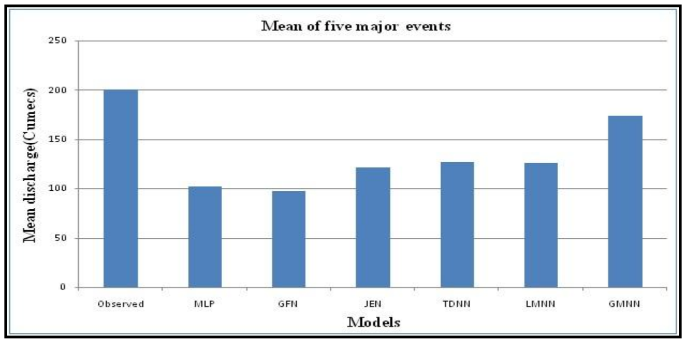

Fig. 8. Model wise simulation of mean of five severe events.

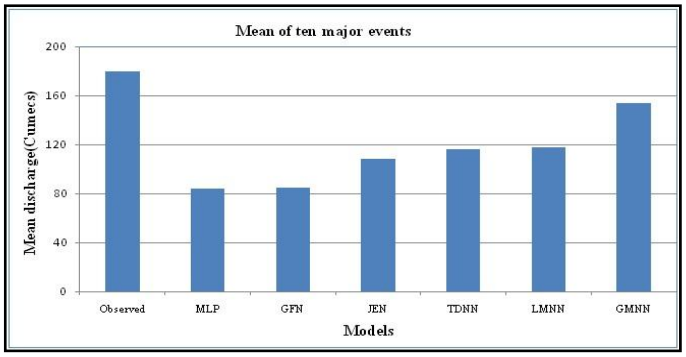

Fig. 9. Model wise simulation of ten most severe events.

IV. CONCLUSION AND FUTURE WORK

The aim of this paper is to analyze the performance of six different ANN models in simulating daily river discharge of Dholai(Rukni) river. The ANN models include both static and dynamic models. Important features of all the selected neural network modeling techniques and basic concepts of these were introduced. We have compared the simulation ability of all the ANN models with the observed mean daily data for test data series. Dynamic models such as GMNN model outperformed static and other dynamic models. The simulating ability of models are accessed in terms of training min MSE; CV min MSE; testing MSE; NMSE; MAE; Min Abs Error; Max Abs Error; RMSE and r respectively. The model performances are also analyzed using model output data in simulating major events of test series. The severe events of test data series are selected as most severe day, mean of five severe events and mean of ten severe events respectively. From the analysis, it is found that the GMNN model dominated other models in all the cases.

The area of neural networks is very diverse and opportunities for future research exist in many aspects.. Overall, GMNN model proved to be an efficient tool among the other chosen ANN models and information derived from the present study would be useful in planning, management and operation of the selected river and its main river.

ACKNOWLEDGEMENT

We are grateful to the authority of National Institute of Technology Silchar for providing necessary facilities and support for the research work. We are also grateful to Regional Meteorological Centre, Guwahati, Assam,India for providing necessary data.

REFERENCE

[1] Dibike Y.B., and Coulibaly, P. "Temporal neural networks for downscaling climate variability and extremes," *Neural Networks 19 (2006) 135–144*.

[2] Parthasarathi Choudhury and Parthajit Roy, Forecasting Concurrent Flows in a River System Using ANNs, J. Hydrol. Eng., 2015, 20(8): 06014012.

[3] French, M. N., Krajewski, W. F., and Cuykendall, R. R. (1992). ‘‘Rainfall forecasting in space and time using a neural network.’’ *J. Hydro*., Amsterdam, 137, 1–31.

[4] Hsu, K. L., Gupta, H. V., and Sorooshian, S. (1995). ‘‘Artificial neural network modeling of the rainfall-runoff process.’’ *Water Resour. Res*., 31(10), 2517–2530.

[5] Karunanithi, N., Grenney, W. J., Whitley, D., and Bovee, K. (1994). ‘‘Neural networks for river flow prediction.’’ *J. Comp. in Civ. Engrg*., ASCE, 8(2), 201–220.

[6] Zealand, C. M., Burn, D. H., and Simonovic, S. P. (1999). ‘‘Short term stream-flow forecasting using artificial neural networks.’’ *J. Hydro*., Amsterdam, 214, 32–48.

[7] Lippmann, R. (1987). An introduction to computing with neural nets. IEEE Acoustics, Speech, and Signal Processing Magazine, 4(2), 4–22.

[8] John Makhoul, “Pattern Recognition Properties of Neural Networks”, BBN Systems and Technologies.

[9] Lapedes, A. and Farber, R. (1988). "How Neural Networks Works," in *Neural Information Processing Systems* (Denver 1987), D. Z. Anderson, Editor, 442-456. American Institute of Physics, New York.

[10] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986a). Learning internal representations by error propagation. In D. E. Rumelhart, & J. L. McClelland (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition. Vol. 1: Foundations* (pp. 318–362). Cambridge, MA:MIT Press.

[11] Kisi, O., 2007. Streamflow forecasting using different artificial neural network algorithms, *ASCE J. of Hydrol. Eng*., 12(5), 532-539.

Author Profile

Shyama Debbarma received the **B.E.** degree in Agricultural engineering from the North-Eastern Regional Institute of Science and Technology (NERIST), Arunachal Pradesh, India, in 2004 and M.Tech in Water Resources Engineering from NIT Silchar. Currently pursuing PhD in NIT Siclahr, India. Research interest includes water resources management, river flow modeling, climate change, Neural Networks.

Dr. Parthajit Roy received the **B.E.** degree in Civil engineering from REC Silchar, India, in 1990.Currently working as Associate Professor in NIT Silchar, India. His research interest includes Water Resources Engineering, Open Channel Flow, Hydraulic Structures, Sediment Transport, and Hydrology.