(22)

Project Report

on

Rainfall-Runoff Modeling of Sutlej River Basin, India using Soft Computing Techniques

Conducted at National Institute of Hydrology, Roorkee

Uttarakhand

Gautam Buddha University

School of Engineering, Greater Noida

Submitted By

Jatin Kumar Singh

11/ICE/049

Under the guidance of

DR. A. R. SENTHIL KUMAR
SCIENTIST-E
SWH DIVISION

CERTIFICATE

This is to certify that **Mr. Jatin Kumar Singh** has undergone a project work on "**Rainfall-Runoff Modeling of Sutlej River Basin using Soft Computing Techniques**" from 1st November, 2015 to 30th April, 2016 as six months training submitted to the Research Management and Outreach Division, National Institute of Hydrology, Roorkee, in partial fulfillment of the requirement for the award of degree of 5 year Integrated Dual Degree Programme B.Tech (Civil Engineering)+M.Tech(Environmental Engineering) is an authentic work carried out by him under my supervision and guidance.

Date: 09/05/16

A. R. Senthilkumar

DR. A. R. SENTHIL KUMAR

SCIENTIST-E

SWH DIVISION

ACKNOWLEDGEMENTS

I would like to express my deepest appreciation to all those who provided me the possible help to complete this report. It is my great pleasure to acknowledge those whose active help and support make this report possible in the present form.

I would also like to express my sincere gratitude to my supervisor **Dr. ATHAR HUSSAIN**, Assistant Professor, Department of Civil Engineering, Gautam Buddha University, Greater Noida, for encouraging me to undergo this training programme. I extend my heartfelt thanks to all the teachers of the department and the college for their support, suggestion and blessings.

I express my heartfelt thanks and gratitude to **Dr. A. R. SENTHIL KUMAR**, Scientist E, Surface Water Hydrology Division, National Institute of Hydrology, for his valuable guidance and support on completion of this project in its present form. This would not have been possible without his guidance, support and encouragement. Under his guidance I successfully overcame many difficulties and learned a lot.

I am extremely thankful to **Dr. SURJEET SINGH**, Scientist D, Ground Water Hydrology Division, National Institute of Hydrology, for his steady support, guidance and valuable suggestion.

I thank almighty, my parents for their constant encouragement and financial support. I equally thanks to my **friends** for their co-operation and helping hand at the times of adversities.

JATIN KUMAR SINGH

ABSTRACT

Artificial Neural Network (ANN) is a very useful data modeling tool that is able to capture and represent complex input and output relationships. The advantage of ANN lies in its ability to represent both linear and non-linear relationships and in its ability to learn these relationships directly from the data being modeled. Modeling of rainfall runoff relationship is important in view of the many uses of water resources such as hydropower generation, irrigation, water supply and flood control.

This study is to purposefully develop a rainfall runoff model for rainfall-runoff modeling in Sutlej river basin, India using soft computing techniques such as Artificial Neural Network (ANN), Radial Basis Function (RBF) and Fuzzy Logic. Training and simulation was done using Matlab 6.5.1 software with varying parameters to obtain the optimum result.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

Abbreviations	Description
ANN	Artificial Neural Networks
RBF	Radial Basis Function
USGS	United States Geological Survey
SOFM	Self-Organizing Feature Map
SCE-UA	Shuffled Complex Algorithm Evolution
SPIDA	Simulation Program for Interactive Drainage Analysis
MLP	Multi-Layer Perceptron
FIs	fuzzy implications

CHAPTER 1

INTRODUCTION

1.1 GENERAL

The demand for water has increased due to population growth, urbanization and industrialization as a result of which watersheds and river systems have been altered. This will cause greater damage to property and result in loss of life if flooding occurs. Therefore, it's critically important to successfully plan, design and manage these water resources systems. To determine the relationship of transformation of precipitation to runoff is an important issue in surface hydrology. Forecasted runoff can be used in stream flow measurement and planning for water supply, flood control, irrigation, drainage, power generation, water quality, recreation, etc. A rainfall runoff model is required to obtain the relationship between rainfall and runoff. This model is capable of forecasting river runoff values that can be used in hydrology and hydraulic engineering design and water management purposes. Hence, it is important to hydrologist in determining these two relationships.

However, this relationship is known to be highly non-linear and complex due to large spatial and temporal variability of watershed characteristics and precipitation patterns, and the number of variables involved in the modeling of the physical process (Tokar & Johnson 1999). Generally three types of models, including deterministic (physical) models, conceptual models and empirical/systems theoretic/black-box models, are being used by hydrologists in order to model this relationship. The deterministic (physical) models describe the relationship using physical laws of mass and energy transfer (Dawson & Wilby 2001). In contrast, in conceptual models instead of using physical laws of mass and energy transfer, a simplified, but a plausible or reliable conceptual representation of the underlying physics is adopted (Jain & Srinivasulu 2006).

An alternative modelling approach for hydrological processes such as rainfall-runoff process is the empirical/systems theoretic/black-box models, which try to find a relationship between historical inputs and outputs (ASCE Task Committee 2000a)

without detailed understanding of the physics involved in the process under investigation, such as artificial neural networks (ANNs).

1.2 ARTIFICIAL NEURAL NETWORKS

In the recent past, Artificial Neural Networks (ANN) modeling has gained significant attention due its ability to provide better solutions when applied to complex systems that have been poorly described or understood and where input is incomplete or uncertain by nature. ANN models shows better performance over the than traditional modelling techniques such as empirical models, statistical models (autoregressive, autoregressive moving average models) and physical based models. The advantages of ANN models over physically based models have been described in detail by French et al. (1992).

The applications of ANNs in hydrology can be found in many papers, such as (Shamseldin 1997; Abrahart & Kneale 1997; Dawson & Wilby 1998; Abrahart & See 2000; Coulibaly et al. 2000; Imrie et al. (2000); Govindaraju & Rao 2000; Lekkas et al. 2001; Persson & Berndtsson 2001; Shamseldin & O'Connor 2001; Tayfur 2002).

An ANN is an information-processing system that consists of a number of interconnected processing elements called nodes, analogous to neurons in the brain. Their growth is founded on the following principles:

i. Information processing takes place at many individual elements called nodes, too known as units, cells, or neurons.
ii. Signals are passed between nodes over interconnection links.
iii. A synaptic weight is assigned to each connection link to represent the connection strength between two nodes.
iv. A nonlinear function called an activation function is applied to the net input by each node to determine its output signal.

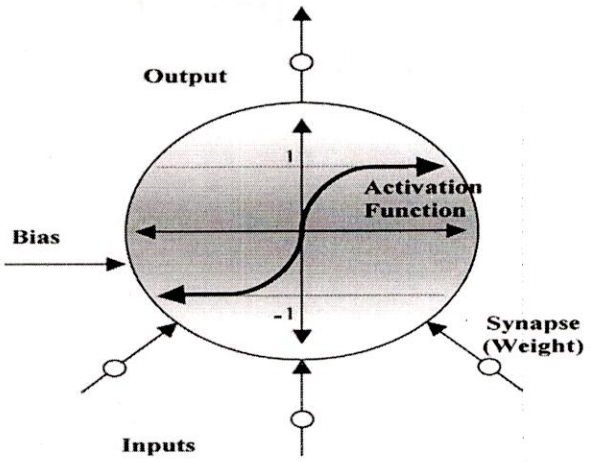

Figure 1.1: A typical artificial neuron

A neural network is characterized by its architecture that represents the pattern of connection between nodes, its method of determining the connection weights, and the activation function (Fausett 1994). Caudill presented a detailed description of neural networks in a series of papers (Caudill 1987, 1988, 1989).

One way of classifying neural networks is by the number of layers: single (Hopfield nets); bilayer (Carpenter/Grossberg adaptive resonance networks); and multilayer (most backpropagation networks). ANNs can also be categorized based on the direction of information flow and processing. A feedforward network consists of nodes that are arranged in layers that begins from a first input layer and ends at the final output layer. The information is passed from the input to the output side. The neurons in one layer are connected to those in the adjacent layers, but not to those in the same layer. Thus, the output of a node in a layer is only dependent on the inputs it receives from previous layers and the corresponding weights.

However, in a recurrent Ann information is passed through the neurons in both directions from the input to the output side and vice versa. This is generally achieved by recycling previous network outputs as current inputs, thus allowing for feedback.

In many previous studies ANN type such as Multilayer Feed foreword back propagation neural network (MLFBPN) commonly adopted and it proved to be most powerful tool to 80 per cent of practical application in all filed of hydrologic engineering and sciences (Hsu et al., 1995; Smith and Sli, 1995). In the present study, a multilayered feed forward backpropagation neural network model is developed with rainfall, evaporation and lagged runoff as input to predict runoff. The number of

hidden layers and the number of nodes in each hidden layer are usually determined by a trial-and-error procedure. The nodes within neighboring layers of the network are fully connected by links. One of the most groundbreaking rediscoveries was that of backpropagation techniques (which were conceived by Rosenblatt) by Rumelhart et al. (1986).

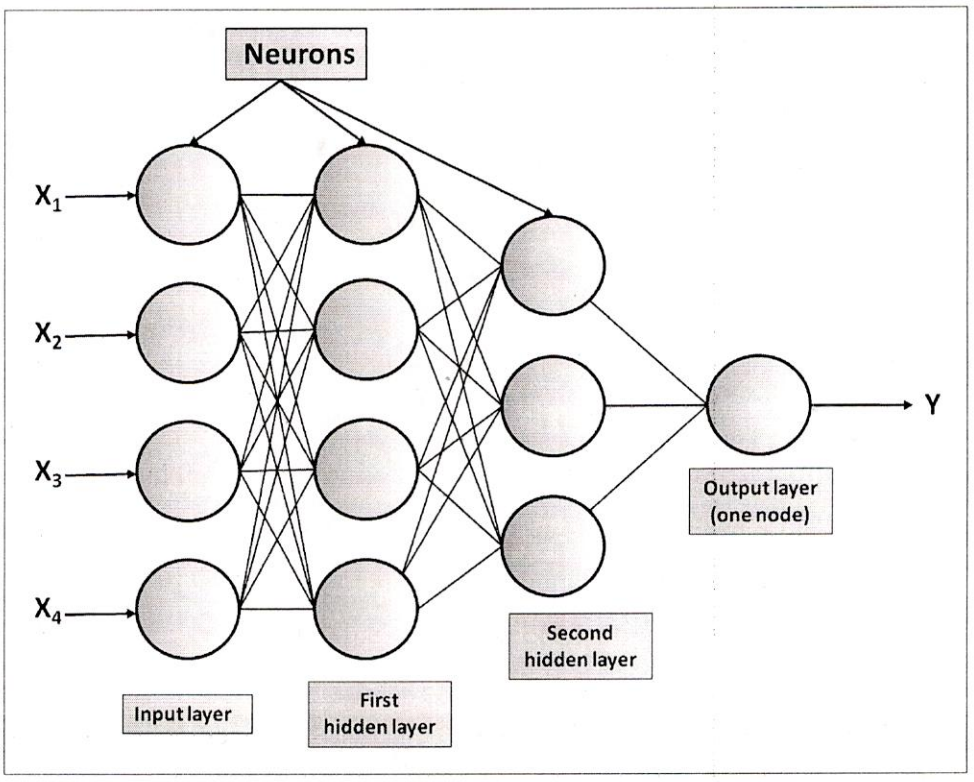

Figure 1.2: An example of a three-layer ANN, showing neurons arranged in layers.

1.3 RADIAL BASIS FUNCTION NETWORK

In this study, Radial Basis Function (RBF) Neural Networks is used to construct a rainfall-runoff model. The RBF network is a variant of the standard feedforward network. It can be considered as a two-layer feedforward artificial neural 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 (Chen et al. 1991).

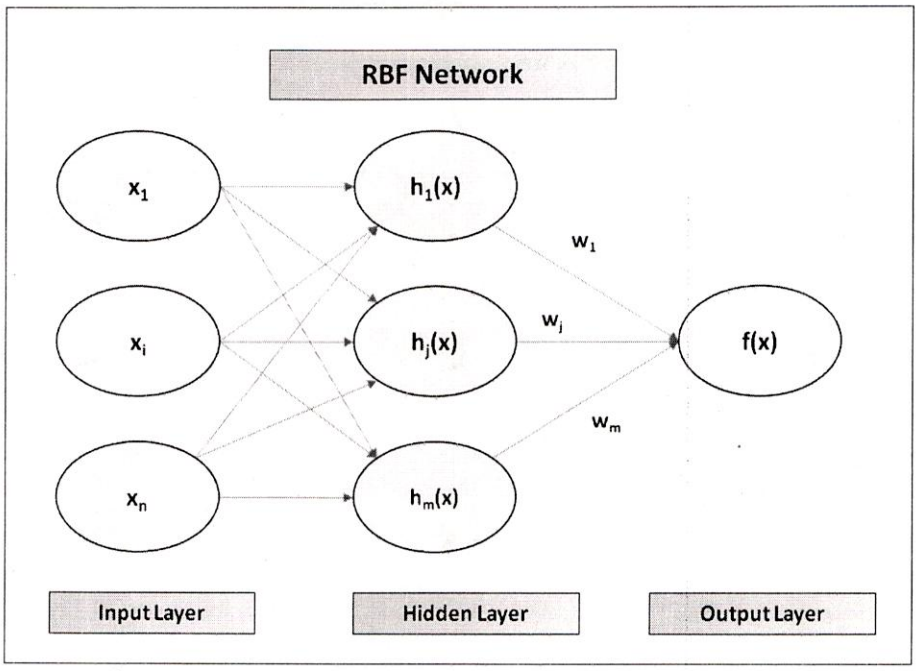

Figure 1.3: Radial basis function network.

Here, $h(x)$ is the Gaussian activation function with the parameters r (the radius or standard deviation) and c (the center or average taken from the input space) defined separately at each RBF unit.

$$h(x) = exp\left(-\frac{(x-c)^2}{r^2}\right) \tag{1.1}$$

and the output layer function is represented by the following equation:

$$f(x) = \sum_{j=1}^{m} w_j h_j(x) \tag{1.2}$$

The learning process is based on adjusting the parameters of the network to reproduce a set of input-output patterns. There are three types of parameters; the weight w between the hidden nodes and the output nodes, the center c of each neuron of the hidden layer and the unit width r.

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 'centres', which can be considered the weight vectors of the hidden neurons. A neuron 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. For example, in one dimension, the Gaussian function is the probability density function of the normal distribution,

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/(2\sigma^2)} \tag{1.3}$$

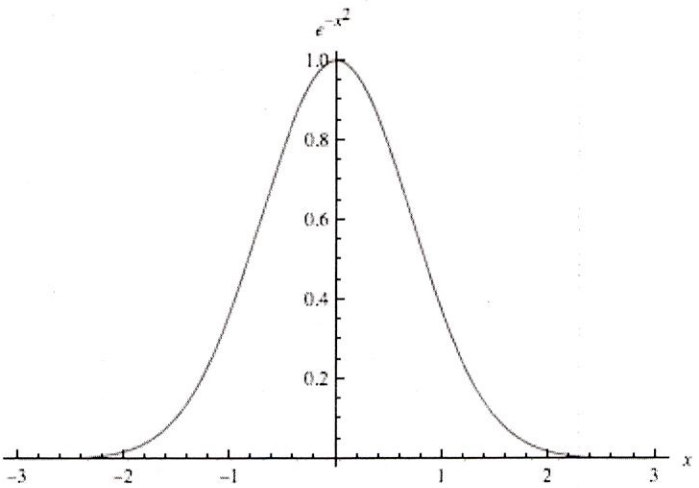

Figure 1.4: Gaussian function curve.

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.

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 centres. It consequently produces an output based on a combination of

activated neurons (i.e. centres that show a likeness) and the weights between these hidden neurons and the output layer.

RBF networks are generally capable of reaching the same performance as feedforward networks while learning faster. On the downside, more data is required to reach the same accuracy as feedforward networks. According to Chen et al. (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.

The primary difference between the RBF network and backpropagation lies in the nature of the nonlinearities associated with hidden neurons. The nonlinearity in backpropagation 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 (Govindaraju 2000). The original RBF method requires that there be as many RBF centres (neurons) as training data points, which is rarely practical, since the number of data points is usually very large (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.

1.4 FUZZY LOGIC

In this thesis, the Fuzzy theory has also been used for rainfall-runoff modeling. Fuzzy logic is a superset of conventional (Boolean) logic that has been extended to handle the concept of partial truth- truth values between "completely true" and "completely false" which provides a convenient framework to map an input domain to output domain. The central concept of Fuzzy set theory is the membership function, which represents numerically the degree to which an element belongs to a set. For example, if an element is a member of a fuzzy set to some degree, the value of its membership function can be between 0 and 1, as determined by eliminating the sharp boundary dividing members of the set from nonmembers (Klir & Foger 1988; Ojha et al. 2007). Fuzzy Logic has to be a useful and practical technique for modeling complex phenomenon that may not yet be fully understood owing to its ability to deal with imprecise, uncertain data, or ambiguous relationships among data sets (Metternicht 2001). Fuzzy Logic theory and Fuzzy set theory provide an excellent means for representing imprecision and uncertainty in the decision-making process and for

defining the reasoning in such processes (Zadeh 1983). The Fuzzy set theory has been used to represent uncertain information in mathematical form (Zadeh 1965) and has also been applied for different purposes in engineering, business, and many other areas.

The fuzzy logic approach has been successfully applied to flood forecasting (Chang et al. 2005); precipitation (Maskey et al. 2004); sediment transport (Tayfur et al. 2003), reservoir operation (Tilmant et al. 2002), and storm water infiltration (Hong et al. 2002).

1.5 OBJECTIVES

Main objective of the present work is to present the development of rainfall-runoff models using different soft computing techniques for predicting runoff at Bhakra located in the Sutlej basin in northern India. The specific objectives are given below:

I. Rainfall-runoff modelling using ANN with backpropagation algorithm, RBF and Fuzzy Logic models.
II. Comparison of results of rainfall-runoff models to determine which model performed better in predicting runoff at Bhakra.

1.6 ORGANIZATION OF WORK

The thesis has been organized in 7 chapters.

Chapter 1 is introduction part of the research work.

Chapter 2 describe literature reviews related to rainfall-runoff modeling, application of ANN's, RBF networks and fuzzy logic in many hydrological processes including rainfall-runoff processes and streamflow prediction.

Chapter 3 describe the study area Sutlej River Basin, topographic information, Sutlej river and its tributaries, various input data collected to model the streamflow at Bhakra using ANN, RBF and fuzzy logic models and presents the detailed procedures followed for the development of rainfall runoff models used for rainfall-runoff modeling and performance evaluation using error statistics for predicting runoff at Bhakra located in the Sutlej basin in northern India.

Chapter 4 represents the result and discussion part, in which the results of rainfall-runoff modeling are discussed in detail. Performance evaluation using error statistics

of results obtained during calibration and validation of the data is also done in this chapter.

Chapter 5 provides the conclusion based on the analysis of rainfall-runoff modeling using ANN, RBF and fuzzy logic models.

CHAPTER 2

LITERATURE REVIEW

2.1 INTRODUCTION

ANNs are a form of computing inspired by the functioning of the brain and nervous system and are discussed in detail in a number of hydrologic papers (Minns & Hall 1996; Dibike & Solomatine 1999; Sajikumar & Thandaveswara 1999; Zealand et al. (1999); ASCE 2000 a,b; Maier & Dandy 2000; Elshorbagy 2000; Rajurkar et al. 2002; Sudheer et al. (2002); Jain & Srinivasulu 2004; Parasuraman et al. (2006); Gill et al. (2007); Srinivasulu & Jain 2009).

The development of artificial neural networks (ANNs) began approximately 50 years ago (McCulloch & Pitts 1943), inspired by a desire to understand the human brain and emulate its functioning. Within the last two decades, it has experienced a huge resurgence due to the development of more sophisticated algorithms and the emergence of powerful computation tools. Extensive research has been devoted to investigate the potential of artificial neural networks (ANNs) as computational tools that acquire, represent, and compute a mapping from one multivariate input space to another (Wasserman 1989). The ability to identify a relationship from given patterns make it possible for ANNs to solve large scale complex problems such as pattern recognition, nonlinear modeling, classification, association, and control.

Since the early nineties, ANNs have been successfully used in hydrology and water resources engineering such as rainfall-runoff modeling, stream flow forecasting, ground-water modeling, water quality, water management policy, precipitation forecasting, hydrologic time series, and reservoir operations. More concepts and application of ANN models in hydrology has been discussed by Govindaraju and Rao (2000) and by the ASCE task committee on application of artificial neural networks in hydrology (2000 a, b).

2.2 LITERATURE ON RAINFALL-RUNOFF MODELING

The relationship of rainfall-runoff is known to be highly non-linear and complex and difficult problem involving many variables, which are interconnected in a very complicated way.

The problem of rainfall-runoff modeling has perhaps received the maximum attention by ANN modelers. This problem lends itself admirably to ANN applications (Hsu et al. 1995). The nonlinear nature of the relationship, availability of long historical records, and the complexity of physically-based models in this regard, are some of the factors that have caused researchers to look at alternative models and ANNs have been a logical choice.

Daniel (1991) introduced the application of ANNs in water resource and hydrologic modelling to the water resource community, he used ANNs to predict monthly water consumption and to estimate flood occurrence.

A number of researchers (Zhu et al. (1994); Dawson & Wilby 1998; Tokar & Johnson 1999; Coulibaly et al. 2000) have investigated the potential of using neural networks in modeling watershed runoff based on rainfall inputs.

In the last decade, ANNs have been successfully employed in modeling a wide range of hydrologic processes, including rainfall–runoff processes. (Smith & Eli 1995; Hsu et al. (1995); Minns & Hall 1996; Shamseldin 1997; Dawson & Wilby 1998; Cigizoglu & Alp 2004) studied on neural-network models of rainfall–runoff process.

Govindaraju (2000) states that a broad classification into two categories of research activities after ANNs in Rainfall –Runoff modelling can be made:

i. The first category of studies are those where ANNs were trained and tested using existing models (e.g., Smith & Eli 1995; Shamseldin 1997). These studies may be viewed as providing a ''proof of concept'' analysis for ANNs. They have laid the foundations for future ANN use by demonstrating that they are indeed capable of replicating model behavior, provided sufficient data is available for training. This requirement is easily met, as the data necessary for training can be generated on a computer relatively easily. In such cases, ANN performance can at best equal the original model that provided the data for training. Such experiments are a first step in evaluating the applicability of ANNs for use in real catchments.
ii. Most ANN-based studies fall into the second category, those that have used observed rainfall-runoff data. Frequently, supplementary inputs such as temperature, snowmelt equivalent, and historical stream flows have been included. In such instances, comparisons with other empirical or conceptual

models have also been provided. These studies provide a more comprehensive evaluation of ANN performance and are capable of establishing ANNs as viable tools for modeling rainfall- runoff. While most studies report that ANNs have resulted in superior performance, they have not been useful for providing any useful insight or furthering our understanding of watershed processes. Using ANNs as a mere black-box to reproduce an input-output sequence well does not help in advancing the scientific understanding of hydrological processes. More creative use of ANNs is modeling the rainfall-runoff process will be needed in the future.

There are many different types of ANN models in practice. Multi-layer feedforward neural networks are perhaps the favorite and perform well in most ANN applications. Maier and Dandy (2000) reported that more than 95% of the ANN related papers they reviewed in the water resources area used feed-forward networks. In forecasting time series, the feed-forward network can be viewed as a general nonlinear auto-regressive model. The linear auto-regressive (AR) models are special cases of ANN without hidden nodes (Zhang et al. 2001).

Rajurkar et al. (2002) studied the application of artificial neural network (ANN) methodology for modelling daily flows during monsoon flood events for a large size catchment of the Narmada River in Madhya Pradesh, India. They found that a linear multiple-input single-output (MISO) model coupled with the ANN provided a better representation of the rainfall-runoff relationship in such large size catchments compared with linear and nonlinear MISO models.

(Jain & Srinivasulu 2004; Rajurkar et al. 2004; De Vos & Rientjes 2005; Ahmad & Simonovic 2005; Tayfur & Singh 2006) found acceptable performance of Artificial Neural Networks (ANNs) in rainfall-runoff modelling.

Solaimani (2009) developed Artificial Neural Network (ANN) to modelling the rainfall-runoff relationship in a catchment area located in a semiarid region of Iran. The applications of the feed forward back propagation for the rainfall forecasting with various algorithms with performance of multilayer perceptions has been illustrated in this study.

2.2.1 Artificial Neural Networks

Determining the relationship between rainfall and runoff for a watershed is one of the most important problems faced by hydrologists and engineers. Information about rainfall and runoff is needed for hydrologic engineering design and management purposes. In addition to rainfall, runoff is dependent on numerous factors such as initial soil moisture, land use, watershed geomorphology, evaporation, infiltration, distribution, duration of the rainfall, and so on. Although many watersheds have been gauged to provide continuous records of stream flow, engineers are often faced with situations where little or no information is available. In such instances, simulation models are often used to generate synthetic flows. A number of researchers have investigated the potential of neural networks in modeling watershed runoff based on rainfall inputs. In a preliminary study, Halff et al. (1993) designed a three-layer feedforward ANN using the observed rainfall hyetographs as inputs and hydrographs recorded by the U.S. Geological Survey (USGS) at Bellvue, Washington, as outputs. The authors decided to use five nodes in the hidden layer. A total of five storm events were considered. On a rotation basis, data from four storms were used for training, while data from the fifth storm were used for testing network performance. A sequence of 25 normalized 5 min rainfalls was applied as inputs to predict the runoff. This study opened up several possibilities for rainfall-runoff application using neural networks.

Bonafe et al. (1994) assessed the performance of a neural network in forecasting daily mean flow from the upper Tiber River basin in central Italy. The previous discharge, daily precipitation, daily mean temperature, total rainfall of the previous five days, and mean temperature over the previous ten days were selected as ANN inputs. They concluded that the ANN was able to yield much better performances than ARMA models.

Back propagation is the most popular algorithm used for the training of the feed forward ANNs (Hsu et al. 1995; Sajikumar & Thandaveswara 1999; Zealand et al. (1999); Thirumalaiah & Deo 2000; ASCE 2000a; Elshorbagy et al. 2000; Maier & Dandy, 2000; Burian et al., 2001; Nagy et al., 2002; Deka & Chandramouli, 2003; Jain & Srinivasulu, 2004; Keskin & Terzi, 2006; Kisi, 2007; Jain, 2008).

Carriere (1996) developed a virtual runoff hydrograph system that employed a recurrent back-propagation artificial neural network to generate runoff hydrographs. A recurrent backpropagation network was utilized, in which input layer feeds back to

itself during training to capture time dependence in the series. The network consisted of 7 input nodes, 35 nodes in hidden layer, and a single node in the output layer. Bipolar linear normalization was used in the input layer, and the logistic function was used for activation in the nodes of the hidden and output layer. Data from 45 laboratory experiments over a small watershed under different conditions of slope and cover were selected to develop the neural network. Out of these, 29 data sets were employed to train the neural network, and the rest were used for testing. The author concluded that the neural network could predict runoff hydrographs accurately, with good agreement between the observed and predicted values.

Dawson and Wilby (1998) used a three-layer back-propagation network to determine runoff over the catchments of the Rivers Amber and Mole. The two catchments are about 140 km2 in size, and are prone to floods. ANN inputs were past flows and averages of past rainfall and flow values. The ANN output consisted of predicting future flows at 15 min intervals up to a lead time of six hours. Their results show that ANNs performed about as well as an existing forecasting system that required more information. When compared with actual flows, the ANNs appeared to overestimate low flows for the Mole River.

Tokar and Johnson (1999) reported that ANN models provided higher training and testing accuracy when compared with regression and simple conceptual models. Their goal was to forecast daily runoff for the Little Patuxent River, Maryland, with daily precipitation, temperature, and snowmelt equivalent serving as inputs. It was found that the selection of training data has a large impact on accuracy of prediction. The authors trained and tested the ANN with wet, dry, and average-year data, respectively, as well as combinations of these, in order to illustrate the impact of the training series on network performance. The ANN that was trained on wet and dry data had the highest prediction accuracy. The length of training record had a much smaller impact on network performance than the types of training data.

Campolo et al., (2003) forecasted flood in Arno River by using feed forward neural network approach with standard back propagation training algorithm. They used the information of rainfall, hydrometric data and dam operation at the basin, Italy, to predict the hourly water level variations. They used two years data with some special treatment with as inputs .

2.2.2 Radial Basis Function Networks

Haykin (1994) showed that design of a supervised neural network might be pursued in a number of different ways. While the back-propagation algorithm for the design of a multilayer perceptron (under supervision) may be viewed as an application of stochastic approximation, radial-basis function (RBF) networks can be viewed as a curve-fitting problem in a high-dimensional space. Therefore, the learning for such networks is equivalent to finding a surface in a multidimensional space that provides a best fit to the training data, with the criterion for ''best fit'' being expressed in a statistical sense.

Mason et al. (1996) used RBF networks for accelerating the training procedure as compared with regular back-propagation techniques. Data were generated using the Simulation Program for Interactive Drainage Analysis (SPIDA) model. The network output was runoff based on inputs consisting of time, rainfall intensity, cumulative rainfall, and derivative of rainfall intensity. The authors briefly discuss network architectures and compositions and tried five different forms of basis functions in their study. Sixty data sets were utilized for network training, and 39 were used for validation of the model. The authors concluded that, while RBF networks did provide for faster training, such networks require the solution of a linear system of equations that may become ill conditioned, especially if a large number of cluster centers are chosen.

Fernando and Jayawardena (1998) studied on runoff forecasting using RBF networks with OLS algorithm. In this study Radial Basis Function (RBF) neural network is used to construct a rainfall- runoff model. To do the training, Matlab 6.5.1 computer software was used. Further the result is compared to that obtained by rainfall- runoff model designed with multilayer percepteron model .

Kumar et al. (2004) have studied the performance of MLP and RBF type neural network models developed for rainfall-runoff modelling of two Indian River basins.

Kumar et al. (2005) developed MLP and RBF type neural network models for rainfall-runoff modelling of two Indian River basins. The performance of both the MLP and RBF network models were comprehensively evaluated in terms of their generalization properties, predicted hydrograph characteristics, and predictive uncertainty. Merits and limitations of networks of both models were discussed.

Senthil Kumar et al. (2005) studied modeling of Suspended Sediment Concentration at Kasol in India using ANN, Fuzzy Logic, and Decision Tree Algorithms. The focus of this paper was to present the development of models using Artificial Neural Network (ANN) with back propagation and Levenberg-Maquardt algorithms, radial basis function (RBF), Fuzzy Logic, and decision tree algorithms such as M5 and REPTree for predicting the suspended sediment concentration at Kasol, upstream of the Bhakra reservoir, located in the Sutlej basin in northern India. It was found that the M5 model performed well compared to other soft computing techniques such as ANN, fuzzy logic, radial basis function, and REPTree investigated in this study, and results of the M5 model indicate that all ranges of sediment concentration values were simulated fairly well. This study also suggests that M5 model trees, which are analogous to piecewise linear functions, have certain advantages over other soft computing techniques because they offer more insight into the generated model, are acceptable to decision makers, and always converge. Further, the M5 model tree offers explicit expressions for use by field engineers.

Fernando and Shamseldin (2009) applied radial basis function neural network for one day ahead flow forecasting. Two RBF networks were trained using daily flow data of two different rivers from different part of the world having different characteristics (i.e. Blue Nile River from Sudan and Brosna River from Ireland). Eight years data were divided into two parts in a ratio 50%, four year for training and testing each. Autocorrelation analysis was examined to select appropriate number of inputs. Present day discharge with two antecedent discharge values were selected to forecast one day ahead discharge in both RBF model architectures. The effect of radial basis functions or hidden neurons in both the models was also investigated. Conjugate gradient descent algorithm was employed to minimize the network error in order to choose the RBF centers, spreads and weights between hidden and output layers. From the inspection of the effect of hidden nodes on outputs, the authors examined that results with hidden node 1 is more dominant at low flow range, hidden node 2 showed dominancy at medium/high flow range and from hidden node 3, the very high zone flow was covered. From this observation, they suggested that RBF model have ability to analytically crumble the flow hydrograph into a number of consequential flow elements in the catchment. Since the authors obtained successful forecasting results of river flow with different flow characteristics, in the mean while they also suggested

that RBF network is not completely slanted, but it produce important information about the natural scenario.

Suhaimi et al. (2009) studied on Rainfall – runoff modelling using Radial Basis Function Neural Network for Sungai Tinjar Catchment, Miri, Sarawak. The RBF network developed in this study has successfully modelled rainfall runoff relationship in Tinjar Catchment, Miri, Sarawak with an accuracy of about 98.3% .

Kagoda et al. (2010) used radial basis function type of neural network for one day ahead forecasting short-term stream flow. Application of RBF neural network for three locations at the Luvuvhu River in South Africa was demonstrated for forecasting stream flows. Daily data of rainfall and stream flow with antecedent conditions were used in the input layer to forecast one day ahead stream flow. Gaussian radial basis function was used during training RBF model. The network training consisted on two stages (i) contain the calibration of Gaussian function parameters and (ii) include the calculation of connection weights. The authors used Self-Organizing Feature Map (SOFM) technique to determine the Gaussian function parameters. While, for calibration of connection weights, Shuffled Complex Algorithm Evolution (SCE-UA) was used. The Performance of the models was evaluated using Nash-Sutcliffe efficiency and root mean square error as statistical measures. Satisfactory results were found at two locations where sufficient data was available, whereas at third location where data was not enough for network training, poor results were observed. Thus, the authors suggested that a good enough length of data is necessary to get satisfactory results from ANN modeling. However, the authors proposed on basis of obtained results that artificial neural networks is promising for forecasting stream flow in South Africa.

Vivekanandan (2014) studied prediction of Rainfall using MLP and RBF networks. This paper illustrates the use of ANN for prediction of rainfall at Atner, Multai and Dharni stations. Multi-Layer Perceptron (MLP) and Radial Basis Function (RBF) networks are applied to train the network data. Model performance indicators such as correlation coefficient, model efficiency and root mean square error are used to evaluate the performance of the MLP and RBF networks. The paper presents the MLP network is better suited for prediction of rainfall for Atner and Multai whereas RBF network for Dharni.

2.2.3 Fuzzy Logic

Fuzzy logic is a mathematical system in which rigorous logical mathematics is used to deal with fuzzy information and data that are difficult to compute using conventional mathematics (Zadeh 1965).

The fuzzy logic approach has also been applied to : rainfall–runoff processes (Abebe et al. 2000; Hundecha *et al.* 2001; Jacquin & Shamseldin 2006); river flow routing (See & Openshaw 2000; Chang & chen 2001); groundwater modelling (Hong *et al.* 2002); water-level prediction in reservoirs (Chang & chen 2006); and time series modelling (Nayak *et al.* 2004). Deka and Chandramouli (2005) proposed a new approach combining FL and ANNs, which is referred to as fuzzy neural networks (FNN), for river flow prediction.

Fuzzy logic based modeling methodologies can be classified into two types (Sugeno & Kang 1988; Pedrycz & Gomide 1994):

i. The first type is used for developing purely linguistic models, based on conventional fuzzy implication and reasoning. The system behavior is described by means of fuzzy relational equations. A typical linguistic model is expressed as `IF x is A THEN y is B', where A and B are fuzzy sets. In this type of model, the fuzzy sets are determined subjectively, on the basis of experience. Quantitative information is seldom directly used for the determination of the model structure and parameters.

ii. Takagi and Sugeno (1985) pioneered the second type of modeling. Their model, referred to as ' the T-S model ' in this text, consists of a number of fuzzy implications (FIs). Each FI is composed of a set of premises in the IF part and a set of consequences in the THEN part. The IF part provides a logic-based guidance to the use of regression models in the THEN part. The T-S model structure has attracted much attention in recent years (Terano et al. 1987; Tan et al. 1995). It has been found, however, that the model-development process is computationally intensive. The model finally derived may be structurally more complicated than is necessary.

Takagi and Sugeno first introduced their method of fuzzy inference in 1985. In recent years the Takagi– Sugeno fuzzy method (TS) has been used extensively in hydrology and has given many satisfactory results (Vernieuwea et al. 2005, Jacquin &

Shamseldin 2006, Lohani et al. 2006). The TS fuzzy system conception was introduced into a combination method by Fiordaliso (1998).

Mamdani (1974) introduced the Mamdani approach as a type of FL modelling by showing its application for simple dynamic plant.

Yu and Yang (2000) presented a fuzzy multi objective function (FMOF) to improve the performance of conventional objective functions of root-mean square error (RMSE) and mean percentage error (MPE) that are used in calibrating rainfall-runoff conceptual models. Using daily rainfall and flow discharge measurements as well as monthly evaporation estimates for calibrating and verifying the rainfall-runoff model, they showed that the FMOF led to improved simulation of a wide range of flow stages as it was capable of combining various objective functions with different acceptable levels.

Ozelkan and Duckstein (2001) proposed a fuzzy conceptual rainfall-runoff framework to deal with parameter uncertainties of conceptual rainfall-runoff models. They concluded that the fuzzy logic framework enabled a decision-maker to gain insight into the model sensitivity and the uncertainty stemming from the elements of the conceptual rainfall-runoff model.

Yeshewatesfa et al. (2001) applied FL model for rainfall streamflow modeling. The past decade has witnessed a applications of fuzzy logic approach in water resources (Nayak, *et al.* 2005). Nayak et al. (2005) used Mamdani approach (Mamdani and Assilian 1975), which has been used in some hydrological applications for rainfall streamflow modeling.

Alvisi et al. (2006) stated that the FL models have a limited capacity for dealing with too detailed information compared to ANN models and this result is in line with other hydrological studies based on the fuzzy rules system that are generally characterized by a low number (from 2 to 5) of input variables.

Recently, FL has been used in the field of hydrology and water resources. Bárdossy (2006) proposed a fuzzy unit hydrograph to account for the number of uncertainties raised from both model assumptions and data acquisition in representing the rainfall–runoff transformation.

Tayfur and Singh (2006) used ANN and fuzzy logic models for simulating event-based rainfall–streamflow.

Firat and Güngör (2007a,b) used streamflow records of two stations in the Great Menderes basin for adaptive estimation by using the Takagi-Sugeno (TS) approach (Takagi & Sugeno 1985). They concluded that the TS-type fuzzy model has better skill in estimating the current value from two antecedent values, compared to ANNs.

Mukarji et al. (2009) apply the ANN, *adaptive neuro-fuzzy inference system* ANFIS and ANFGI mode to forecast stramflow for Ajay River Basin in Jharkhand, India and results observed that ANFIS model predicts better than the ANN model in most of the cases.

Gowda and Mayya (2014) apply fuzzy logic model for predicting streamflow for Nethravathi River basin is located in Dakshina Kannad applying different membership functions and results found that, fuzzy inference system using triangular membership function show a good performance compared to other models developed.

2.3 CONCLUSION

The different types of approaches to the rainfall-runoff modeling have been reported. Taking the old approach in to consideration the development of new approaches conceptualized through these literatures. This study employs Fuzzy logic, ANN and RBF to model the streamflow at Bhakra. In all the models the input vector of the algorithm consists of mean daily rainfall, discharge and evaporation.

CHAPTER 3

MATERIAL AND METHODS

3.1 STUDY AREA

3.1.1 The Sutlej river basin

The Sutlej River is one of the main tributaries of the Indus River System and is located in the western Himalayan Region. The Sutlej River originates from Mansarowar Lake in Tibet at an elevation of about 4572 m and is a major tributary of the River Indus. Sutlej plays a key role in the economy of northern India where two out of three persons depend upon agriculture and allied activities for their livelihood. The entire Sutlej basin lies between latitudes 30°N and 33°N and longitudes 76°E and 83°E. The Sutlej River enters India near Shipkila at an elevation of about 2530 m or 6,608 meters and continues to flow in Himachal Pradesh through Wangtoo and Kian before reaching bhakra reservoir, where the India's highest gravity dam has been constructed. The total length of the river is 1,448 km. The total drainage area of the Sutlej River up to Bhakra Reservoir is about 56,000 km^2. The major part of the basin (35,725 km^2) lies in the Tibetan plateau experiences little precipitation and has cold desert type of climate. The Indian part of Sutlej basin upstream of Bhakra dam is about 20,275 km^2. The principal tributary of Sutlej River, known as Spiti, joins the Sutlej River just after entering India and contributes about two thirds of the total flow at Khab (the confluence of the Sutlej and Spiti). Figure 3.1 shows location of Sutlej basin in India. The present study has been carried out in a part of Sutlej River basin that is confined in the hilly State of Himachal Pradesh, India. The State shares its boundary with four Indian States namely, Jammu and Kashmir from North, Punjab from West, Haryana. from South, Uttarakhand from South-East and has international border with China (Tibet). Figure 3.2 shows Sutlej river basin location in the State of Himachal Pradesh, India. The Digital Elevation Model (DEM) of the Sutlej river basin provide clearer understanding of its topography (Figure 3.4).

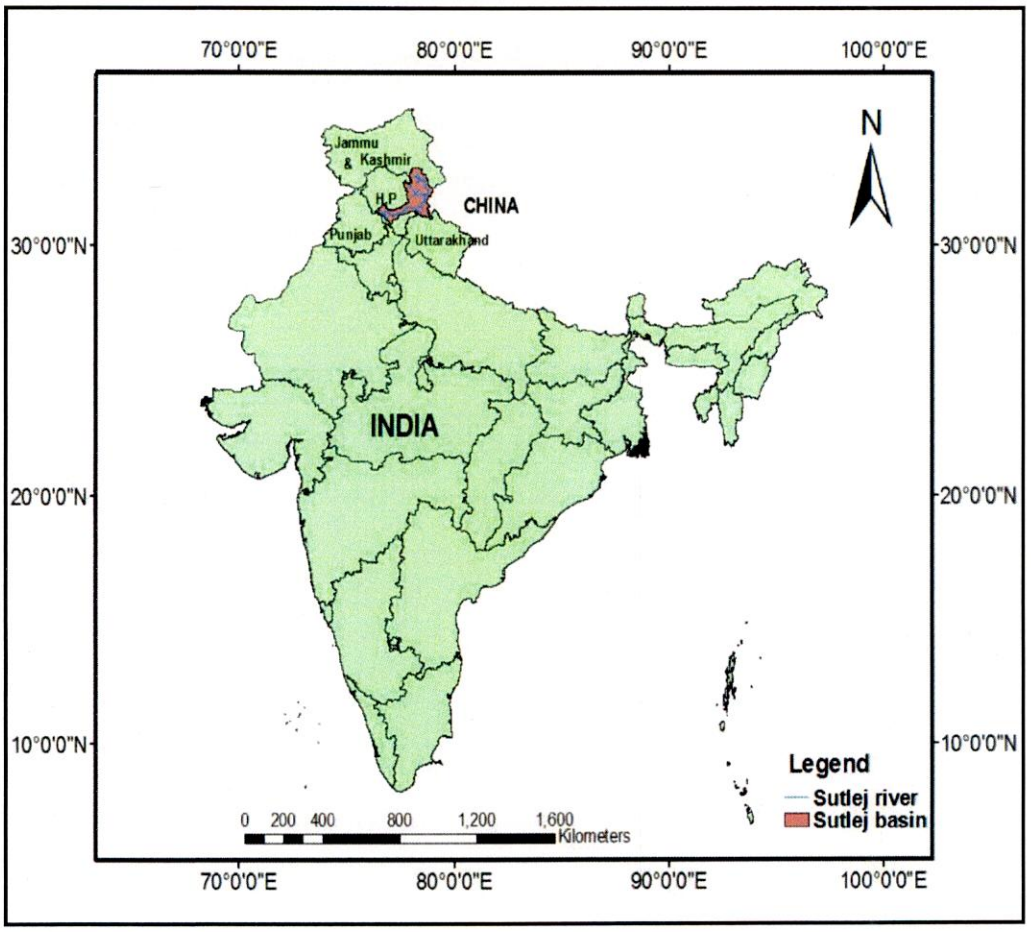

Figure 3.1: Index map of Sutlej basin

The State shares its boundary with four Indian States namely, Jammu and Kashmir from North, Punjab from West, Haryana. from South, Uttarakhand from South-East and has international border with China (Tibet). This basin lies between 30°51′23″N and 33°6′30″N latitudes and 76°26′11″E and 78°59′32″E longitudes as shown in figure 3.3. The elevation of the upper basin varies from about 500 m at Bhakra dam to 7000 m. However, only very small area exists above 6000 m. The mean elevation of the basin is about 3600 m. Although, the basin covers outer, middle and greater Himalayan ranges, the major part of basin lies in the greater Himalayan ranges. Owing to large differences in the relief, the basin is characterized by the diversified climatic patterns. Westerly weather disturbances produce most of the precipitation during winter in the middle and upper parts of the upper basin. Winter precipitation in the upper basin falls mostly as snow. The mean annual rainfall (excluding snow) in the outer, middle and outer Himalayan ranges of the basin is about 1300, 700 and 200 mm, respectively (Singh & Kumar 1997b). The distribution of rainfall indicates that the rainfall is mostly concentrated in the lower part of the basin and has little

influence in the greater Himalayan range. The snowline is highly variable, descending to an elevation of about 2000 m during winter and retreats to above 4500 m after the ablation period. About 65% of the upper basin area is covered with snow during winters (Singh & Jain 2002). The Bhakra Beas Management Board (BBMB) is responsible for collection of hydrometeorological data (snowfall, rainfall, temperature, discharge) for the Sutlej and Beas basins.

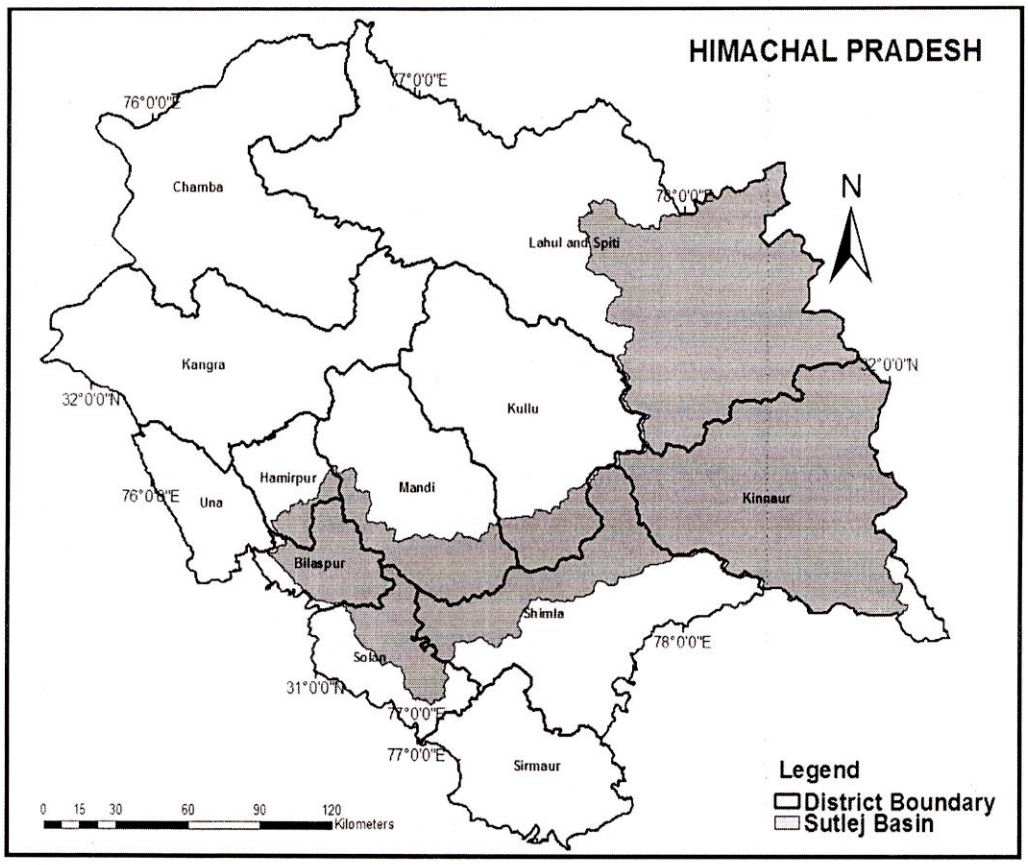

Figure 3.2: Location map of Sutlej basin in Himachal Pradesh

3.1.1.1 Sutlej river and its tributaries

The Sutlej river is the longest and largest river among the five rivers of Himachal Pradesh. It enters Himachal at Shipki (altitude = 6,608 metres) and flows in the South-Westerly direction through Kinnaur, Shimla, Kullu, Solan, Mandi and Bilaspur districts. Its corse in Himachal Pradesh is 320 km from Rakastal, with famous tributaries viz. the Spiti, the Ropa, the Taiti, the Kashang, the Mulgaon, the Yula, the Wanger, the Throng and the Rupi as right bank tributaries, whereas the Tirung, the Gayathing, the Baspa, the Duling and the Soldang are left bank tributaries. The Satluj finally drains into the Indus in Pakistan.

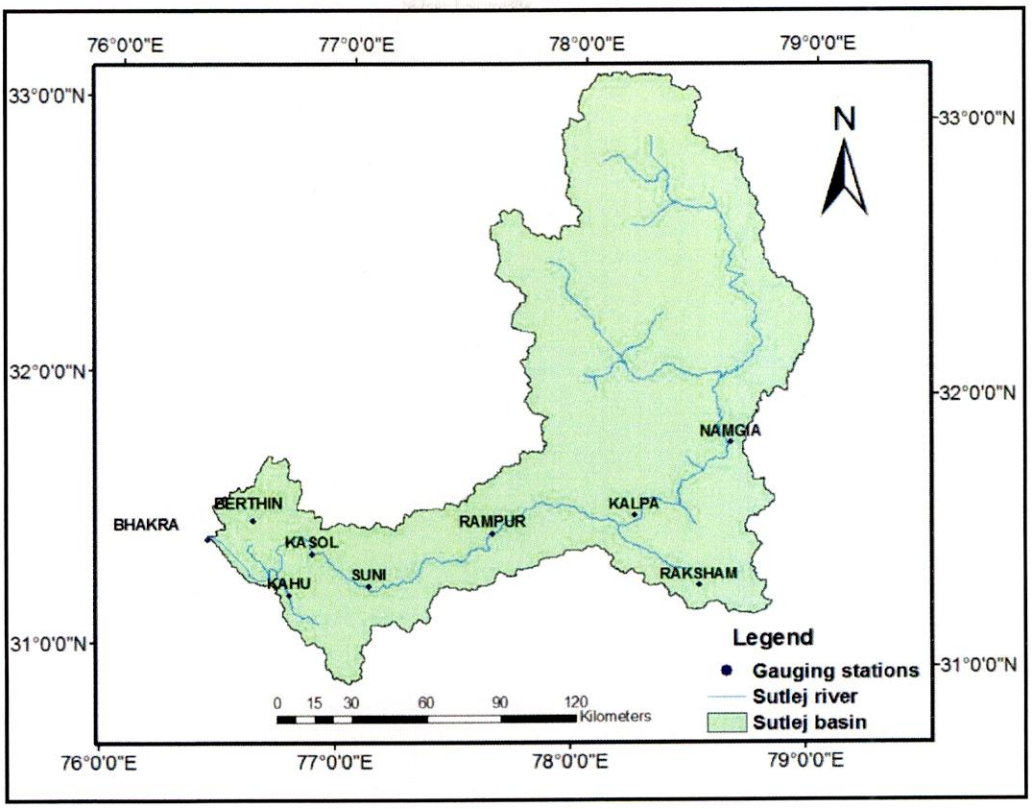

Figure 3.3: Map of Sutlej basin showing Sutlej river system and location of rainfall-discharge stations

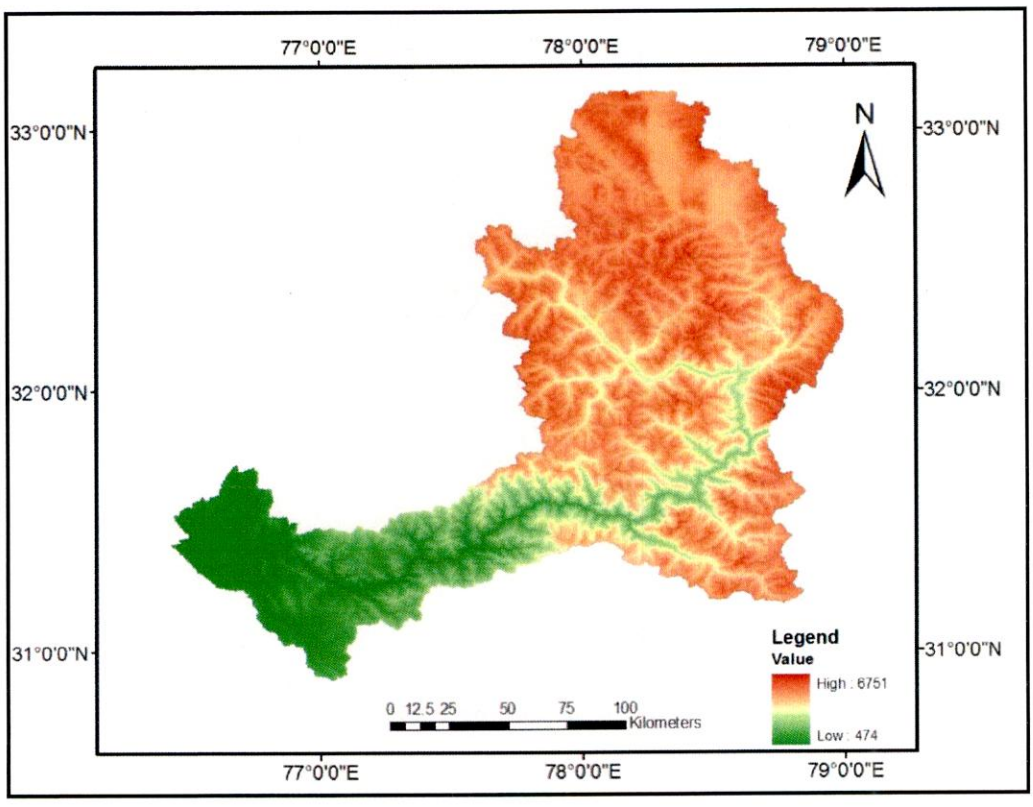

Figure 3.4: Digital elevation model for Sutlej river basin

3.1.1.2 Geology of Sutlej river

The Satluj, together with all the rivers in Punjab, is considered to have sapped east into the Ganges before 5 Mya. There is considerable geographical proof to show that before 1700 BC at the most recent, Sutlej was a major tributary of the Ghaggar-Hakra River (probably through the Saraswati River) instead of the Indus with different writers mentioning the channelization from 2500-2000 BC or 5000-3000 BC. Geological scientists assume that tectonic movement resulted in altitude variations, which rerouted the discharge of Sutlej from the southeast to the southwest. Subsequently, the potent Saraswati started to desiccate, resulting in transformation of Cholistan and the eastern portion of the present state of Sindh into desert. The desertification led to desertion of many prehistoric human colonies beside the riverbanks of Saraswati.

There is certain proof that the escalating rate of wearing down created by the present Sutlej River has regulated the cracks in restricted parts and speedily unearthed stones over Rampur. This will be comparable to, but on a much lower extent than the digging up of rocks by the Indus River in Nanga Parbat, Pakistan. In addition, the Sutlej River also exhibits a twofold reversed metamorphic slope.

3.1.2 DATA COLLECTION AND ANALYSIS

The basic data required for the rainfall-runoff modeling are rainfall information, time series of discharge data and evaporation data of the study area. The long-term rainfall, discharge and evaporation data used in the present study has been collected and supplied by Bhakra Beas Management Board (BBMB) for eight gauging sites; Berthin, Bhakra, Kahu, Kasol, Namgia, Raksham, Rampur, Suni. Daily rainfall data for sixteen years starting from 1, January, 1988 to 31, December, 2004 have been collected for eight raingauge stations whereas the discharge for the same period were collected from four discharge measuring stations and evaporation was collected for one measuring station. These data sets were analyzed and transformed for proper use as input to the models. The locations of given stations have been shown in Figure 3.3 earlier while the details of the data have been described in Table 3.1. Figure 3.5 illustrates the evaporation data, figure 3.6 illustrates the rainfall data and Figure 3.7 discharge data for the study period.

Table 3.1: Location details of the stations considered for the study in Sutlej basin.

S.No.	Station	Height above M.S.L. (in feet)	Latitude	Longitude	Data Availability
1.	Bhakra	1700	31°24'32"N	76°26'28"E	**1988 – 2004**
2.	Berthin	2155	31°28'15"N	76°37'20"E	**1988 – 2004**
3.	Kasol	2170	31°21'25"N	76°52'42"E	**1988 – 2004**
4.	Kahu	2130	31°12'13"N	76°47'15"E	**1988 – 2004**
5.	Suni	2150	31°14'15"N	77°06'30"E	**1988 – 2004**
6.	Rampur	3200	31°27'15"N	77°38'40"E	**1988 – 2004**
7.	Raksham	10269	31°17'48"N	78°32'10"E	**1988 – 2004**
8.	Namgia	9547	31°48'36"N	78°39'23"E	**1988 – 2004**
NOTE: M.S.L. = **Mean Sea Level**					

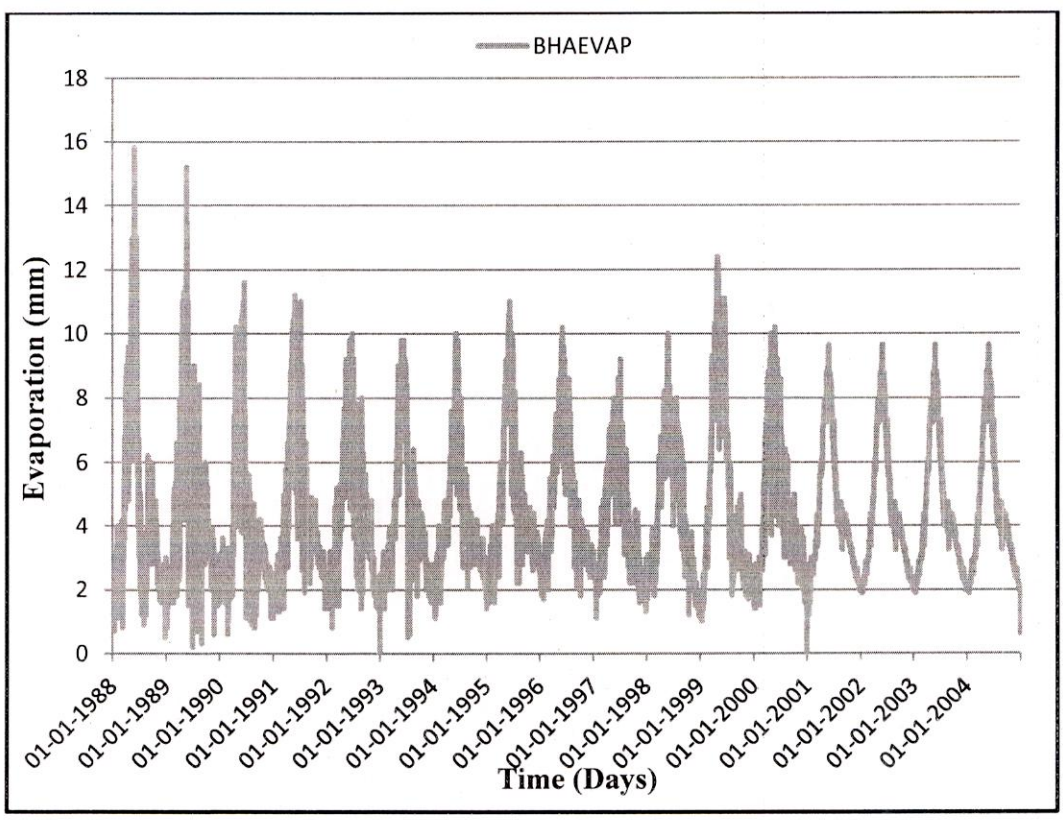

Figure 3.5: Observed daily evaporation data at Bhakra station of Sutlej basin

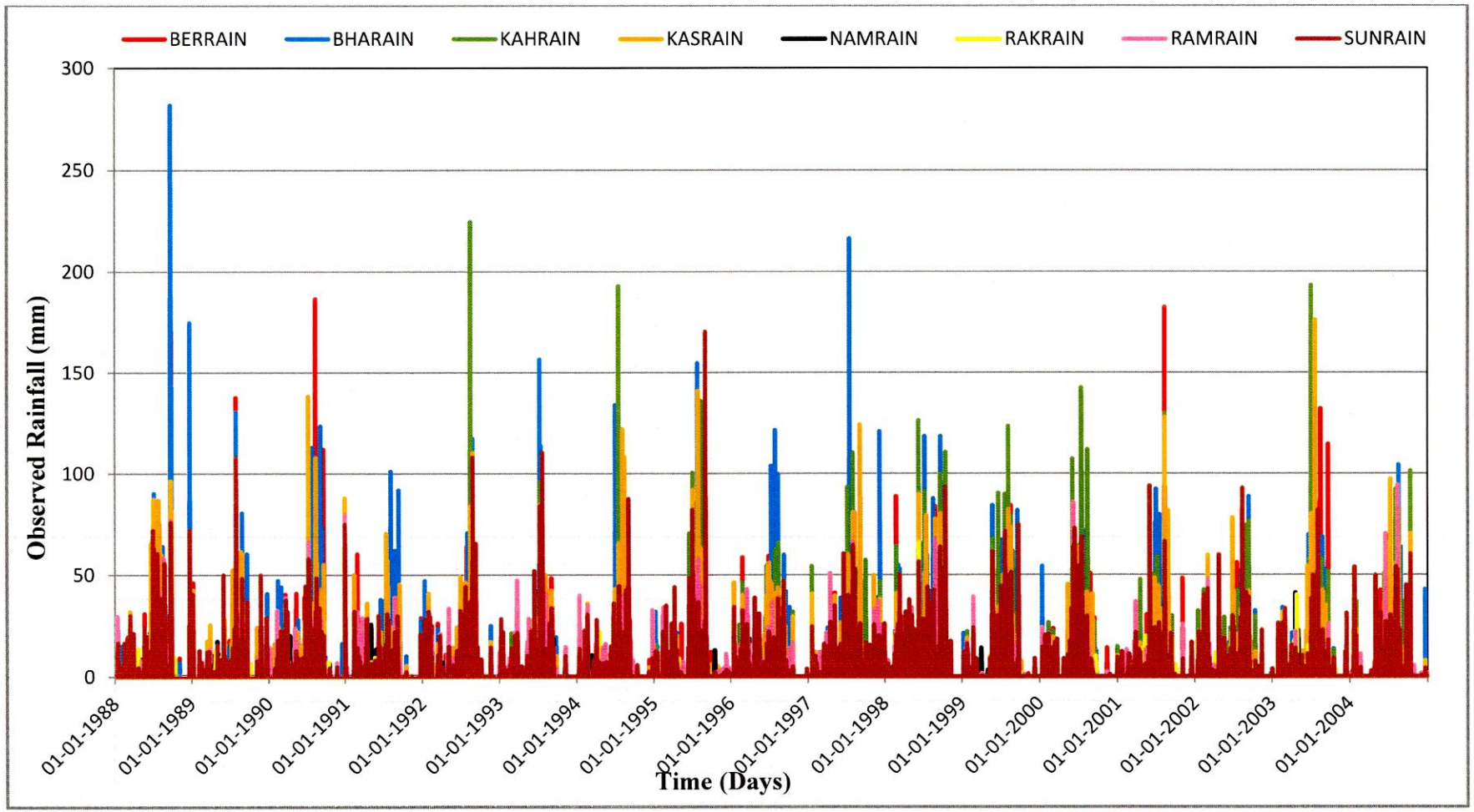

Figure 3.6: Observed daily rainfall data at nine stations of Sutlej basin

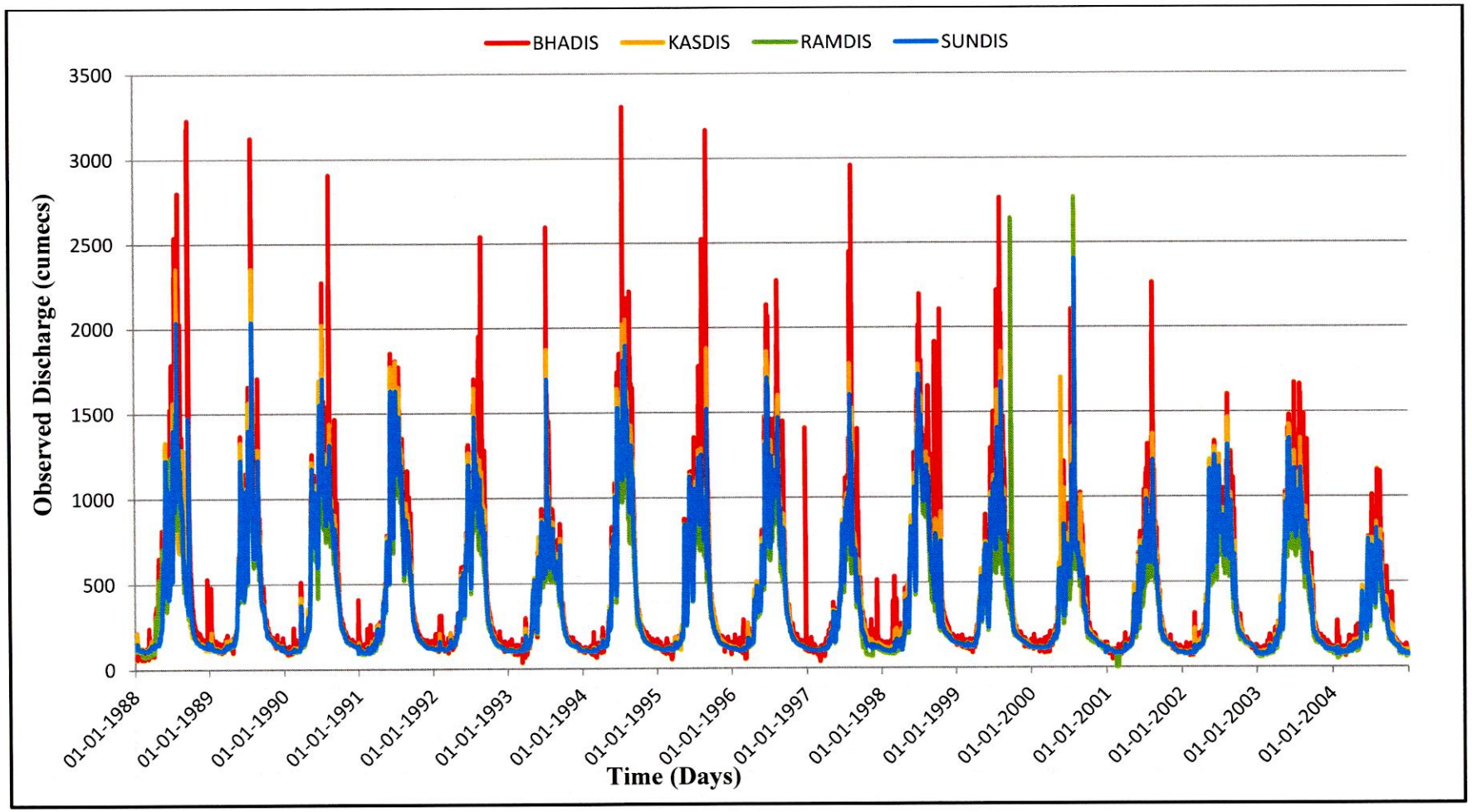

Figure 3.7: Daily discharge data at three stations of Sutlej basin

3.2 METHODOLOGIES

3.2.1 Framework for ANNs

ANNs are the best-known examples of information processing structures that have been conceived in the field of Neurocomputing. Neurocomputing is the technological discipline concerned with information processing systems that autonomously develop operational capabilities in adaptive response to an information environment (Hecht-Nielsen 1990). Neurocomputing is also known as parallel distributed processing.

In other words, ANNs are models that use dense interconnection of simple computational elements in combination with specific algorithms to make their structure (and therefore their response) adapt to information that is presented to them.

Hecht-Nielsen (1990) proposed the following formal definition of an ANN:

> *"A neural network is a parallel, distributed information processing structure consisting of processing elements (which can possess a local memory and can carry out localized information processing operations) interconnected via unidirectional signal channels called branches ('fans out') into as many collateral connections as desired; each carries the same signal – the processing element output signal. The processing element output signal can be of any mathematical type desired. The information processing that goes on within each processing element can be defined arbitrarily with the restriction that it must be completely local; that is, it must depend only on the current values of the input signals arriving at the processing element via impinging connections and on values stored in the processing element's local memory."*

From a mathematical point of view, ANNs can be called universal approximators, because they are often able to uncover and approximate relationships in different types of data. Even though an underlying process may be complex, an ANN can approximate it closely, provided that sufficient and appropriate data about the process is available to which the model can adapt.

Let us assume a set of processing elements (neurons); at each point in time, each neuron u_i has an activation value, denoted in the diagram as $a_i(t)$; this activation value is passed trough a function f_i to produce an output value $o_i(t)$. This output value can

be seen as passing through a set of unidirectional connections to other neurons in the system. What is associated with each connection is a real number – usually called the weight of the connection, designated w_{ij} - which determines the amount of effect that the first neuron has on the second. All of the inputs must then be combined by some operator (usually addition), after which the combined inputs to a neuron, along with its current activation value, determine its new activation value via a function F_i. Finally, the weights of these systems can undergo modification as a function of experience. This is the way the system can adapt its behavior, aiming for a better performance.

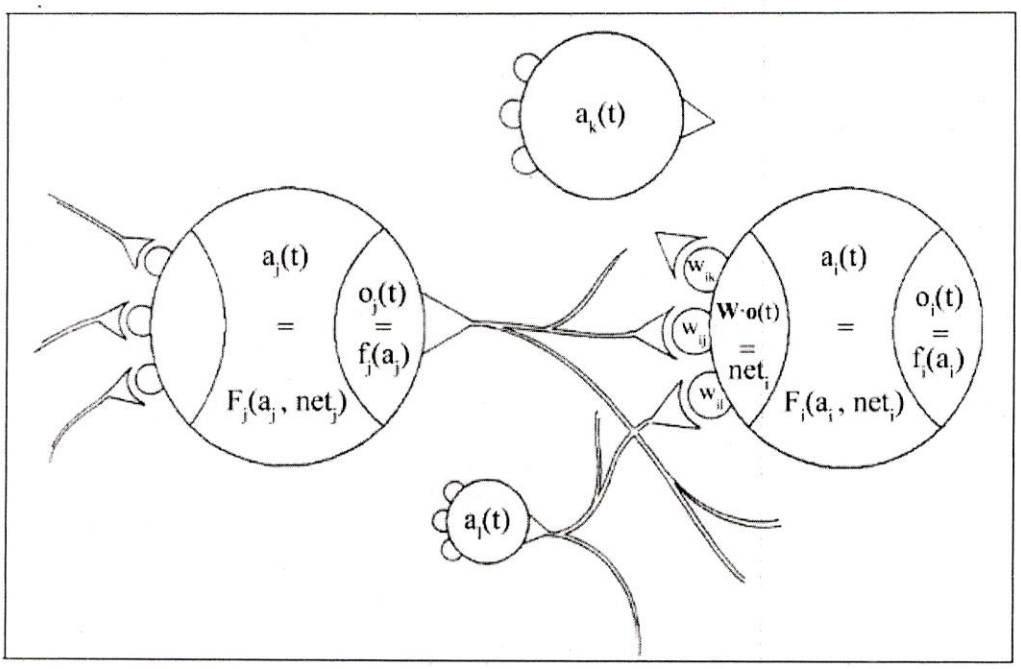

Figure 3.8: Schematic representation of two artificial neurons and their internal processes (Rumelhart et al. 1986)

Characteristics and examples of the above mentioned components of ANNs will be presented in the following subsections in more detail. The basic structure of these sections is also based on the work of Rumelhart et al. (1986).

3.2.1.1 Neurons and layers

Neurons are the relatively simple computational elements that are the basic building blocks for ANNs. Neurons can also be referred to as processing elements or nodes. They are typically arranged in layers (see Figure 1.2). By convention the inputs that receive the data are called the input units, and the layer that transmits data out of the ANN is called the output layer. Internal layers, where intermediate internal processing

takes place, are traditionally called hidden layers (Dhar & Stein 1997). There are as many input units and output neurons as there are input and output variables respectively. Hidden layers can contain any number of neurons. Not all networks have hidden layers.

Neurons are usually indicated by circles in diagrams, and connections between neurons by lines or arrows. Input units will be depicted as squares or small circles to make a clear differentiation between these units and hidden or output neurons.

3.2.1.2 State of activation

The state of the system at a certain point in time is represented by the state of activation of the neurons of a network. If we let N be the number of neurons, the state of a system can be represented by a vector of N real numbers, $\mathbf{a}(t)$, which specifies the state of activation of the neurons in a network.

Depending on the ANN model, activation values may be of any mathematical type (integer, real number, complex number, Boolean, et cetera). Continuous activation types may be bounded within a certain interval.

3.2.1.3 Output of the neurons

Neurons interact by transmitting signals to their neighbors. The strength of their signals is determined by their degree of activation. Each neuron has an output function that maps the current state of activation to an output signal:

$$o_i(t) = f(a_i(t)) \qquad (3.1)$$

This output function is often either the identity function $f(x) = x$ (so that the current activation value is passed on to other neurons), or some sort of threshold function (so that a neuron has no effect on other neurons unless its activation exceeds a certain value). The set of current output values is represented by a vector $o(t)$.

3.2.1.4 Pattern of connectivity

Neurons are connected to one another. Basically, it is this pattern of connectivity that determines how a network will respond to an arbitrary input.

The connections between neurons vary in strength. In many cases we assume that the inputs from all of the incoming neurons are simply multiplied by a weight and summed to get the overall input to that neuron. In this case the total pattern of

connectivity can be expressed by specifying each of the weights in the system. It is not necessary for a neuron to be connected to all neurons in the following layer. Therefore, zero values for these weights can occur.

It is often convenient to use a matrix **W** for expressing all weights in the system, as the figure 3.9 shows. Weight w_{21} is the weight by which the output of the first node in a layer is multiplied with when it is transmitted to the second node in the successive layer.

Sometimes a more complex pattern of connectivity is required. A given neuron may receive inputs of different kinds whose effects are separately summated. In such cases it is convenient to have separate connectivity matrices for each kind of connection.

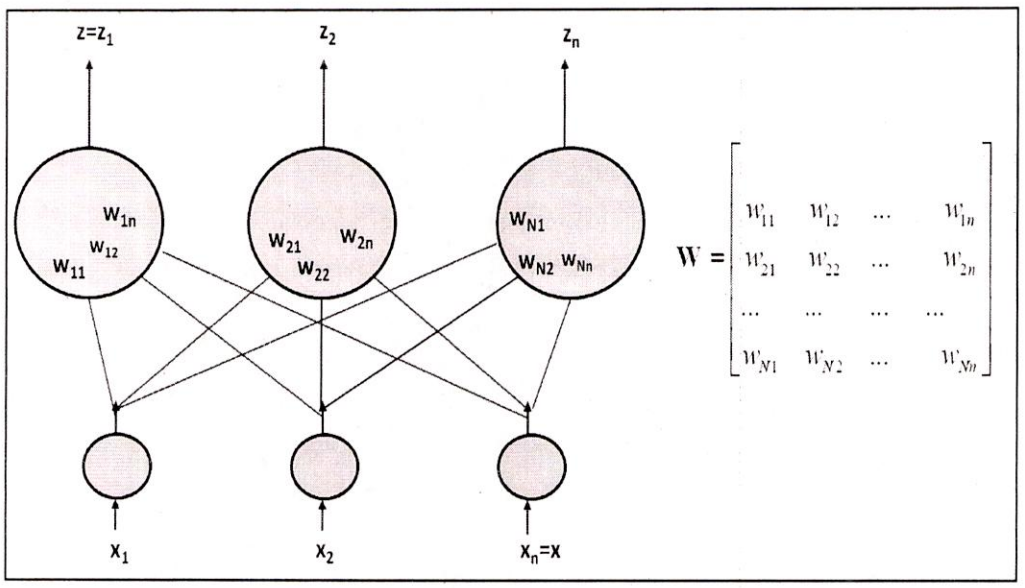

Figure 3.9: Illustration of network weights and the accompanying weight matrix

Connections between neurons are often classified by their direction in the network architecture:

i. Feedforward connections are connections between neurons in consecutive layers. They are directed from input to output.
ii. Lateral connections are connections between neurons in the same layer.
iii. Recurrent connections are connections to a neuron in a previous layer. They are directed from output to input.

3.2.1.5 Propagation rule

The propagation rule of a network describes the way the so-called net input of a neuron is calculated from several outputs of neighboring neurons. Typically, this net input is the weighted sum of the inputs to the neuron, i.e. the output of the previous nodes multiplied with the weights in the weight matrix:

$$net(t) = \mathbf{W}.\mathbf{o}(t) \tag{3.2}$$

3.2.1.6 Activation rule

The activation rule (often called transfer function) determines the new activation value of a neuron based on the net input (and sometimes the previous activation value, in case a memory is used). The function **F** , which takes $\mathbf{a}(t)$ and the vectors **net** for each different type of connection, produces a new state of activation.

F can vary from a simple identity function, so that $\mathbf{a}(t+1) = \mathbf{net}(t) = \mathbf{W}.\mathbf{o}(t)$, to variations of linear and even non-linear functions like sigmoid functions. Transfer function calculate a layer's output from its net input. In the present study, log-sigmoid transfer function is used to perform calculations in the hidden layer and linear transfer function is used to generate the output. Graph and symbol of log-sigmoid transfer function and linear transfer function are shown in figure 3.10 and 3.11 respectively.

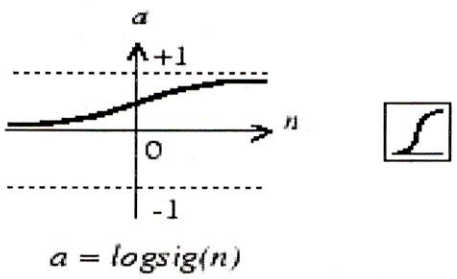

Figure 3.10: Log-sigmoid transfer function

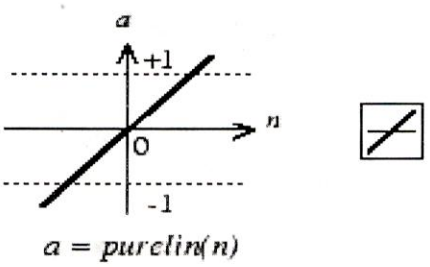

Figure 3.11: Linear transfer function

3.2.2 Model Development

The most important steps in the ANN model development process is the selection of significant input variables. Usually, not all of the potential input variables will be equally informative, because some may be correlated, noisy, or have no significant relationship with the output variable being modeled (Maier & Dandy 2000). Input variables were selected based on cross-correlation, autocorrelation and partial autocorrelation technique. Many researchers such as (Nayak et al. 2006) have been successfully using correlation analysis for selection of input variables.

The auto-correlation coefficient (Salas et al. 1980) is defined as:

$$r_k = \frac{\sum_{t=1}^{N-k}(x_t - \bar{x}_t)(x_{t+k} - \bar{x}_{t+k})}{\left[\sum_{t=1}^{N-k}(x_t - \bar{x}_t)^2 \sum_{t=1}^{N-k}(x_{t+k} - \bar{x}_{t+k})^2\right]^{1/2}} \quad (3.3)$$

Where r_k is called the *lag-k* correlation coefficient, the serial correlation coefficient or the auto correlation function (ACF), x_t is the time series for $t = 1,..,N$, x_{t+K} is the lagged time series for $t = 1,..., N-k$, $\overline{x_t}$ is the sample mean for $t = 1,...,N$, $\overline{x_{t+k}}$ is the sample mean *for* $t = 1,...,N-k$, N is the sample size.

The partial auto-correlation coefficient (Salas et al. 1980) may be obtained recursively by the Durbins relations as given below:

$$\emptyset_1(1) = \rho_1, \emptyset_1(2) = \frac{\rho_1(1-\rho_2)}{(1-\rho_1^2)}, \emptyset_2(2) = \frac{\rho_2-\rho_1^2}{(1-\rho_1^2)} \quad (3.4)$$

$$\emptyset_k(k) = \frac{\rho_k - \sum_{j=1}^{k-1} \emptyset_j(k-1)\rho_{k-j}}{1 - \sum_{j=1}^{k-1} \emptyset_j(k-1)\rho_j} \quad (3.5)$$

$$\emptyset_j(k) = \emptyset_j(k-1) - \emptyset_k(k)\emptyset_{k-j}(k-1) \quad (3.6)$$

To determine the partial auto-correlation function from a sample series $x_1,.....,x_N$, the sample autocorrelation the ρ*'s* are replaced by r*'s*. ρ*'s* are auto-regression coefficients.

The cross-correlation coefficient (Salas *et al.*, 1980) is defined as:

$$r_k^{ij} = \frac{\sum_{t=1}^{N-k}(x_t^{(i)} - \bar{x}_t^{(i)})(x_{t+k}^{(i)} - \bar{x}_{t+k}^{(i)})}{\left[\sum_{t=1}^{N-k}(x_t^{(i)} - \bar{x}_t^{(i)})^2 \sum_{t=1}^{N-k}(x_{t+k}^{(j)} - \bar{x}_{t+k}^{(j)})^2\right]^{1/2}} \quad (3.7)$$

Where r_k^{ij} is the lag-k cross-correlation coefficient, $x_t^{(i)}$ is the time series values of series *i*, $x_t^{(j)}$ is the time series values of series *j*, $\overline{x_t^{(i)}}$ is the mean of the first N-k values of series *i*, and $\overline{x_{t+k}^{(j)}}$ is the mean of the last N-k values of series *j*.

3.2.3 Normalization of Input Data

The input values should be normalized to the range between 0 and 1 before passing into a neural network since the output of sigmoidal function is bound between 0 and 1. Dawson and Wilby (1998) and many others have emphasized the importance of the normalization of data and have given the procedure to normalize. The output from the ANN should be denormalised to provide meaningful results. In this study, equation 3.8 is used to normalize the data set:

$$N_i = (R_i - Min_i) / (Max_i - Min_i) \quad (3.8)$$

Where R_i is the real value applied to neuron *i*; N_i is the subsequent normalized value calculated for neuron *i*; Min_i is the minimum value of all values applied to neuron *i*; Max_i is the maximum value of all values applied to neuron *i*.

3.2.4 ANN models

ANNs consist of a large number of simple processing elements called *neurons* or *nodes*. Each neuron is then connected to other neurons by means of direct links, each being associated with a weight that represents information being used by the network in its effort to solve the problem. The neural network can be in general characterized by architecture (the patterns of connection between the neurons), training or learning algorithm (the methods of determining the weights on the connections) and activation function.

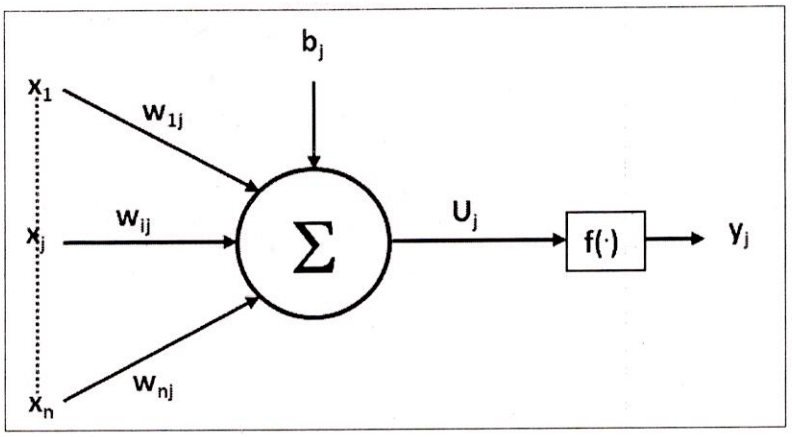

Figure 3.12: Schematic diagram of an artificial neuron

The architecture of a typical neural network with a single neuron is shown in Figure 3.12. It consists of five basic elements:

i. Input nodes for receiving input signals $x_1, \ldots, x_P$;
ii. A set of connecting links (often called *synapses*), each of which is characterized by a weight w_{ij};
iii. Aggregating function to sum the input signals;
iv. Activation function that calculates the activation level of the neuron; and
v. Output nodes $y_1, \ldots, y_l$.

The processing of each neuron is carried out in two steps:

i. Summing of the weighted input signals.
ii. Applying activation function to the sum for limiting the amplitude of the output of a neuron.

Mathematically it can be described by the following two equations:

$$u_j = \sum_{i=1}^{p} w_{ij} x_i \tag{3.9}$$

$$y_j = f(u_j + b_j) \tag{3.10}$$

where w_{ij} is the weight connecting the input i to the neuron j. The effective incoming signal u_j and bias b_j is passed through activation function $f(.)$ to produce the output signal y_j. The main difference between the neurons in common use lies in the type of the activation function. Their functional form determines the response of a node to the

total input signal, however there is one thing common among these activation functions – they all restrict the input signals to certain limits. Some commonly used activation functions are linear, binary, sigmoid and tangent hyperbolic.

In the present work, both the Multilayer Perceptron (MLP) network and the Radial Basis Function (RBF) network have been considered for rainfall-runoff modeling. The RBF and MLP networks are usually used in the same kind of applications (nonlinear mapping approximation and pattern recognition), however their internal calculation structures are different.

3.2.4.1 Artificial Neural Network (ANN)

Feedforward artificial neural network is one of the most popular and successful neural network architectures. The ANN consists of an input layer, an output layers and at least one intermediate layer between input and output layer. The first layer is the input layer, which receives the input signal. The intermediate layers are known as hidden layers, which do not have direct connection to the outer world. The last layer is the output layer at which the overall mapping of the network input is made available and thus represents the model output. The nodes in one layer are connected to those in the next, but not to those in the same layer. Thus, the information or signal flow in the network is restricted to a flow, layer by layer, from the input to output through hidden layers. Figure 3.13 shows the example of a three layer feedforward ANN with one hidden layer. Training (i.e. computing the weights) of the ANN networks is done with the backpropagation algorithm which is the most popular algorithm capable of capturing a variety of non-linear error surfaces. It is essentially a gradient descent technique that minimizes the network error function. The back propagation algorithm involves two steps: the first step is feed forward pass, in which the input vectors of all the training examples are fed to the network and the output vectors are calculated. The performance of the network is evaluated using an error function that is based on the target and network outputs. After the error is computed, the back propagation step starts, in which the error is propagated back to adjust the network weights and bias. The iteration continues until the outputs of the network will match with the targets with the desired degree of accuracy. In a typical application, the weight update loop in back propagation may be iterated thousands of times. A variety of termination condition can be used to halt the iteration. One may choose to halt after a fixed number of iterations or once the error on the training examples falls below some

threshold or once the error on separate validation set of examples meets some criterion. The choice of termination or stopping criterion is important and discussed by many authors (see, e.g., Mitchell 1997). Although the back propagation method does not guarantee convergence to an optimal solution since local minima may exist, it appears in practice that it leads to solutions in almost every case. In fact, standard multi-layer, feed-forward networks with only one hidden layer have been found capable of approximating any measurable function to any desired degree of accuracy. Detailed description of the back propagation algorithm can be found in (Haykin 1999; Principe et al. 1999) among others.

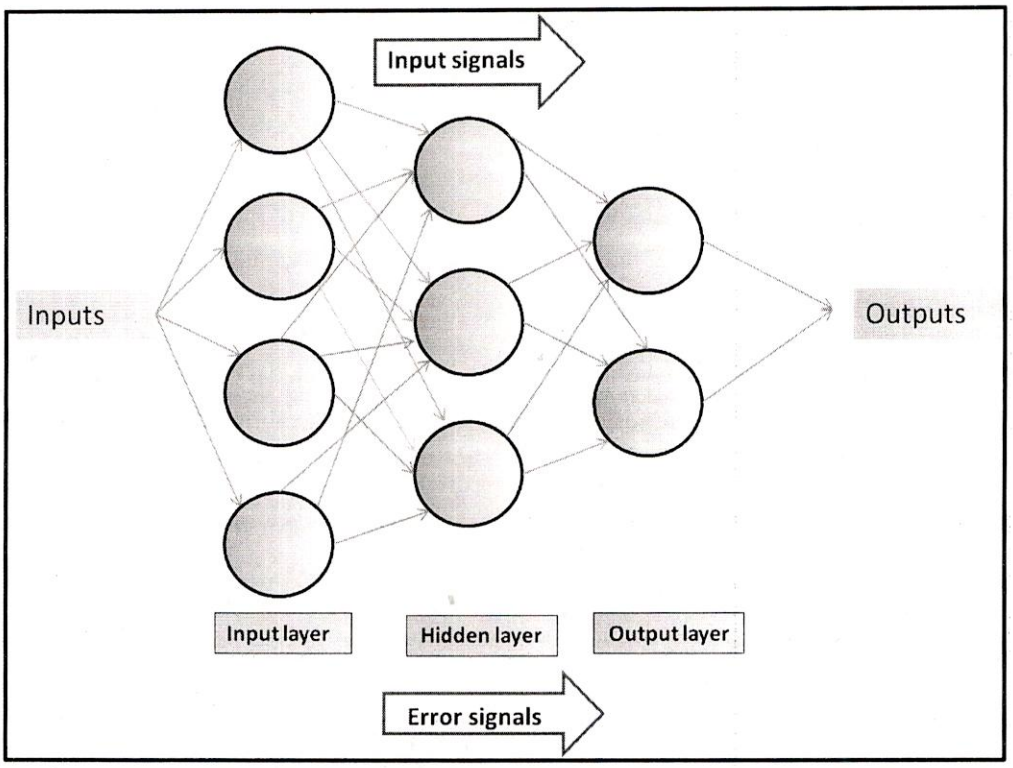

Figure 3.13: A three layer feedforward artificial neural network architecture

3.2.4.2 Radial Basis Function (RBF) network

Based on Demuth and Beale (2001), Radial basis networks consist of two layers: a hidden radial basis layer of S^1 neurons and an output linear layer of S^2 neurons as presented in figure 3.14.

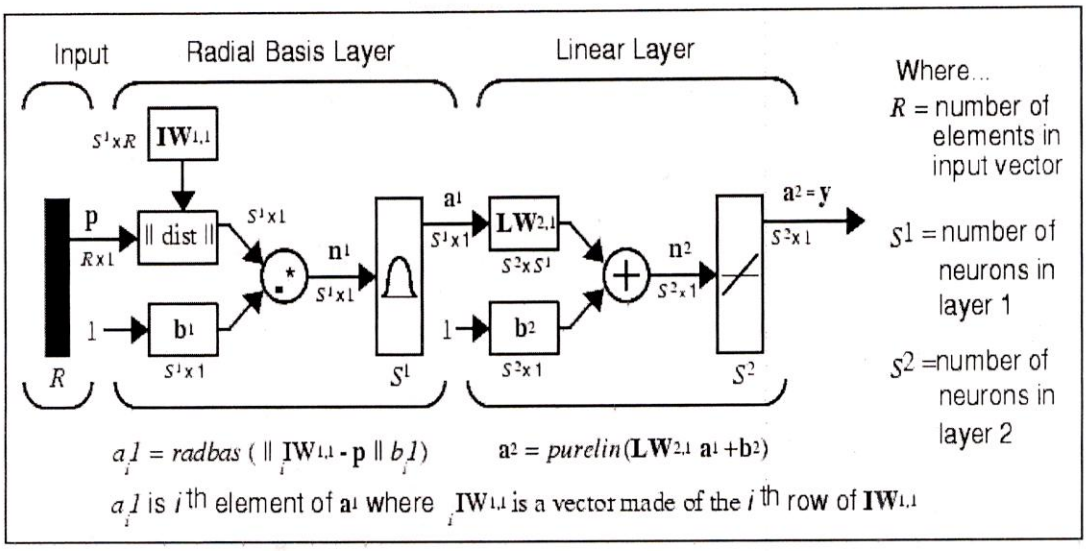

Figure 3.14: Radial Basis Function (Demuth & Beale 2001)

In order to design the network, a design function newrb will be the net code and their outputs can be obtained with 'sim'. This network will behave by following an input vector '**p**' through the network to the output $\mathbf{a}^2$. When an input vector is present to the network, each neuron in the radial basis layer will output a value according to how close the input vector is to each neurons's weight vector. Typically several neurons are always firing, to varying degrees. The bias **b** allows the sensitivity of the radbas neuron to be adapted.

The function newrb creates a two layer network. The first layer has a radbas neurons and calculates its weighted inputs with dist, and its net input with netprod. Number of neurons in the hidden layer must be sufficient enough to withhold the mass of inputs. Number of hidden neurons in the hidden layer was investigated by trial and error method. The values investigated are 10, 20, 30, 40, 50, 60, 70, 80, 90 and 100. While the second layer has purelin neuron that calculates its weighted input with dotprod and its net input with netsum. Both of the layers have biases. The training process will keep generating until the neurons networks mean squared error goal is met or the maximum number of neuron is achieved.

A smoothing parameter known as spread, need to be set. Basically, the larger the value of spread, the smoother the function approximation will be. However, if the value of the spread is too large, many neurons will be required to fit a fast changing function. Meanwhile, if it is too small, many neurons will be required in order to fit a smooth function and the networks will not be generalized well. Therefore, in this

project a set of trial and error of spread value will be tested to find the optimum value for determining the best network for radial basis function network. The spread values tested are 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 and 1.

3.2.5 Training algorithms

The way a network is trained is a basic property of an ANN; the values of several neuron properties and the manner in which the neurons of an ANN are structured are closely related to the chosen algorithm. The algorithm that is used to optimize these weights and biases is called training algorithm or learning algorithm. Training algorithms can be classified broadly into those comprising supervised learning and unsupervised learning.

i. Supervised learning works by presenting the ANN with input data and the desired correct output results. This is done by an external 'teacher', hence the name of this method. The network generates an estimate, based on the given input, and then compares its output with the desired results. This information is used to help guide the ANN to a good solution. Some learning methods do not present the actual desired value of the output to the network, but rather give an indication of the correctness of the estimate (Dhar & Stein 1997).

ii. ANNs being trained using an unsupervised learning paradigm are only presented with the input data but not the desired results. The network clusters the training records based on similarities that it abstracts from the input data.

In the present study, backpropogation algorithm is used which is a supervised learning technique. An ANN that is trained using a supervised learning method attempts to find optimal internal parameters (weights and biases) by comparing its own approximations of a process with the real values of that process and subsequently adjusting its weights (and biases) to make its approximation closer to the real value. The aforementioned comparison is based upon an evaluation using a performance function (hence the name performance learning). The author will refer to this function as error function.

Suppose a network is trying to approximate a certain process, which can be characterized by a number of n variables (see Figure 3.15). The network input is a vector $\mathbf{x}$ and the weights of the network form a matrix $\mathbf{W}$). The approximation of the network is a vector of n variables called $y=(y_1,y_2,\ldots y_n)$ (which is a function of $\mathbf{x}$ and

W) and the real values of the variables are included in a vector called $t=(t_1,t_2,\ldots.t_n)$. The difference between the two is used to calculate an approximation error **E** . In order for an ANN to generate an output vector **y** that is as close as possible to the target vector **t** , an algorithm is employed to find optimal internal parameters that minimize an error function. This function usually has the form:

$$E = \sum_{h=1}^{n} (t_h - y_h)^2 \tag{3.11}$$

where n is the number of output neurons (Govindaraju 2000).

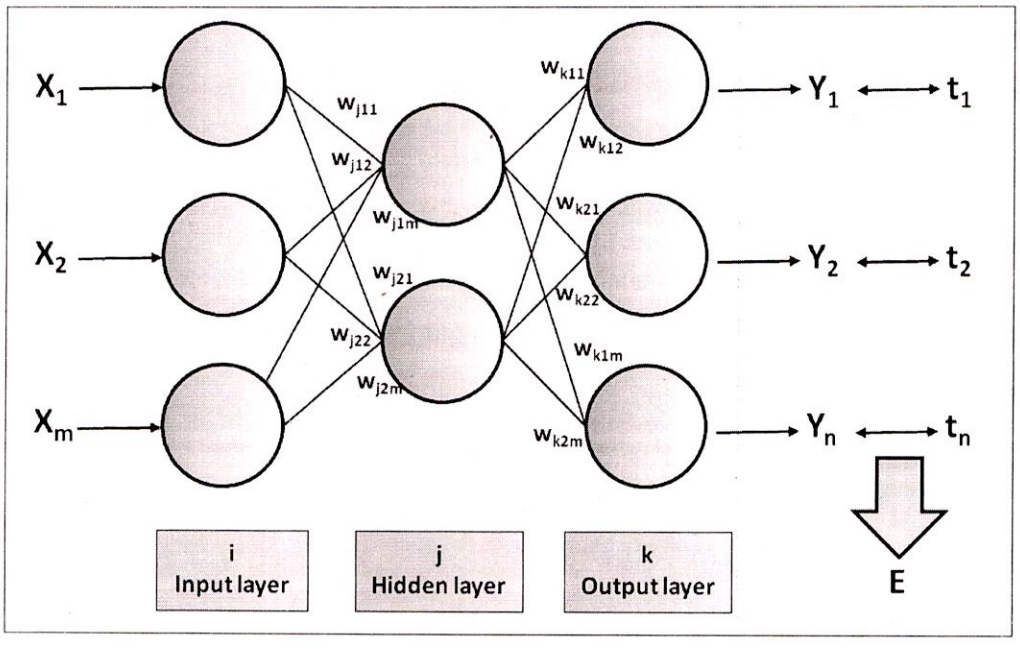

Figure 3.15: Example of a two-layer feedforward network

Equation 3.11 is based on the error expression called Mean Square Error (MSE). The MSE error measurement scheme is often used, because it has certain advantages. Firstly, it ensures that large errors receive much greater attention than small errors, which is usually what is desired. Secondly, the MSE takes into account the frequency of occurrence of particular inputs. The MSE is best used if errors are near normally distributed. Because **y** is a function of the weights in **W** the error function (**E**) also becomes a function of **W** of the network being evaluated.

3.2.5.1 Backpropagation algorithm

The best-known algorithm for training ANNs is the backpropagation algorithm. It essentially searches for minima on the error surface by applying a steepest-descent gradient technique. The algorithm is linearly convergent. The backpropagation architecture described here and in the accompanying appendices is the basic, classical version, but many variants of this basic form exist. Basically, each input pattern of the training data set is passed through a feedforward network from the input units to the output layer. The network output is compared with the desired target output, and an error is computed based on an error function. This error is propagated backward through the network to each neuron, and correspondingly the connection weights are adjusted.

Backpropagation is a first-order method based on the steepest gradient descent, with the direction vector being set equal to the negative of the gradient vector. Consequently, the solution often follows a zigzag path while trying to reach a minimum error position, which may slow down the training process. It is also possible for the training process to be trapped in a local minimum. (Govindaraju 2000).

A possible way of preventing overtraining is called regularization. This method involves modifying the error function of performance learning algorithms. For example, if the MSE is used as error function, generalization can be improved by adding a term that consists of the mean of the sum of squares of the network weights and biases:

$$MSEREG = \gamma . MSE + (1 - \gamma) . MSW \tag{3.12}$$

Where

$$MSW = \frac{1}{n} \sum_{j=1}^{n} {w_j}^2 \tag{3.13}$$

Using this performance function will cause the network to have smaller weights and biases, and this will force the network response to be smoother and less likely to overtrain. Bayesian regularization algorithm is used in this study in order train the given network more efficiently.

3.2.5.2 Bayesian Regularization Algorithm (BR)

The Bayesian regularization is an algorithm that automatically sets optimum values for the parameters of the objective function. In the approach used, the weights and biases of the network are assumed to be random variables with specified distributions. In order to estimate regularization parameters, which are related to the unknown variances, statistical techniques are being used. The advantage of this algorithm is that whatever the size of the network, the function won't be over-fitted. Bayesian regularization has been effectively used in literature (Anctil et al. 2004; Coulibaly et al. 2001a; Porter et al. 2000).

3.2.6 Fuzzy logic

A Takagi-Sugeno Fuzzy Inference System (FIS) has been developed by using the subtractive clustering (Chiu 1994) algorithm integrated with a linear least squares estimate algorithm for rainfall- runoff modeling.

The TS model was proposed by Takagi and Sugeno (1985). Figure 3.16 explains the concept of a first-order TS model with one input variable X and one output variable Y. The input variable is partitioned into n fuzzy sets $A_1,..,A_n$, the antecedent fuzzy sets. This results in n fuzzy rules of the form:

$$R_i: \text{IF } X \text{ is } A_i \text{ THEN } Y = a_iX+b_i ,$$

with a_i and b_i the parameters of the consequent part of rule R_i. Given a value x of the input variable X, the resulting value y of the output variable Y is computed as:

$$y = \frac{\sum_{i=1}^{n} A_i(x)(a_i x + b_i)}{\sum_{i=1}^{n} A_i(x)} \qquad (3.14)$$

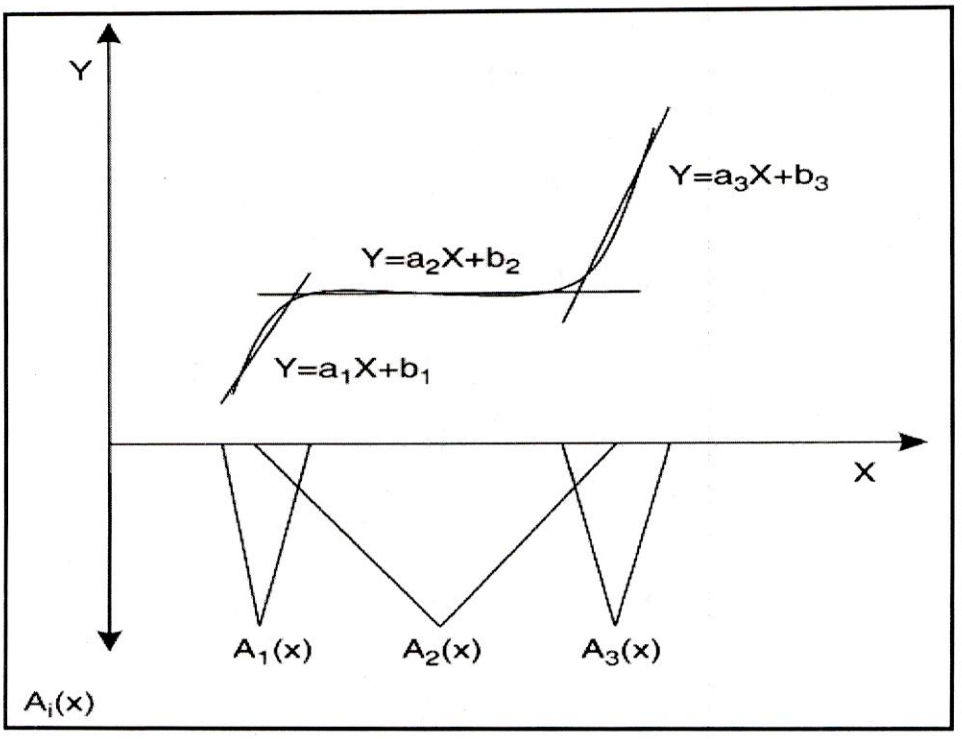

Figure 3.16: Schematic representation of a Takagi–Sugeno model

In order to apply a TS model to a p-dimensional input space, in particular if A_i is a Cartesian product $A_i = A_{1,i1}*\ldots*A_{p,ip}$, with $i_1 \in \{1,\ldots,n_1\},\ldots, i_p \in \{1,\ldots,n_p\}$, and $n_1,n_2,\ldots,n_p$, the number of fuzzy sets each input variable is partitioned into, the rule reads:

$$R_i: \text{IF } (X_1,\ldots,X_p) \text{ is } A_i \text{ THEN}$$

$$Y = a_{1,i}X_1 + a_{2,i}X_2 + \ldots + a_{p,i}X_p + b_i,$$

For a p-dimensional input vector $\mathbf{x} = (x_1,\ldots,x_p)$, $A_i(\mathbf{x})$ is then usually realized as:

$$A_i(x) = A_{1,i_1}(x_1)\, A_{2,i_2}(x_2) \ldots \ldots Ap,_{i_p}(x_p) \tag{3.15}$$

when the above type of rules are used, $A_i(\mathbf{x})$ is also the degree of fulfillment (DOF), $w_i(\mathbf{x})$, of rule i.

The resulting output value y is then computed as:

$$y = \frac{\sum_{i=1}^{n} w_i(x)(\, a_{1,i}x_1 + \cdots \ldots \ldots a_{p,i}x_p + b_i)}{\sum_{i=1}^{n} w_i(x)} \tag{3.16}$$

In this way, a weighted average of the individual rule outputs is computed and a nonlinear function can be approximated.

3.2.6.1 Subtractive clustering

SC was introduced by Chiu (1994). Subtractive clustering is a fast and robust method for estimating the number and location of clusters present in a collection of data points. From figure 3.17, *di* denotes the projected cluster center and c_i denotes the data point. Clustering is processing which deals with the task of partitioning a set of patterns into a number of homogeneous classes (clusters) with respect to a suitable similarity measure.

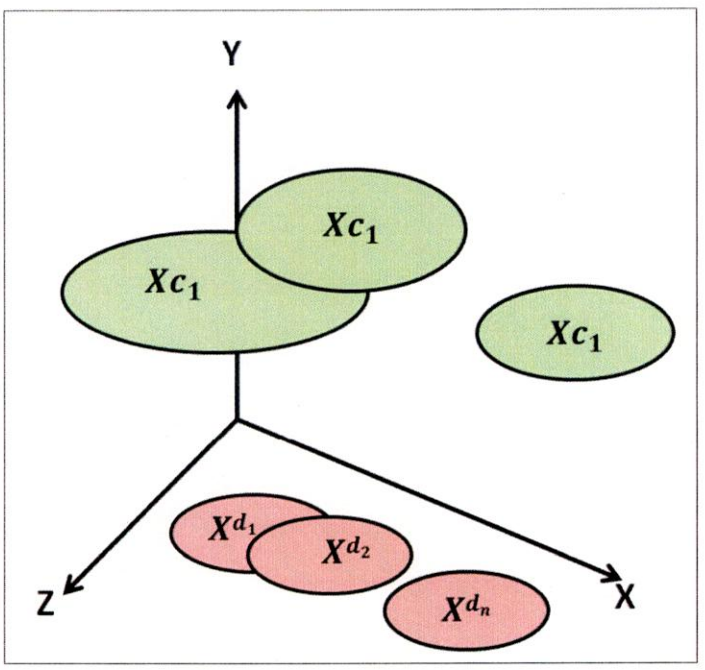

Figure 3.17: Projection of the fuzzy clusters onto the antecedent space in the case of a three-dimensional Input -Output space

Clustering is processing which deals with the task of partitioning a set of patterns into a number of homogeneous classes (clusters) with respect to a suitable similarity measure. Patterns belonging to any one of the clusters are similar, and the patterns of different clusters are as dissimilar as possible. In classical cluster analysis, the boundary of different clusters is crisp such that one pattern is assigned to exactly one cluster. In that case, where data distribution is not good, cluster boundary may not be precisely defined. Hence, a data point could belong to tow or more clusters with different degree of membership. Clustering technique is used for structure

identification based on input-output data. In this study, fuzzy substractive clustering-based system identification and a sugeno-type fuzzy inference system was the basis of our approach to predict soil moisture from SAR, vegetation and soil type data. The procedure used to define the modeling process are as following:

i. Select factors to be involved in the process and choose the levels of these factors.
ii. Conduct the experiments randomly at all possible factor-level combinations.
iii. Construct the fuzzy model using a subtractive clustering-based system identification algorithm and a Sugeno-type fuzzy inference system.
iv. Search through clustering parameters to obtain a model with minimum error.

The Least Square Estimation algorithm used for the overall optimization of the regression parameters for a given set of clusters. The optimization of the fuzzy model depends mainly on finding the optimum range of the clustering parameters such as squash factor (s_f), cluster radius (r_a), acceptance ratio, and rejection ratio (η). The models which result in an acceptable error are selected for further validation with the testing set.

The genfis2 algorithm provided by MATLAB software uses a subtractive clustering method to generate Fuzzy Inference System (FIS). The genfis2 function uses the subclust function to estimate an antecedent membership function and set of rules. Further, the subclust function uses the linear least-square method to determine each rule's consequent equation, and returns FIS structure that contains a set of fuzzy rules. The subclust function assumes each data point is a potential cluster center and calculates a measure of the likelihood that each data point will define the cluster center, based on the density of the surrounding data points.

The subtractive clustering approach is used to identify cluster centers using r_a parameter. This value is the maximum distance between any two points within the same cluster, yet less than the distance between distance between any two points from different clusters where each point belongs. Cluster center selection criteria are based on acceptance and rejection ratios. Acceptance ratio can be determined by fractions of the potential first cluster center over another potential cluster center data that has been accepted. The rejection ratio is the condition required to reject a data point. A data point will be rejected as a cluster center if it is below the rejection ratio obtained from

fractions of the potential first cluster center. The acceptance ratio is selected as 0.5 for the first cluster center and rejection ratio (η) between 0.15-0.5 to derive other cluster centers. Detailed descriptions of steps followed during the subtractive clustering are giver below:

i. Compute the initial potential value for each data point (x_i)

$$P_t = \sum_{j=1}^{n} e^{-a\,\|x_i - x_j\|^2} \tag{3.17}$$

Where, $\alpha = 4/r_a^2$.

$\| x_i - x_j \|$ is the Euclidean distance and r_a is a positive constant representing a normalized neighborhood data radius. Any point falling outside this circle region will have little influence on the potential point. The point with the highest potential value is selected as the first cluster center. This tentatively defines the first cluster center.

ii. A point is qualified as the first cluster center if its potential value ($P^{(1)}$) is equal to the maximum of initial potential value ($P^{(1)*}$).

$$P^{(1)*} = \max_i(P^{(1)}(x_i)) \tag{3.18}$$

iii. Define a specific threshold δ to make decision to either continue or stop the cluster center search. This process continues until current maximum potential remains greater than δ.

δ = (rejection ratio) * (potential value of the first cluster center)

Where the rejection ratio (η) used in this work is 0.15-0.5, and $P^{(i)*}$ is the potential value of the first cluster center.

iv. Remove the previous cluster center for further consideration.

v. Revise the potential value of the remaining points according to the following equation.

$$P_i = P_i - P_k^* * e^{-\beta\|x_i - x_k^*\|^2} \tag{3.19}$$

X_k^* is the point of the k^{th} cluster center, P_k^* is its potential value, and r_b is a positive constant. Thus, the potential value of each data point is revised by subtracting an amount of potential from each data point as a function of its

distance from the first cluster center. Therefore, data point near the first cluster center will have greatly reduced potential, and will unlikely be selected as the future cluster center. The constant r_b typically selected as $r_b = s_f * r_a$ is the radius defining the neighborhood which will have measurable reductions in potential. The squash factor (s_f) is a positive constant greater than 1.

vi. For the point having the maximum potential value, calculate the acceptance ratio. If this value is greater than the predefined constant (0.5), the point is accepted as the next cluster center. Otherwise compute the rejection ratio. If the rejection ratio is greater than the predefined threshold (η = 0.15-0.5), this point is accepted.

3.2.7 Performance evaluation

The results obtained from calibration and validation are evaluated to determine the difference between observed and predicted values. In most of studies in this thesis, root mean square error (RMSE), model efficiency (EFF) (Nash & Sutcliffe 1970), and coefficient of correlation (CORR) were used as performance criteria to evaluate the various ANN models. They are defined as follows:

$$Root\ Mean\ Squared\ Error\ (RMSE) = \sqrt{\frac{\sum_{k=1}^{K}(t-y)^2}{K}} \tag{3.20}$$

$$Efficiency = 1 - \frac{\sum(t-y)^2}{\sum(t-\bar{t})^2} \tag{3.21}$$

$$Coefficient\ of\ Correlation\ (CORR) = \frac{\sum TY}{\sqrt{\sum T^2 \sum Y^2}} \tag{3.22}$$

CHAPTER 4

RESULTS AND DISCUSSION

The results of the study with the defined objectives to develop Artificial neural network (ANN), RBF and Fuzzy logic models for Rainfall – Runoff Modeling of the study area and the analysis of the results is presented in this chapter.

4.1 SELECTION OF INPUT VECTOR

The input vector is selected generally by trial and error procedure. The simple correlation between the dependent and independent variables helps in selecting the significant input vector to the model. The correlation analysis helps to find out the possible input variable for the modeling, but it does not give the exact lag values.

Sudheer et al. (2002) have suggested a statistical procedure that avoids the trial and error procedure. They have reported that the statistical parameters such as auto-correlation function (ACF), partial auto-correlation function (PACF) and cross-correlation function (CCF) could be used to find out the significant lag values of input variables.

The ACF and PACF of discharge at Bhakra are presented in Figure 4.1 and 4.2 respectively. The CCF between discharge at Bhakra and rainfall at Bhakra, Berthin, Kahu, Kasol, Namgia, Raksham, Rampur and Suni are presented in Figure 4.3. The CCF between discharge and evaporation at Bhakra is presented in Figure 4.4. The CCF between discharge at Bhakra and discharge at Kasol, Rampur and Suni are presented in Figure.

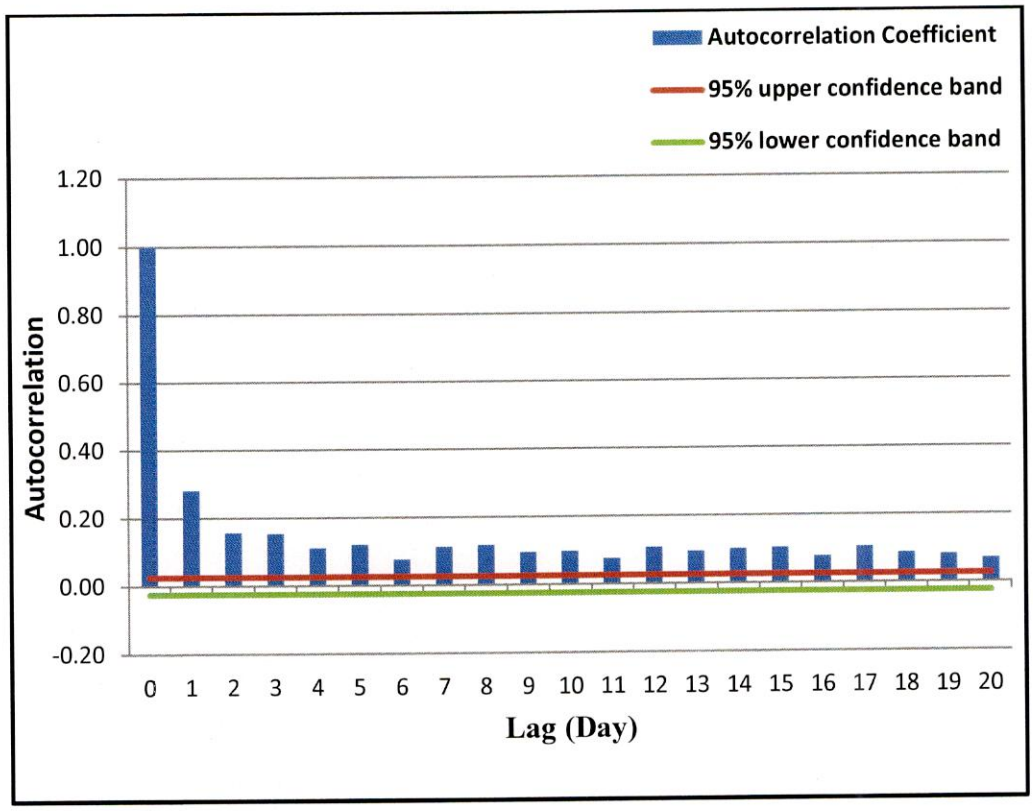

Figure 4.1: Autocorrelation of discharge at Bhakra

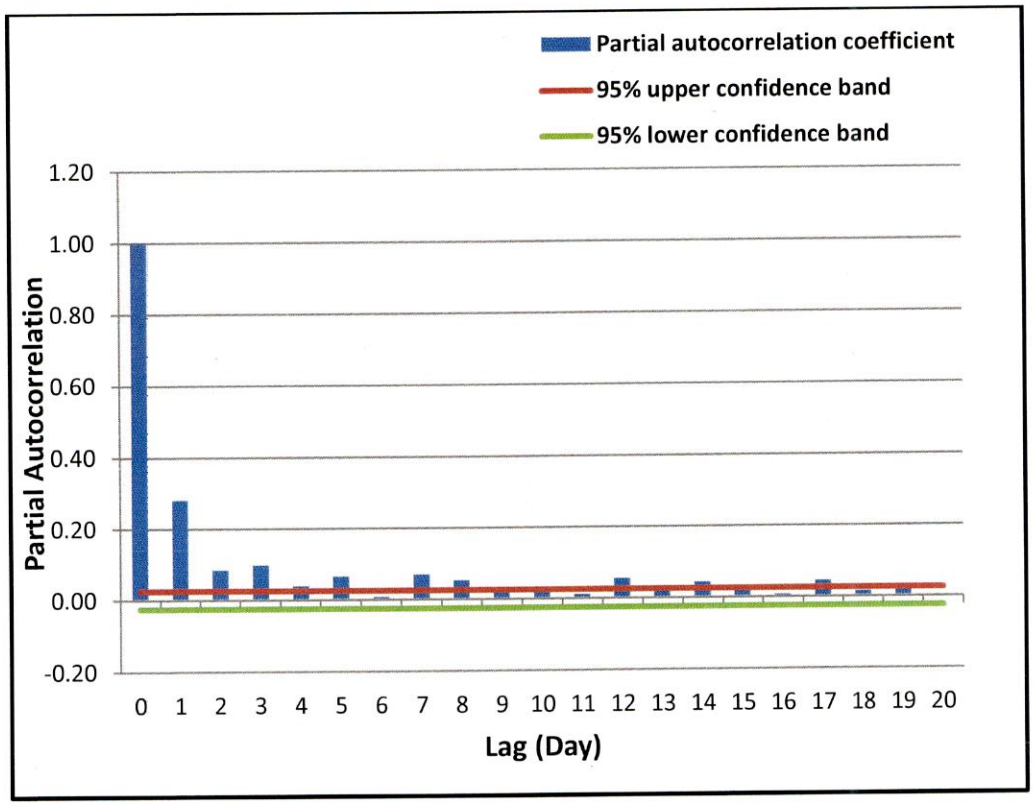

Figure 4.2: Partial autocorrelation of discharge at Bhakra

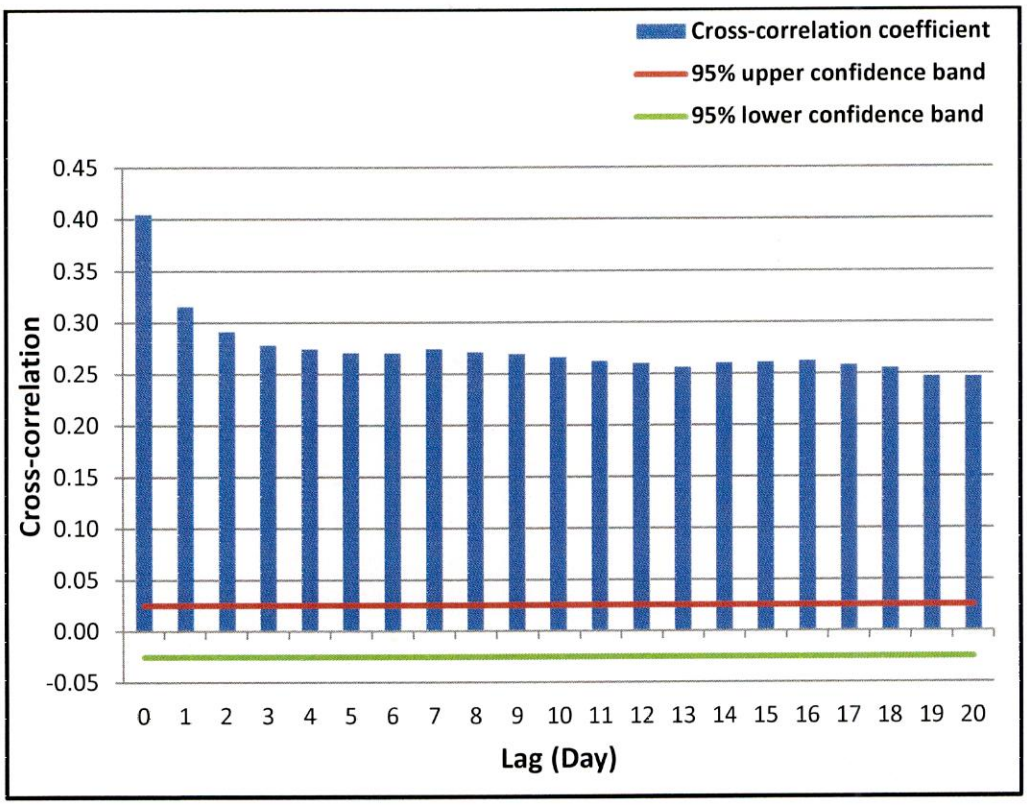

Figure 4.3: Cross-correlation of discharge at Bhakra with rainfall at Bhakra

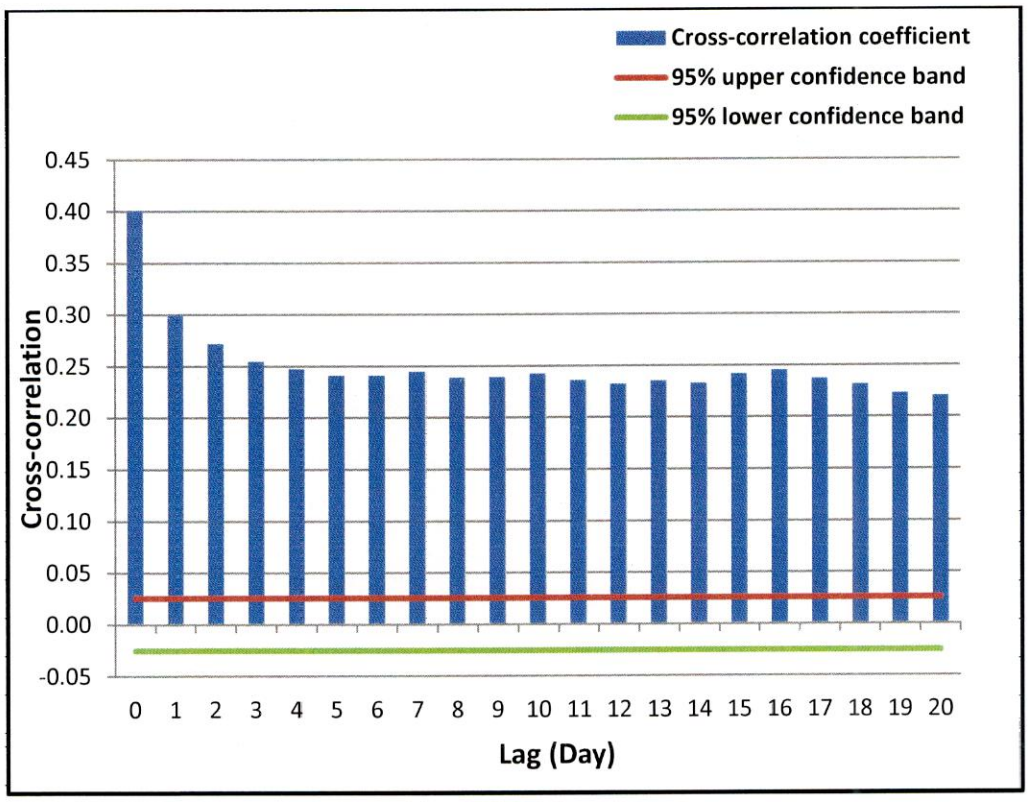

Figure 4.4: Cross-correlation of discharge at Bhakra with rainfall at Berthin

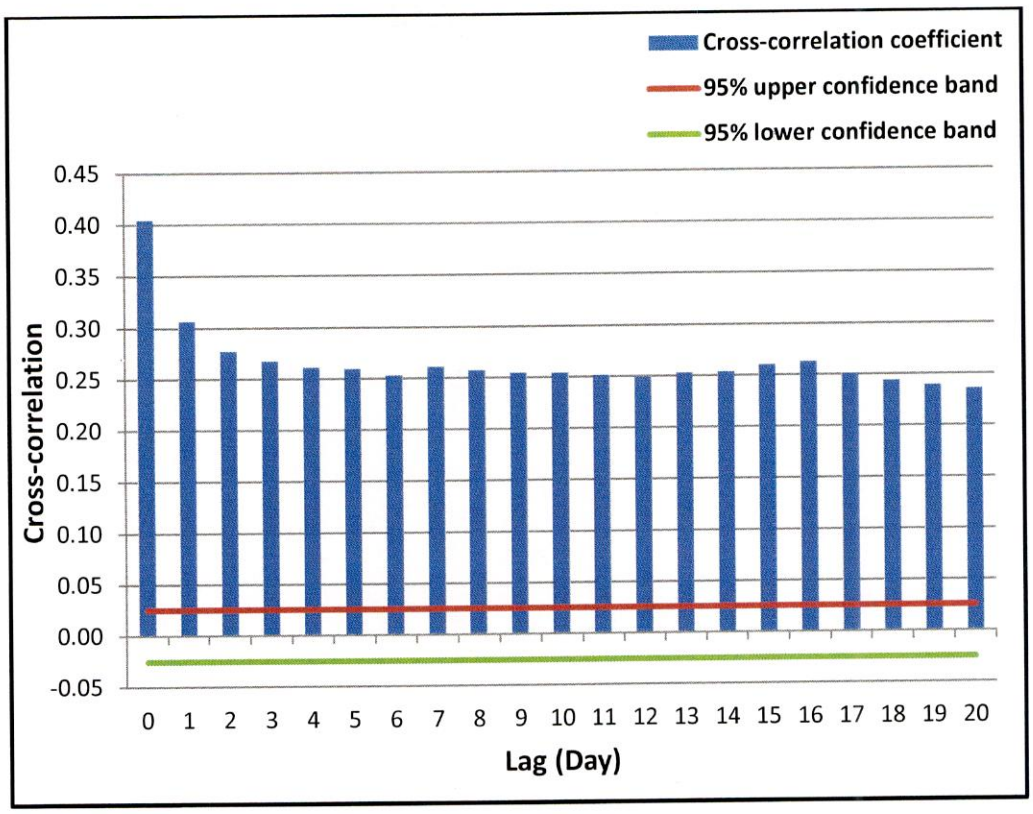

Figure 4.5: Cross-correlation of discharge at Bhakra with rainfall at Kahu

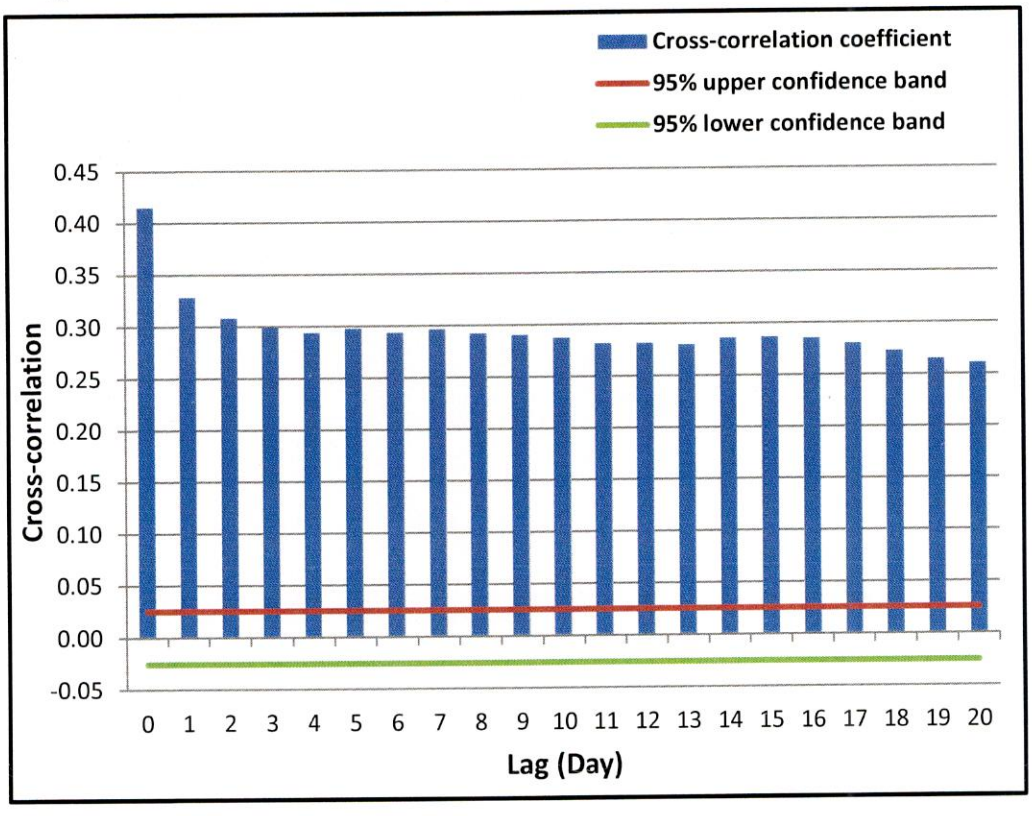

Figure 4.6: Cross-correlation of discharge at Bhakra with rainfall at Kasol

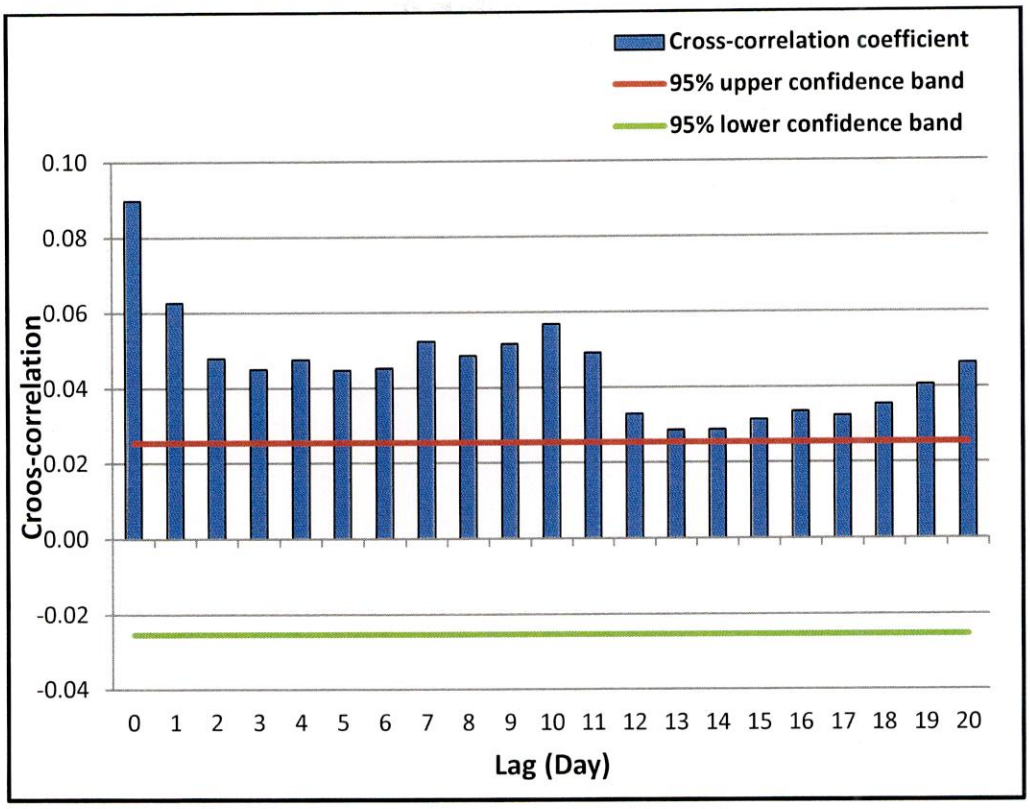

Figure 4.7: Cross-correlation of discharge at Bhakra with rainfall at Namgia

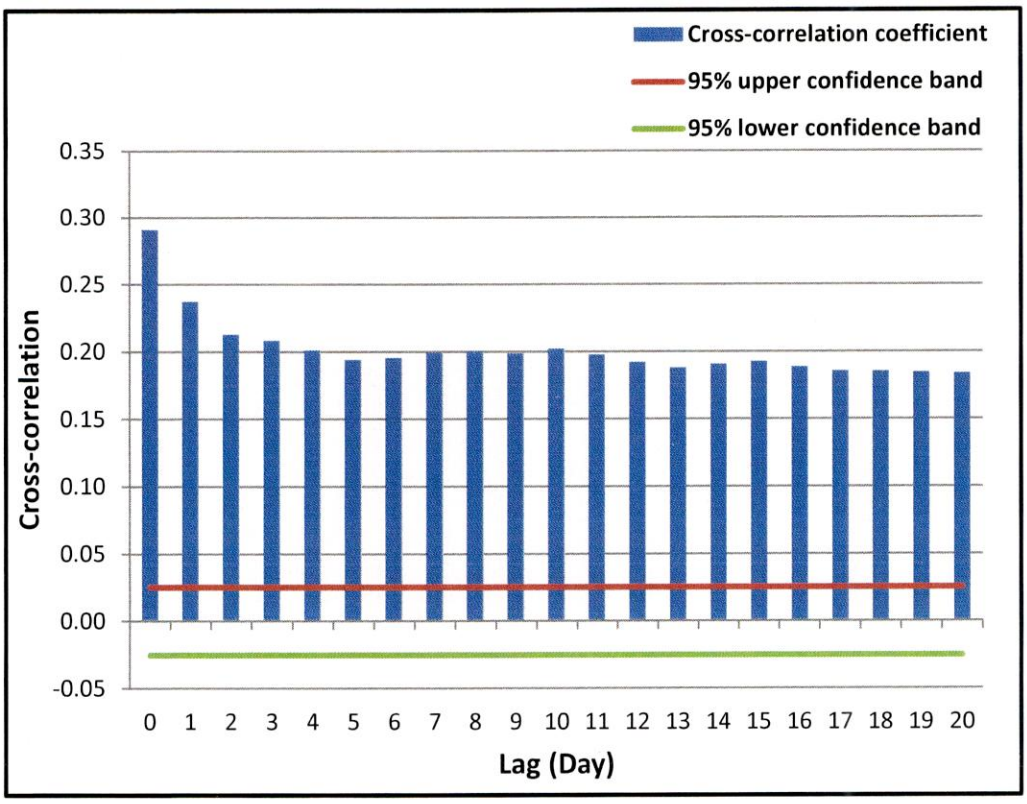

Figure 4.8: Cross-correlation of discharge at Bhakra with rainfall at Raksham

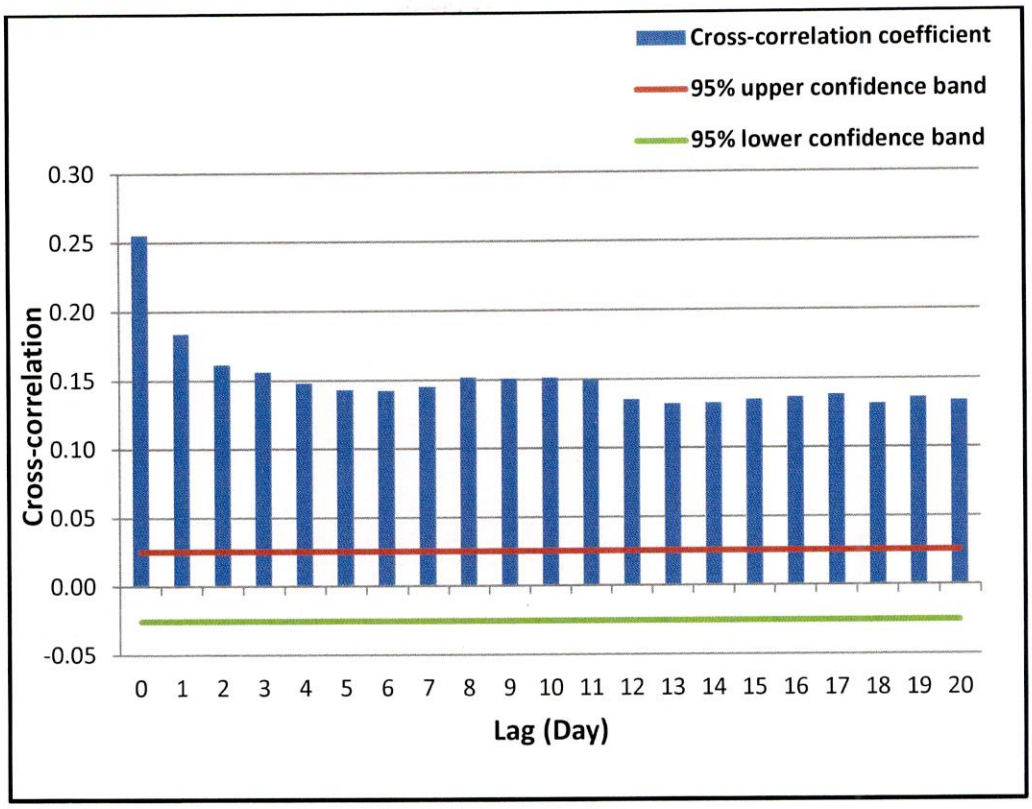

Figure 4.9: Cross-correlation of discharge at Bhakra with rainfall at Rampur

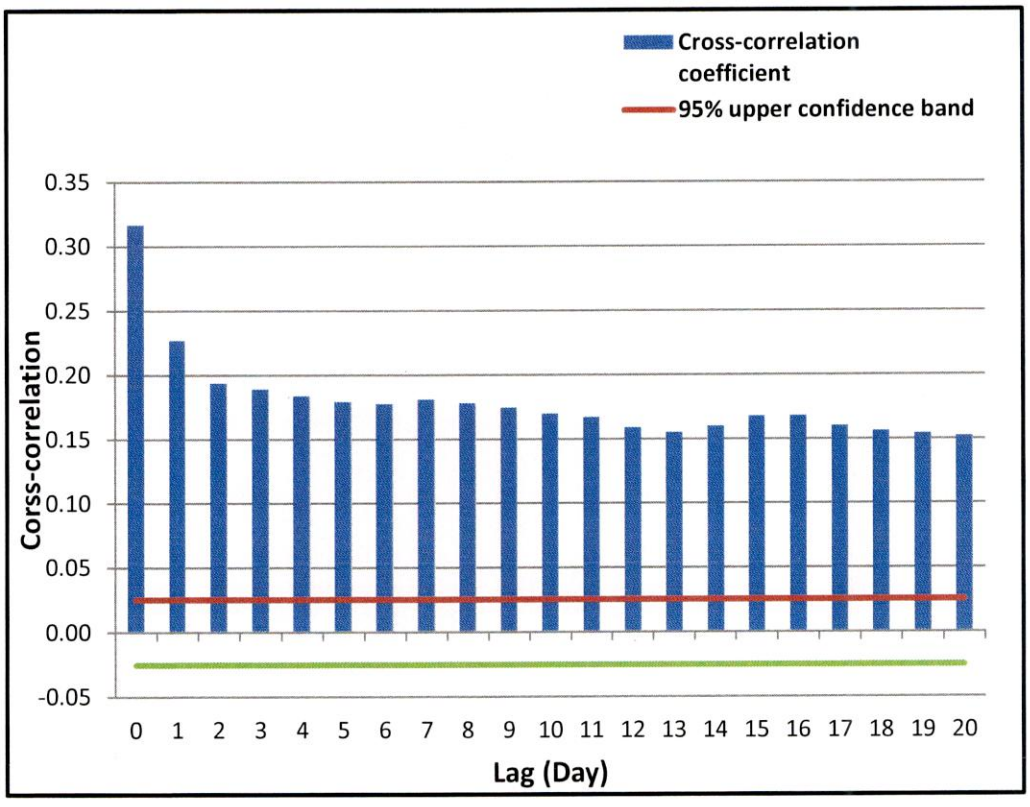

Figure 4.10: Cross-correlation of discharge at Bhakra with rainfall at Suni

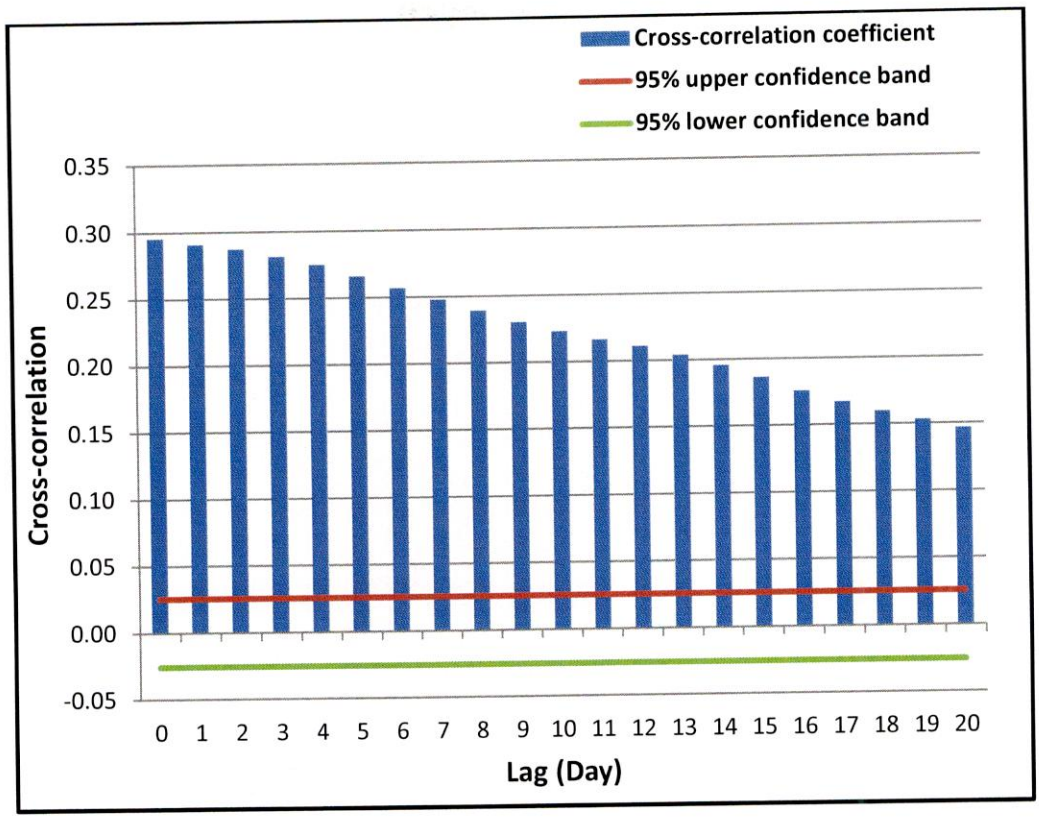

Figure 4.11: Cross-correlation between evaporation and discharge at Bhakra

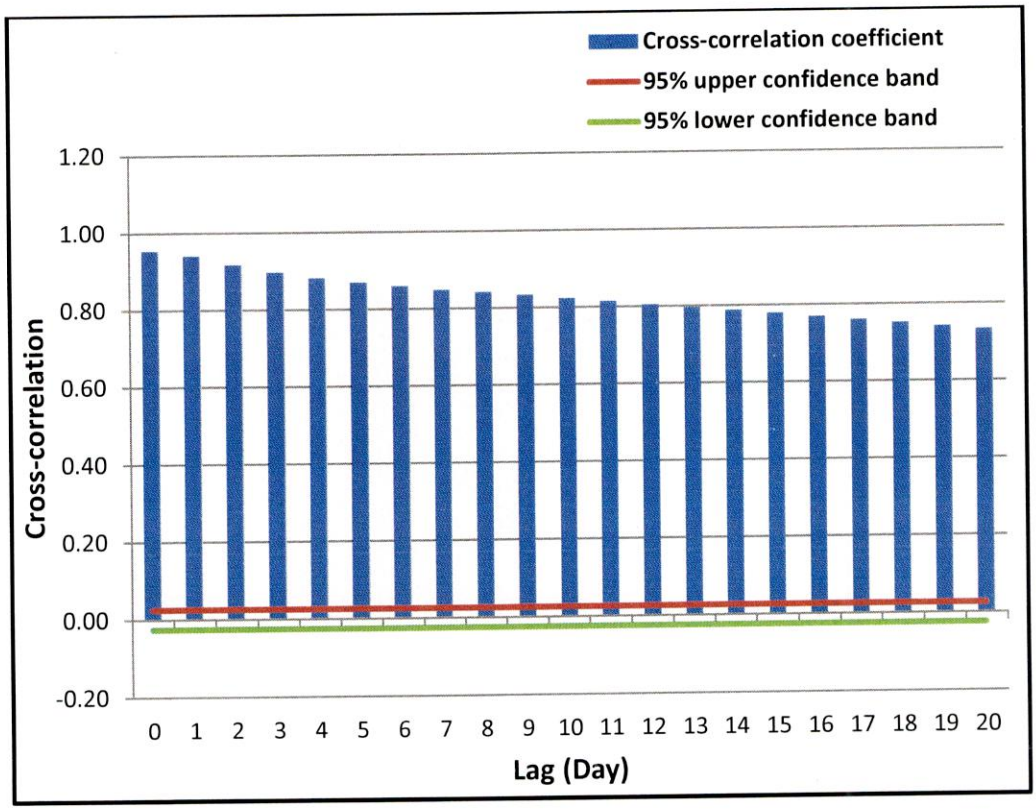

Figure 4.12: Cross-correlation of discharge at Bhakra with discharge at Kasol

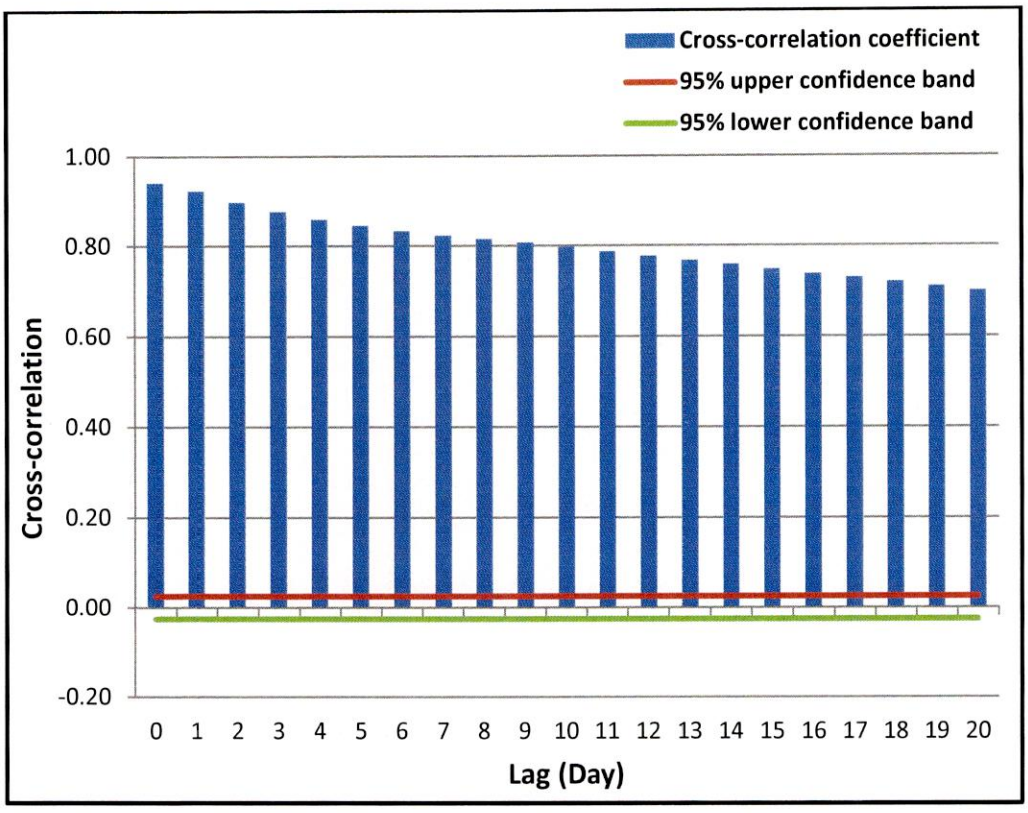

Figure 4.13: Cross-correlation of discharge at Bhakra with discharge at Rampur

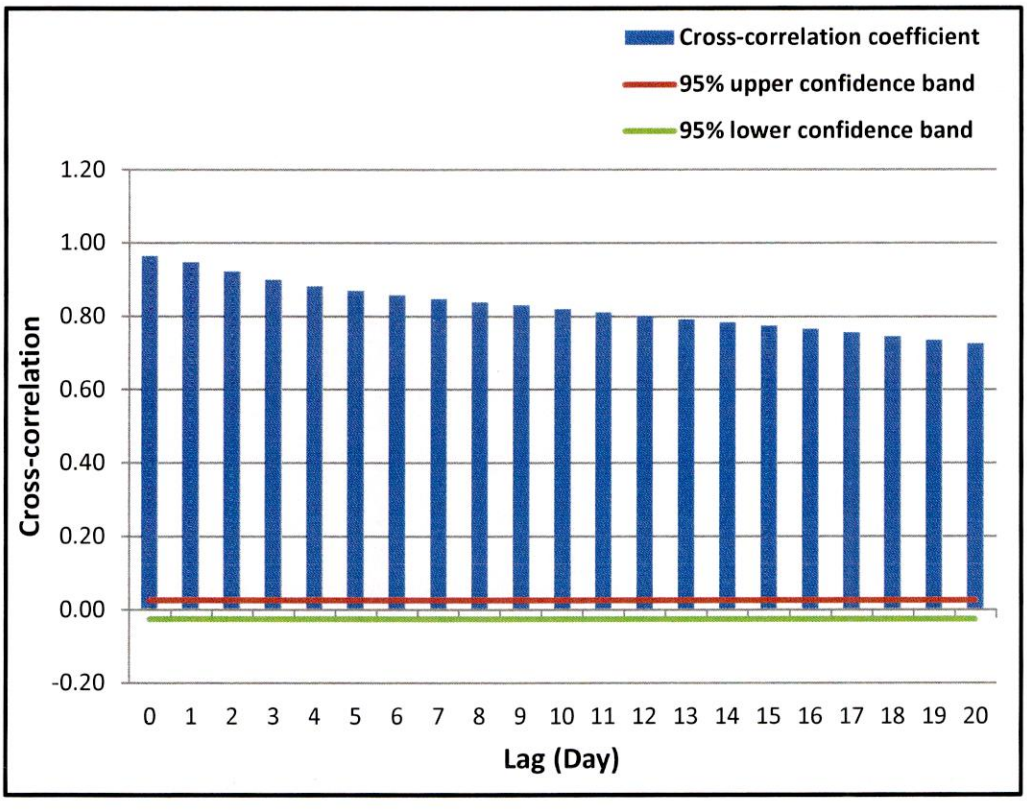

Figure 4.14: Cross-correlation of discharge at Bhakra with discharge at Suni

On the basis of the values of PACF and CCF of the data series as per values mentioned, the following input vector was selected for neural network training:

bhadis(t) = f(berrain(t), bharain(t), kahrain(t), kasrain(t), namrain(t), rakrain(t),

ramrain(t), sunrain(t), bhaevap(t), kasdis(t), ramdis(t), sundis(t), bhadis (t 1)) (4.1)

In equation , bhadis, kasdis, ramdis, sundis are discharge at Bhakra, Kasol, Rampur and Suni respectively. Similarly, bharain, berrain, kahrain, kasrain, namrain, rakrain, ramrain, sunrain are rainfall at Berthin, Bhakra, Kahu, Kasol, Namgia, Raksham, Rampur, Suni and bhaevap is evaporation at Bhakra.

4.2 TRAINING AND VALIDATION OF THE DATA

This is similar to the idea of calibration that is an integral part of most hydrologic modeling studies. The purpose of training is to determine the set of connection weights and nodal thresholds that cause the ANN to estimate outputs that are sufficiently close to target values. The dataset reserved for training is used to achieve this goal. This fraction of the complete data to be employed for training should contain sufficient patterns so that the network can mimic the underlying relationship between input and output variables adequately. The weights and threshold values are assigned small random values initially . During training, these are adjusted based on the error, or the difference between ANN output and the target responses. This adjustment can be continued recursively until a weight space is found, which results in the smallest overall prediction error.

Similar to other modeling approaches in hydrology, the performance of a trained ANN can be fairly evaluated by subjecting it to new patterns that it has not seen during training. The performance of the network can be determined by computing the percentage error between predicted and desired values. In addition, plotting the model output versus desired response can also be used to assess ANN performance. Since finding optimal network parameters is essentially a minimization process, it is advisable to repeat the training and validation processes several times to ensure that satisfactory results have been obtained. The ANN models were trained by using the functions of MATLAB (The Mathworks, Inc., 2001).

The whole data has been divided into two data sets for the training and validation of the models.

i. The daily data from 1, January 1988 to 25, November, 1999 (i.e. 70 % of the total available data) were considered for the calibration of the model.

ii. The data from 26, November, 1999 to 31, December, 2004 (i.e. 30 % of the total available data) were considered for the validation of the model.

4.3 MODEL PERFORMANCE

The performance of the MLP, RBF and FUZZY models during calibration and validation with the input combination derived from statistical procedure given by Sudheer et al. (2002) is shown in Tables 4.1, 4.2 and 4.3 respectively. The statistical procedure uses the ACF, PACF and CCF of the time series to find out significant lag values of input variable. The selection of input variables from ACF, PACF and CCF coefficients also requires the modeler's ability to assess the significant lagged variables.

4.3.1 Artificial Neural Network

The number of the neurons in the hidden layer is found by trial and error procedure starting with one hidden neuron initially and then increasing it up to 10 hidden neuron, based on the performance criteria of the model.

The transfer function for the hidden and output layer are log sigmoid and pure linear respectively in the training of the models.

Table 4.1: Results of ANN model during Calibration and Validation

Model No.	Input Combinations	ANN Structure	Calibration			Validation		
			CORR	RMSE	EFF (%)	CORR	RMSE	EFF (%)
ANN1	bhaR(t), berR(t), kahR(t), kasR(t), namR(t), rakR(t), ramR(t), sunR(t), bhaE(t), kasD(t), ramD(t), sunD(t), bhaD(t-1)	13-1-1	0.98	90.55	96.47	0.99	61.05	97.23
ANN2		13-2-1	0.98	83.83	96.98	0.99	60.66	97.26
ANN3		13-3-1	0.99	74.94	97.58	0.98	63.92	96.96
ANN4		13-4-1	0.99	70.64	97.85	0.97	94.54	96.76
ANN5		13-5-1	0.99	69.77	97.91	0.98	67.28	96.63
ANN6		13-6-1	0.99	64.14	98.23	0.96	100.43	92.50
ANN7		13-7-1	0.99	58.41	98.53	0.97	88.26	94.20
ANN8		**13-8-1**	**0.99**	**54.24**	**98.56**	**0.98**	**56.18**	**97.69**
ANN9		13-9-1	0.99	54.42	98.73	0.97	96.03	93.14
ANN10		13-10-1	0.99	57.84	98.69	0.96	104.94	96.04

Note: bhaR = rainfall at Bhakra; berR = rainfall at Berthin; kah:rainfall at KAhu; kasR = rainfall at Kasol; namR = rainfall at Namgia; rakR = rainfall at raksham; ramR = rainfall at rampur; sunR = rainfall at Suni; bhaE = evaporation at Bhakra; kasD = discharge at Kasol; ramD = discharge at Rampur; sunD = discharge at Suni; bhaD = discharge at Bhakra.

The model ANN8 performed better than other models during calibration (CORR = 0.99, RMSE = 54.24, EFF = 98.56%) and validation (CORR = 0:98, RMSE = 56.18, EFF = 97.69%) because the coefficients of correlation and Efficiency (%) of ANN8 is higher among all other ANN models and RMSE of ANN8 is lower among all other ANN models as shown in table 4.1 The optimum structure of the ANN model was found to be 8 neurons in the hidden layer.

4.3.2 Radial Basis Function Network

The function 'newrb' iteratively creates a radial basis network with one neuron at a time. Neurons are added to the network until the sum-squared error falls beneath an error goal or a maximum number of neurons has been reached. The call for this function is :

net = newrb(P,T,GOAL,SPREAD)

The function 'newrb' takes matrices of input and target vectors P and T, and design parameters GOAL and SPREAD, and returns the desired network. At each iteration the input vector that results in lowering the network error the most is used to create a radbas neuron. The error of the new network is checked, and if low enough newrb is finished. Otherwise the next neuron is added. This procedure is repeated until the error goal is met or the maximum number of neurons is reached. It is important that the spread parameter be large enough that the radbas neurons respond to overlapping regions of the input space, but not so large that all the neurons respond in essentially the same manner.

Table 4.2: Results of RBF model during Calibration and Validation

Model number	Input calibration	ANN structure	Goal	Spread	Calibration			Validation		
					CORR	RMSE	EFF (%)	CORR	RMSE	EFF (%)
RBF1	bhaR(t), berR(t), kahR(t), kasR(t), namR(t), rakR(t), ramR(t), sunR(t), bhaE(t), kasD(t), ramD(t), sunD(t), bhaD(t-1)	13-10-1	0.1	0.1	0.41	440.35	16.54	0.44	341.01	13.48
RBF2		13-20-1	0.1	0.2	0.71	340.13	50.21	0.76	248.00	54.24
RBF3		13-30-1	0.1	0.3	0.81	300.85	61.04	0.87	189.78	73.20
RBF4		13-40-1	0.1	0.4	0.89	216.87	79.76	0.93	140.94	85.22
RBF5		13-50-1	0.1	0.5	0.92	188.11	84.77	0.95	123.28	88.69
RBF6		13-60-1	0.1	0.6	0.94	163.05	88.56	0.96	106.51	91.56
RBF7		13-70-1	0.1	0.7	0.95	147.07	90.69	0.97	92.71	93.60
RBF8		13-80-1	0.1	0.8	0.97	120.04	93.50	0.98	79.65	95.28
RBF9		13-90-1	0.1	0.9	0.97	111.87	93.61	0.97	87.99	97.24
RBF10		13-100-1	0.1	1	0.97	111.98	93.60	0.98	73.04	96.03

Note: bhaR = rainfall at Bhakra; berR = rainfall at Berthin; kah:rainfall at Kahu; kasR = rainfall at Kasol; namR = rainfall at Namgia; rakR = rainfall at raksham; ramR = rainfall at rampur; sunR = rainfall at Suni; bhaE = evaporation at Bhakra; kasD = discharge at Kasol; ramD = discharge at Rampur; sunD = discharge at Suni; bhaD = discharge at Bhakra.

The model RBF9 performed better than other models during calibration (CORR = 0.97, RMSE = 111.87, EFF = 93.61%) and validation (CORR = 0:97, RMSE = 87.99, EFF = 97.24 %) because the coefficients of correlation and Efficiency (%) of RBF9 is higher among all other RBF models and RMSE of RBF9 is lower among all other RBF models as shown in table 4.2. The optimum structure of the RBF model was found to be 90 neurons in the hidden layer.

4.3.3 Fuzzy Logic

The Radius is found by trial and error procedure starting with 0.1 and then increasing it up to 1, based on the performance criteria of the model.

Table 4.3: Results of Fuzzy logic model during Calibration and Validation

Model No.	Input Combinations	Radius	Calibration			Validation		
			CORR	RMSE	EFF (%)	CORR	RMSE	EFF (%)
FUZZY1	bhaR(t), berR(t), kahR(t), kasR(t), namR(t), rakR(t), ramR(t), sunR(t), bhaE(t), kasD(t), ramD(t), sunD(t), bhaD(t-1)	0.1	0.27	638.69	75.57	0.21	493.96	81.54
FUZZY2		0.2	0.98	88.53	96.63	0.99	62.47	97.10
FUZZY3		0.3	0.98	90.77	96.45	0.99	61.46	97.19
FUZZY4		0.4	0.98	90.77	96.45	0.99	61.46	97.19
FUZZY5		0.5	0.98	97.90	96.67	0.99	58.44	97.46
FUZZY6		0.6	0.98	87.65	96.69	0.99	57.59	97.52
FUZZY7		0.7	0.98	87.78	96.68	0.99	57.64	97.53
FUZZY8		0.8	0.98	87.42	96.71	0.99	57.03	97.58
FUZZY9		0.9	0.98	87.58	96.70	0.99	57.18	97.57
FUZZY 10		1.0	0.98	90.77	96.45	0.99	61.46	97.19

Note: bhaR = rainfall at Bhakra; berR = rainfall at Berthin; kah:rainfall at Kahu; kasR = rainfall at Kasol; namR = rainfall at Namgia; rakR = rainfall at raksham; ramR = rainfall at rampur; sunR = rainfall at Suni; bhaE = evaporation at Bhakra; kasD = discharge at Kasol; ramD = discharge at Rampur; sunD = discharge at Suni; bhaD = discharge at Bhakra.

The model FUZZY8 performed better than other models during calibration (CORR = 0.98, RMSE = 87.42, EFF = 96.71%) and validation (CORR = 0:99, RMSE = 57.03, EFF = 97.58 %) because the coefficients of correlation and Efficiency (%) of FUZZY8 is higher among all other FUZZY models and RMSE of FUZZY8 is lower among all other FUZZY models as shown in table 4.3. The optimum structure of the FUZZY model was found to be 90 neurons in the hidden layer.

4.4 ANALYSIS OF RESULTS OF ANN, RBF AND FUZZY MODELS

The performance of the best ANN, RBF and FUZZY models for the prediction of runoff at Bhakra during calibration and validation is presented in Figure 4.15 to 4.20. The scatter plots clearly demonstrate the potentiality of the developed ANN, RBF and FUZZY models in the prediction of runoff.

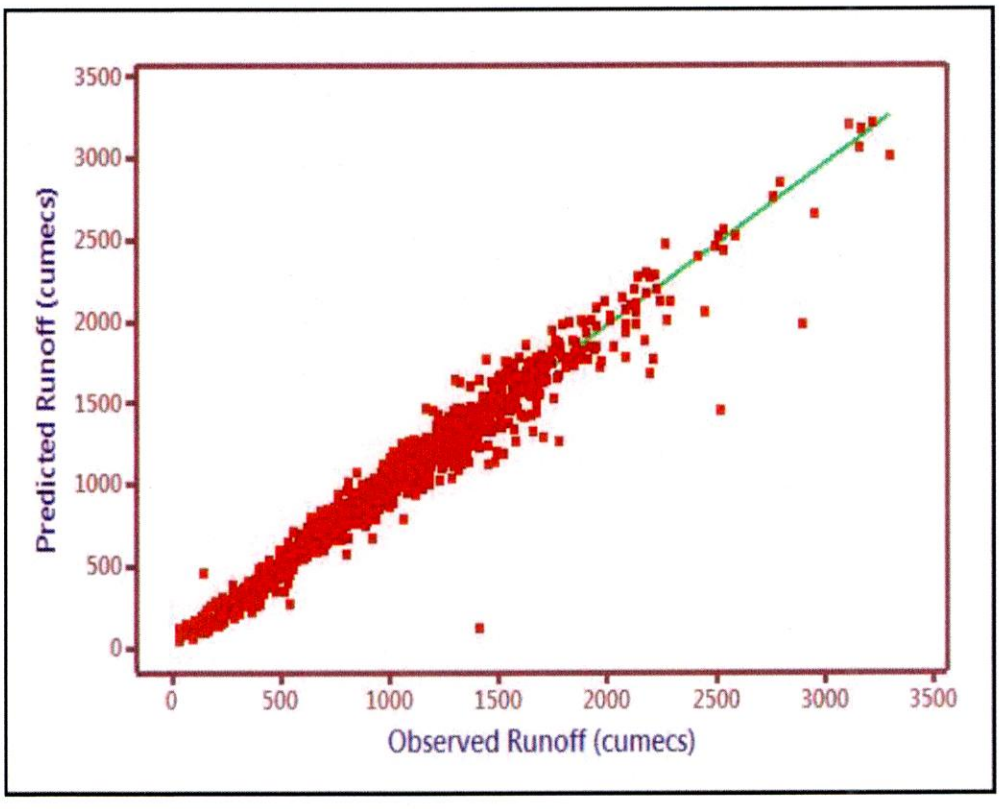

Figure 4.15: Scatter plot for the result of best ANN model during calibration

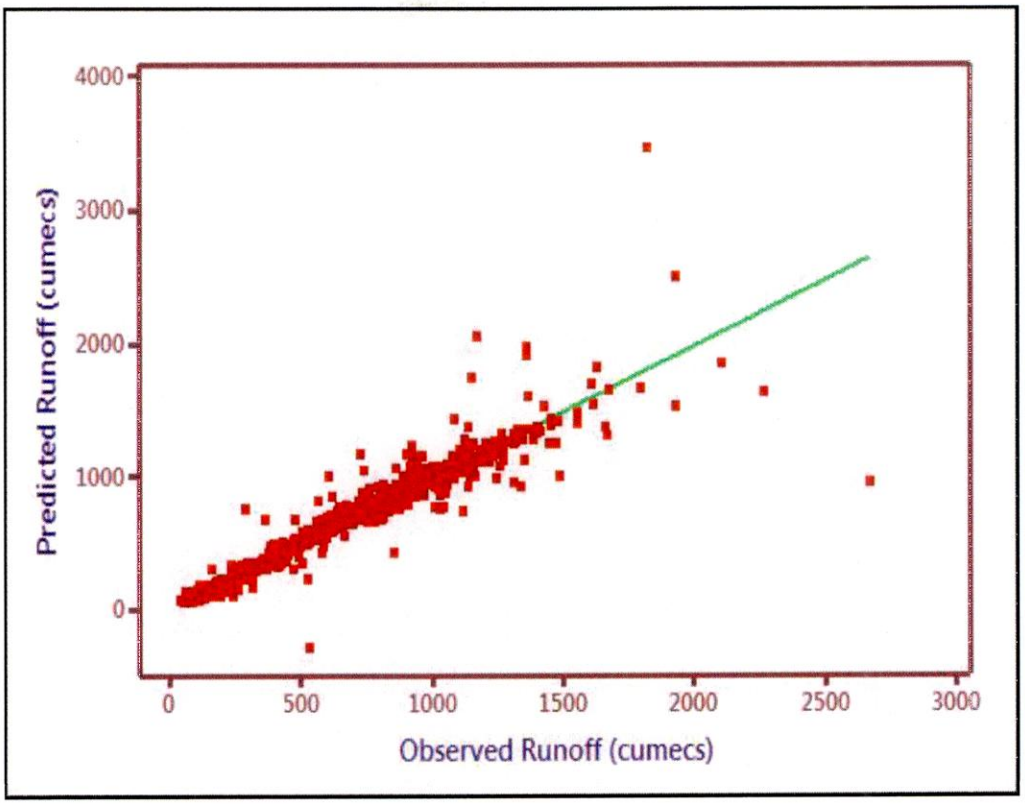

Figure 4.16: Scatter plot for the result of best ANN model during validation

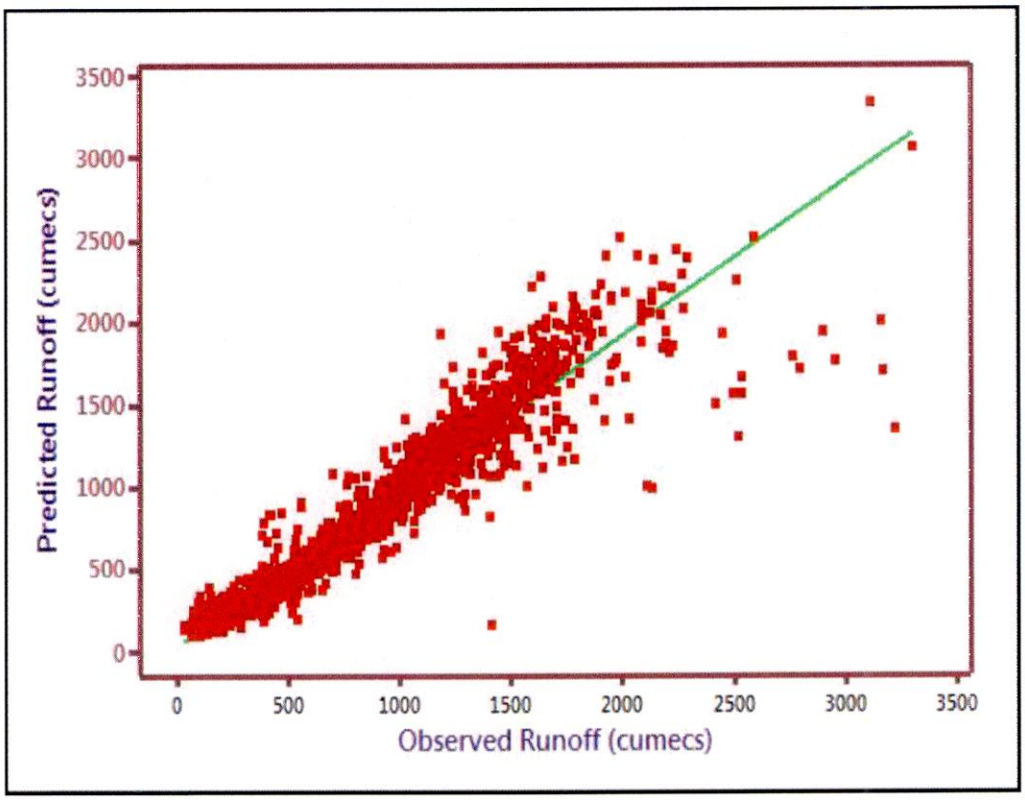

Figure 4.17: Scatter plot for the result of best RBF model during calibration

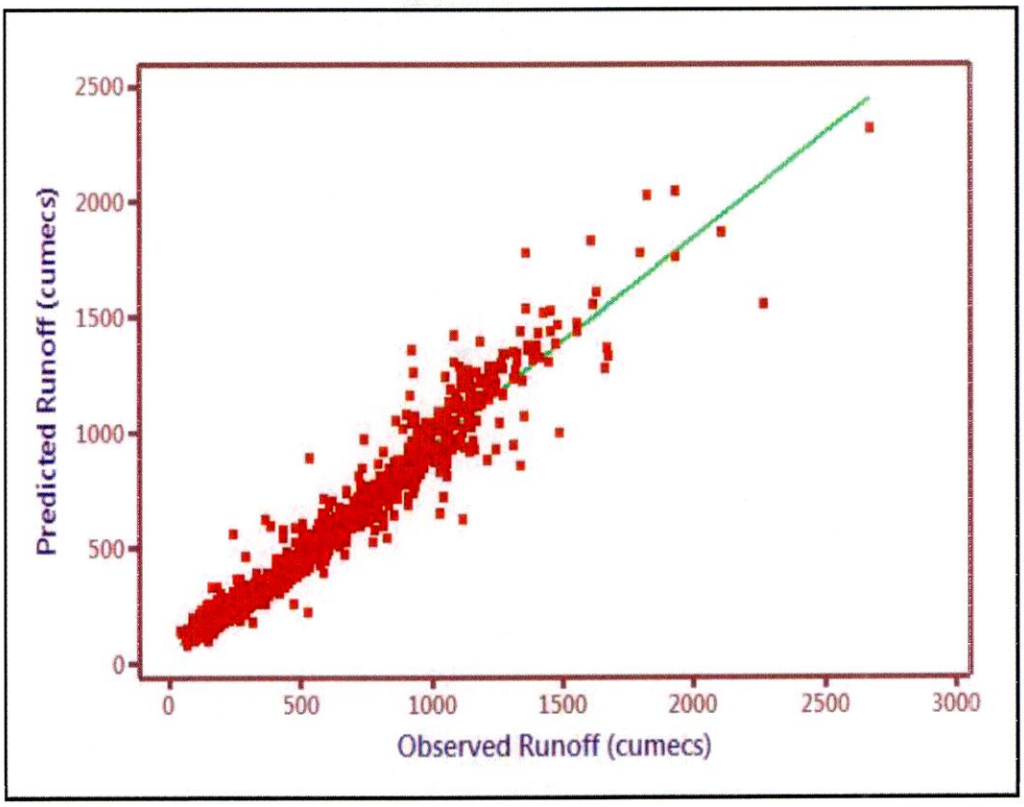

Figure 4.18: Scatter plot for the result of best RBF model during validation

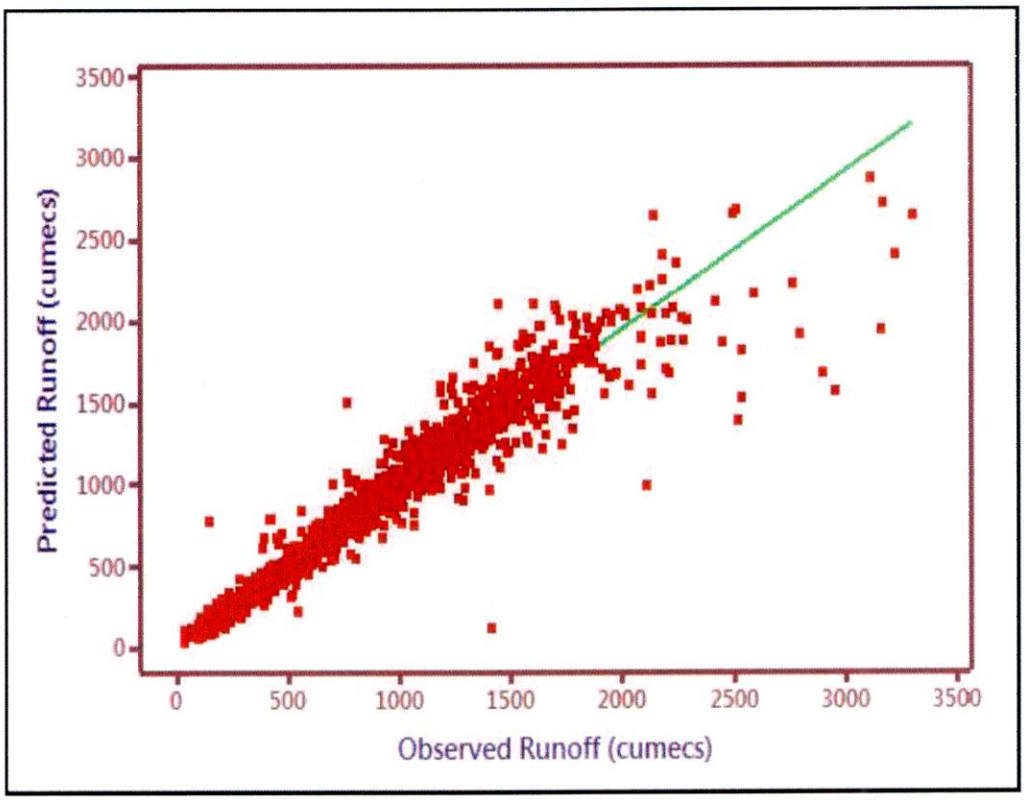

Figure 4.19: Scatter plot for the result of best FUZZY model during calibration

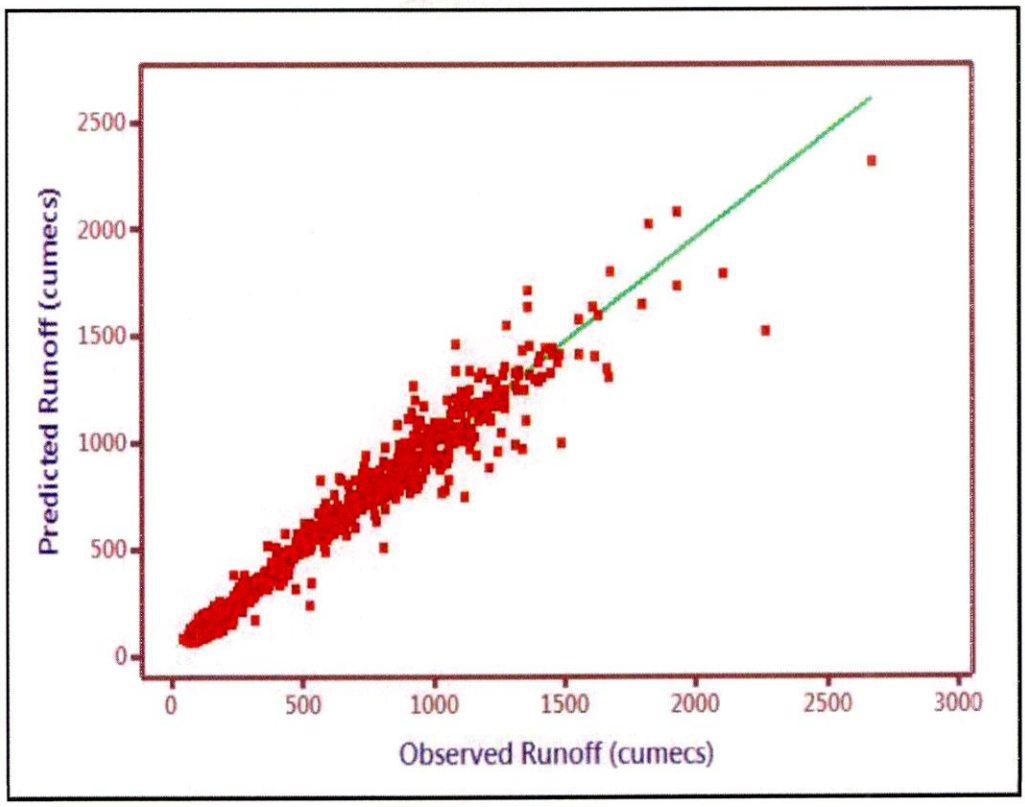

Figure 4.20: Scatter plot for the result of best FUZZY model during validation

4.5 COMPARISON OF RESULTS AMONG BEST ANN, RBF AND FUZZY MODELS.

4.5.1 Calibration/Training Results

During calibration, the coefficient of correlation for the ANN model (ANN8) is 0.99, whereas, the value of the coefficient of correlation for the RBF model (RBF9) is 0.97 and for the FUZZY model (FUZZY8) is 0.98. RMSE for ANN model (ANN8) is 54.24 and for RBF model (RBF9) is 111.87; whereas, the FUZZY model (FUZZY8) is 87.42. Therefore, the ANN model performed best (54.24) and RBF model (RBF9) performed worst (0.82) than all other models investigated in this study. Regarding efficiency, the ANN model (ANN8) performed the best (98.56%), whereas RBF model (RBF9) performed the worst (93.61%).

4.5.2 Validation/Testing Results

During Validation, the coefficient of correlation for the ANN model (ANN8) is 0.98, whereas, the value of the coefficient of correlation for the RBF model (RBF9) is 0.97 and for the FUZZY model (FUZZY8) is 0.99. RMSE for ANN model (ANN8) is 56.18 and for RBF model (RBF9) is 87.99; whereas, the FUZZY model (FUZZY8) is

57.03. Therefore, the ANN model performed best (56.18) and RBF model (RBF9) performed worst (87.99) than all other models investigated in this study. Regarding efficiency, the ANN model (ANN8) performed the best (97.69%), whereas RBF model (RBF9) performed the best (97.24%).

4.5.3 Overall results

From performance indexes, it was observed that the ANN model learned the process better than any other model during calibration and validation. Thus, it can be concluded from the overall performance of the models that the ANN model (ANN8) performed the best and the RBF (RBF9) model performed the worst during both calibration and validation.

The results of the calibration and validation of the best ANN, RBF and FUZZY models in terms of various statistical indices are presented in the Table 4.4. The performance of the best ANN, RBF and FUZZY models in terms of observed and predicted runoff at Bhakra during calibration and validation is presented in Figure 4.21 and 4.22 respectively.

Table 4.4: Comparison of results among the best ANN, RBF and FUZZY Logic models during calibration and validation.

Model	Calibration			Validation		
	CORR	RMSE	EFF (%)	CORR	RMSE	EFF (%)
ANN8	0.99	54.24	98.56	0.98	56.18	97.69
RBF9	0.97	111.87	93.61	0.97	87.99	97.24
FUZZY8	0.98	87.42	96.71	0.99	57.03	97.58

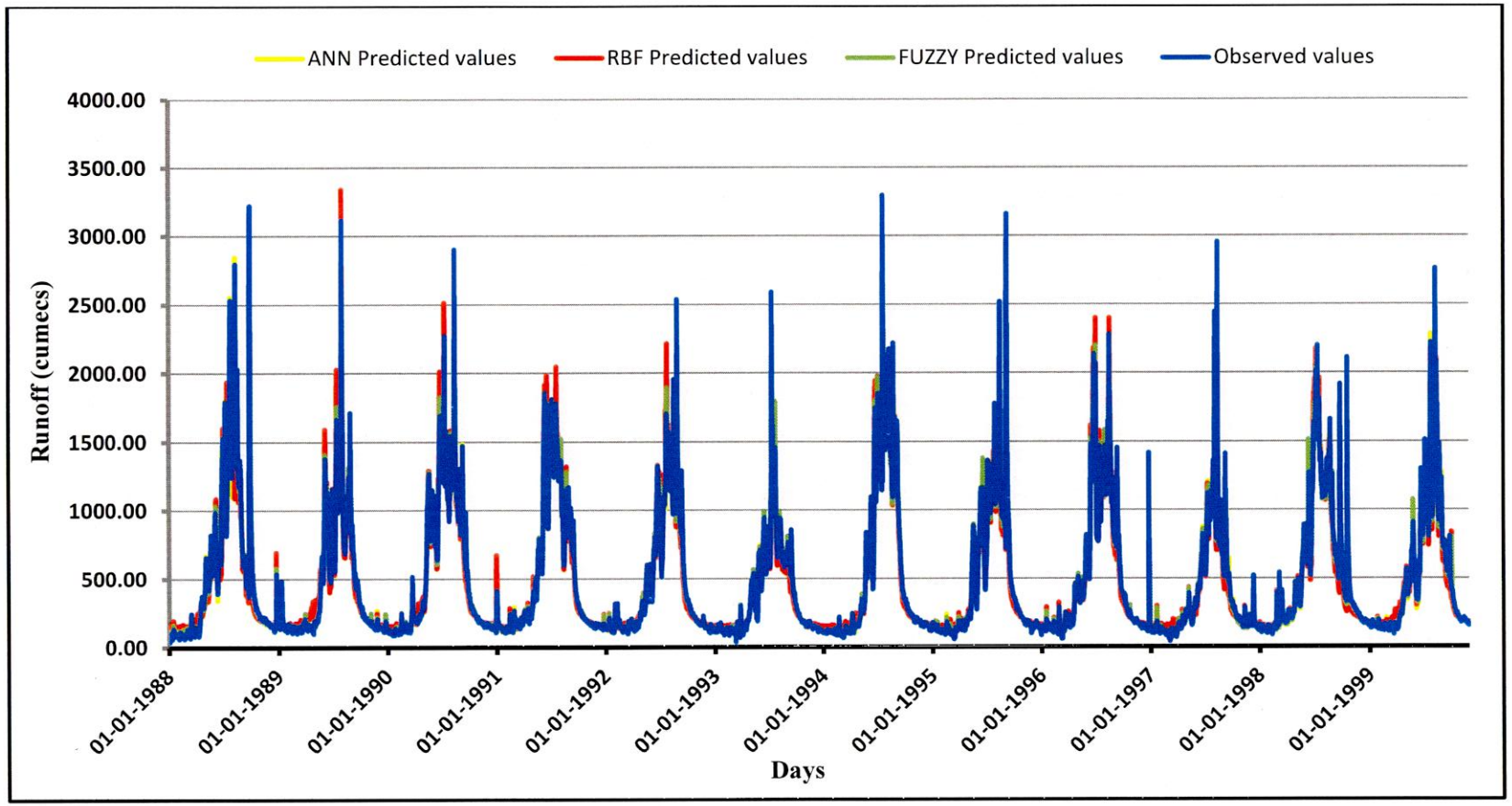

Figure 4.21: Graph showing predicted runoff values by ANN, RBF and Fuzzy logic against observed runoff values during calibration at Bhakra

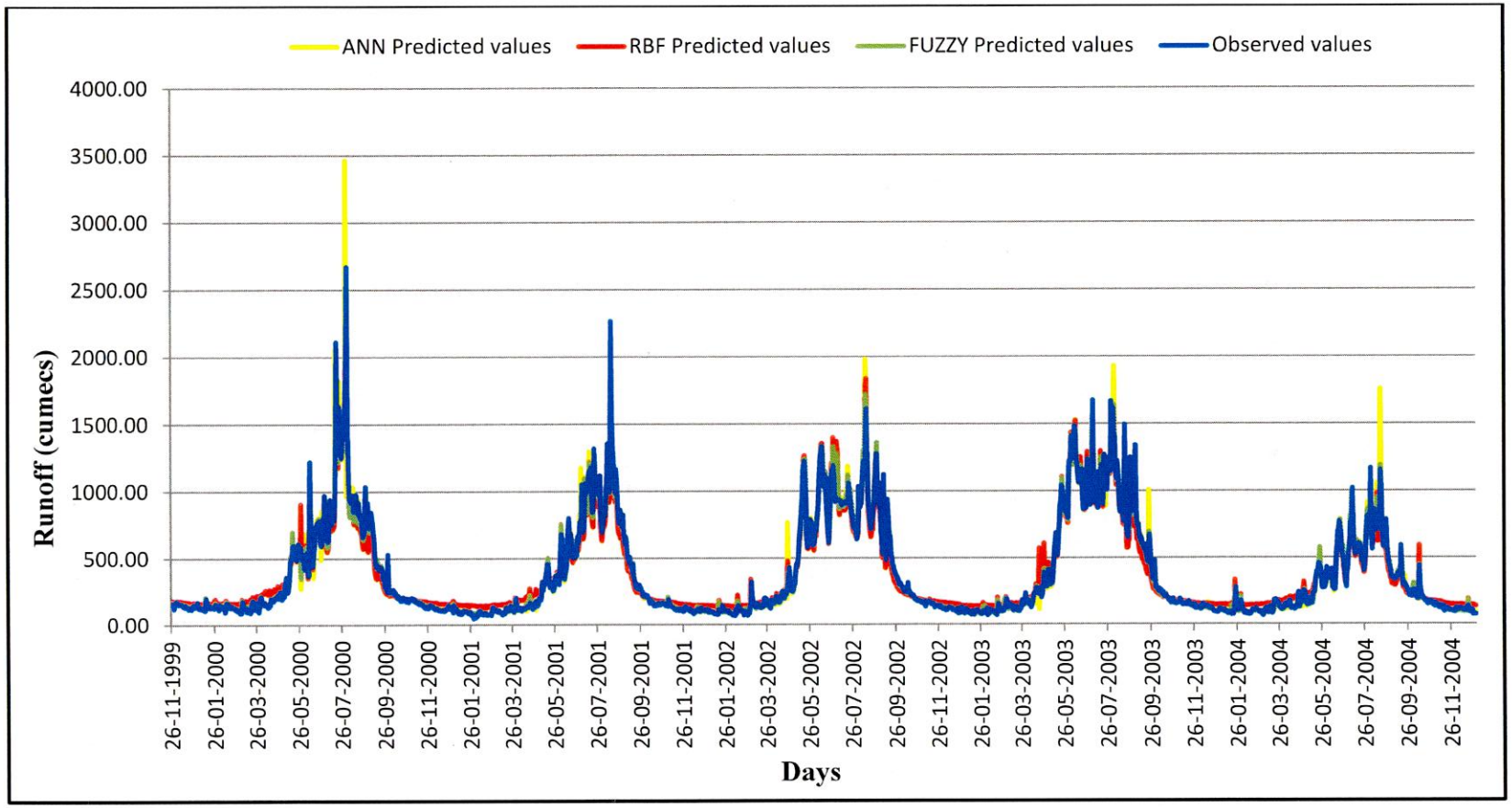

Figure 4.22: Graph showing predicted runoff values by ANN, RBF and Fuzzy logic against observed runoff values during validation at Bhakra

CHAPTER 5

CONCLUSIONS

Rainfall - runoff prediction models using Artificial Neural Network (ANN) with back propagation algorithm, Radial Basis Function (RBF), Fuzzy Logic are developed. After training and validation of these models, comparison of results among the best ANN model by Back Propagation, Radial Basis Function and FUZZY Logic during calibration and validation were made.

The analysis of the performance of ANN model by Back Propagation, FUZZY Logic and Radial Basis Function clearly indicate that the application of ANN model by Back Propagation helps in the better prediction of runoff at Bhakra in Sutlej river basin.

The performance of the models reveal that these models are able to predict the evaporation and runoff with adequate accuracy.

REFERENCES

Abebe, A.J., Solomatine, D.P. and Venneker, R.G.W. (2000) 'Application of adaptive fuzzy rule-based models for reconstruction of missing precipitation events', *Hydrological Sciences-Journal-des Sciences Hydrologiques*, 45 (3), pp. 425–436.

Abrahart, R. J. and Kneale, P. E. (1997) 'Exploring neural network rainfall–runoff modelling', *Sixth National Hydrology Symposium.* Manchester, UK, 15-18 September 1997. Greater Manchester: University of Salford.

Abrahart, R. J. & See, L. (2000) 'Comparing neural network and auto regressive moving average techniques for the provision of continuous river flow forecasts in two contrasting catchments', *Hydrological Processes*, 14, pp. 2157–2172.

Ahmad, S. and Simonovic, S. P. (2005) 'An Artificial Neural Network Model for Generating Hydrograph From Hydro-Meteorological Parameters', *Journal of Hydrology*, 315(1), pp. 236-251.

Alvisi S., Mascellani G., Franchini M. and Bárdossy, A. (2006) 'Water level forecasting through fuzzy logic and articial neural network approaches', *Hydrology and Earth System Sciences*, 10(1), pp. 1–17.

Anctil, F., Perrin, C., Andreassian, V. (2004) 'Impact of the length of observed records on the performance of ANN and of conceptual parsimonious rainfall-runoff forecasting models', *Environmental Modelling* and *Software*, 19(4), pp. 357–368.

ASCE Task Committee on Application of Artificial Neural Networks in Hydrology (2000a) 'Artificial neural networks in hydrology-I: Preliminary concepts', *Journal of Hydrologic Engineering*, 5(2), pp. 115- 123.

ASCE Task Committee on Application of Artificial Neural Networks in Hydrology (2000b) 'Artificial neural networks in hydrology-II: Hydrologic applications', *Journal of Hydrologic Engineering*, 5(2), pp. 124–137.

Bárdossy, A., Mascellani, G. and Franchini, M. (2006) 'Fuzzy unit hydrograph', *Water Resources Research*, 42(2), pp. 3-10.

Bonafe, A., Galeati, G. and Sforna, M. (1994) 'Neural networks for daily mean flow forecasting', *Hydraulic Engineering Software*, 1, pp. 131–138.

Burian, J.S., Durrans, S.R., Nix, S.J. and Pitt, R.E. (2001) 'Training artificial neural networks to perform rainfall Disaggregation', *Journal of Hydrologic Engineering*, 6(1), pp. 43-51.

Campolo, M., Soldati, A. & Andreussi, P. (2003) 'Artificial neural network approach to flood forecasting in the River Arno', *Hydrological Sciences Journal*, 48(3), pp. 381-398.

Carriere, P., Mohaghegh, S. and Gaskari, R. (1996) 'Performance of a virtual runoff hydrograph system.', *Journal of Water Resources Planning and Management*, 122(6), pp. 421–427.

Caudill, M. (1987) 'Neural networks primer I', *AI Expert.*

Caudill, M. (1988) 'Neural networks primer II, III, IV and V', *AI Expert.*

Caudill, M. (1989) 'Neural networks primer VI, VII and VIII', *AI Expert.*

Chang, F.J. and Chen, Y.C. (2001) 'A counter propagation fuzzy neural network modeling approach to real time streamflow prediction', Journal of *Hydrology*, 245, pp. 153–164.

Chang, L.C., Chang F. J. and Sai, Y. H. T. (2005) 'Fuzzy exemplar-based inference system for flood forecasting', *Water Resources* Research, 41(2), pp. 41.

Chen, S., Cowan, C. F. N. and Grant, P. M. (1991) 'Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks', *IEEE Transactions on Neural Networks*, 2(2), pp. 302-309.

Chiu, S.L. (1994) 'Fuzzy model identification based on cluster estimation', *Journal of Intelligent and Fuzzy Systems*, 2, pp. 267–278.

Cigizoglu, H.K. and Alp, M. (eds.), (2004) *Artificial Intelligence and Soft Computing-ICAISC 2004*. Zakopane, Poland, 7-11 June. Zakopane: Springer.

Coulibaly, P., Anctil, F., Aravena, R. and Bobée B. (2000a) 'Artificial neural network modeling of water table depth fluctuations', *Water Resources Research,* 37(4), pp. 885–896.

Coulibaly, P., Anctil, F., Rasmussen, P. and Bobée, B. (2000) 'A recurrent neural network approach using indices of low frequency climatic variability to forecast regional annual runoff', *Hydrological Processes*, 14, pp. 2555–2577.

Daniel, T. M. (1991) 'Neural networks-applications in hydrology and water resources engineering', *Proc., International* Hydrology and *Water Resource Symposium*. Perth: Institution of Engineers.

Dawson, C. W. and Wilby, R.L. (1998) 'An artificial neural network approach to rainfall–runoff modeling', *Hydrological Sciences Journal*, 43(1), pp. 47–66.

Dawson, C.W. and Wilby, R.L. (2001) 'Hydrological modelling using artificial neural networks', *Progress in Physical Geography*, 25(1), pp. 80-108.

Deka P. and Chandramouli V. (2003) 'A fuzzy neural network model for deriving the river stage-discharge relationship', *Hydrological Sciences Journal,* 48(2), pp. 197-209.

Deka, P. and Chandramouli, V. (2005) 'A fuzzy neural network model for deriving the river stage—discharge relationship', *Hydrological sciences journal,* 48(2), pp. 197-209.

De Vos N. J. and Rientjes T. H. M. (2005) 'Constraints of artificial neural networks for rainfall-runoff modelling: trade-offs in hydrological state representation and model evaluation', *Hydrology and Earth System Sciences,* 9, pp. 111–126.

Dhar, V., Roger, S. (1997) *Seven methods for transforming corporate data into business intelligence.* New Jersey: Prentice-Hall.

Dibike, Y. B. and Solomatine, D. P. (1999) 'River flow forecasting using artificial neural network', *European Geophysical Society XXIV General Assembly*. The Hague, The Netherlands.

Elshorbagy, A., Simonovic, S. P. and Panu, U. S. (2000) 'Performance evaluation of artificial neural networks for runoff prediction', *Journal of Hydrologic Engineering*, 5, pp. 424-427.

Fausett, L. (1994) *Fundamentals of Neural Networks*. New Jersey: Prentice Hall.

Fernando, D. A. K. and Jayawardena, A. W. (1998) 'Runoff forecasting using RBF networks with OLS algorithm', *Journal of Hydrologic Engineering*, 3(3), pp. 203–209.

Fernando, D. A. K. and Shamseldin, A. Y. (2009) 'Investigation of internal functioning of the radial-basis-function neural network river flow forecasting models', *Journal of Hydrologic Engineering*, 14(3), pp. 286–292.

Fiordaliso, A. (1998) 'A nonlinear forecasts combination method based on

Takagi–Sugeno fuzzy systems', *International Journal of Forecasting,* 14, pp. 367–369.

Firat, M. and Güngör, M. (2007) 'River flow estimation using adaptive neuro-fuzzy inference system', *Mathematics and Computers in Simulation*, 75 (3-4), pp. 87–96.

French, M. N., Krajewski, W. F., and Cuykendall, R. R. (1992) 'Rainfall forecasting in space and time using neural network', *Journal of Hydrology*, 137, pp. 1–31.

Gill, S.E., Handley J.F., Ennos, A.R. and Pauleit, S. (2007) 'Adapting cities for climate change: the role of the green infrastructure', *Indoor and Built Environment,* 33(1), pp. 115–133.

Govindaraju, R.S. (2000) 'Artificial neural networks in hydrology II: hydrologic Applications', *Journal of Hydrologic Engineering*, 5(2), pp. 124-137.

Govindaraju, R. S. and Rao, A. R. (eds.) (2000) *Artificial Neural Network in Hydrology.* Dordrecht: Kluwer.

Gowda, C. C. and Moyya, S. D. (2014) 'Runoff Modeling using different member ship functions in Adaptive Neuro fuzzy inference system', *International Journal of Advances in Engineering Sciences*, 4(4), pp. 48-51.

Halff, A. H., Halff, H. M. and Azmoodeh, M. (1993) 'Predicting from rainfall using neural networks', *Proceedings of Engineering Hydrology*, pp. 760-765.

Haykin, S. (1994) *Neural networks: a comprehensive foundation.* New York: Mac-Millan.

Haykin, S. (1999) *Neural networks: A comprehensive foundation.* New Jersey: Prentice Hall.

Hecht-Nielsen, R. (1990) *Neurocomputing.* Addison-Wesley.

Hong, Y. S., Rosen, M. R. and Reeves, R. R. (2002) 'Dynamic fuzzy modeling of storm water infiltration in urban fractured aquifers', *Journal* of *Hydrologic Engineering*, 7(5), pp. 380–391.

Hsu, K., Gupta, H. V. and Sorooshian, S. (1995) 'Artificial neural network modeling of the rainfall–runoff process', *Water Resources Research*, 31, pp. 2517–2530.

Hundecha, Y., Bárdossy, A. and Theisen, H. (2001) 'Development of a fuzzy logic-based rainfall–runoff model', *Hydrological Sciences-Journal-des Sciences Hydrologiques,* 46 (3), pp. 363–376.

Imrie, C. E., Durucan, S. and Korre, A. (2000) 'River flow prediction using artificial neural network generalisation beyond calibration range' *Journal of Hydrology*, 233, pp. 138–153.

Jacquin, A.P. and Shamseldin, A.Y. (2006) 'Development of rainfall–runoff models using Takagi–Sugeno fuzzy inference systems', *Journal of Hydrology,* 329, pp. 154–173.

Jain, A., and Srinivasulu, S. (2004) 'Development of effective and efficient rainfall-runoff models using integration of deterministic, real-coded genetic algorithms and artificial neural network techniques', *Water Resources Research*, 40(4), pp. 1–12.

Jain, S. K. (2008) 'Development of integrated sediment rating curves using ANNs', *Journal of Hydrologic Engineering*, 13(3), pp. 124–131.

Kagoda, P. A., Ndiritu, J., Ntuli, C. and Mwaka, B. (2010) 'Application of radial basis function neural network to short term stream flow forecasting', *Physics and Chemistry of the Earth*, 35, pp. 571-581.

Keskin, M. E. and Terzi, Ö. (2006) 'Artificial Neural Network Models of Daily Pan Evaporation', *Journal of Hydrologic Engineering,* 11(1), pp. 65-70.

Kisi, Ö. (2007) 'Streamflow forecasting using different artificial neural network algorithms', *Journal of Hydrologic Engineering,* 12(5), pp. 532–539.

Klir, J. and Foger, T. A. (1988) *Fuzzy sets, uncertainty, and information*. New Jersey: Prentice-Hall.

Kumar, D.N., Raju, K.S. and Sathish, T. (2004) 'River flow forecasting using recurrent neural networks', *Water Resources Management*, 18, pp. 143-161.

Lekkas, D. F., Lees, M. J. and Imrie, C. E. (2001) 'Improved nonlinear transfer function and neural network methods of flow routing for real-time forecasting', *Journal of Hydroinformatics*, 3(3), pp. 153–164.

Lohani, A.K., Goel, N.K. and Bhatia, K.K.S. (2006) 'Takagi–Sugeno fuzzy inference system for modeling stage–discharge relationship', *Journal of Hydrology,* 331, pp. 146–160.

Maier, H. R., and Dandy, G. C. (2000) 'Neural networks for the prediction and forecasting of water resources variables: a review of modeling issues and applications', *Environmental* Modelling & *Software*, 15, pp. 101–124.

Mamdani, E.H. (1974) 'Application of fuzzy algorithms for control of simple dynamic plant', *Proceedings IEE*, 121 (12), pp. 1585–1588.

Mamdani E. H. and Assilian, S. (1975) 'An experiment in linguistic synthesis with a fuzzy logic controller', *International Journal of Man–Machine Studies*, 7(1), pp. 1–13.

Maskey, S., Guinot, V., Proce, k.k. (2004) 'Treatment of precipitation uncertainty in rainfall-runoff modeling: A fuzzy set approach', *Advances in Water Resources*, 27(9), pp. 889–898.

Mason, J. C., Price, R. K. and Tem'me, A. (1996) 'A neural network model of rainfall-runoff using radial basis functions', *Journal of Hydraulic Research*, 34(4), 537–548.

Minns, A. W. & Hall, M. J. (1996) 'Artificial neural network as rainfall–runoff model', *Hydrological Sciences Journal*, 41(3), pp. 399–417.

Mitchell, T.M. (1997). *Machine learning*. Singapore: McGraw-Hill.

McCulloch, W. and Pitts, W. (1943) 'A logical calculus of the ideas immanent in nervous activity', *Bulletin of Mathematical Biophysics*, 5(4), pp. 115-133.

Mukarji A., Chatterjee C. and Raghuwanshi, N. S. (2009) 'Flood Forecasting Using ANN, Neuro-Fuzzy, and Neuro-GA Models', *Journal of Hydrologic Engineering*, 14(6), pp. 647-652.

Nagy, H. M.,Watanabe, K. and Hirano, M. (2002) 'Prediction of sediment load concentration in rivers using artificial neural networks', *Journal of Hydrologic Engineering*, 128(6), pp. 588–595.

Nash, J. E. and Sutcliffe, J. V. (1970) 'River flow forecasting through conceptual models: 1. A discussion of principles', *Journal of Hydrology*, 10(3), pp. 282–290.

Nayak, P. C., Satyaji Rao, Y. R. and Sudheer, K. P. (2006) 'Groundwater level forecasting in a shallow aquifer using artificial neural network approach', *Water Resources Management*, 20, pp. 77–90.

Nayak, P.C., Sudheer, K.P., Rangan, D.M. and Ramasastri, K.S. (2004) 'A neuro-fuzzy computing technique for modelling hydrological time series', *Journal of Hydrology*, 291, pp. 52–66.

Ojha, C. S. P., Goyal, M. K. and Kumar, S. (2007) 'Applying Fuzzy logic and the point count system to select landfill sites', *Environmental Monitoring* and *Assessment*, 135(1–3), pp. 99–106.

Ozelkan, E.C. and Duckstein, L. (2001) 'Fuzzy conceptual rainfall–runoff models', *Journal of Hydrology*, 253, pp. 41–68.

Parasuraman, K., Elshorbagy, A. and Carey, S.K. (2006) 'Spiking modular neural networks: A neural network modeling approach for hydrological processes', *Water Resources Research*, 42, pp. 21-24.

Pedrycz, W. and Gomide, F. (1994) 'A generalized fuzzy Petri net model', *IEEE Transactions on Fuzzy Systems*, 2(4), pp. 295-301.

Persson, M. and Berndtsson, R. (2001) 'Using neural networks for calibration of time-domain reflectometry measurements', *Hydrological Sciences Journal*, 46(3), pp. 389–398.

Porter, D.W., Gibbs, P.G., Jones, W.F., Huyakorn, P.S., Hamm, L.L. and Flach, G.P. (2000) 'Data fusion modeling for groundwater systems', *Journal of Contaminant Hydrology*, 42, pp. 303–335.

Principe, J., Euliano, N. and Lefebvre, W. (1999) *Neural and adaptive systems: Fundamentals through simulations with CD-ROM.* New York: John Wiley & Sons.

Rajurkar, M. P., Kothyari , U. C. and Chaube, U. C. (2002) 'Artificial neural networks for daily rainfall–runoff modellin', *Hydrological Sciences Journal*, 47(6), pp. 865–877.

Rajurkar, M. P., Kothyari, U. C. and Chaube, U. C. (2004) 'Modeling of the daily rainfall– runoff relationship with artificial neural network', Journal of *Hydrology*, 285, pp. 96–113.

Rumelhart, D. E., Hinton, G. E. and McClelland, J. L. (1986) 'A general framework for parallel distributed processing', *Parallel Distributed Processing: explorations in the microstructure of cognition*, 1, pp. 45-76, Cambridge: MIT Press.

Sajikumar, N. and Thandaveswara, B. S. (1999) 'A non-linear rainfall–runoff model using artificial neural network', *Journal of Hydrology*, 214, pp. 32–48.

Salas, J. D., Delleur, J. W., Yevjevich, V. and Lane, W. L. (1980) *Applied Modeling of Hydrological Time Series*. Denver: Water Resources Publications.

See, L. and Openshaw, S. (2000) 'A hybrid multi-model approach to river level forecasting', *Hydrological Sciences-Journal-des Sciences Hydrologiques,* 45 (4), pp. 523–536.

Senthil Kumar, A. R., Ojha, C. S. P., Manish Kumar Goyal, Singh, R. D. and Swamee, P. K. (2012) 'Modeling of Suspended Sediment Concentration at Kasol in India Using ANN, Fuzzy Logic, and Decision Tree Algorithm', *Journal of Hydrologic Engineering*, 17(3), pp. 295-403.

Senthil Kumar, A.R., Sudheer, K.P., Jain, S.K. and Agarwal, P.K. (2005) 'Rainfall-runoff modelling using artificial neural networks: comparison of network types', *Hydrological Process*, 19, pp. 1277-1291.

Shamseldin, A. Y. (1997) 'Application of a neural network technique to rainfall–runoff modelling', *Journal of Hydrology,* 199, pp. 272–294.

Shamseldin, A. Y. and O'Connor, K. M. (2001) 'A non-linear neural network technique for updating river flow forecasts', *Hydrology and Earth System Sciences*, 5(4), pp. 577–597.

Singh, P. and Jain, S. K. (2002) 'Snow and glacier contribution in the Sutlej river at Bhakra Dam in the Western Himalayan region', *Hydrological Sciences Journal,* 47(1), pp. 93–106.

Singh, P. and Kumar, N. (1997b) 'Effect of orography on precipitation in the western Himalayan region', *Journal of Hydrology,* 199, pp. 183–206.

Smith, J. and Sli, R. N. (1995) 'Neural network model for rainfall runoff process', *Journal of Water Resources Planning and Management*, 121, pp. 49-508.

Solaimani, K. (2009) 'A Study of Rainfall Forecasting Models Based on Artificial Neural Network', *Asian Journal of Applied Sciences,* 2(6), pp. 486-498.

Srinivasulu, S. and Jain, A. (2006) 'A comparative analysis of training methods for artificial neural network rainfall–runoff models', *Applied Soft Computing* 6, pp. 295–306.

Srinivasulu, S. and Jain, A. (2009) 'River flow prediction using an integrated approach', *Journal of Hydrologic Engineering*, 14(1), pp. 75–83.

Sudheer, K. P., Gosain, A. K. and Ramasastri, K. S. (2002) 'A data-driven algorithm for constructing artificial neural network rainfall-runoff models', *Hydrological Processes,* 16(6), pp. 1325–1330.

Suhaimi, S. and Bustami, R.A. (2009) 'Rainfall Runoff Modeling using Radial Basis Function Neural Network for Sungai Tinjar Catchment, Miri, Sarawak', *UNIMAS E-Journal of Civil Engineering,* 1(1).

Sugeno, M. and Kang, G. (1988) 'Structure identification of fuzzy model', *Fuzzy Sets and Systems*, 26(1), pp. 15–33.

Takagi, T. and Sugeno, M. (1985) 'Fuzzy identification of systems and its applications to modeling and control', *IEEE Transactions* on *Systems, Man, and Cybernetics: Systems*, 15 (1), pp. 116–131.

Tan, J., Xie, H. and Lee, Y.C. (1995) 'Efficient establishment of a fuzzy logic model for process modeling and control', *The IEEE Transactions* on *Semiconductor Manufacturing,* 8 (1), pp. 50-61.

Tayfur, G. (2002) 'Artificial neural networks for sheet sediment transport', *Hydrological Sciences Journal*, 47(6), pp. 879–892.

Tayfur G., Moramorco T. and Singh, V.P. (2003) 'Predicting and forecasting flow discharge at sites receiving significant lateral inflow', *Hydrological Processes*, 21, pp. 1848–1859.

Tayfur, G. and Singh, V. P. (2006) 'ANN and fuzzy logic models for simulating event-based rainfall–runoff', *Journal of Hydrologic Engineering*, 132, pp. 1321–1330.

Terano, T., Asai, K. and Sugeno, M. (1987) *Fuzzy Systems Theory and its Applications*. San Diego: Academic Press.

Thirumalaiah, K. and Makarand, D. (2000) 'Hydrological forecasting using neural networks', *Journal of Hydrologic Engineering*, 5(2), pp. 180-189.

Tilmant, A., Vanclooster, M., Duckstein, L. and Persoons, E. (2002) 'Comparison of fuzzy and nonfuzzy optimal reservoir operating policies', *Journal* of *Water Resources Planning* and *Management*, 128(6), pp. 390–398.

Tokar, A.S. and Johnson P.A. (1999) 'Rainfall–runoff modeling using artificial neural networks', *Journal of Hydrologic Engineering*, 4, pp. 232–239.

Vernieuwe, H., Georgieva, O., Baets, B.D., Pauwels, V.R.N., Verhoest, N.E.C. and Troch, D.F.P. (2005) 'Comparison of data-driven Takagi–Sugeno models of rainfall–discharge dynamics', *Journal of Hydrology,* 302, pp. 173–186.

Vivekanandan, N. (2014) 'Prediction of Rainfall Using MLP and RBF Networks', *International Journal of Advanced Networking and Applications*, 5(4), pp. 1974-1979.

Wasserman, P.D. (1989) *Neural computing: theory and practice*. New York: Van Nostrand Reinhold.

Yu, P. and Yang, T. (2000) ' Fuzzy multi-objective function for rainfall-runoff model calibration', *Journal of Hydrology*, 238, pp. 1-14.

Zadeh L. A. (1965) 'Fuzzy sets', *Information and Control*, 8(3), pp. 338–353.

Zealand, C. M., Burn, D. H. & Simonovic, S. P. (1999) 'Short term streamflow forecasting using artificial neural network', *Journal of Hydrology*, 216, pp. 32–55.

Zhang, G. P., Patuwo, B. E. and Hu, M. Y. (2001) 'A simulation study of artificial neural networks for nonlinear time-series forecasting', *Computers & Operations Research,* 28, pp. 381-396.