

COMPARISON OF VARIOUS OPTIMIZATION METHODS FOR TRAINING OF ANN RIVER FLOW MODEL

Ganti RAVIKUMAR¹ and Ashu JAIN²

- (1) Research Scholar, Department of Civil Engineering, Indian Institute of Technology Kanpur, Kanpur, email: gravi@iitk.ac.in
(2) Professor, Department of Civil Engineering, Indian Institute of Technology Kanpur, Kanpur, email: ashujain@iitk.ac.in

Abstract

Artificial neural network (ANN) models provide huge potential for simulating the behaviour of nonlinear physical systems e.g. hydrological systems. The training of an ANN is primarily a non-linear optimization problem in which the objective is to determine the values of decision variables (i.e. connection weights) such that the global error at the output layer (normally mean square error, MSE) is minimized. Most of the ANN applications in hydrology reported in literature have employed the popular back-propagation algorithm (BPA), which is a gradient search method, for training purposes. In this paper, the results from an investigation on the use of different optimization methods for the training of feed-forward neural network model structure are presented. Two classical (gradient based) and two evolutionary (non-gradient based) optimization methods were investigated for the training of the ANN river flow model. The gradient based classical optimization methods investigated include Back Propagation Algorithm (BPA) and Levenberg-Marquardt Algorithm (LMA), while genetic algorithm (GA) and particle swarm optimization (PSO) were the two non-gradient based evolutionary optimization methods adopted for training in this study. The daily river flow data derived from Godavari Basin @ Polavaram, Andhra Pradesh, India have been employed to develop all the models included here. A wide range of error statistics have been used to evaluate the performance of all the models developed in this study. It has been found that LMA performed slightly better than the BPA and PSO methods, and the GA performed the best amongst all other methods investigated. In terms of computational effort LMA was found to be most efficient as compared to other algorithms. It is concluded that LMA or GA to be adopted in training of ANN models for hydrological applications for capturing the complex, dynamic, and non-linear nature inherent in the hydrological systems.

1. INTRODUCTION

Over the past 15 years, there has been a growing interest in artificial neural networks (ANNs) for simulating, forecasting and predicting hydrological variables (ASCE Task Committee, 2000; Maier and Dandy, 2000; Govindaraju and Rao, 2000). The majority of hydrological processes are highly nonlinear in nature and, in many cases, modelling

these variables with more conventional physically based or conceptual models may be limited by a poor understanding of the complex interactions that are involved in the process. In such cases, ANNs are often viewed as an appealing alternative, as they have the ability to extract a nonlinear relationship from the data without requiring an in depth knowledge of the physics occurring within the hydrological system (Zhang and Eddy Patuwo, 1998)

Despite the increasing use of ANNs as hydrological models, they are still viewed with some scepticism. This is primarily due to their 'black-box' nature, where solutions to a problem are automatically generated and no insight is gained into how the ANN arrived at its solution. Traditionally, the primary purpose of an ANN has been to provide a prediction of system response rather than to gain an understanding of the causal interactions that generate the hydrological occurrence, and therefore, neural network practitioners have only been concerned with prediction accuracy (Maier and Dandy, 2000; Yitian and Gu, 2003). However, if ANNs are to become more widely accepted and reach their full potential as hydrological models, they should not only provide a good fit to the calibration and validation data, but the predictions should also be plausible in terms of the relationship modelled and robust under a wide range of conditions.

Most of the ANN models presented in the literature use the popular back-propagation algorithm (BPA) for training (Rumelhart et al., 1986). The BPA uses gradient search technique to determine an optimal set of weight parameters in order to match the known target values at the output layer of an ANN. The use of gradient search techniques often results in inconsistent and unpredictable performance of neural networks (Curry and Morgan, 1997). Thus, there is a strong need to explore alternative training methods to train an ANN to develop better models for streamflow prediction for operational purposes (Srinivasulu and Jain, 2006).

The objectives of the current study are to (a) investigate the technique of ANNs for modelling the complex hydrological process, and (b) to evaluate various methods available for training the ANN models. The training methods investigated include the supervised training methods of BPA, LMA, GA and PSO. The daily streamflow data derived from the Godavari River basin have been employed to develop all the ANN models. A wide variety of standard statistical performance evaluation measures have been employed to validate all the models investigated. The paper begins with a brief introduction of the soft computing techniques of ANN followed by the details of the model development before discussion the results and making concluding remarks.

2. ARTIFICIAL NEURAL NETWORKS (ANN)

An ANN is a highly interconnected network of many simple processing units called neurons, which are analogous to the biological neurons in the human brain. Neurons having similar characteristics in an ANN are arranged in groups called layers. The neurons in one layer are connected to those in the adjacent layers, but not to those in the same layer. The strength of connection between the two neurons in adjacent layers is represented by what is known as 'connection strength' or 'weight'. An ANN normally consists of three layers, an input layer, a hidden layer, and an output layer. In feed-forward network, the weighted connections feed activations only in the forward direction from an input layer to the output layer. The structure of a feed-forward ANN

is shown in Figure. 1. An important step in developing an ANN model is the determination of its weights matrix through training. There are primarily two types of training mechanisms, supervised and unsupervised. A supervised training algorithm requires an external teacher to guide the training process. The primary goal in supervised training is to minimize the error at the output layer by searching for a set of connection strengths that cause the ANN to produce outputs that are equal to or closer to the targets.

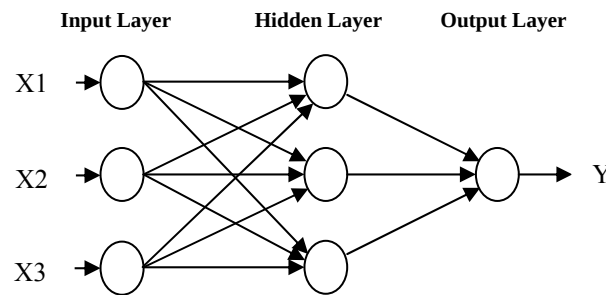


Figure.1.Structure of a Feed Forward ANN

3. MODEL DEVELOPMENT

Four different types of ANN models have been developed that differ in the manner of the training method employed. The first type of ANN model employs the popular BPA which is first order gradient based method, the second type of ANN model employs the LMA which is second order gradient based method, the third type of ANN model employs GA which is population based non-gradient method and the fourth type of ANN model employs PSO which is also population based non-gradient method. The daily streamflow data derived from the Godavari river basin have been employed to develop all the ANN models. A brief overview of the study area and data employed is first provided before giving the details of the ANN model development.

3.1. Study area and data

The streamflow data derived from Godavari river, India were used to test the proposed methodologies in this study.

Polavaram is located within the Godavari River Basin in Andhra Pradesh, India (Figure. 2). Streamflow data are recorded for this catchment at Polavaram station. The catchment has an area of 307800 km² approximately.

Nine years of daily streamflow data from 01-06-1986 to 31-05-1995, were available for training, validating and testing of the ANN model's developed. The entire data set was divided into training, validation and testing set based on statistical similarity. Descriptive statistics of the data sets is been presented in Table 1. Refer to Figure. 3 for the timeseries plot and the data division considered. The data were standardized using linear normalization procedure, normalised between 0.1 to 0.9.

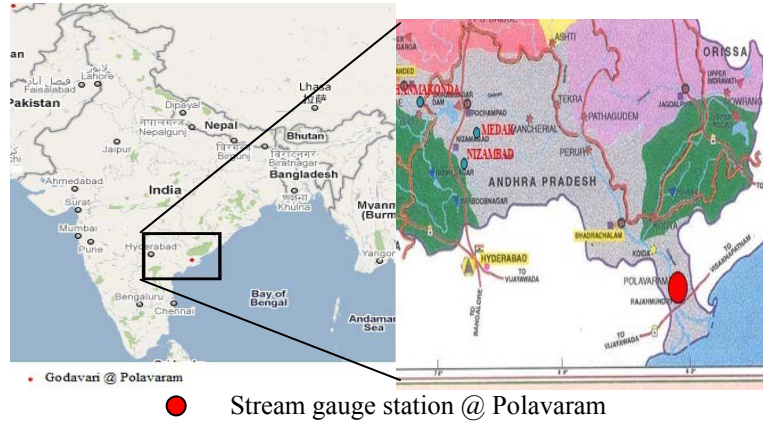


Figure 2. Godavari River Basin, Andhra Pradesh, India

Table 1 Descriptive Statistics for streamflow data

Data Set	Qmin, cumec	Qmax, cumec	Qmean, cumec	Qstdev, cumec	Qskewness	Qkurtosis	Number of Data points
Entire	60.90	87250	3262.10	7448.70	4.53	30.64	3277
Training	68	87250	2954.10	7409.30	5.48	45.19	1451
Validation	323.30	60700	5668.20	10412	2.92	1.78	375
Testing	60.90	47900	2948.20	6387.40	3.99	21.73	1451

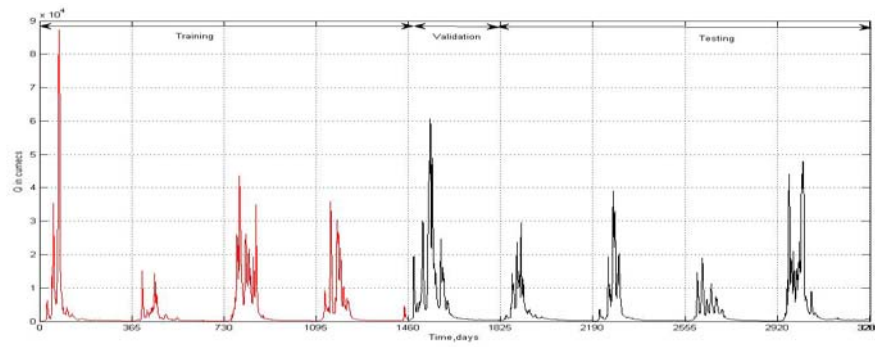


Figure 3 Time Series Plot of daily streamflow , Godavari River @ Polavaram (Time window: 01-06-1986 to 31-05-1995, 9 Years)

3.2 Input Variable Selections (IVS)

The first step in the development of any mathematical model is to determine the significant inputs for the model. In this study, the significant inputs were determined by calculating the auto-correlation coefficients for the daily lagged streamflow data at Polavaram, lag up to 30 days was considered. The inputs having $ACF > 0.5$ have been considered for the ANN model development. Flow values lagged up to 10 days, Q_{t-1} ,

Qt-2, Qt-3, Qt-4, Qt-5, Qt-6, Qt-7, Qt-8, Qt-9 and Qt-10 are considered as inputs for predicting Qt.

3.3 Model calibration (training)

ANNs work on the assumption that there is a function underlying a system that relates a set of independent predictor variables to one or more depended variables of interest. The aim of ANN ‘training’ is to infer an acceptable approximation of this relationship from a set of calibration (training) data, so that the model can be used to produce accurate predictions when presented with new data. During training, the connection weights are iteratively adjusted such that a predetermined error; or objective function is minimized and the best fit between the model predictions and the observed data is obtained.

Local or global optimization algorithms may be used to train an ANN. Local methods search for an optimum in a downhill direction from their initial position on the error surface: whereas, global methods employ directed random search techniques to allow the simultaneous search for an optimum in several directions. In this paper, the training algorithms described in the following subsections are compared in their ability to find the best estimation of weights.

3.3.1 Backpropagation Algorithm (BPA)

Backpropagation is a gradient based local search method that uses the first derivative of the objective function to direct the weight changes downhill along the error surface. This requires that the objective function is continuous and differentiable. The algorithm works by computing the error between the network output and the corresponding target value and propagating this backward through the network to update the weights. A detailed description of this algorithm can be found in (Rumelhart et al., 1986) .

3.3.2 Levenberg Marquardt Algorithm (LMA)

Training neural network is essentially a non-linear least squares problem, and thus can be solved by a class of non-linear least squares algorithms. Among them, the Levenberg-Marquardt is a trust region based method with hyper-spherical trust region. This method works extremely well in practice, and is considered the most efficient algorithm for training artificial neural networks. The Levenberg Marquardt algorithm is a modification of the Gauss Newton algorithm. Cauchy’s method (steepest descent or BPA) works well when the initial point is far away from the minimum point and Newton’s method works well when the initial point is near the minimum point. In any given problem, it is usually not known whether the chosen initial point is away from the minimum or close to the minimum, but wherever be the minimum point, LMA is devised to take advantage of both these methods. In LMA, Cauchy’s method is initially followed. Thereafter, Newton’s method is adopted. The transition from Cauchy’s method to Newton’s method is adaptive and depends on the history of the obtained intermediate solutions. For details of the algorithm refer to (Hagan and Menhaj, 1994).

One of the key aspect in LMA is the adjustment of its training parameter, μ . Thus, the Hessian matrix has little effect on the determination of the search direction. Initially, the search is similar to that in Cauchy’s method. If the solution in the current iteration is

less than the previous iteration then μ value is reduced by a constant multiplying factor else, it is increased by constant multiplying factor. After a number of iterations (when hopefully the current solution has converged close to the minimum), the value of μ becomes small and the effect is more like that in Newton's method.

The μ parameter is automatically adjusted at each iteration in order to secure convergence. μ is decreased by a predefined multiplication factor ($\mu_{dec} * \mu$) after each successful step (reduction in performance function) and is increased by a predefined multiplication factor ($\mu_{inc} * \mu$) only when a tentative step would increase the performance function. In this way, the performance function will always be reduced at each iteration of the algorithm. The Levenberg-Marquardt optimization technique is more powerful than the conventional gradient descent techniques (Kisi and Uncuoglu, 2005). The LMA is now considered as the most efficient algorithm (Brighton et al., 1992; Balakrishnan and Honavar, 1992). It combines the speed of the Newton algorithm with the stability of the steepest descent method.

3.3.3 Genetic Algorithm (GA)

The genetic algorithm (GA) is one of the population based evolutionary or nontraditional optimization algorithms. GA is a commonly explored alternative to using backpropagation for ANN training (Sexton and Gupta, 2002). Unlike backpropagation, GAs are based on the theory of natural selection and do not require any information about the gradient of the objective function (Yao, 1999). GAs search for the optimal set of weights in a number of different directions simultaneously. This gives them the ability to escape local minima and makes them less sensitive to initial conditions than gradient based techniques. Additionally, because no gradient information is required, the objective function does not need to be differentiable or continuous, making GAs more applicable to large, complex, non-differentiable search spaces that are typical of the real world.

GAs are driven by simple operators that attempt to mimic the way populations of species genetically evolve over many generations. To initiate this process, a search space is set by placing upper and lower limits on the possible values of the weights, and an initial population of possible solutions is generated randomly within this space. Each individual in the population is called a chromosome, which is made up of a number of genes and is encoded to represent the variables being optimized. In the optimization of ANNs, each chromosome represents a weight vector and each gene in the chromosome is one connection weight. In this paper, real value coding was used to represent each gene. GA consists of three operators: selection, crossover and mutation. Selection is the process of survival of the fittest where the chromosomes compete to fill the population of parent chromosomes. Once the parent chromosomes are selected, crossover is applied between pairs of parents to produce the next generation of chromosomes. There are a number of different forms of crossover operator. In this paper, uniform crossover operator was applied (Goldberg, 1989; Sastry et al., 2005). Mutation is the final step in the generation of child chromosomes. This ensures that the evolution does not get trapped in unpromising regions of the search space. For details of the GA implementation methodologies, interested readers are referred to (Goldberg, 1989), (V tkovský et al., 2000) and (Deb, 2004).

3.3.4 Particle Swarm Optimisation (PSO)

PSO is a population based evolutionary or non-traditional heuristic search optimization algorithm. The PSO was first proposed by (Kennedy and Eberhart, 1995) and has deserved some attention during the last years in the global optimization field. PSO was discovered through simplified social model simulation. PSO is based on the population of agents or swarm and tries to simulate its social behaviour in optimal exploration of problem space. During time (iterations in the optimization context) each swarm possesses a velocity vector that is a stochastic combination of its previous velocity and the distances of its current position to its own best ever position and to the best ever swarm position. The weights of the last two directions are controlled by two parameters called cognitive and social parameters (Bergh, 2001; Shi and Eberhart, 1998). As an algorithm, the main strength of the PSO is its fast convergence, which compares favourably with many global optimization algorithms like genetic algorithm (GA), simulated annealing (SA) and other global optimization algorithms. For applying PSO successfully, one of the key issues is finding how to map the problem solution into the PSO particle, which directly affects its feasibility and performance.

A simple explanation of the PSO's operation is as follows. Each particle represents a possible solution to the optimization task at hand. For the remainder of this paper, reference will be made to unconstrained minimization problems. During each iteration each particle accelerates in the direction of its own personal best solution found so far, as well as in the direction of the global best position discovered so far by any of the particles in the swarm. This means that if a particle discovers a promising new solution, all the other particles will move closer to it, exploring the region more thoroughly in the process. For more details of the algorithm refer to (Van den Bergh and Engelbrecht, 2004).

Inertia weight (w), cognitive parameter ($C1$) and social parameter ($C2$) are the three main parameters of the PSO. Whose values are fixed by trial and error approach. Empirical results have shown that a constant inertia of $w=0.7298$ and acceleration coefficients with $C1=C2=1.49618$ provide good convergent behaviour (Eberhart and Shi, 2000).

3.4 Model Performance Evaluation

One of the important steps of any mathematical model development is the validation of the model. The model validation can be done by calculating standard statistical errors both during training and testing data sets. Many graphical methods such as scatter and time series plots can also be used. To evaluate the performance of each model developed in this study, six different types of statistical parameters have been employed. In addition, Scatter and time-series plots are used to assess model performance graphically. The various error statistics used in this study are Nash-Sutcliffe Coefficient of Efficiency (E), Correlation Coefficient (R), Normalized Root Mean Square Error (NRMSE), Average Absolute Relative Error (AARE) and Threshold Statistics (TS10, TS25, TS50, TS75, and TS100). Refer to (Jain and Srinivasulu, 2006; Srinivasulu and Jain, 2006) for better understanding of these performance measures.

4. RESULTS AND DISCUSSIONS

Single layer ANN models were developed using the four training algorithms discussed in the earlier sections. Trial and error approach was adopted in fixing the ANN architecture. The hidden neurons in single hidden layer were varied between 1 to 20 with the increment of 1 for every trial. The architecture giving minimum mean square error (MSE) during training is considered for testing. The parameters adopted for training the ANN is given in Table 3 for each of the individual training algorithm.

A comparative study has been made to evaluate the strength of each of the individual training algorithm. Model performance measure during training and testing is presented in Table 4 and Table 5, respectively. Analyzing the results during training, it can be observed that the ANN model trained using the popular BPA performed the worst while the performances of the ANN models trained using GA was moderate and PSO and LMA were the best. The ANN model trained using LMA obtained the best AARE, NRMSE, E and R statistics of 11.96%, 0.4628, 0.9659 and 0.9828, respectively. Analyzing the results during testing, it can be observed that the ANN model trained using LMA and GA consistently outperformed ANN models developed using BPA and PSO. ANN model trained using GA obtained the best AARE, NRMSE, E and R statistics of 9.05%, 0.5141, 0.9361 and 0.9680, respectively, during testing. Thus, it can be said that when the overall performance is considered, the ANN model trained using LMA and GA performed the best, the ANN model trained using BPA performed the worst, and the performance of the ANN model developed using PSO was moderate. The scatter and timeseries plots for all the training algorithms during testing time window (4 years) are shown in Figure. 4. It can be observed that LMA and GA have better scatter and timeseries plots which justifies the performance statistics obtained.

Table 3. User defined parameters employed for training

Training Algorithm	Parameter	Value adopted
BPA	Learning rate	0.01
	Momentum correction factor	0.08
	Maximum iterations	50000
LMA	Initial learning rate	10000
	Learning rate decremental factor	0.1
	Learning rate incremental factor	10
	Final learning rate	0.001
	Maximum iterations	5000
GA	Population size	1600
	Crossover probability	0.8
	Mutation probability	0.2
	Elitism, elite count	2
	Maximum generations	10000
PSO	Inertia weight	0.7298
	Cognitive parameter	1.49618
	Social parameter	1.49618
	Maximum iterations	50000

Table 4. Results of different training algorithms during training in terms of model performance measures

Training Algorithm	E	R	NRMSE	AARE	TS10	TS25	TS50	TS75	TS100
BPA (10-2-1)	0.9514	0.9754	0.5128	43.39	16.41	40.64	66.74	84.97	91.58
LMA (10-6-1)	0.9659	0.9828	0.4628	11.96	33.43	61.47	89.39	98.55	99.45
GA (10-5-1)	0.9617	0.9807	0.4550	29.68	26.10	55.29	84.31	92.90	97.52
PSO (10-4-1)	0.9552	0.9774	0.4918	17.56	34.75	80.40	96.31	99.34	99.72

Table 5. Results of different training algorithms during testing in terms of model performance measures

Training Algorithm	E	R	NRMSE	AARE	TS10	TS25	TS50	TS75	TS100
BPA (10-2-1)	0.9287	0.9648	0.5783	20.63	38.38	74.16	91.52	96.28	98.28
LMA (10-6-1)	0.9352	0.9671	0.5210	9.39	72.71	91.87	98.48	99.72	100.00
GA (10-5-1)	0.9361	0.9680	0.5141	9.05	72.92	90.9	97.93	99.66	99.79
PSO (10-4-1)	0.9210	0.9612	0.6088	12.17	55.20	91.11	98.21	99.66	99.93

5. SUMMARY AND CONCLUSIONS

This paper presents the findings of a study of comparison of the available gradient and non-gradient training algorithms of the ANN models. The gradient based training methods investigated include the popular back-propagation algorithm (BPA) and Levenberg-Marquardt Algorithm (LMA), and in non-gradient based methods genetic algorithm (GA) and particle swarm optimization (PSO) were investigated. The daily streamflow data derived from the Godavari River basin, INDIA, were employed to develop all ANN models investigated in this study. A wide variety of standard statistical performance evaluation measures were employed to evaluate the performances of various ANN models developed.

The findings of the study reported in this paper suggest that the LMA is the best in gradient based methods because of its predictive capability and faster convergence where as GA is the best in non-gradient based method which is not sensitive to initial weights because it is population based approach and reaches the solution close to global minima very quickly in few generations but it definitely takes more time to reach the solution close to global optima. The predictive capability of the LMA and GA trained ANN models were found to be much superior to those trained using PSO and BPA. The study presented in this paper employs PSO apart from GA to the field of hydrology and water resources; however, the approach adopted here can be easily extended to other branches of engineering involving complex physical systems. The suitability of the LMA and GA in training the ANN models needs to be explored further by applying this approach to different areas in engineering and sciences in order to exploit its advantages over the traditional approach of using BPA in learning the complex relationships. It is hoped that future research efforts will focus in these directions to improve the accuracy of the forecast models for better planning, design, operation, and management of various engineering systems.

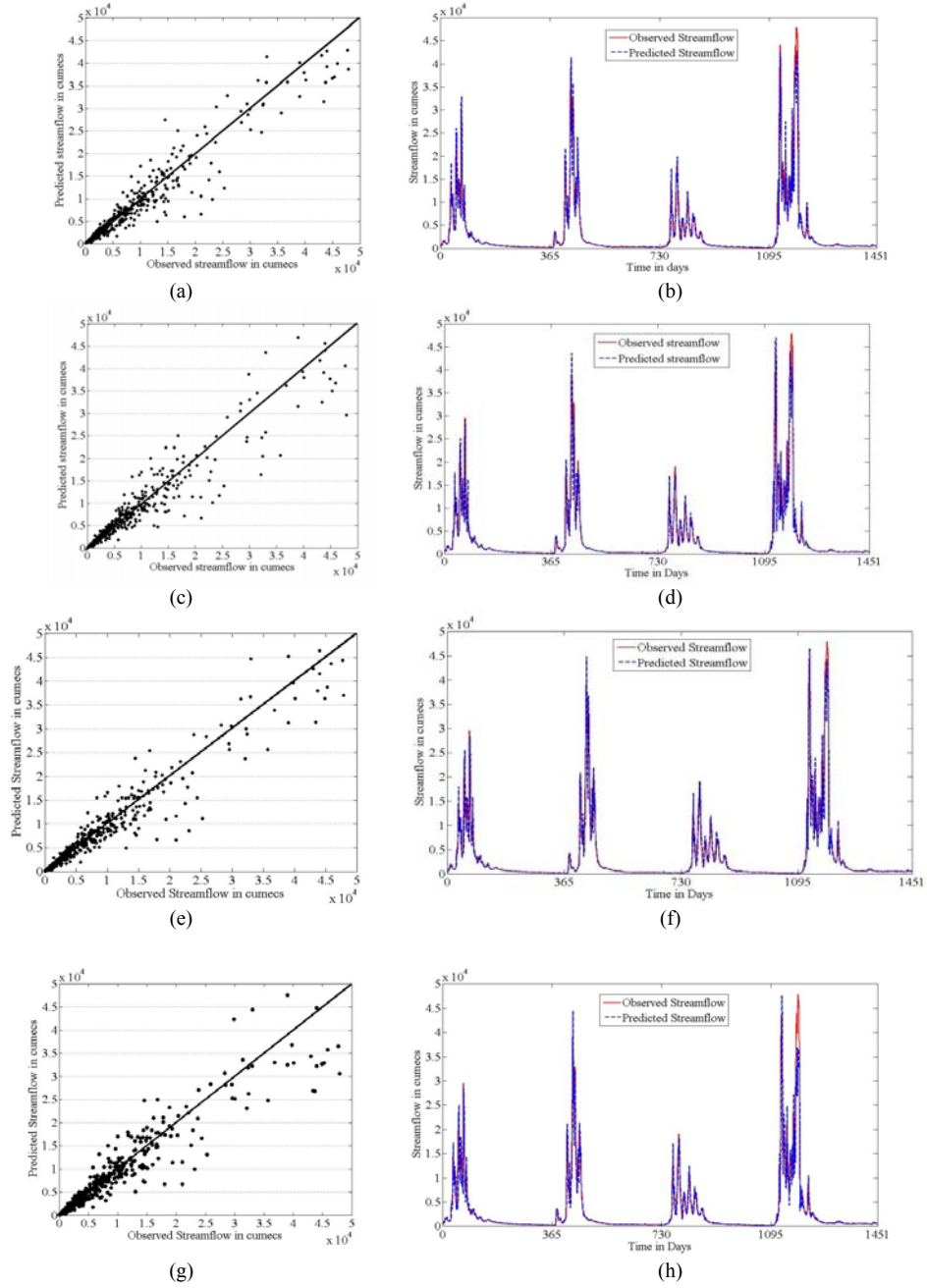


Figure.4. Scatter and timeseries plots of BPA ((a), (b)), LMA((c), (d)), GA ((e), (f)) and PSO ((g), (h)) during ANN testing

REFERENCES

- ASCE Task Committee on Application of Artificial Neural Networks in Hydrology, 2000 Task, 2000. Artificial neural networks in hydrology. I: preliminary concepts. *Journal of Hydrologic Engineering*, ASCE 5(2), 115-123.
- Bergh, F., 2001. An analysis of particle swarm optimizers. South Africa: University of Pretoria, Vol. 200, No. pp.
- Curry, B. and Morgan, P., 1997. *Neural networks: a need for caution*. Omega, Vol. 25, No. 1 pp. 123-133.
- Deb, K., 2004. *Optimization for engineering design: Algorithms and examples*. PHI Learning Pvt. Ltd.
- Eberhart, R.C. and Shi, Y., 2000. Comparing inertia weights and constriction factors in particle swarm optimization. *IEEE*, pp. 84-88 vol. 1.
- Goldberg, D.E., 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley. Reading, Ma, Vol. 131, No. pp.
- Govindaraju, R.S. and Rao, A.R., 2000. *Artificial neural networks in hydrology*. Springer Netherlands.
- Hagan, M.T. and Menhaj, M.B., 1994. Training feedforward networks with the Marquardt algorithm. *Neural Networks, IEEE Transactions on*, Vol. 5, No. 6 pp. 989-993.
- Jain, A. and Srinivasulu, S., 2006. Integrated approach to model decomposed flow hydrograph using artificial neural network and conceptual techniques. *Journal of Hydrology*, Vol. 317, No. 3-4 pp. 291-306.
- Kennedy, J. and Eberhart, R., 1995. Particle swarm optimization. *IEEE*, pp. 1942-1948 vol. 4.
- Maier, H.R. and Dandy, G.C., 2000. Neural networks for the prediction and forecasting of water resources variables: a review of modelling issues and applications. *Environmental Modelling and software*, Vol. 15, No. 1 pp. 101-124.
- Rumelhart, D.E., Hinton, G.E. and Williams, R.J., 1986. Learning representations by back-propagating errors. *Nature*, Vol. 323, No. 6088 pp. 533-536.
- Sastry, K., Goldberg, D. and Kendall, G., 2005. Genetic algorithms. *Search Methodologies*, Vol., No. pp. 97-125.
- Shi, Y. and Eberhart, R., 1998. A modified particle swarm optimizer. *IEEE*, pp. 69-73.
- Srinivasulu, S. and Jain, A., 2006. A comparative analysis of training methods for artificial neural network rainfall-runoff models. *Applied Soft Computing*, Vol. 6, No. 3 pp. 295-306.
- Vtkovský, J.P., Simpson, A.R. and Lambert, M.F., 2000. Leak detection and calibration using transients and genetic algorithms. *Journal of Water Resources Planning and Management*, Vol., No. pp. 263.
- Van den Bergh, F. and Engelbrecht, A.P., 2004. A cooperative approach to particle swarm optimization. *Evolutionary Computation, IEEE Transactions on*, Vol. 8, No. 3 pp. 225-239.
- Yitian, L. and Gu, R.R., 2003. Modeling flow and sediment transport in a river system using an artificial neural network. *Environmental Management*, Vol. 31, No. 1 pp. 122-134.
- Zhang, G. and Eddy Patuwo, B., 1998. Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting*, Vol. 14, No. 1 pp. 35-62.