DEVELOPMENT RADIAL BASIS FUNCTION ARTIFICIAL NEURAL NETWORK AND ITS APPLICATION

Avinash Agarwal
Scientist 'F'

Introduction

Radial basis function artificial neural network RBANN is very similar in topology to the multi layer BPANN. Unlike, multi layer BPANN in which the dimensionality of the data is reduced by projecting a large input space on to a smaller number of hidden units and forcing the data through a bottle neck, the RBANN does precisely the opposite. In RBANN network the function nodes replaces the hidden nodes of BPANN. These function nodes do not implement the same multiply-and-add (weighted summation) as the hidden nodes in a BPANN but computes a respective field and the respective fields from individual function overlaps.

Artificial Neural Network (ANN) models are being widely used in many hydrological applications in recent years by many researchers. However, the study of the internal characteristics of radial basis type ANN model is very limited. The internal characteristics are the data domain (length and statics), optimal ANN structure, suitable learning rate and optimal number of iteration. An improper selection of these parameters leads to error in model development. This lecture investigates the suitable selection of optimal ANN structure, learning rates and optimum number of iteration required for RBANN to model the rainfall-runoff process. The daily rainfall and runoff data of Vamsadhara basin, Andhra Pradesh, were used to develop the model. The optimum number of iteration was identified through appropriate selection of learning rates ALR and ALRG values and the model for the rainfall-runoff process was developed. The performance of the RBANN rainfall-runoff model was compared with back propagation artificial neural network BPANN and support vector machines (SVM) model. It is has been found that the ANN has potential for successful application to the problem of runoff modelling. The back propagation artificial neural network BPANN model was best among all.

Mathematical Modelling

A RBF network with input, function, and output layers of nodes respectively with j, i, and k are shown in figure 1. The structure of RBFANN shows jj-dimensional input pattern (x) being mapped to kk-dimensional output (o). The values j and k are problem dependent, the value i is to be determined by the network designer.

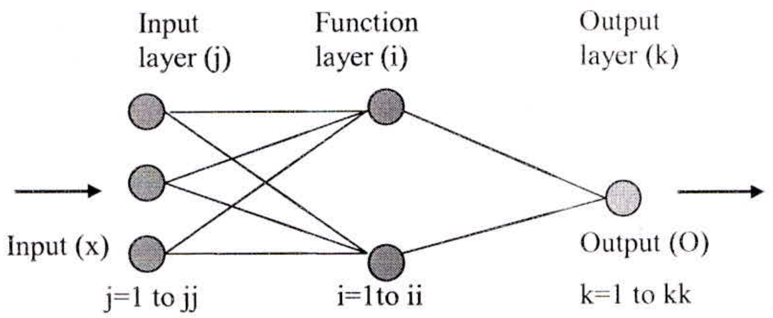

Figure 1: Structure and notations in a radial basis function ANN.

In RBANN operation input of nth pattern with each pattern made up of jj variables represents a point in the jj-dimensional input space. It enters the network at the input layer such that one variable is fed into one node. The input layer does not transform the pattern, but forward an image of variables to each node in the function layer. The nodes in the function layer are each specified by a transfer function f(d), which radially transforms the incoming information.

For n input patterns x having jj dimensionality (x^n_{jj}), the response of O_i of hidden layer through radial transformation describe can be expressed in mathematical terms as;

$$O_i = f(d)$$

where, O_i is the output of function layer and f(d) a nonlinear function.

The non-linearity with in a radial basis function network can be chosen from a few typical nonlinear functions as explained below;

(a) Thin plate spline function:
$f(d) = d^2 \log(d)$; When, $d \Rightarrow 0$ then $f(d) \Rightarrow 0$
$d \Rightarrow \propto$ then $f(d) \Rightarrow \propto$

(b) Gaussian function:
$f(d) = \exp - [d^2/\sigma^2]$; When, $d \Rightarrow 0$ then $f(d) \Rightarrow 1$
$d \Rightarrow \propto$ then $f(d) \Rightarrow 0$

(c) Multi quadric function:
$f(d) = [d^2 + \sigma^2]^{1/2}$; When, $d \Rightarrow 0$ then $f(d) \Rightarrow \sigma$
$d \Rightarrow \propto$ then $f(d) \Rightarrow \propto$

(d) Inverse multi quadric function:
$f(d) = [d^2 + \sigma^2]^{-1/2}$; When, $d \Rightarrow 0$ then $f(d) \Rightarrow 1/\sigma$
$d \Rightarrow \propto$ then $f(d) \Rightarrow 0$

The most commonly used transfer function of the RBF network is radially symmetric or Gaussian function shown in figure 2.

In above equation σ is a measure of spread of data x_j in cluster association also called normalization factor. Typically five alternatives for estimation of σ are considered and the FORTRAN programme is developed for each cased for calculations.

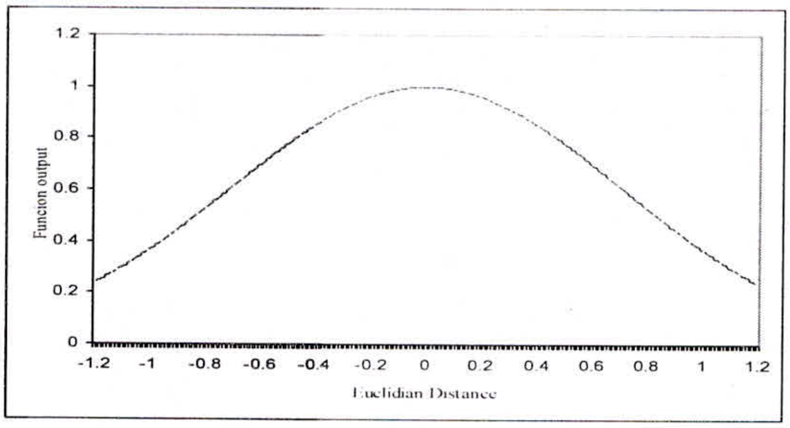

Figure 2: A view of Gaussian function.

Case I:

It is commonly described as the average distance between the cluster center and training instances (number of input variables) in that cluster.

$$\sigma_i^2 = \frac{1}{\text{jj}} \sum_{j=1}^{\text{jj}} (x_j\, w_{ij} - C_i)^2$$

where, w_{ij} is receptive weight of interconnection and C_i is respective center of variable.

Case II:

In this approach, the width value is calculated from the input data points itself. So for all the iteration, the width value will be same.

$$\sigma^2 = \frac{\Sigma(x_i - \bar{x})^2}{n}$$

where, $\bar{x}$ is the mean of input data x_j and n is the total number of input variables.

Case III:

This approach consists in taking the widths equal to a constant for all Gaussian functions for example, the widths are fixed as follows:

$$\sigma_i = \frac{d_{\max}}{\sqrt{2M}}$$

where, M is the total number of centers and $d_{\max}$ is the maximum distance between those centers. Since, the maximum distance between centers reduces with increasing optimization, the value of σ becomes low and finally the condition of math floating point error is obtained during calculation.

Case IV:

In this method, the width is calculated from the maximum and minimum value of input data points. The width value will not get changed in further iteration. But the width value will change when the cluster will move from one hidden neuron to another hidden neuron.

The standard deviation (i.e. width) of the jth neuron is;

$$\sigma_i = \sqrt{\frac{d_{\max}^2}{i+1}}$$

where, $d_{\max}$ is maximum distance between training data set.

Case V:

In this case, the spread value is fixed constantly. A hidden neuron is more sensitive to data points near its center. For Gaussian RBF this sensitivity may be tuned by adjusting the spread (σ), where a larger spread implies less sensitivity and smaller spread implies good sensitivity. In general, the value of spread $2\sigma^2$ is taken as 1.

The Euclidean distance (d) between the set of inputs and respective center of variable is given as,

$$d_{ij} = || x_j - C_i ||$$

An RBF is represented by a center (C) where the function value is highest and a spread (σ) that is indicative of the radial distance from the RBF center, within which the function value is significantly different from zero. Of the several possible radial functions, the most common choice is the Gaussian function. The Gaussian RBF center of the ith function node can be specified by mean C_i and deviation σ_i also known as measure of spread of the respective field of the function.

The performance of radial basis function network critically depends upon the chosen center. A radial basis function network for exact interpolation would interpolate every data point and the number of nodes in the function layer would be equal to the number of input-output patterns resulting in large complex network. In order to reduce the complexity of the network the centers of the variable are chosen as to be a subset of the data set. 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. The center obtained through the mean of data point of the subset can be given as,

$$C_i = \sum_{j=1}^{jj} \frac{w_{ij}}{jj}$$

Each RBF center i then compute through a radially symmetric function usually Gaussian, with the output a maximum value when the input patterns x_j is near Euclidean distance. The response O_i of the function unit i is calculated by using equation;

$$O_i = \exp - [d^2/2\sigma^2]$$

$$O_i = \exp - [\{\sum_{j=1}^{jj} ||x_j - C_i||^2\} / 2\sigma_i^2]$$

In case the center C_i is considered as the value of respective weight the above equation may be written as,

$$O_i = \exp - [\{\sum_{j=1}^{jj} ||x_j - w_{ij}||^2\} / 2\sigma_i^2]$$

In learning strategy, the weights between input and function layer are updated using supervised or unsupervised training. In supervised training a standard gradient descent procedure can be used. This involves the minimization of an objective function with respect to the parameters C and σ. However, such procedures are liable to be trapped in local minimum of the parameter space. The other method is the orthogonal least square technique. The procedure starts with the single basis function network. All the input vectors are trained for the center, and one resulted the best performance is retained. The number of basis function is increased incrementally in steps of one. A search is conducted for the best center among the remaining input vectors. This procedure can be repeated until a desired level of performance is achieved.

The other method is the clustering algorithm. Clustering technique 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 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. At the beginning of training process, these weights are randomly initialized. Present a set of input x_j and compute the magnitude of Eucilidian distance for each neuron i. Select the neuron having minimum distance. All connecting weights adjacent to the winner node are adjusted by making a weight movement proportional to a Mexican hat function.

The second derivative of Gaussian function ($\exp(-d^2/2)$) is $f^{('')}(d) = (d_{ij}^2 - 1) * \exp - (d_{ij}^2/2)$ as Mexican hat function as shown in figure 3.

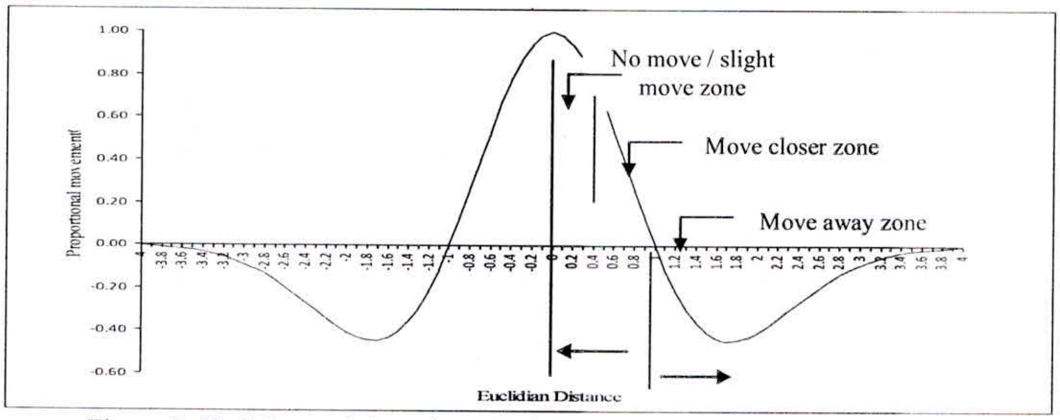

Figure 3: Training activity using Mexican hat function

The proportional movement related to the Mexican hat function may be explained with the third derivative of the Gaussian function as shown in figure 4.

$$f^{(''')}(d) = \Delta w_{ij} = (3d_{ij}^2 - d_{ij}^3) * \exp - (d_{ij}^2/2)$$

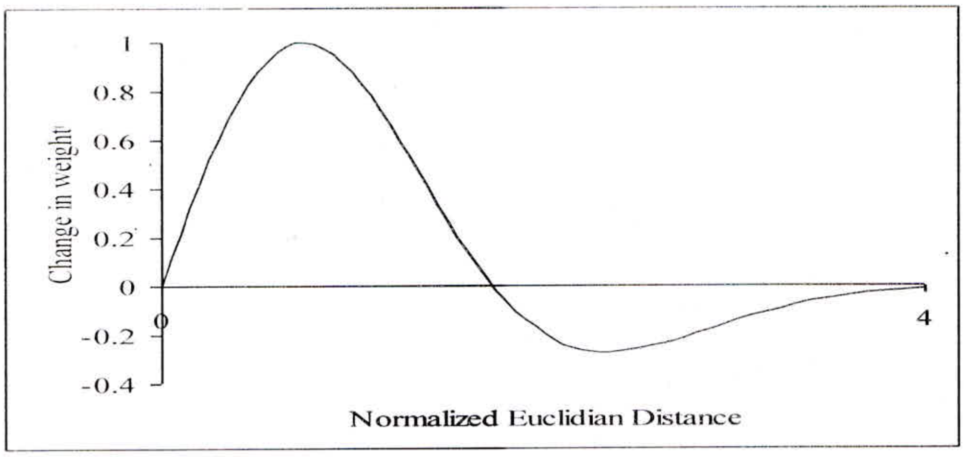

Figure 4: Proportional movement using Mexican hat function

The function is reduced to zero to one range, so that the relative movement from the function will be zero to one.

$f(\cdot\cdot\cdot)(d) = \Delta w_{ij} = (12d_{ij}^2 - 64d_{ij}^3) * \exp - (8d_{ij}^2)$

The Mexican hat function has the effect in moving near neighbors close or no movement while neurons slightly away moved closer and the neurons still further away will have their weights moved away from the input space as shown in figure 3.

Based on the change in weight from the Mexican hat function, the move is calculated.

$\text{move}_{ij} = (x_j - w_{ij}) * \Delta w_{ij} * \alpha$

The new updated weight for the next iteration,

$w_{ij}(t) = w_{ij}(t-1) + \text{move}_{ij}$

Training of weights between the function and the output layer nodes are weighted according to their strengths. The responses of the function layer neurons are summed up according to theses output layer weights by the nodes in the output layer. Therefore, optimization of weights between function and output layer and often de-coupled and accomplished independently.

Learning in radial basis network can be divided into two stages. For any iteration, first the learning is carried out in function layer that is followed by learning in output layer. The learning in function layer is performed using unsupervised method, such as the k-means clustering algorithm. While learning in the output layer uses supervised methods. After the initial solution is obtained by this approach, a supervised learning algorithm (back propagation) could be applied in both the layers to fine-tune the weights of the network as an optional strategy.

Study Area and Data Used

The area selected for the study is the Vamsadhara river basin situated in between well known 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 5). However, the catchment upstream to the last gauging and discharge measurement station of the river at Kashinagar, comprises of 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°C to 43°C and humidity during the monsoon is above 95 percent. 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 study area and data used has been reported by Agarwal et al., (2005).

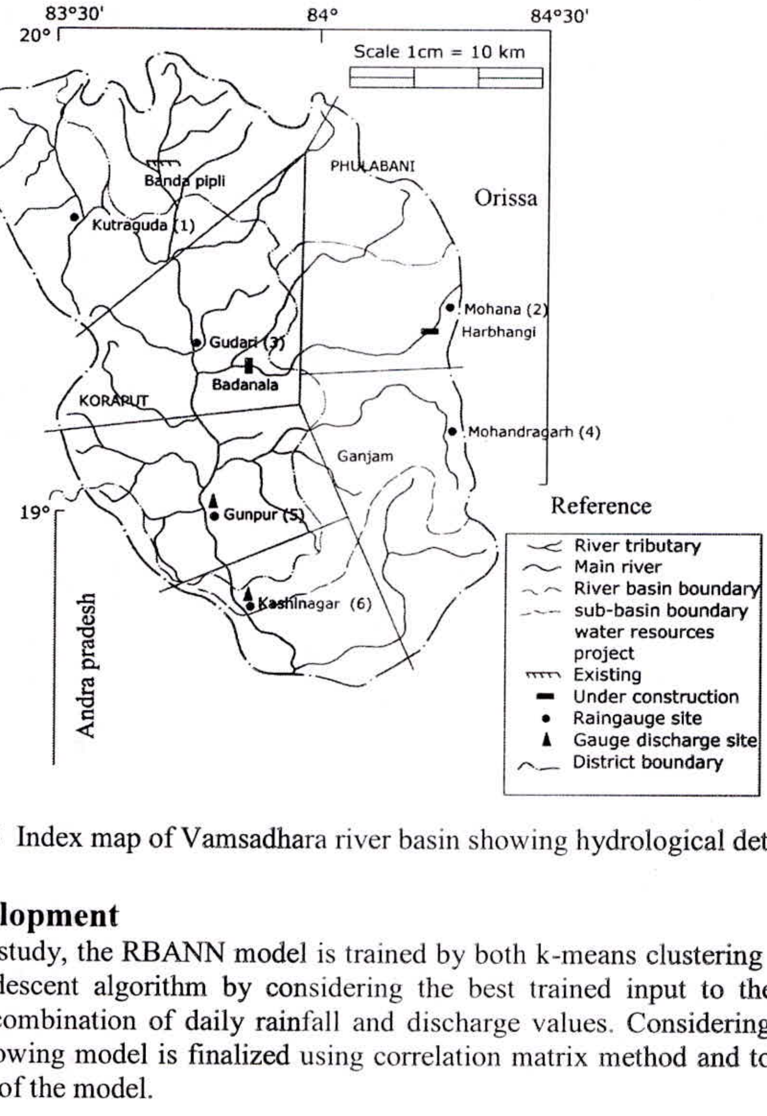

Figure 5 Index map of Vamsadhara river basin showing hydrological details.

Model Development

In this study, the RBANN model is trained by both k-means clustering algorithm and gradient descent algorithm by considering the best trained input to the network consists of a combination of daily rainfall and discharge values. Considering different inputs the following model is finalized using correlation matrix method and to maintain the parsimony of the model.

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

where, Q_t represents the runoff at time (t); R_t represents rainfall at time (t).

The daily rainfall, runoff data of monsoon period (June 1st to October 31st) for the years 1984-1989 and 1992-1995 were used for the development of rainfall-runoff models. In which, the data from 1984 to 1987 were used for the calibration of the model where as the data from1988-1989 and 1992-1995 were used for the cross validation and verification of the model respectively. The best performance of model is obtained by choosing proper learning rate ALR and ALRG and optimum number of iteration required.

Selection of ALR

The first step in development of RBANN model is to select the values for learning rates as ALR, ALRG and optimum number of iteration. In literature, it has been mentioned that learning rate ALRG as 0.5 for the gradient decent algorithm is good for the proper convergence of the network (Agarwal A, 2002) and hence the learning rate in output layer (ALRG) is fixed as constant value of 0.5 for all cases in the beginning. In present case the learning rate (ALR) in function layer is varied from 0.5 to 15 with keeping constant spread value of 1.0 and 500 iterations. The performance results for network 5-4-1, 5-16-1 and 5-32-1 is presented in table 1.

Table 1: Performance of model for fixed ALRG as 0.5, fixed iterations as 500 spread 1.0 and for varying ALR as 0.5 to 15.

Network Structure	**ALR**	**Model performance in different period**								
		Calibration (1984-1987)			**Cross Validation (1988-1989)**			**Verification (1992-1995)**		
		CC (%)	**CE (%)**	**EV (%)**	**CC (%)**	**CE (%)**	**EV (%)**	**CC (%)**	**CE (%)**	**EV (%)**
5-4-1	0.5	65.9	13.3	-26.2	75.5	-9.9	-50.3	75.7	-2.7	-57.2
	1	-76.6	-32.3	-28.2	-74.8	-63.7	-60.4	-46.2	-62.0	-71.0
	2	85.8	70.9	3.9	84.8	66.2	-12.8	88.5	68.6	-17.4
	5	86.0	72.9	4.3	84.6	68.6	-10.4	88.5	68.6	-17.4
	10	86.2	69.0	26.6	84.1	69.7	7.8	87.5	71.8	-3.5
	15	78.3	-1363.3	429.4	73.9	-674.0	236.2	74.7	-231.1	151.1
5-16-1	0.5	65.4	10.9	-31.0	75.4	-14.3	-53.5	75.7	-2.8	-57.3
	1	-75.9	-39.0	-49.8	-73.9	-79.4	-72.9	-46.2	-62.0	-71.0
	2	85.8	70.9	3.6	84.8	66.2	-13.1	88.5	68.6	-17.4
	5	86.0	72.6	3.3	84.7	68.0	-11.6	88.5	68.6	-17.4
	10	85.9	73.6	4.1	84.7	70.5	-9.0	87.5	71.9	-3.5
	15	85.2	65.4	4.5	84.8	60.5	-16.1	74.8	-231.2	151.2
5-32-1	0.5	66.0	13.7	-25.5	75.6	-9.3	-49.9	75.8	-2.3	-56.9
	1	-75.7	-41.2	-54.7	-73.6	-83.2	-75.7	-40.9	-75.7	-82.2
	2	85.8	70.9	3.2	84.8	66.1	-13.4	88.6	67.9	-18.5
	5	86.0	72.6	3.2	84.7	68.0	-11.8	88.6	67.9	-18.5
	10	85.8	73.6	3.0	84.7	70.6	-9.3	88.3	73.0	-14.2
	15	85.3	66.2	4.5	84.8	61.3	-15.6	89.0	60.6	-23.5

It can be clearly seen that the performance of learning rate (ALR) is good for the value of 2 to 10 and results reasonably good for correlation coefficient, coefficient of efficiency with minimum volumetric error for the network 5-4-1. The coefficient of efficiency is consistently good for the ALR value of 10 in all calibration, verification and cross validation periods giving the value of 73.6%, 70.5% and 71.9% for the network 5-16-1. The performance of model is very poor for the value of ALR as 0.5, 1, and 15. In network 5-32-1, the ALR values higher or lower than 10 affecting the model performance drastically and leads poor model convergence and results in the similar response as the network 5-16-1. Finally, it suggests that the model performance is best from the lower network (5-4-1) to higher network (5-32-1).

Selection of ALRG

The next step of the model development is identification of learning rate in output layer (ALRG), which is initially taken as 0.5 referring the literature. To identify the proper value of output layer leaning rate (ALRG), the function layer learning rate (ALR) is fixed constant to a value of 10 as identified previously and the network is varied from 5-4-1 to 5-32-1 with varying ALRG. Initially the ALRG is fixed to a value of 0.5 and that increased up to 10 for a fixed 500 iterations and spread as 1.0. The model performance is analyzed and is presented in table 2.

Table 2: Performance of model for fixed ALR as 10, fixed iterations as 500 spread 1.0 and for varying ALRG as 0.5 to 15.

Network Structure	**ALRG**	**Model performance in different period**								
		Calibration (1984-1987)			**Cross Validation (1988-1989)**			**Verification (1992-1995)**		
		CC (%)	**CE (%)**	**EV (%)**	**CC (%)**	**CE (%)**	**EV (%)**	**CC (%)**	**CE (%)**	**EV (%)**
	0.5	86.2	69.1	26.7	84.2	69.8	7.8	87.5	71.9	-3.5
	1	86.2	73.4	11.6	84.3	70.7	-2.6	87.8	71.8	-10.7
5-4-1	2	86.2	74.1	4.3	84.4	70.2	-7.6	87.9	71.3	-14.2
	5	86.2	74.2	1.7	84.4	69.9	-9.2	87.9	71.2	-15.2
	10	86.2	74.2	1.3	84.3	69.9	-9.4	87.9	71.3	-15.2
	0.5	85.9	73.6	4.1	84.7	70.5	-9.0	88.3	72.2	-14.6
	1	85.8	73.6	0.6	84.7	69.9	-11.3	88.3	72.0	-16.1
5-16-1	2	85.7	72.9	-5.3	84.7	67.8	-16.3	88.5	69.8	-20.3
	5	85.5	69.7	-14.0	84.7	62.1	-24.3	88.8	64.1	-27.5
	10	85.4	65.5	-19.2	84.6	55.4	-29.8	89.0	57.2	-33.2
	0.5	85.8	73.6	3.0	84.7	70.6	-9.3	88.3	73.0	-14.2
	1	85.8	73.5	0.3	84.7	69.9	-11.5	88.3	72.2	-16.1
5-32-1	2	85.6	72.7	-5.6	84.7	67.7	-16.6	88.5	69.8	-20.5
	5	85.4	69.5	-14.2	84.7	62.0	-24.5	88.8	64.1	-27.6
	10	85.3	65.3	-19.3	84.6	55.2	-30.0	89.0	57.2	-33.3

It can be seen that the performance of model 5-4-1 is good for ALRG varying from 2 to 10 and the value of ALRG as 5 could be the best selection based on CC and CE values (Table 2). The error in volume (EV) is not varying high. Other two values of ALRG as 0.5 and 1.0 results in lower performance and calibration error of volume are also high for these cases. For the network 5-16-1, based on the values of CC, CE and EV the performance of the model is good for the ALRG value of 0.5 to 1.0. The model performance is poor for the ALRG value as 1 to 10. The value of CC is almost constant.

The value CE is decreasing with increasing ALRG and EV is increasing for all calibration, cross validation and verification period. The performance of model for the higher network (5-32-1) is better for the lower value of ALRG in comparison with higher value of ALRG. For the value of ALRG as 0.5, the coefficient of efficiency is good during calibration, cross validation and verification as 73.6%, 70.6% and 73%. It can be seen that the selection of ALRG as reported in literature as 0.5 is not suitable in present case and is found varying with network selection. For a lower network 5-4-1, the value of ALRG is 5.0. However, for network 5-16-1 to 5-32-1, the value of ALRG is equal to 0.5 as reported in literature.

Selection of iteration required

After fixing the value of ALR and ALRG, the performance of the model was checked for the number of iteration which was taken initially as 500 fixed. The models with suitable ALR and ALRG were developed for varying iteration from 100 to 1000 and are reported in table 3. For the network (5-4-1) during the calibration period, the model efficiency is increased from 73.0% to 74.2%. On the other hand the cross-validation and verification efficiencies of the model getting reduced from 70.8 to 69.9% and 72.1 to 71.2%. This is due to the over learning of the system. Thus the performance of model is good for a 100 iteration. For increased iteration 500 and beyond it, there is no significant variation for 5-4-1 network and showing equal performance as obtained in 500 iteration. Further in higher network (5-16-1) and (5-28-1), the model performance is optimized in 500 iteration and increase of iteration does not show any remarkable variation in model efficiency (Table 3).

Table 3: Performance of model for fixed ALR as 10 spread 1.0, varying ALRG as 0.5 to 5 and for varying iterations as 100 to 1000.

Network Structure	ALRG	Iter-ation	Model performance in different period								
			Calibration (1984-1987)			Cross Validation (1988-1989)			Verification (1992-1995)		
			CC (%)	CE (%)	EV (%)	CC (%)	CE (%)	EV (%)	CC (%)	CE (%)	EV (%)
		100	86.2	73.0	13.8	84.3	70.8	-0.7	87.7	72.1	-9.2
5-4-1	5	500	86.2	74.2	1.7	84.4	69.9	-9.2	87.9	71.2	-15.2
		1000	86.2	74.2	1.5	84.4	69.9	-9.4	87.9	71.2	-15.3
		100	85.5	62.8	9.0	84.8	57.9	-14.1	89.1	56.1	-23.9
5-16-1	0.5	500	85.9	73.6	4.1	84.7	70.5	-9.0	88.3	72.2	-14.6
		1000	85.9	73.7	3.1	84.7	70.6	-9.2	53.3	72.8	-14.3
		100	85.9	72.2	6.9	84.8	68.9	-9.5	88.6	69.1	-16.6
5-28-1	0.5	500	85.8	73.6	3.0	84.7	70.6	-9.3	88.3	73.0	-14.2
		1000	85.8	73.6	2.9	84.7	70.6	-9.3	88.3	73.0	-14.2

Selection of spread

In literature, it has been suggested that the RBANN behaves better for the spread value of 0 to 1 (Shahsavand and Ahmadpour, 2005). In this study, the initial selection of spread value was 1.0. In order to finalize the spread value, the network 5-4-1, 5-16-1 and 5-28-1 were selected for fixed ALR as 10 and suitable ALRG as identified earlier for the structures. The range of spread value varied from 0.8 to 1.2 and the results obtained are reported in table 4. Based on CC, CE, and EV values it can be very well assessed that the performance of all three model is the best for spread value of 1.0. A spread value less than or higher than 1.0, results in lower performance of the model in all calibration, cross validation as well as verification period.

Table 4: Performance of model for fixed ALR as 10, varying ALRG as 0.5 to 5, fixed iterations as 500 and for the varying spread as 0.8 to 1.2.

Network Structure	ALRG	Spread	Model performance in different period								
			Calibration (1984-1987)			Cross Validation (1988-1989)			Verification (1992-1995)		
			CC (%)	CE (%)	EV (%)	CC (%)	CE (%)	EV (%)	CC (%)	CE (%)	EV (%)
		0.8	85.2	70.2	17.4	83.6	69.2	5.6	86.5	68.3	-3.3
5-4-1	5	1.0	86.2	74.2	1.7	84.4	69.9	-9.2	87.9	71.2	-15.2
		1.2	86.0	73.0	3.0	84.7	68.4	-11.2	86.0	68.5	-18.0
		0.8	86.1	72.7	6.9	84.6	68.6	-8.9	88.4	68.2	-16.5
5-16-1	0.5	1.0	85.9	73.6	4.1	84.7	70.5	-9.0	88.3	72.2	-14.6
		1.2	85.6	73.2	3.6	84.8	70.7	-9.2	88.5	73.6	-14.0
		0.8	84.1	71.0	1.2	84.5	70.2	-9.3	88.0	71.6	-14.8
5-28-1	0.5	1.0	85.8	73.6	3.0	84.7	70.6	-9.3	88.3	73.0	-14.2
		1.2	85.5	72.1	1.2	84.5	70.2	-9.2	88.6	71.8	-15.0

Selection of final RBANN models

Based on the finding, eight RBANN models with different network varying 5-4-1 to 5-32-1 are developed for fixed ALR as 10 spread 1.0 and varying ALRG as 0.5 to 5 and are reported in table 5.

Table 5: Performance of model for fixed ALR as 10, varying ALRG as 0.5 to 5, the constant spread as 1.0 and for iteration 100 to 500

Model and Network Structure	ALRG	Iter-Ation	Model performance in different period								
			Calibration (1984-1987)			Cross Validation (1988-1989)			Verification (1992-1995)		
			CC (%)	CE (%)	EV (%)	CC (%)	CE (%)	EV (%)	CC (%)	CE (%)	EV (%)
RBNN 5-4-1	5	100	86.2	73.0	13.8	84.3	70.8	-0.7	87.7	72.1	-9.2
RBNN 5-8-1	4.5	200	86.2	74.2	3.9	84.4	70.1	-8.0	88.0	71.2	-14.4
RBNN 5-12-1	4	300	85.6	63.0	-9.5	84.6	52.0	-25.7	88.8	42.4	-35.6
RBNN 5-16-1	2	500	85.7	72.9	-5.2	84.7	67.8	-16.3	88.5	69.8	-20.3
RBNN 5-	0.5	500	85.9	73.7	3.6	84.7	70.5	-9.0	88.3	72.4	-

20-1											14.5
RBNN 5-	0.5	500	85.8	73.6	3.0	84.7	70.6	-9.3	88.3	73.0	-
24-1											14.2
RBNN 5-	0.5	500	85.8	73.6	3.0	84.7	70.6	-9.3	88.3	73.0	-
28-1											14.2
RBNN 5-	0.5	500	85.8	73.6	3.0	84.7	70.6	-9.3	88.3	73.0	-
32-1	0.5	750	90.5	81.6	-1.9	86.3	73.7	-9.4	90.1	72.7	14.2
BPNN 5-						86.3	73.7		90.3	72.7	-
10-1											17.5
SVM											

From the network (5-4-1 to 5-8-1), there is a slight increase in performance during calibration period but cross validation and verification results are biased. Further, the function node is increased to 12 and the results show that there is a lesser performance compared to previous network. From the network (5-16-1) to (5-32-1), the performance of the model increased and producing suitably good results in calibration as well as in cross validation and verification periods. In all network structure, during cross validation and verification, a negative value of volumetric error shows the difference of data domain of the system. From the whole observation, the network initially needs higher learning rate (ALRG as 5) and lesser number of iteration when the numbers of function nodes increased, it needs lesser learning rate (ALRG as 0.5) and higher iteration. Based on the results it can be seen that the model performance is better for a lower network structure. With increase in network structure, the performance of model is biased and indicates for an over learning of system and that is supported by a higher efficiencies during model calibration.

BPANN and SVM Models

Considering the methodology described in section 2.2 and 2.3, the models were developed and available in literature (Agarwal A, 2002; Debasmita Misra et al., 2009) for the same area using the same data utilized for the development of RBANN.

Conclusion

In this paper, the selection of suitable learning rates and optimum number of iteration for training static RBANN has been proposed. Function layer needs higher learning rate may be due to the k-means clustering algorithm performed in unsupervised fashion and not depends on observed value to minimize the error function. Output layer needs higher learning rate for the lower network and lower learning rate for the higher network. In accordance with iteration, the lower network needs less number of iteration and higher network needs more number of iteration. The simulation results suggest that the selected learning rates and iteration provides fast and stable learning and modelling. The RBANN models are compared to BPANN and SVM models available in literature for the same area (Agarwal A, 2002; Debasmita Misra et al., 2009) presented in table 5. A comparison of three methods suggests that BPANN and SVM models are superior to RBANN. The best model is the BPANN considering the easiness of model.

References

1. Agarwal A. 2002. Artificial Neural Networks and their application for simulation and prediction of runoff and sediment yield. Ph. D. thesis, Department of soil and Water Conservation Engineering, G.B. Pant University of Agriculture and Technology, Pantnagar, India.
2. Agarwal A., Singh R.D., Mishra S.K.,and Bhunya P.K. 2005. ANN based sediment yield models for Vamsadhara river basin (India). Journal of Bio Systems Engineering, 31(1), 95-100.
3. Debasmita Misra, Thomas Oommen, Avinash Agarwal, Surendra K. Mishra, 2009. Application and analysis of support vector machine based simulation for runoff and sediment yield. Biosystems Engineering. 103, 527-535.
4. Shahsavand A., Ahmadpour A., 2005. Application of optimal RBF neural networks for optimization and characterization of porous materials. Journal of Computers and Chemical Engineering, 29(10), 2134-2143.

SOLVED EXAMPLE

The input patterns for a RBANN and respective weights are given tabular form. Find out structure of RB and find first update of weights for first pattern, first iteration of the process.

Input/Output variables as:

	In puts		Out puts	
Pattern 1	6	1	3	9
Pattern 2	4	12	5	11
Pattern 3	8	6	18	16
Pattern 4	6	1	2	7

Weight as:

w_{ij}	W_{i1}	w_{i2}	w_{ij}
W_{1j}	.243	.145	.139
W_{2j}	.127	.399	.212
W_{3j}	1	2	

W_{ki}	W_{k1}	W_{k2}
W_{1j}	.469	.444

Solution: For the normalization of Input and Output for four numbers of patterns, the maximum and minimum values are first estimated and by using these values the normalization of input and output is done.

Max	8	12	18	16
Min	4	1	2	7

Normalized values of input and output are as follows:

	In puts			Out puts
Pattern 1	.75	.08	.17	.56
Pattern 2	.50	1.0	.28	.69
Pattern 3	1.0	0.5	1.0	1.0
Pattern 4	.75	.08	.11	.44

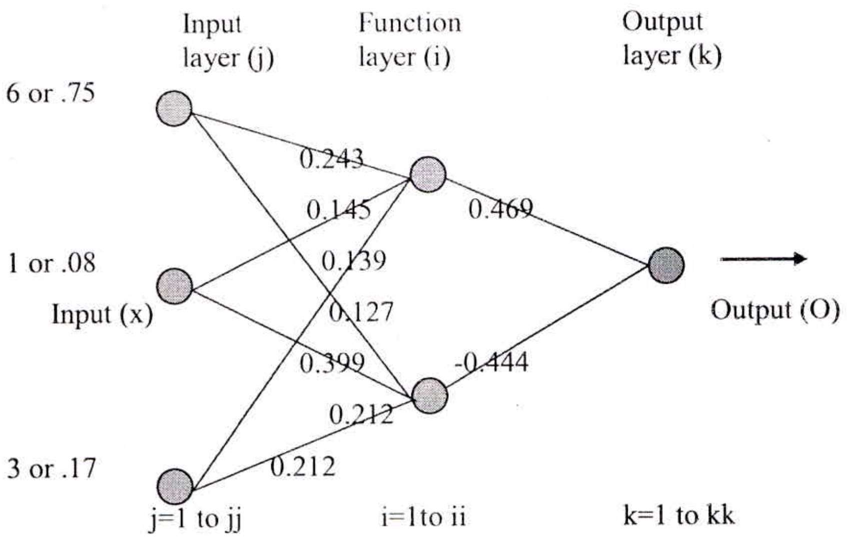

Figure 6: Structure and notations and values in a radial basis function ANN analysis.

In RBF ANN patterns are given one by one. Here the first pattern is given for analysis and the analysis sequence is presented. The calculation for the first case is discussed here.

Case I: (i) Input layer to hidden layer calculation

$C_i = \sum_{j=1}^{jj} \frac{w_{ij}}{jj}$ $C_1 = (.243 + .145 + .139) / 3 = .527 / 3 = .1756$

$(\text{ROS1})^2$ $\sigma^2 = \frac{\Sigma(x_i - \bar{x})^2}{n}$

$\sigma_1^2 = (0.75*0.243 - 0.1756)*2 + (0.08*0.145–0.1756)*2 + (0.17*0.139 – 0.1756)\ 2$

$= (0.00665)^2 + (-0.164)^2 + (-0.15197)$

$= 0.0000442225 + 0.026896 + 0.02309488$

$= 0.050035 / 3 \quad = 0.016678$

(ED_{ij}) $d_{ij} = || x_j - C_i ||$

(ED_{11}) $d_{11} = \text{ABS}(0.75 - .1756) = 0.5744$

(ED_{12}) $d_{12} = \text{ABS}(0.08 - .1756) = 0.0956$

(ED_{13}) $d_{12} = \text{ABS}(0.17 - .1756) = 0.0056$

SUM $= (0.5744)^2 + (0.0956)^2 + (0.0056)^2$

$= .3299 + .00914 + .00003 \quad = .3391$

$O_i = \exp - [\{\sum_{j=1}^{jj} ||x_j - w_{ij}||^2\} / 2\sigma_i^2]$

$O_i = \exp - [d^2/2\sigma^2]$

$= \exp- (20.34)$

$= 0.000038$

$EDN_{ij} = ABS\ (ED_{ij} - w_{ij})$

$EDN_{11} = ABS\ (.5744 - .243) = .3314$

$EDN_{12} = ABS\ (.0956 - .145) = .0494$

$EDN_{13} = ABS\ (.0056 - .139) = .1334$

Change in weight,

$$\Delta w_{ij} = \{(12EDN_{ij}^2 - 64EDN_{ij}^3) * \exp - (8EDN_{ij}^2)\}/1.371$$

$\Delta w_{11}=\{(12*0.\ 3314^2 -64*0.\ 3314^3)*\exp-(8*0.\ 3314^2)\}/1.371 = 0.4991$

$\Delta w_{12}=\{(12*0.\ 0494^2 -64*0.\ 0494^3)*\exp-(8*0.\ 0494^2)\}/1.371 =0.4185$

$\Delta w_{13}= \{(12*0.\ 1334^2 -64*0.\ 1334^3)*\exp-(8*0.\ 1334^2)\}/1.371=0.9166$

$move_{ij} = (x_j-w_{ij})* \Delta w_{ij}*\alpha$

$move_{11} = (.75-.243)* (0.\ 4991)*0.5= 0.1265$

$move_{12} = (.08-.145)* (0.\ 4185)* 0.5= -0.0136$

$move_{13} = (.17-.139)* (0.\ 9166)* 0.5= 0.0142$

New weights in the second iteration

$w_{ij}(t) = w_{ij}(t-1)+ move_{ij}$

$w_{11}(t) = .243+.1265= .3695$

$w_{12}(t) = .145-.0136= .1314$

$w_{13}(t) = .139+.0142= .1532$

(i) Hidden layer to output layer calculation

The output from the Gaussian function in hidden layer is as follows:

There are two neurons in the hidden layer

$O_1 = 0.000038$

$O_2=.021868$

$w_{ki}=0.469$

w_{ki}=-0.444

weighted sum, S={(0.000038*0.469)+(0.02868* -0.444)}

O_k = -0.01272

Finally, the output from the output layer

O_k = (1/1+exp(-S))

O_k= (1/1+exp(-(-0.01272))=0.4968