STM JOURNALS

Journal of Water Resource Engineering and Management
ISSN: 2349-4336 (online)
Volume 1, Issue 2
www.stmjournals.com

Radial Basis Function Neural Networks for Rainfall-Runoff Modeling

K. S. Kasiviswanathan[1]*, *Avinash Agarwal*[2], *A. R. Senthil Kumar*[2]
[1]School of Mechanical and Building Sciences, VIT University, Chennai, India
[2]Surface Water Division, National Institute of Hydrology, Roorkee, India

Abstract

Rainfall-runoff process is purely nonlinear and varies spatially as well as temporally. Any hydrological model requires many parameters which represent different components of the process. Availability of all the parameters is difficult for any catchment and probabilistic generation of such type of data is impossible. Under such circumstances, artificial neural networks (ANNs) have proven to be a better tool to model the rainfall-runoff process with minimum available data. The present study is to compare the performance of the model trained with K-means clustering algorithm and modified K-means clustering algorithm. The potential of these two algorithms was tested by developing rainfall runoff models for Vamsadhara river basin located in Andhrapradesh, India. Results of these two models were compared with observed data of Vamsadhara river basin. It is shown that modified K-means clustering algorithm results are more generalized than K-means clustering algorithm results.

Keywords: *K-mean clustering algorithm, non-linear, radial basis function artificial neural network (RBFANN)*

****Author for Correspondence*** E-mail: vishwaiitr@gmail.com

INTRODUCTION

Data driven approach such as computational intelligence models are powerful search algorithms that can be used for modeling of hydrological phenomena. Artificial neural networks (ANNs) are one such technique that takes care of nonlinear behavior of the system. Numerous researchers have involved in devising different learning algorithms for ANNs according to various applications. Jalili and Kharaajoo [1] proposed a uniform weight learning algorithm to improve fault tolerance of neural network which improved the performance with lesser simulation time. Peralta *et al.* [2] coupled ANN with genetic algorithm for the direct encoding scheme. In direct encoding system, the information is placed in chromosomes. Golak [3] designed induced-weights artificial neural network to reduce the time-consuming task of pre-processing the patterns. Vieira [4] proposed iterative neural network approach for high-dimensional data analysis and concluded that it is robust, relatively simple to implement and it can handle many features, even if they are irrelevant for the solution. The state-of-the-art review presented by ASCE Task Committee [5] and Maier *et al.* [6] can be referred to know about the different ANN networks, transfer functions, training algorithms and real world applications in the field of hydrology and water resources. The application of back propagation neural network (BPANN) is reported in many cases [7–9]. One of the major problems with BPANN might get trapped in local minima while training. To circumvent this issue, radial basis function neural network uses clustering algorithm-based training procedure to optimize the parameters which lead to global minima. In general, hydrological system has higher to lower extreme events which lead high dimensionality in data patterns. RBFANN has potential to handle high-dimensional data with lesser computational cost. Many of the case studies pertaining to RBF [10–12] consider model-setting parameters such as spread, center as a constant value. This study aims to change these values dynamically using real code algorithm to improve its generalization capacity and termed as modified K-means clustering algorithm. The results of RBFANN with K-means clustering and modified K-means clustering are compared with actual output on the basis of root mean square error (RMSE), coefficient of correlation (CC), Nash and Sutcliffe [13] model efficiency (CE).

RADIAL BASIS FUNCTION ARTIFICIAL NEURAL NETWORK

An RBFANN has input nodes (juju), function nodes (ii), and output nodes (kaki) in layers. Each layer has different number of neurons or nodes for its operation. Input and output neuron is problem-dependent and is fixed. Functional layer neurons are assigned by the network designer based on behavior of network during training. The functional nodes consists a parameter vector called as center (c_i) and spread σ. The performance of radial basis function artificial network critically depends upon the chosen center. In general, the selection of center could be through an arbitrary selection from the data points of the subset or the mean of data points of the subset or ordinary least square of subset or orthogonal least square of subset. In this problem, it is estimated as a mean of randomly generated weight vectors.

The Euclidean distance is used to measure the distance between input vector and center. The Euclidean distance can be written as

$$d_{ij} = \left\| x_j - c_i \right\| \quad (1)$$

The main objective of the function is to minimize the Euclidean distance to get maximum function response from the function node.

In RBFANN, Gaussian function is often used as an activation function 13] and it is represented as,

$$\Phi(x) = \exp\text{-}\frac{\sum_{j=1}^{jj} d_{ij}^{\ 2}}{2\sigma^2} \quad (2)$$

where, $\Phi(x)$ is the response from the functional node.

From the function layer to output layer, the linear summation of weight into function response from the hidden layer is calculated initially and the Sigmoid function is used to project the output value in the output layer node.

$$Sum = \sum w_n \Phi(x) \quad (3)$$

where, w_n is the randomly generated weight vector. Finally, the output *(y)* is obtained from the following sigmoidal function.

$$y = \frac{1}{1+e^{\text{-}Sum}} \quad (4)$$

In learning strategy, the weights between input and function layer are updated using unsupervised training and function layer to output layer uses supervised training. The most popular unsupervised algorithm is K-means clustering algorithm. In clustering technique, it attempts to find centers for basis function in a manner that it reflects the distribution of input vectors over the input space. This can be accomplished in an unsupervised fashion using a variant of nearest neighbor analysis or by the Kohonen self-organizing feature maps. The Kohonen self-organizing feature maps are used for projecting patterns from high dimensional to low dimensional space. All connecting weights are adjusted by making a weight movement proportional to a Mexican hat function [14]. In supervised training a standard gradient descent procedure is used [15, 16].

In order to select the spread value of algorithm, the traditional K-means clustering algorithm is modified with Eq. (5). Based on that, the model is classified as static and dynamic model and can be seen from the flow chart in Figure 1. Static model (K-means clustering algorithm) uses constant spread value in all iterations. Dynamic model (modified K-means clustering algorithm) uses varying spread value by the weight vector and input patterns and is represented by Eq. (5).

$$\sigma_i^2 = \frac{1}{M}\sum_{j=1}^{jj}(x_j w_{ij} \text{-} c_i)^2 \quad (5)$$

where, w_{ij} is receptive weight of interconnection, M is the number of input vector and c_i is respective center of variable.

STUDY AREA AND DATA USED

The area selected for the study is the Vamsadhara river basin situated in between wellknown Mahanadi and Godavari river basins of south India. The total catchment area to the point where the river joins the Bay of Bengal is 10830 km^2 and is situated within the geographical coordinates of 18°15' to 19°55' north latitudes and 83°20' to 84°20' east longitudes (Figure 2). However, the catchment upstream to the last gauging and discharge measurement station of the river at Kashinagar, comprising 7820 km^2 is considered as the study area. The basin is narrow and highly undulated. A greater part of the catchment falls on the left side of the river. The temperature variation in the plains of basin is in between 10 to 43 °C and humidity during the monsoon is above 95%. The daily rainfall data (mm) and runoff (m^3/s) of the active period (June 1 to October 31) for years 1984 to 1989 and 1992 to 1995 were available and collected by India Meteorological Department (IMD) and Central Water Commission (CWC). The detailed description about the same study area and data used has been reported by Agarwal *et al.* [17].

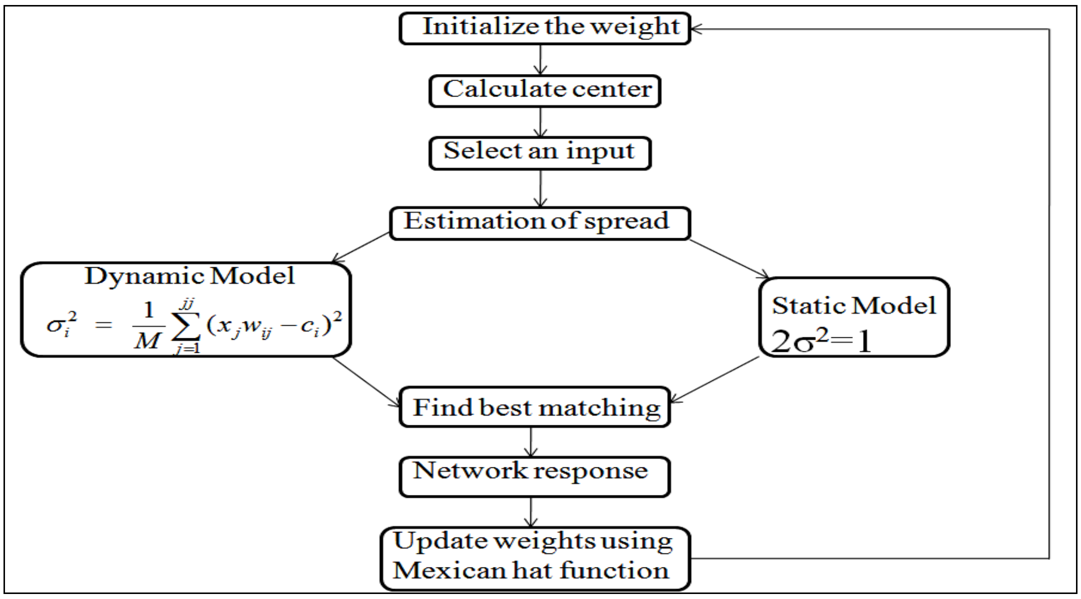

Fig. 1: *Flow Chart of Model Outline.*

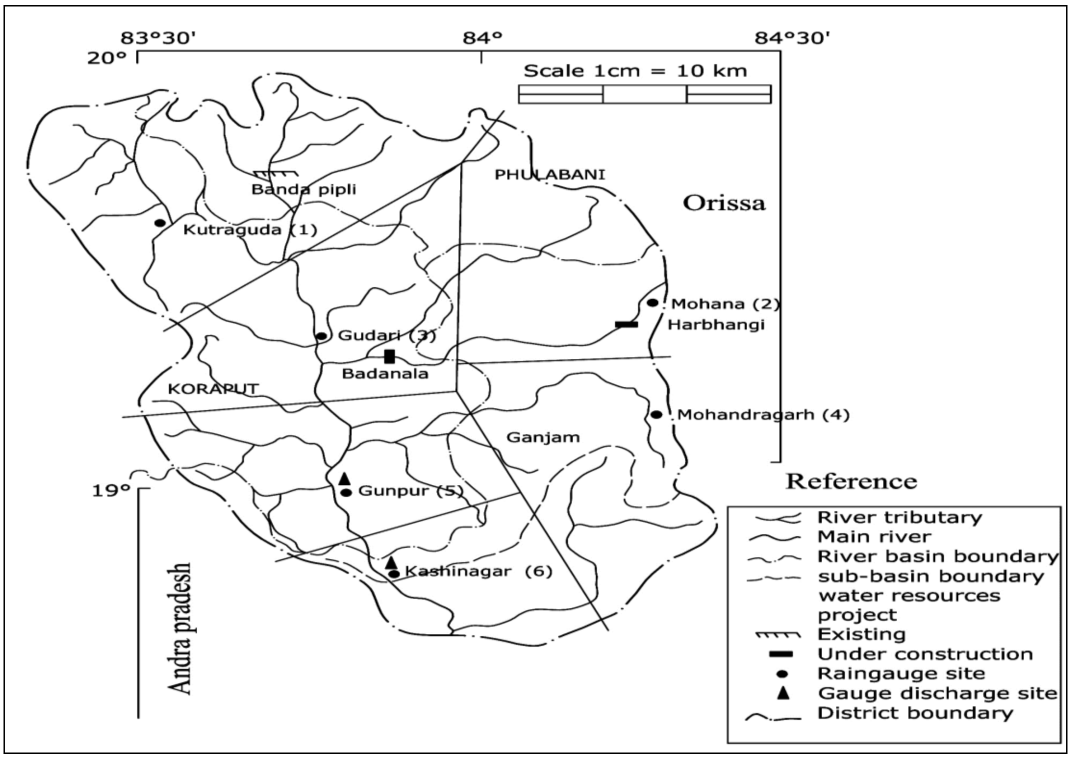

Fig. 2: *Index Map of Vamsadhara River Basin Showing Hydrological Details.*

MODEL DEVELOPMENT

The model is developed using K-means clustering and modified K-means clustering algorithm for Vamsadhara river catchment. The daily rainfall, runoff data of monsoon period (June 1 to October 31) for the years 1984–1989 and 1992–1995 are used for the development of rainfall-runoff modeling in which data from 1984–1987 are used for the calibration of the model whereas the data from1988–1989 and 1992–1995 are used for the cross validation and verification of the model respectively. Model is trained using calibration data and along with calibration, the obtained optimized weight of the neural network is used to check the performance verification data. Cross validation data is used to avoid any over-fitting of the model found in cross validation. Data normalization is performed using the Eq. (6).

$$x_n = \frac{x_0}{x_{\max}} \tag{6}$$

From the above equation, the data will fall from 0 to 1. De-normalization is carried out at the output nodes to get the actual value.

Considering different inputs, the following model is finalized using correlation matrix method and to maintain the parsimony of the model.

$$Q_t = f(R_t, R_{t-1}, Q_{t-1}, Q_{t-2}, Q_{t-3}) \tag{7}$$

where, Q_t represents the runoff at time (*t*). *Rt* represents rainfall at time (*t*) and *t–1, t–2, t–3* represent lagged values of inputs.

PERFORMANCE EVALUATION OF ANN MODEL

The performance of the proposed methodologies during calibration and validation is evaluated by performance indices such as root mean square error (RMSE), and coefficient of correlation (CC). They are defined as with the following equations:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \tag{8}$$

$$CC = \frac{\sum_{i=1}^{n}[(y_i - \bar{y}_i).(\hat{y}_i - \tilde{y}_i)]}{\sqrt{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2 . \sum_{i=1}^{n}(\hat{y}_i - \tilde{y}_i)^2}} \tag{9}$$

$$CE = \left\{1 - \frac{\sum_{j=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{j=1}^{n}(y_i - \bar{y}_i)^2}\right\} x\,100 \tag{10}$$

where, y_i is the observed runoff in m^3/s, $\bar{y}_i$ is the mean observed runoff in m^3/s, $\hat{y}_i$ is the estimated runoff in m^3/s and $\tilde{y}_i$ is the mean of estimated runoff in m^3/s.

RESULTS AND DISCUSSION

K-means Clustering Algorithm

Static RBFANN model critically depends on spread and center value. Change in weight controls the input vector to move closer to the center and hence producing maximum function output from the function layer. The weight change is carried out by suitable selection of learning rates. Static RBFANN model developed in MATLAB using inbuilt function *newrbe,* in which, the function newrbe takes input vectors (x_{ij}) and output vectors (O_k), and the spread value is fixed constantly to the value of 0.8, 1.0 and 1.2. The number of iterations should be optimal to avoid any over fitting. In this study, it was found that 500 iterations were sufficient to train the neural network for better prediction. This was found using various trials (100 to 5000).

In order to select the spread value, the network 5-4-1, 5-12-1 and 5-20-1 are selected. The range of spread values is varied from 0.8 to 1.2 and the results obtained are reported in Table 1. Based on CC, CE and RMSE values, it can be very well assessed that the performance in calibration of all three models is almost equally good for spread values (0.8, 1.0, and 1.2) considered. The cross validation and verification results are not good as compared to calibration for spread value as 0.8. A selection of spread value 1.0 or 1.2 could not be differenced based on cross validation and verification results. Based on literature, spread value of 1 as suggested by Shahsavand and Ahmadpour, 2005 could be a best choice to develop the final model.

The performance of five models with different networks 5-4-1, 5-8-1, 5-12-1, 5-16-1 and 5-20-1 is reported in Table 2. Based on the RMSE, CC, CE values, the model performance is continuously increased from smaller network to larger network during the calibration period. The efficiency of the model is 82.2% for the network 5-4-1, and it is increased to a value of 88.5% for the network 5-20-1. Cross validation results show that the performance of the models starts increasing for the network 5-4-1, 5-8-1, 5-12-1 and best performance for the model 5-16-1. The performance of the model 5-20-1 is deteriorated during the cross validation of the model. Verification results are showing decrease in performance by the increase of function nodes. The efficiency of the model with four function nodes is 81.6% and it is decreased to the value of 79.7% with 12 function node. For the network 5-16-1, the coefficient of efficiency is increased to a value 80.3%. Further increase of function node decreases the model performance. Thus, the overall performance is good for network 5-16-1 during calibration, cross validation and verification period.

Modified K-means Clustering Algorithm

In dynamic model (modified K-means clustering algorithm), there are two learning rates (i.e., function layer learning rate (α) and output layer learning rate (β)) used for the effective training with reduced computational time. Based on experience, the initial iterations were finalized and fixed nearly 2000. To ensure the proper selection of α from smaller network to larger network, three network structure (5-4-1, 5-16-1 and 5-32-1) are selected. The 'α' is varied from 0.5 to 15 and for the entire selected network; the model performance is good for 'α' value of 10 with minimum error. After fixing α value as 10, the emphasis has been focused towards the selection of learning rate (β) in output layer. To identify proper value of β, different values varying from 0.5 to 10 have been tried and found that the performance of model is good for the value of 0.5 in the entire selected network. To fix the optimum number of iterations, the system is run from lower to higher values of iterations and for three different network (5-4-1, 5-16-1 and 5-32-1). The number of iterations is varied from 100 to 10000 and it can be seen that number of iterations required for best optimization of network 5-4-1 is around 1000. After 1000 iterations, it was found that there is no remarkable variance as shown in Figure 3 during calibration period. The number of iterations required for the best optimization of networks 5-16-1 and 5-32-1 are 100 to 500. Figure 4 shows the convergence of network with 516-1 with respect to efficiency of the model during calibration of the model. Finally, it can be concluded that for a smaller network higher number of iterations required and with increase in network to 5-16-1 to 5-32-1 the number of iterations required for best optimization is reduced. Based on the finding, five models with different network varying 5-4-1 to 5-20-1 are developed and the results are reported in Table 3.

Table 1: *Performance of MATLAB RBFANN Model for the Varying Spread Value as 0.8 to 1.2.*

Network structure	Spread value	Model performance in different periods								
		Calibration (1984–1987)			Cross validation (1988–1989)			Verification (1992–1995)		
		RMSE (Abs.)	CC (%)	CE (%)	RMSE (Abs.)	CC (%)	CE (%)	RMSE (Abs.)	CC (%)	CE (%)
	0.8	59.3	97.0	82.2	87.6	78.5	57.6	84.4	91.3	81.6
5-4-1	1.0	62.3	89.6	83.0	88.0	79.6	57.1	76.1	93.2	88.5
	1.2	59.3	97.0	82.2	81.4	83.2	63.4	80.1	92.9	83.4
	0.8	52.9	92.7	85.9	92.5	76.9	52.7	87.7	90.3	79.2
5-12-1	1.0	51.0	93.2	86.9	89.3	78.7	56.0	88.7	90.4	79.7
	1.2	53.4	92.5	85.6	90.0	78.7	55.3	88.0	91.6	80.0
	0.8	48.4	93.4	87.9	91.6	87.3	53.7	104.7	88.3	71.7
5-20-1	1.0	47.6	94.1	88.5	102.3	81.9	42.2	108.3	87.8	69.7
	1.2	49.0	93.7	87.8	88.1	83.0	59.0	100.4	87.9	73.9

Table 2: *Performance of MATLAB RBFANN Model.*

Network structure	Model performance in different periods								
	Calibration (1984–1987)			Cross validation (1988–1989)			Verification (1992–1995)		
	RMSE (Abs.)	CC (%)	CE (%)	RMSE (Abs.)	CC (%)	CE (%)	RMSE (Abs.)	CC (%)	CE (%)
5-4-1	59.3	97.0	82.2	87.6	78.5	57.6	84.4	91.3	81.6
5-8-1	54.4	92.2	85.0	97.9	73.1	47.0	85.8	90.9	81.2
5-12-1	51.0	93.2	86.9	89.3	78.7	56.0	88.7	90.4	79.7
5-16-1	49.2	93.7	87.8	82.0	94.6	62.9	87.2	91.2	80.3
5-20-1	47.6	94.1	88.5	102.3	81.9	42.2	108.3	87.8	69.7

Table 3: *Performance of Dynamic RBFANN Model for Fixed α as 10, Fixed β as 0.5 and for Varying Iterations as 1000 to 500.*

Network structure		Model performance in different period								
	Iter ation	Calibration (1984–1987)			Cross validation (1988–1989)			Verification (1992–1995)		
		RMSE (Abs.)	CC (%)	CE (%)	RMSE (Abs.)	CC (%)	CE (%)	RMSE (Abs.)	CC (%)	CE (%)
5-4-1	1000	66.9	83.0	62.5	82.7	81.4	62.3	112.3	82.8	67.5
5-8-1	1000	67.3	81.6	62.0	79.9	82.2	64.7	117.8	81.6	64.2
5-12-1	800	52.8	87.6	76.7	79.0	81.8	65.5	111.9	87.6	67.7
5-16-1	500	52.6	87.7	76.8	81.2	80.6	63.6	120.8	86.0	62.3
5-20-1	500	51.6	88.2	77.7	82.1	80.2	62.8	122.2	85.2	61.4

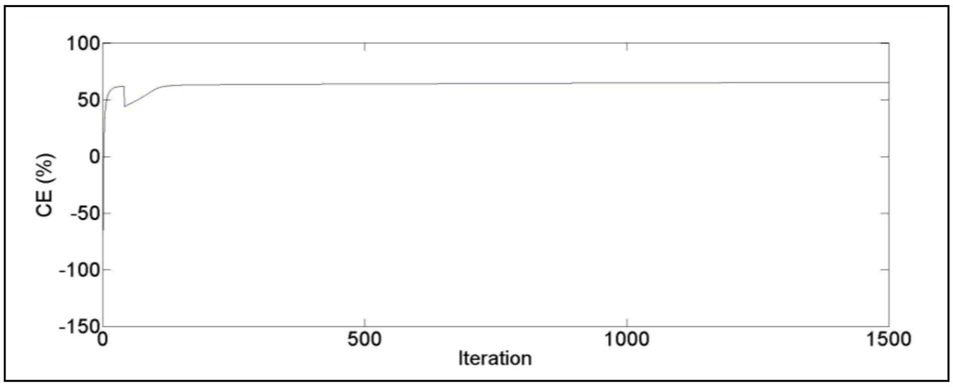

Fig. 3: *Model Performance vs No. of Iterations (5-4-1).*

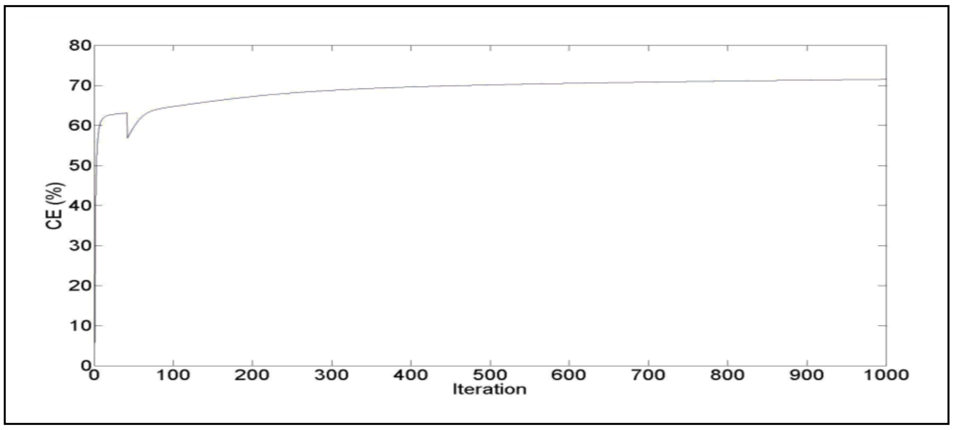

Fig. 4: *Model Performance vs No. of Iterations (5-16-1).*

The model performance in calibration increases with increase in function node. For a network (5-8-1), the performance in calibration is lower than cross validation but closely matches with verification results. It suggests that the model development is biased towards the properties of cross validation data set and supports the over learning of the model. Similarly, for a network 5-12-1, the performance is increased in calibration, cross validation and verification. From network 5-12-1 to 5-20-1, the performance of model increased in calibration and decreased in cross validation and verification period due to the over learning of the network. Based on the results it is observed that the model performance is better for a smaller network structure. With increase in network structure, the performance of model increases and is best for model structure as 5-12-1. A further increase of structure improves the model performance in calibration but not in cross validation and verification period. Thus the increase in calibration indicates biasness leads over learning of model during in development and that is supported by higher efficiencies during model calibration.

Model Comparison

In static model, the smaller network needs lesser number of iterations and larger network needs more number of iterations thus reducing modeling time. This may be due to the static model's fixed spread value and hence for the smaller network, the model can reach optimum path to activate the function node with limited number of iterations. If the number of function nodes is higher, the cluster formed will be high. Thus the model has to search the optimum path to activate the function node with more number of iterations. On the other hand dynamic model needs more iterations for the smaller network and less iterations for larger network. This can be seen that the dynamically changing spread value needs more iterations to optimize the system for the smaller network and less number of iterations for the larger network. Both models require same value function layer learning rate (α). Dynamic model needs constant β value for all networks. However, static model needs varying β as smaller network needs higher value and larger network needs lower value. In static model, the calibration results are good over cross validation and verification results for the entire network.

CONCLUSIONS

In K-means clustering algorithm, static and dynamic models are equally performing better and behave oppositely with respect to learning rate and iterations required for the selected network. Comparing the K-means clustering and modified K-means clustering results shows that K-means clustering algorithm results are good for the entire selected network during the calibration and verification period but not in cross validation period. From this, K-means clustering algorithm approach could not be a generalized model compare to modified K-means clustering algorithm. Thus the developed modified K-means clustering algorithm could be a better choice for the rainfall-runoff modeling of a selected basin.

REFERENCES

1. Mahdi Jalili K, Kharaajoo. Proposing a new learning algorithm to improve fault tolerance of neural networks. *ICCS*, *LNCS.* 2004; 3037: 717–21p.
2. Juan Peralta, German Gutierrez, Araceli Sanchis. Design of artificial neural networks based on genetic algorithms to forecast time series. *Innovations in Hybrid Intelligent Systems* ASC. 2007; 44: 231–8p.
3. Slawomir Golak. Induced weights artificial neural network. *ICANN* 2005, *LNCS.* 2005; 3697; 295–300p.
4. Armando Vieira. An iterative artificial neural network for high dimensional data analysis. *ICANN* 2005, *LNCS.* 2005; 3697: 691–6p.
5. ASCE Task Committee on Application of Artificial Neural Networks in Hydrology (2000) Artificial neural networks in hydrology–I: Preliminary concepts. *Journal of Hydrologic Engineering.* 2000; 5(2): 115–23p.
6. Maier HR, Ashu J, Graeme CD, et al. Methods used for the development of neural networks for the prediction of water resource variables in river systems: Current status and future directions. *Environmental Modelling and Software.* 2010; 25(8): 891–909p.

7. Arslan CA. Rainfall–Runoff Modeling Based on Artificial Neural Networks(ANNs). *European Journal of Scientific Research*. 2011; 64(4): 490–506p.
8. Kumar ARS, Sudheer KP, Jain SK, et al. Rainfall–runoff modelling using artificial neural networks: comparison of network types. *Hydrol. Process* 2005; 19: 1277–91p.
9. Sudheer KP, Gosain AK, Ramasastri KS. A data–driven algorithm for constructing artificial neural network rainfall–runoff models. *Hydrol. Processes*. 2002; 16(6): 1325–30p.
10. Nor NIA, Harun S, Kassim AHM. Radial basis function modeling of hourly streamflow hydrograph. *Journal of Hydrologic Engineering*. 2007; 12(1): 113–23p.
11. Suhaimi S, Bustami RA. rainfall runoff modeling using radial basis function neural network for Sungai Tinjar catchment, Miri, Sarawak. *UNIMAS E–Journal of Civil Engineering*. 2009; 1(1).
12. Fernando DAK, Jayawardena AW. Runoff forecasting using RBF networks with OLS algorithm. *J.Hydrol. Engng* ASCE. 1998; 3(3): 203–9p.
13. Nash JE, Sutcliffe JV. River flow forecasting through conceptual models. *Hydrological Science Journal*. 1970; 41(3): 399–417p.
14. Ralph B. On the convergence of derivatives of B–splines to derivatives of the Gaussian function. *Computational and Applied Mathematics*. 2008; 27(1): 79–92p.
15. Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back–propagation errors. *Nature*. 1986; 323: 533–6p.
16. Agarwal A, Singh RD. Runoff modeling through back propagation artificial neural network with variable rainfall–runoff data. *Water Resources Management*. 2004; 18(3): 285–300p.
17. Agarwal A, Singh RD, Mishra SK, et al. ANN–based sediment yield models for Vamsadhara river basin (India). *Journal of Biosystems Engineering*. 2005; 31(1): 95–100p.