ANN TOOLS - A CASE STUDY OF RAINFALL-RUNOFF MODELLING USING NEURAL POWER

Archana Sarkar
Scientist 'C'

INTRODUCTION

In recent years ANNs have shown exceptional performance as regression tools, especially when used for pattern recognition and function estimation. They are highly non-linear, and can capture complex interactions among the input variables in a system without any prior knowledge about the nature of these interactions (Hammerstrom, 1993). The main advantage of ANNs is that one does not have to explicitly assume a model form, which is a prerequisite in conventional modelling approaches. Indeed, in ANNs the data points themselves generate a relationship of possibly complicated or orthodox shape. In comparison to the conventional methods, ANNs tolerate imprecise or incomplete data, approximate results, and are less vulnerable to outliers (Haykin, 1994). They are highly parallel, i.e.; their numerous independent operations can be executed simultaneously. These characteristics render ANNs to be very suitable tools for handling various hydrological modelling problems. Although application of ANN approach rainfall-runoff process is recent, it has already produced very encouraging results.

The objective of this presentation is to provide information about some freeware and shareware packages for ANN simulation available on the internet and demonstration of one such package with a case study of ANN modeling of the event-based rainfall-runoff process for a selected Indian catchment.

A rainfall-runoff model can be one of two types - an event based rainfall-runoff model or a continuous rainfall-runoff model. Application of ANN approach for modelling the continuous rainfall-runoff process are numerous but that of modelling the event based process are limited. The present study focusses on the modelling of an event-based rainfall-runoff process. Such ANN models are very useful in real time flood forecasting. In the present study, back propagation ANN models have been designed and developed for the runoff simulation of Ajay river basin at Sarath gauging site in Jharkhand (South Bihar).

The presentation starts with general information on various shareware ANN packages. Then, the salient features of the ANN package Neural Power used in the present case study are described followed by the implementation of the software through development of ANN models for event-based rainfall-runoff process.

FREEWARE AND SHAREWARE PACKAGES FOR ANN SIMULATION

Some ANN packages freely available on the internet alongwith few details and site addresses are given below:

Neural Power

NeuralPower is an easy-to-use yet powerful Artificial Neural Network (ANN) program. NeuralPower can be used in most research fields. Some examples where NeuralPower can be applied are the following: multi-nonlinear regression, forecasting, curve fit, pattern recognition, classification, decision making, problem optimization, time series analysis, etc. Free demo version can be downloaded from http://www.geocities.com/neuralpower/

SNNS

SNNS (Stuttgart Neural Network Simulator) is a software simulator for neural networks on Unix workstations developed at the Institute for Parallel and Distributed High Performance Systems (IPVR) at the University of Stuttgart. The goal of the SNNS project is to create an efficient and flexible simulation environment for research on and application of neural nets. Currently the network architectures and learning procedures included are: Backpropagation (BP), Counterpropagation, Quickprop, Generalized radial basis functions (RBF), ART, Cascade Correlation, Dynamic LVQ, Self-organizing maps, (Kohonen maps). SNNS web page: http://www-ra.informatik.uni-tuebingen.de/SNNS

PDP++

The PDP++ software is a neural-network simulation system written in C++. It represents the next generation of the PDP software released with the McClelland and Rumelhart "Explorations in Parallel Distributed Processing Handbook", MIT Press, 1987. It is easy enough for novice users, but very powerful and flexible for research use. Supported algorithms include: Feedforward and recurrent error backpropagation. Constraint satisfaction algorithms and associated learning algorithms, Self-organizing learning, Mixtures-of-experts using backpropagation experts, EM updating, and a SoftMax gating module. The software can be obtained by anonymous ftp from: ftp://grey.colorado.edu/pub/oreilly/pdp++

NeurDS

Neural Design and Simulation System. This is a general purpose tool for building, running and analysing Neural Network Models in an efficient manner. NeurDS will compile and run virtually any Neural Network Model using a consistent user interface that may be either window or "batch" oriented. HP-UX 8.07 source code is available from http://hpux.u-aizu.ac.jp/hppd/hpux/NeuralNets/NeurDS-3.1/

ALN Workbench (a spreadsheet for Windows)

ALNBench is a free spreadsheet program for MS-Windows (NT, 95) that allows the user to import training and test sets and predict a chosen column of data from the others in the training set. It is an easy-to-use program for research, education and evaluation of ALN technology. Anyone who can use a spreadsheet can quickly understand how to use it. The program can be downloaded from http://www.dendronic.com/main.htm

Multi-Module Neural Computing Environment (MUME)

MUME is a simulation environment for multi-modules neural computing. It provides an object oriented facility for the simulation and training of multiple nets with various architectures and learning algorithms. MUME includes a library of network architectures including feedforward, simple recurrent, and continuously running recurrent neural networks. Each architecture is supported by a variety of learning algorithms. MUME can be used for large scale neural network simulations as it provides support for learning in multi-net environments. It also provide pre- and post-processing facilities. For more information, see

http://www-2.cs.cmu.edu/afs/cs/project/airepository/ai/areas/neural/systems/mume/0.html

Nevada Backpropagation (NevProp)

NevProp, version 3, is a relatively easy-to-use, feedforward backpropagation multilayer perceptron simulator-that is, statistically speaking, a multivariate nonlinear regression program. NevProp3 is distributed for free under the terms of the GNU Public License and can be downloaded from http://brain.cs.unr.edu/publications/NevProp.zip

Matrix Backpropagation

MBP (Matrix Back Propagation) is a very efficient implementation of the back-propagation algorithm for current-generation workstations. The algorithm includes a per-epoch adaptive technique for gradient descent. All the computations are done through matrix multiplications and make use of highly optimized C code. The software is available by anonymous ftp from ftp.esng.dibe.unige.it as /neural/MBP/MBPv11.zip

BIOSIM

BIOSIM is a biologically oriented neural network simulator. Public domain, runs on Unix (less powerful PC-version is available, too), easy to install, bilingual (german and english), has a GUI (Graphical User Interface), designed for research and teaching, provides online help facilities, offers controlling interfaces, batch version is available, a DEMO is provided. Available for ftp from ftp.uni-kl.de in directory /pub/bio/neurobio: Get /pub/bio/neurobio/biosimpc.readme (2 kb) and /pub/bio/neurobio/biosimpc.zip (150 kb) for the PC version.

AINET

AINET is a probabilistic neural network application which runs on Windows 95/NT. It was designed specifically to facilitate the modeling task in all neural network problems. It is lightning fast and can be used in conjunction with many different programming languages. It does not require iterative learning, has no limits in variables (input and output neurons), no limits in sample size. It is not sensitive toward noise in the data. The database can be changed dynamically. It provides a way to estimate the rate of error in your prediction. It has a graphical spreadsheet-like user interface. You can get a full working copy from: http://www.ainet-sp.si/

Trajan 2.1 Shareware

Trajan 2.1 Shareware is a Windows-based Neural Network simulation package. It includes support for the two most popular forms of Neural Network: Multilayer Perceptrons with Back Propagation and Kohonen networks. Trajan 2.1 Shareware concentrates on ease-of-use and feedback. It includes Graphs, Bar Charts and Data Sheets presenting a range of Statistical feedback in a simple, intuitive form. It also features extensive on-line Help. See Trajan Software's Home Page at http://www.trajan-software.demon.co.uk/ for further details, and a free copy of the Shareware version.

Neural Networks at your Fingertips

Neural Networks at your Fingertips" is a package of ready-to-reuse neural network simulation source code which was prepared for educational purposes by Karsten Kutza. The package consists of eight programs, each of which implements a particular network architecture together with an embedded example application from a typical application domain. The applications demonstrate use of the networks in various domains such as pattern recognition, time-series forecasting, associative memory, optimization, etc.The programs are coded in portable, self-contained ANSI C and can be obtained from the web pages at http://www.geocities.com/CapeCanaveral/1624.

NNFit

NNFit (Neural Network data Fitting) is a user-friendly software that allows the development of empirical correlations between input and output data. Multilayered neural models have been implemented using a quasi-newton method as learning algorithm. Early stopping method is available and various tables and figures are provided to evaluate fitting performances of the neural models. The software is available for most of the Unix platforms with X-Windows. Informations, manual and executable codes (english and french versions) are available at http://www.gch.ulaval.ca/~nnfit

Lens

Lens (the light, efficient network simulator) is a fast, flexible, and customizable neural network package written primarily in C. It currently handles standard backpropagation networks, simple recurrent (including Jordan and Elman) and fully recurrent nets, deterministic Boltzmann machines, self-organizing maps, and interactive-activation models. Lens runs under Windows as well as a variety of Unix platforms. It includes a graphical interface and an embedded script language (Tcl). Lens is available free-of-charge to those conducting research at academic or non-profit institutions from http://www.cs.cmu.edu/~dr/Lens

EasyNN

EasyNN is a neural network system for Microsoft Windows. It can generate multi layer neural networks from text files or grids with minimal user intervention. The networks can then be trained, validated and queried. Network diagrams, graphs, input/output data and all the network details can be displayed and printed. Nodes can be added or deleted while the network is learning. The graph, grid, network and detail displays are updated dynamically so you can see how the neural networks work. EasyNN runs on Windows 95, 98, ME, NT 4.0, 2000 or XP. URL: http://www.easynn.com/

FEATURES OF NEURALPOWER

NeuralPower is suitable for:

- People who have to work with or analyze data in any form.
- People working on problems that are complex, laborious, 'fuzzy' or simply un-resolvable using present methods.
- People who simply want to improve on their current techniques and gain competitive advantage.

System Requirements

Running the 32-bit version of NeuralPower requires a minimum 486 processor and Windows 95/98/2000/ME/XP or NT 4.0. A complete installation requires about 10 MB of hard disk space. 64 MB or high RAM is recommended.

Getting Started

The first screen which appears on running Neuarl power is shown in Fig.1. There are three main procedures: "Data Files", "Learning/Training" and "Applications" for being selected. If "Don't show this window in future" is checked, this form will be no longer visible when next running NeuralPower, instead, the "Data Files" editor will be the default start procedure.

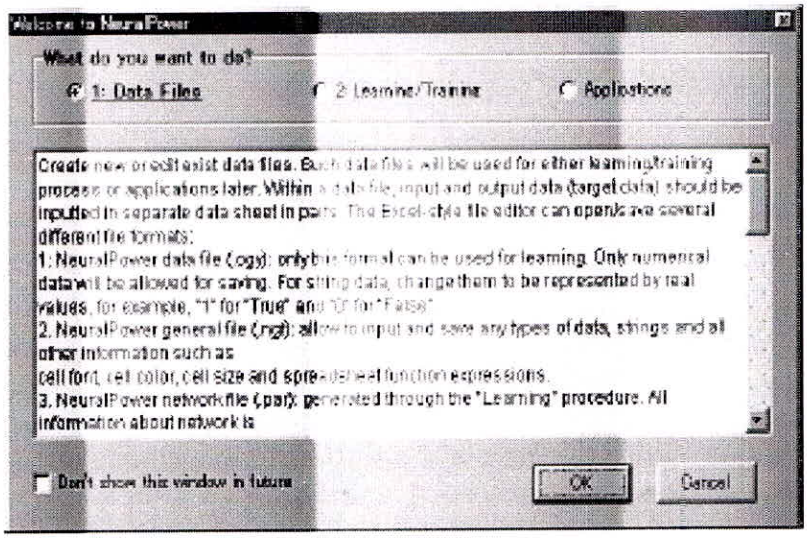

Fig. 1: Task Selection Form

Data Files Editor

The Data file editor provides a spreadsheet-like tool for easy pre- and post-processing of data. In addition, many extra jobs can be performed here also, i.e. charting, regression.

Interface

The Excel-style data file editor, is shown in Fig.2.

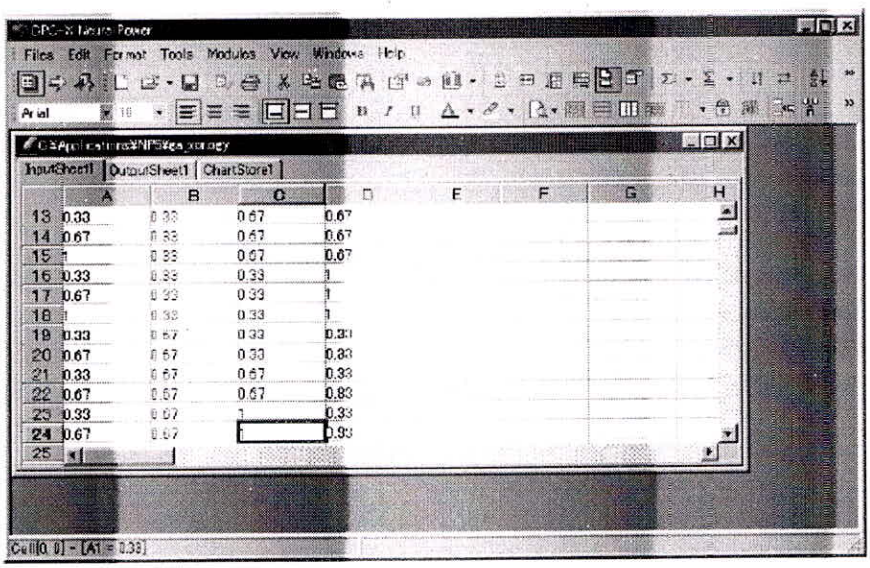

Fig. 2: Data File Editor

Interface is mainly used for creating new or viewing/editing existed data files. Such data files will be used for either the learning/training process or for applications later. "InputSheet" is especially for input data entry of learning data; while "OutputSheet" is for output data corresponded. "ChartStore" is for chart store, view and edit. The charts can be created with the data either from input data or output data, or copy from result charts of Learning or Application procedures.

File Format

- NeuralPower data file (.ogy): any numerical data or strings may be contained. However, if this file is to be used for learning later, be sure no string data is contained within the file except for the column title. For string data, represent them by real values, for example, "1" for "True" and "0" for "False".
- Excel file (.xls): Read/save MS Excel data file directly, without OLE linker. Data values only, all other information will be ignored. If there are more than two worksheets in "xls" file, only first two will be inputted.
- Lotus 1-2-3 file (.wks, .wk1) and Quattro Pro file (.wq1): Direct read without OLE.
- Comma-delimited text file (.csv, .txt): Save and load most general data file format. Reading data with separate of tab space, space, ";" and "," automatically; while saving data with separate of ",".
- Dbase file: (.dbf): Save and load popular desktop database directly
- MS Access file (.mdb): Save and load another popular desktop database files through DAO
- NeuralPower network file (.par): Load NeuralPower network file for viewing connection weights

Learning/Training

The Learning/training process shows that the free parameters of a neural network are adapted through observed data (data file). The objective is to obtain "Network Weights File" which will be used for Applications later.

Learning files

To start "Learning", you need to load one or more data files (.ogy) initially. Multi-files are allowed in the meantime, if they are all for the same problem and have similar data structures. The files with "Checked" will be used for learning; while, "No checked" files are for testing/verification.

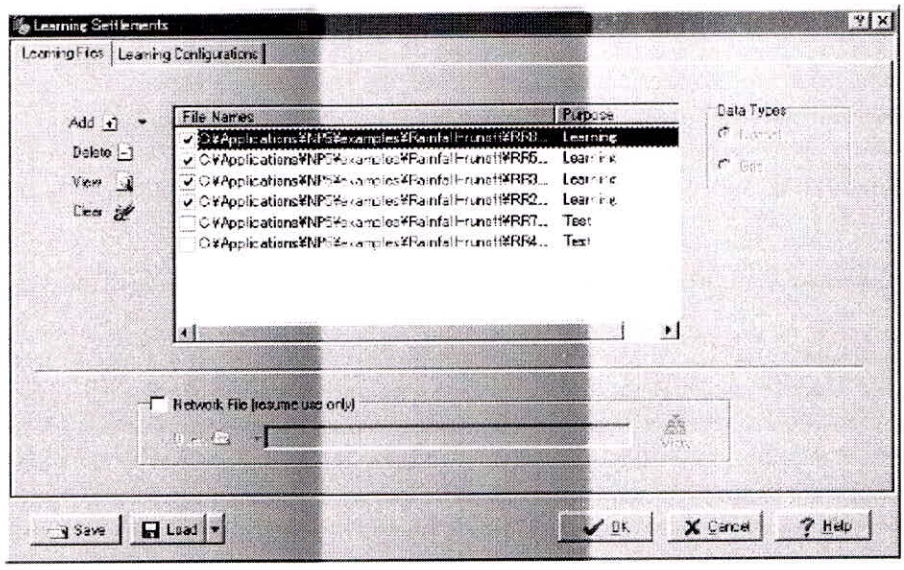

Fig. 3: Load Files for Learning

Resume

In default, "Learning" will begin with new weights either specified by custom or at random. However, "Learning" can also be resumed by loading an existed network file (.par), which has been generated previously with compatible data files.

Data types

Data types are classified into two catalogs: "Normal Type" and "Grid Type":

- Normal type: one pattern consists of the data placed in any row of the output sheet that correspond to the data in some row of input sheet. Within a data file, the patterns number is the same as the row number, and the row number in the output sheet must be identical with that in input sheet. If using a different data file and the same question, however, the row number can be various.

- Grid type: all input data and output data in one file are considered as one pattern. Row numbers are not necessary identical between input and output. However the row and column number should be matched for different data file correspondance within the same problem, e.g. in input and output, respectively.

Layer and structures

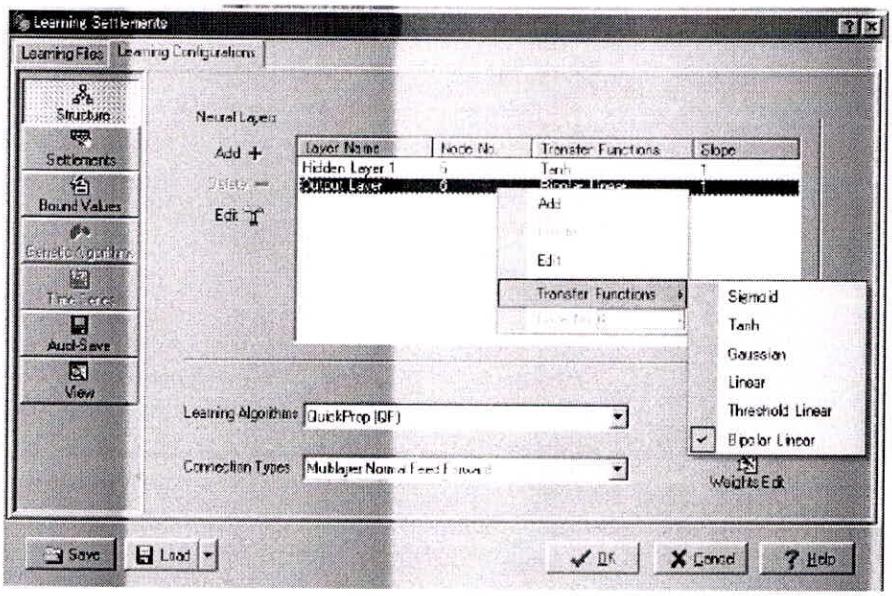

Fig. 4: Settlement of Structures

Usually, a network with one hidden layer will be used more often for general problems. So you should always try a network with one hidden layer initially. Occasionally two or more hidden layers will be needed. There are no limits on layer numbers and node numbers of each layer.

Learning Algorithms:

There are five learning algorithms supported in NeuralPower currently:

- Incremental Back Propagation (IBP): the network weights are updated after presenting each pattern from the learning data set, rather than once per iteration. This is referred to as Standard Back Propagation, and is the most preferred algorithm for large data sets
- Batch Back Propagation (BBP): the network weights update takes place once per iteration, while all learning data pattern are processed through the network.
- Quickprop (QP): Quick propagation is a heuristic modification of the back propagation algorithm. It's proved to be much faster than standard back-propagation for many problems.
- Genetic Algorithms (GA): Genetic algorithm is employed to find optimal connections weights.
- Levenberg-Marquardt Algorithm.

Connection types

Two kinds of connections see difference between Fig.5 and 6.

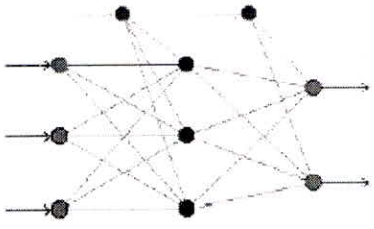

Fig.5 Multilayer Normal Feed Forward

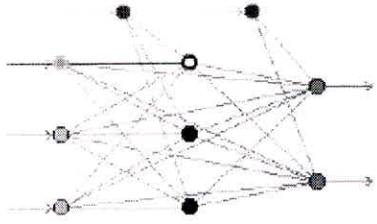

Fig. 6 Multilayer Full Feed Forward

Multilayer Normal Feed Forward: a multilayer feed forward network consists of one or more layers. Each layer receives an input vector, which is either the external input vector or the output vector of the prior layer. The layers are placed in a linear order, so that the input to the first layer is the external input, the input to the second layer, if it exists, is the output of the first layer, and so on, with layers 2,3,..., receiving their inputs from the previous layer. The last layer in the sequence is the output layer; its output vector represents the network output. Every other layer besides the output layer is referred to as a "hidden" layer, because their activation is not externally visible. This is most commonly used and is generally recommended for most applications.

Multilayer Full Feed Forward: The Multilayer Full Feed forward architecture is similar to Multilayer Normal Feed Forward, except for two differences. First, each layer is provided with the external input. Second, each layer receives input from every layer below it in the linear ordering. For example, if the network has 3 layers, the first one receives the input vector, as usual. The second one receives an input vector that is the concatenation of the input vector and the output of the first layer. The third layer receives inputs from the input vector, and the output vectors of both prior layers.

Transfer Functions

The transfer function, denoted by $\xi(\mu_k)$, defines the output of a neuron in terms of the activity level at its input. Six commonly used functions are predefined as below:

- The Sigmoid function, ranged from 0 to 1, is defined as:

$$\xi(\mu_k) = \frac{1}{1+\exp(-a\mu_k)}$$

- The Hyperbolic Tanh function, ranged form -1 to 1, is defined by:

$$\xi(\mu_k) = \frac{1-\exp(-a\mu_k)}{1+\exp(-a\mu_k)}$$

- The Gaussian function, ranged from 0 to 1, defined as:

$$\xi(\mu_k) = e^{(-a \cdot \mu_k)^2}$$

- The Linear function: ranged from $-\infty$ to $+\infty$

$$\xi(\mu_k) = a \cdot \mu_k$$

- The Threshold Linear function: ranged from 0 to +1:

$$\begin{cases} \xi(\mu_k) = 0 & \text{if } \mu_k < 0 \\ \xi(\mu_k) = 1 & \text{if } \mu_k > 1 \\ \xi(\mu_k) = \alpha\mu_k & \text{if } 0 \le \mu_k \le 1 \end{cases}$$

- The Bipolar Linear function: ranged from -1 to +1:

$$\begin{cases} \xi(\mu_k) = -1 & \text{if } \mu_k < -1 \\ \xi(\mu_k) = 1 & \text{if } \mu_k > 1 \\ \xi(\mu_k) = \alpha\mu_k & \text{if } -1 \le \mu_k \le 1 \end{cases}$$

Where, α is a parameter called "Slope" of transfer functions.

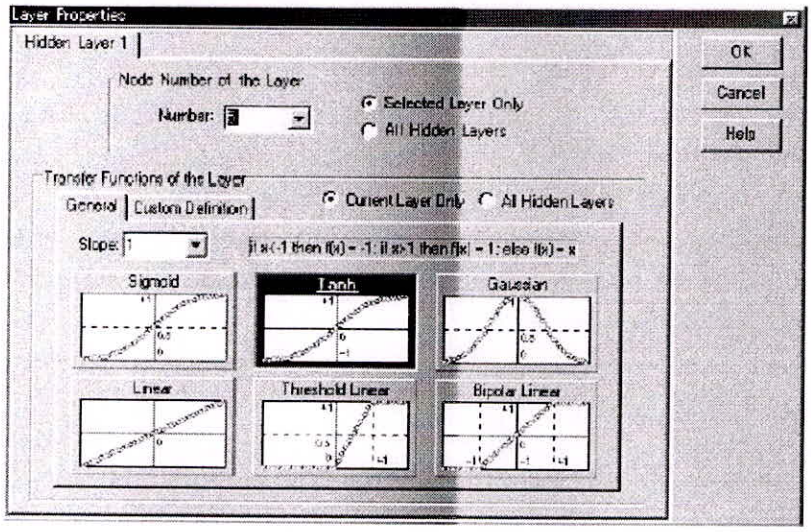

Fig. 7: Six Most often Used Transfer Function

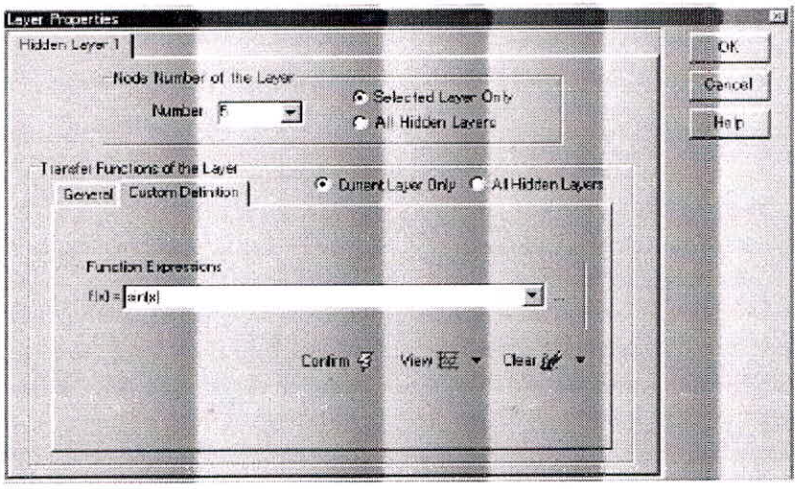

Fig. 8: User Self-defined Transfer Function

The default, for general problems, the "Tanh" function will be used. However, for classification-like problems either a i.e. 1 or b i.e. 0, the Bipolar Linear function will be adopted in output layer.

Connections Editor

As displayed in Fig.9, connection editor provides a easy and graphical mode for users to enable or disable connections among nodes, so some specified structures could be designed, for example, in Fig.10.

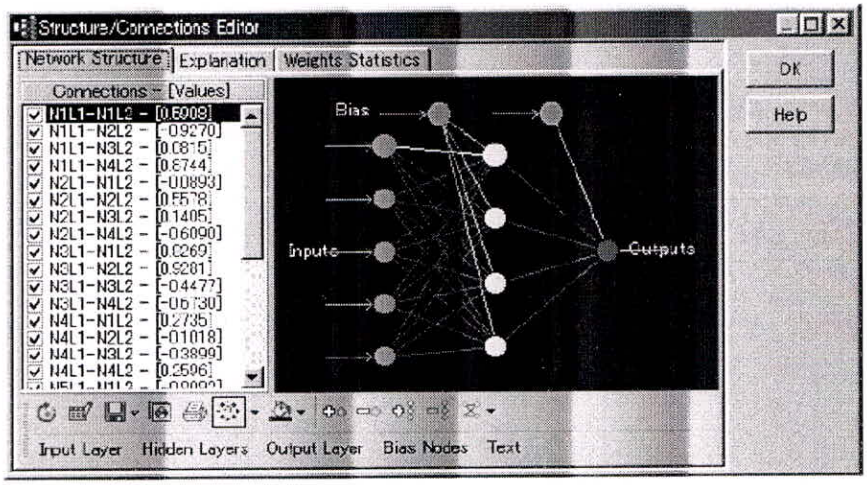

Fig. 9: Connection Editor

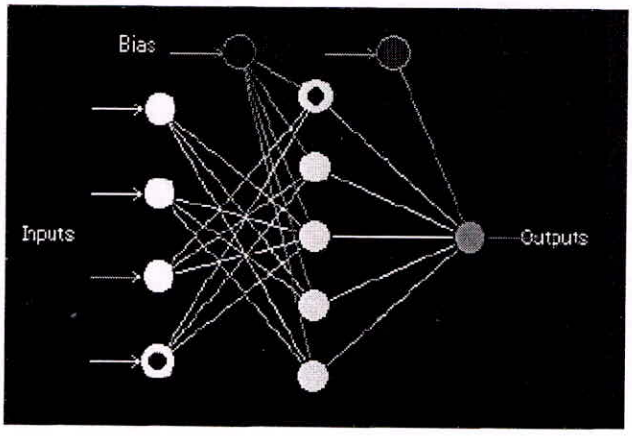

Fig. 10: Neural Network Structure Design

Weights Editor

Default, connection weights will be started from random values. However, user may start learning with specified weight values by utilizing the weights editor.

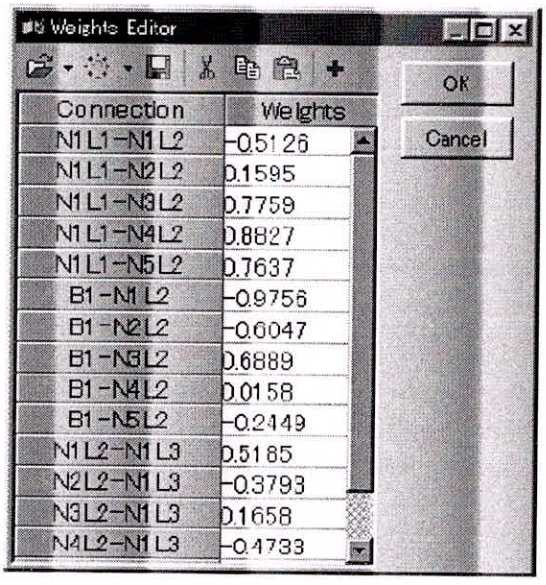

Fig. 11: Weight Editor

Settlements

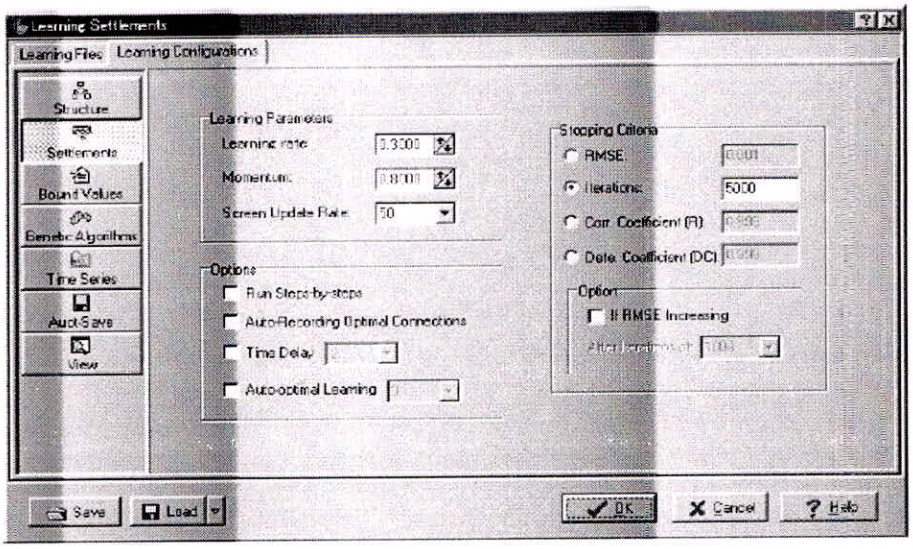

Fig. 12: Learning Settlements

In the "Settlements" section, user can set parameters such as learning rate, momentum, screen update rate and stop criteria. All of those parameters may be modified at any time during learning.

Bound values

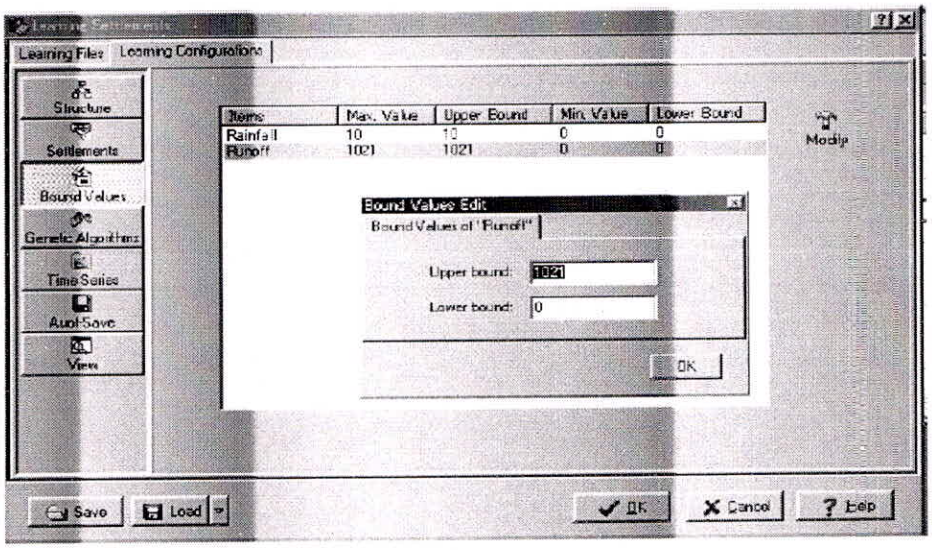

Fig. 13: Bound Values Edit

Bound Values include Upper bound values and Lower bound values. They are the possible maximum/minimum data values of each input/output data series, which are not contained in current data files, but may be encountered in the future. By adjusting Bound values, the learning may be fasted greatly in some cases. In default, Bound Values equal to maximum/minimum values

Auto-Save

You may decide here whether or not to save the network file, or changes consisting of RMSE, R, and DC, as the going of learning progress.

View

You may decide which output variables will be visible or not, and whether or not to actively monitor a window for real time monitoring of the changes of RMSE, R, DC or any weights.

"Fix Length" means the chart displayed in the monitor window will contain data numbers no more than this value. So as learning proceeds, if the data number is bigger than "Fix Length", the first data of the chart series will be deleted. However, "All Data" means all the data will be added to chart.

Project file

Project file (.pjs) contain all information settled above. This is used for a quick load next time.

Learning Process

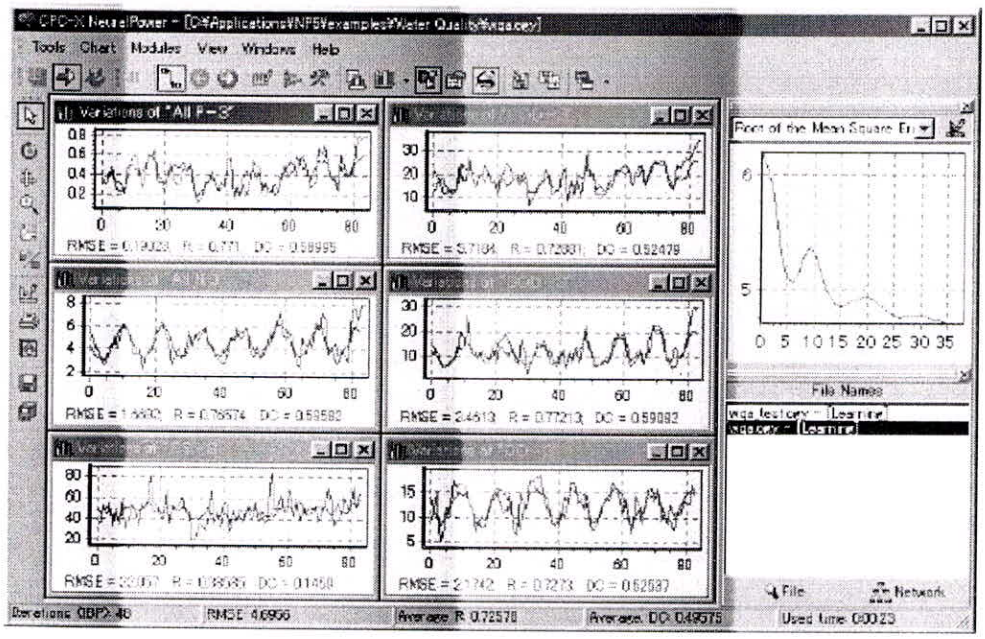

Fig. 14: Learning Process

During the learning process, user will see the observed data (red line) vs. calculated data (green line) for all output variables as the learning progresses. Learning will be auto stopped if "Stop Criteria" is reached. Learning can also be suspended at any time one wishes

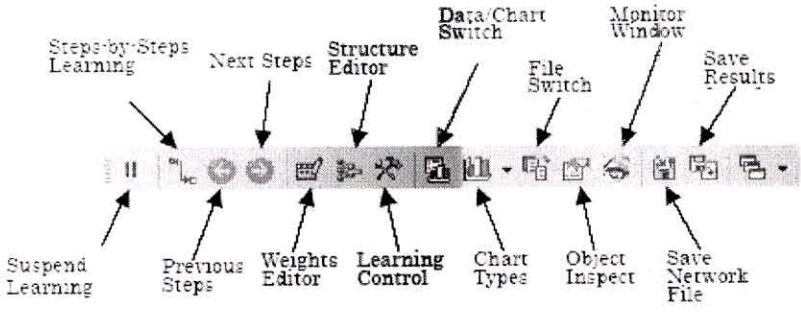

Fig. 15: Learning Control Toolbar

During learning, most of the learning parameters can be adjusted, for example, learning rate, learning algorithms, node number and transfer function of the hidden layer. Learning results, i.e. network file (.par) and result files, can be saved automatically in certain iterations, or saved manually at any time, or saved when learning stopped. Network file is important for used in "Application Model" later, for either forecasting/verification or other analysis. So if you achieve successful learning, save network file please!

Applications

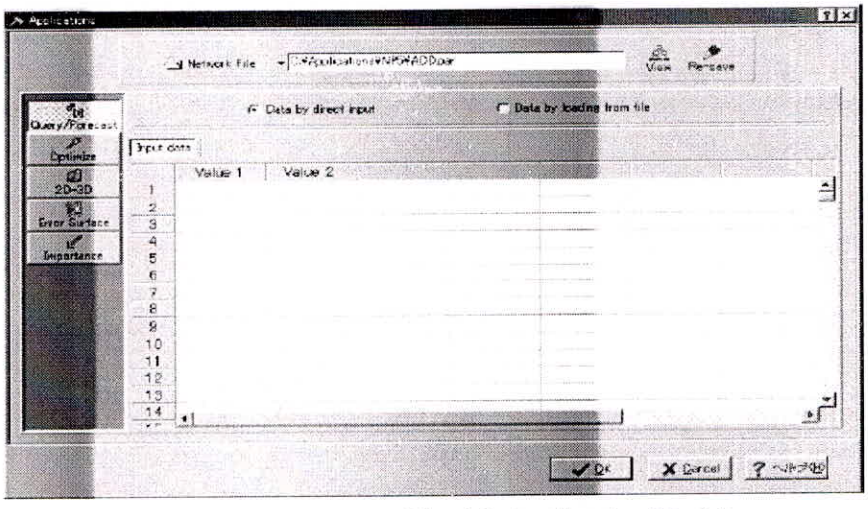

Fig. 16: Application Model

In the application module, the network file (.par) obtained through the learning progress will be used for further analysis:

- Query/Forecasting
- Optimize analysis

- 2-D and 3-D analysis
- Error surface analysis
- Importance analysis

Before carrying out above analysis, a network file (.par) must be loaded firstly.

Query/Forecasting

Forecasting or verification can easy be done here. The data values can be either inputted directly or loaded from a data file. In the situation where the time series items are included in the output data series (recorded in Network File), and if you have input data values directly, the initial data must be inputted corresponding to the output items that contain the time series.

IMPLEMENTATION OF NEURAL POWER PACKAGE

Implementation of the Neural Power package has been done through a case study of rainfall-runoff modeling problems.

STUDY AREA AND DATA AVAILABILITY

For the present study, Ajay river basin upto Sarath gauging site forms the study area. The Ajay river system originates in the low hills near Deoghar in the Santhal Pargana district of Jharkhand (South Bihar) and flows in a South-Easterly direction passing through Monghyr district of Jharkhand (South Bihar) and Birbhum and Burdwan district of West Bengal. Ajay river ultimately falls into the river Bhagirathi at Katwa about 216 Km above Calcutta. The Ajay river system lies between the Mayurakshi on the North and the Damodar and the Banka/Khari river system on the South. The Sarath gauging site, establised near village Sarath and about 160 m upstream of Sarath – Madhupur road bridge, is maintained by Water resources Department, Govt of Jharkhand (South Bihar). The geographical location of the site is 24^0 13' 45" N latitute and 86^0 50' 43" E longitude which is approachable by road in all seasons. The catchment area upto the site is 1,191.40 sq km. The length of Ajay river up to Sarath gauging site is 82.18 sq km. Table1 gives the details of periods of various storms whose rainfall-runoff data have been used in the study.

Table 1: Periods of various rainfall-runoff events

S. No.	Period of the Events
1	13.08.1977 at 09 hrs. to 13.08.1977 at 20 hrs.
2	05.08.1978 at 21 hrs. to 06.08.1978 at 02 hrs.
3	16.08.1979 at 05 hrs. to 16.08.1979 at 16 hrs.
4	26.08.1980 at 15 hrs. to 26.08.1980 at 07 hrs.
5	22.08.1982 at 24 hrs. to 23.08.1982 at 06 hrs.
6	12.09.1987 at 13 hrs. to 12.09.1987 at 24 hrs.

INPUT VARIABLES

The first step in developing an ANN model is to identify the input and output variables. The output from the model is the runoff at time step t, R_t. The input variables have been selected based on the concepts of time of concentration and recession of a storm hydrograph. The time of

concentration of the Ajay river basin was observed to lie between 21 to 22 hours. With a time of concentration of 22 hours and a time interval of 1 hour, the number of time steps for the past for which rainfall must be considered as input in the ANN models should be 22 (=22/1). Further it was found that the runoff in the immediate past was a more significant variable compared to the runoff in the distant past, and hence it has been applied in all the ANN models. Therefore, there were 26 input variables (P_t, P_{t-1}, P_{t-2},.....P_{t-22}, R_{t-1}, R_{t-2}) and one output variable (R_t). Using these lagged inputs, data files were prepared in the Neural Power software.

ANN MODEL DEVELOPMENT

In estimation of parameters of a hydrologic model, the available data are divided in two parts. The first part is used to calibrate the model and the second, to validate it. This practice is known as 'split-sample' test. The length of calibration data depends upon the number of parameters to be estimated. The general practice is to use half to two-third of the data for calibration and the remaining for validation.

Six isolated storm events were chosen for the study. The 1-hourly rainfall runoff data were available for flood seasons. ANN models have been developed considering hourly data for four flood events for training and two flood events for testing. On a rotation basis, data from four storms have been used for training, while data from two storms have been used for testing network performance. Various combinations of the flood events considered for training and testing are given in Table 2.

Table 2 – Description of various ANN models for Training and Testing

ANN Model	Events used in Training (calibration)	Events used in Testing (Validation)
ANN - 1	Events 1, 2 , 3, 4	Events 5,6
ANN - 2	Events 2, 3, 4, 5,	Events 6, 1
ANN - 3	Events 3, 4, 5, 6	Events 1,2
ANN - 4	Events 4, 5, 6, 1	Events 2, 3
ANN - 5	Events 5, 6, 1, 2	Events 3, 4
ANN - 6	Events 6, 1, 2, 3	Events 4, 5

The ANN package Neural Power (2000) downloaded from the internet has been used for the ANN model development. The structure for all simulation models are three layer BPANN which utilizes a non-linear sigmoid activation function uniformly between the layers. Nodes in the input layer are equal to number of input variables, nodes in hidden layer are varied from 18 (default value by the NP package for 26 input nodes) to approximately double of input nodes (Zhu et al., 1994) and the nodes in the output layer is one as the models provide single output. It was found that 18 number of hidden nodes give the best results. So for all the ANN models, 18 nodes in the hidden layer have been considered.

Number of input nodes in input layer	=	26
Number of hidden layers	=	1
Number of hidden nodes	=	18
Number of nodes in output layer	=	1

According to Hsu et al. (1995), three-layer feed forward ANNs can be used to model real-world functional relationships that may be of unknown or poorly defined form and complexity. Therefore, only three-layer networks were tried in this study.

The modelling of ANN initiated with the normalization (re-scaling) of all inputs and output with the maximum value of respective variable reducing the data in the range 0 to 1 to avoid any saturation effect that may be caused by the use of sigmoid function (accomplished through the Neural Power package). All interconnecting links between nodes of successive layers were assigned random values called weights. A constant value of 0.15 and 0.8 respectively has been considered for learning rate α and momentum term β selected after hit and trials. The quick propagation (QP) learning algorithm has been adopted for the training of all the ANN models. The criteria selected to avoid over training was through generalization of ANN for which the developed model was simultaneously checked for its improvement on verification data on each iteration. The training was continued until there was an improvement in the performance of the model in both calibration and verification periods. The performance of various models were tested through the criterion discussed below.

The performance evaluation statistics used for ANN training and testing in the present application are root mean square error (RMSE), coefficient of correlation (R) and determination coefficient (DC). These parameters have been determined using the following equations (Neural Power, 2003).

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(Q_i - q_i)^2}{n}} \quad (1)$$

$$R = \frac{\sum_{i=1}^{n}(Q_i - \bar{Q})(q_i - \bar{q})}{\sqrt{\sum_{i=1}^{n}(Q_i - \bar{Q})^2 (q_i - \bar{q})^2}} \quad (2)$$

$$DC = \frac{\sum_{i=1}^{n}(Q_i - \bar{Q})^2 - \sum_{i=1}^{n}(Q_i - \bar{q})^2}{\sqrt{\sum_{i=1}^{n}(Q_i - \bar{Q})^2}} \quad (3)$$

Where $\bar{Q} = \frac{1}{n}\sum_{i=1}^{n} Q_i$, $\bar{q} = \frac{1}{n}\sum_{i=1}^{n} q_i$, Q= observed, q=calculated

DISCUSSION OF RESULTS

The values of the performance criteria from various models for both training (calibration) and testing (validation) data sets are presented in Table 3. It can be seen from Table 5 that, in

general, RMSE is found to be smaller (lowest for ANN-4) and the ANN estimates are closer to the observed values. Coefficient of correlation (R) is another indicator of goodness of fit and it is seen from Table 5 that, R is also quite high in all the cases of training and tested data sets (highest for ANN-2). The determination coefficient (DC) is also closer to unity, for all the cases of training test data (highest for ANN-2). Thus, the estimations by ANN are found to yield all the three indices with acceptable accuracy.

Table 3: Comparative Performance of Various ANN models

ANN	Calibration (Training)			Verification(Testing)	
Model	RMSE	R	DC	R	DC
ANN – 1	16.382	0.998	0.996	0.832	0.61
ANN – 2	8.375	0.999	0.999	0.989	0.977
ANN – 3	11.685	0.999	0.997	0.941	0.675
ANN - 4	7.775	0.999	0.998	0.975	0.933
ANN - 5	12.78	0.998	0.997	0.968	0.934
ANN - 6	11.913	0.999	0.998	0.952	0.766

From Table-3, it is evident that the ANN-2 model outperforms all the other models. This model consists of four flood events namely, events 2, 3, 4, and 5 for training and events 6 and 1 for testing. It is to be noted that the flood event 2 consists of the lowest as well as the highest numerical values of runoff. Moreover, the patterns covered by the four flood events of ANN-2 for training are also highest (270) compared to all the other models. So the performance of ANN-2 model is the best. This conforms to the general fact that an ANN is better trained as more input data are used.

Figures 17 and 18 present the results of ANN-2 graphically during training and testing respectively and show a close proximity between observed and simulated runoff values during calibration as well as validation. Figure 19 presents a plot between observed and simulated runoff for ANN-2 model during testing and shows a high correlation between the two. Figure 20 shows the error in each of the 217 simulated patterns during testing of ANN-2 model. Comparatively more errors in the first 90 patterns are visible.

The observed and simulated flood hydrographs for the events 1 to 6 are shown in Figures 21 to 26 respectively. These simulated flood hydrographs are based on the best performing ANN model, i.e., ANN-2 model. It can be seen that there is a perfect match between the observed and simulated flood hydrographs for the flood events 2, 3, 4 and 5. This is because these four events together have been used for training the ANN-2 model. However, very good match between the observed and simulated flood hydrographs is also there for the events 1 and 6 which were not used for the training. The coefficient of correlation is as high as 0.925 and 0.927 for flood events 1 and 6 respectively.

CONCLUSIONS

The ANN approach has been successfully used in many hydrological studies especially the rainfall-runoff modelling using continuous data, and this was motivating factor for its application to the present study to examine its applicability to model the event-based rainfall-runoff process. This study evaluated the applicability of artificial neural networks in event-based rainfall runoff process modeling through an ANN software namely, Neural Power. A case study has been done for Ajay river basin to develop event-based rainfall-runoff model for the basin to simulate the hourly runoff at Sarath. The feed forward error back propagation algorithm has been used for training of the ANN model. Out of the data of 6 flood events, 4 were used for training and 2 for testing the model. Various combinations of the flood events were considered during training. The performance of each model structure was evaluated using common performance criteria, i.e., root mean square error (RMSE), coefficient of correlation (R), and coefficient of determination (DC).

The results obtained in the present study have been able to demonstrate that the ANN models are able to provide a good representation of an event-based rainfall-runoff process. And it has been found that Neural power is an easy to use software for ANN model development for rainfall-runoff modeling.

REFERENCES

- Hammerstrom, D., Neural Networks at Work. *IEEE Spectrum*, 1993, 30, 46-53.
- Haykin, S., Neural Networks - a Comprehensive Foundation. *Macmillan*, 1994, New York.
- Hsu, K., Gupta, H. V., and Sorooshian, S. 1995. " Artificial neural network modeling of the rainfall-runoff process." Water Resou7: Res., 31(10),2517-2530.Hydrological Sciences Journal, 41(3), 399-417.
- Neural Power, (2003). Neural networks Professional version 2.5 CPC-X Software,Copyright: 1997-2003. A Demo version downloaded from the Internet.
- Zhu, M., Fujita, M., and Hashimoto, N. 1994. Application of Neural Networks to Runoff Prediction. Stochastic and Statistical Method in Hydrology and Environmental Engineering, vol. 3, K.W. Hipel et al., eds., Kluwer, Dordrecht, The Netherlands, 205-216.

राष्ट्रीय जल विज्ञान संस्थान
National Institute of Hydrology

आपो हिष्टा मयोभुवः