A new hybrid artificial neural networks for rainfall–runoff process modeling

Article *in* Neurocomputing · September 2013

DOI: 10.1016/j.neucom.2013.05.023

CITATIONS
139

READS
1,396

4 authors, including:

Jamal Shahrabi
Amirkabir University of Technology
67 PUBLICATIONS **1,836** CITATIONS

SEE PROFILE

Peyman Abbaszadeh
Princeton University
34 PUBLICATIONS **1,347** CITATIONS

SEE PROFILE

Neurocomputing 121 (2013) 470–480

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

A new hybrid artificial neural networks for rainfall–runoff process modeling

Shahrokh Asadi [a], Jamal Shahrabi [a,*], Peyman Abbaszadeh [b], Shabnam Tabanmehr [a]

[a] Department of Industrial Engineering, Amirkabir University of Technology, P.O. Box 15875-4413, Tehran, Iran
[b] Department of Civil and Environmental Engineering, Amirkabir University of Technology, Tehran, Iran

ARTICLE INFO

Article history:
Received 1 December 2012
Received in revised form
16 April 2013
Accepted 12 May 2013
Communicated by W.S. Hong
Available online 29 June 2013

Keywords:
Rainfall–runoff modeling
Genetic algorithms
Levenberg–Marquardt algorithm
Data pre-processing
Data clustering

ABSTRACT

This paper proposes a hybrid intelligent model for runoff prediction. The proposed model is a combination of data preprocessing methods, genetic algorithms and levenberg–marquardt (LM) algorithm for learning feed forward neural networks. Actually it evolves neural network initial weights for tuning with LM algorithm by using genetic algorithm. We also use data pre-processing methods such as data transformation, input variables selection and data clustering for improving the accuracy of the model. The capability of the proposed method is tested by applying it to predict runoff at the Aghchai watershed. The results show that this approach is able to predict runoff more accurately than Artificial Neural Network (ANN) and Adaptive Neuro Fuzzy Inference System (ANFIS) models.

1. Introduction

The rainfall–runoff process modeling plays an important role in optimizing and planning water resources, managing reservoirs particularly during drought periods and also has become a fundamental area of research in hydrology recently.

Over the past couple of decades, there have been wide attempts to identify and profound the understanding of rainfall–runoff process. Modeling techniques can be broadly classified into two categories: the theory-driven (conceptual and physical-based) approach and the data-driven (empirical and black box) approach [1]. In this context, Data-Driven Models (DDMs) may be preferable to discover relationships from input–output data even when the user does not have a complete physical understanding of the underlying processes [2]. While such models are very useful for river flow forecasting, the main concern is to have accurate predictions of flow at specific river locations [3].

Auto Regressive Integrated Moving Average (ARIMA) or Seasonal ARIMA with exogenous input (SARIMAX) as conventional black box time series models are widely used for hydrological time series forecasting. These models use piecewise linear function as basic element of prediction model [4,5]. Time series models are more practical than conceptual models because it is not essential to understand the internal structure of the physical processes that are taking place in the system being modeled [6]. However, they are basically linear models assuming that data are stationary, and have limited ability in capturing that which is non-stationary and non-linearity in hydrologic data [7]. To overcome the mentioned shortage and due to the complexity of runoff phenomena in the rivers, most of the times there are not any close matches between observed and predicted data by the mentioned model, thus Artificial Intelligence (AI) approaches are more and more popular for the engineers and play an increasingly important role in the water resource as well as rainfall–runoff modeling.

AI is one of the intelligent systems that computerizes human reasoning and found a challenging test bed for various paradigms in many areas including hydrological time series prediction. Extensive research has resulted in numerous prediction applications using artificial neural networks (ANN), fuzzy logic and genetic algorithms (GA), data mining and other techniques [8–23]. Most of the progresses in AI have been made in the areas of problem solving; concepts and methods for building programs that reason about problems rather than calculating a solution for them. Based on the success of ANN in the field of water resources and assuming that the simplest model that can satisfactorily describe the system for the given input data should be used, ANNs, which belong to the class of black box models [24], are explored in this paper for the development of runoff prediction. ANNs are flexible mathematical structures, which are capable of identifying complex non-linear relationships between input and output data [25]. In recent years numerous papers have been published on the successful application of ANNs for modeling rainfall–runoff as one of the highly stochastic, non-linear and complex natural phenomenon [26–32].

* Corresponding author. Tel.: +98 2164545348.
E-mail addresses: s.asadi@aut.ac.ir, s.asadi@gmail.com, s.asadi520@gmail.com (S. Asadi), jamalshahrabi@aut.ac.ir (J. Shahrabi) Peyman.aut@gmail.com (P. Abbaszadeh), shabnam@aut.ac.ir (S. Tabanmehr).

0925-2312/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2013.05.023

ANNs are one of the strongest AI models which can learn the complex nature of the relationship between inputs and outputs. The feed forward back propagation (BP) network is a common architecture among different types of neural networks between engineers. So-called BP network models, three-layer BP network (single hidden layer) as a general approximator is satisfying for forecasting and simulating in the science of water [33,34]. Three-layered feed forward neural networks (FFNNs), which have been widely chosen to forecast hydrologic time series, provide enough complexity to accurately simulate the set of input and output variables.

When developing a feed forward neural network model for prediction purposes, the architecture identification includes determining number of input, hidden, and output neurons and weight training which are important tasks. Weight training in ANNs is usually formulated as a minimization of an error function, such as the mean square error between target and actual outputs averaged overall training data by iteratively adjusting connection weights. Most training algorithms, such as back propagation and conjugate gradient are based on gradient descent [35].

Among literatures regarding using the ANNs as the prediction tool, most of them focus on back propagation Neural Network (NN). BP is characterized by very poor convergence. Several improvements for BP, such as the quick-propagation (QP) algorithm, resilient error back propagation, etc. were developed. Much better results can be obtained using second order methods such as Newton or Levenberg–Marquardt (LM). The Levenberg–Marquardt back propagation (LMBP) is a powerful optimization technique that was introduced to the neural net research because it provided methods to accelerate the training and convergence of the algorithm. It utilized the BP procedures in which derivatives were processed from the last layer of the network to the first [36].

Yet, two shortcomings exist in using LMBP procedure: firstly, despite the fact that the LMBP has shown success in some areas, the algorithm often gets trapped in a local minimum of the error function and when the error function is multimodal and/or non-differentiable, it is not capable of finding a global minimum [37], secondly, it cannot function well in networks with more than two or three hidden layers [38]. The aforementioned issues plus other problems have guided the researches toward employing evolutionary techniques to find the best set of network weights [39].

Using GA to train and optimize neural networks has been an active area of research since mid-90s [40–45]. Dorsey and Mayer [45] succeeded in using GA to find the global solution when optimizing complicated non-linear functions. Since NN is a complicated non-linear function, their approach was applied by Sexton et al. [44] to train and optimize a neural network. That study showed that NN trained and optimized using a GA outperformed NN trained using back propagation in a range of various problems and datasets [44]. In the hydrological forecasting context, recent experiments have reported that combination of evolutionary techniques and ANNs may offer a promising alternative for hydrologic process modeling [29–31].

Evolutionary techniques have several obvious advantages over BP: genetic algorithms and other evolutionary approaches are able to find global minima in complex, multimodal spaces, and do not require a differentiable error function and are more flexible allowing the fitness evaluation to be changed in order to consider extra factors that are not easy to be incorporated in the BP algorithm [46]. Many researchers have used GA for learning neural networks and they have found that GAs have better performance compared with BPs [47–49].

There are several studies on the applicability of evolutionary neural networks (ENN) in the hydrological sciences that have been manifested in recent years [50]. These works suggest that some evolutionary computation methods may be more suitable for ANN training than the others [51]. Cheng et al. [52] used a hybrid fuzzy optimal genetic algorithm to solve multi-objective rainfall–runoff model calibration. Jain and Srinivasulu [53] combined non-binary genetic algorithm and artificial neural network techniques to develop rainfall–runoff models. Some investigations into neural network training, determination of nodes number and exploration of input variables using genetic algorithms as evolutionary neural network have been successfully employed in hydrological forecasting [54–56].

Due to the local convergence of the LMBP, it can be illustrated that solutions are highly dependent upon the initial random draw of weights. If these initial weights are imposed on a local region, which is probable, the BP algorithm will likely become trapped in a local solution that may or may not be the global solution. When NNs are used in real-world applications the local convergence can cause serious problems. So global search techniques must be applied in order to network training. Considering the aforementioned points, in the current research the powerful combination of positive aspects of genetic algorithms and LMBP algorithm is presented to predict runoff one-step ahead. Proposed method is a combination of genetic algorithms and LMBP artificial neural networks. Global search capabilities of genetic algorithms with the ability of LMBP algorithm in the local search are combined. At first, genetic algorithm is used with a broad search to find weights of artificial neural network. After making the searching space smaller, then these weights are used as initial weights for the LMBP algorithm. LMBP algorithm around a global search with a local search acquires the best possible or weights of network [39].

Data quality is a key issue in prediction concepts. To increase the accuracy of the prediction, we perform data pre-processing techniques such as data transformation, input selection and data clustering. Atsalakis and Valavanis [57] pointed out that input data pre-processing may affect prediction performance. In many cases input data have a large range of values reducing the effectiveness of training procedures. One solution to this impediment is applying data transformation techniques such as data scaling. The process of choosing variables as inputs through sensitivity analysis may help eliminate redundant inputs. Besides, in recent years a mathematical tool called Wavelet Transform (WT) as a data preprocessing technique has been widely used in improving capability of the AI models in hydrological time series forecasting [7]. Nourani et al. [7] presented Wavelet-ANN (WANN) and Wavelet-Adaptive Neuro Fuzzy Inference System (WANFIS) in order to Aghchai watershed runoff prediction.

One of the best ways to improve accuracy of a forecasting model is using modular approaches. In modular models, a task or problem is decomposed into a number of subtasks and each module handles a subtask of the global task [58]. There are different motivations for using modular approaches which are as: to improve performance, to reduce model complexity to simplify the problem to recombine sensory information [59]. Using data clustering, we can modularize the prediction problem and improve the accuracy of the model [60].

Data clustering is used in different studies to divide the data into sub-populations and reduce the complexity of the whole data space to something more homogeneous and reduce the effects of noisy data [15]. All have reported that using the data clustering model improves the forecasting accuracy [60].

There are some investigations into neural network training using genetic algorithms which have been successfully employed to overcome the inherent limitations of the BP [29,40]. Genetic algorithms have been used with neural network to search for input variables [48] or to determine the number of nodes or connections in the network [49]. However, there are still few articles concerning evolutionary neural network in hydrological forecasting that have been published in the literature. Data pre-processing, such as data clustering, plays a crucial role in upgrading the model

accuracy. Thus, this paper demonstrates a hybrid intelligent model for runoff prediction in which the proposed model called PELMNN (pre-processed evolutionary LM neural networks) is a combination of genetic algorithms and LM neural networks equipped with pre-processing and also post-processing concepts. The capability of the proposed model has been tested by applying it to predict Aghchai watershed runoff. Moreover, in order to have a hard test for the proposed model during the modeling process more days ahead forecasting (e.g. 3-day and 7-day) was investigated.

2. Methodology

A time series is a time dependent sequence of points, generally equidistant, such as$Y_t = \{y_t \in R | t = 1, 2, 3, \cdots, M\}$, where t is the chronological or temporal index and M is the number of observations. Therefore, $y_t$is a sequence of temporal observations equally spaced. The main purpose of time series prediction techniques is to estimate the future values (unknown) of the series based on the past observations either from the series itself or from exogenous data. This can be accomplished by identifying certain regular patterns presented in the historical data. In this context, a crucial factor for a good forecasting performance is the correct choice of the time lags considered for representing the series [61]. A time series is considered as nonlinear function of several input variable as $y_t^* = f(x_1^*, x_2^*, x_3^*, \cdots, x_n^*)$, Where f is a nonlinear function determined by the neural network, x^* is the normalized value of x_{old}(input variable) and n is integer [62].

This paper presents four-stage architecture to develop a hybrid intelligent model for the rainfall–runoff modeling. The first stage is data pre-processing. In this stage, we apply a data transformation technique to scale data then stepwise regression analysis (SRA) is used to choose the key variables that are to be considered in the model. After that K-means clustering is used to divide the data into sub-populations and reduce the complexity of the whole data space to something more homogeneous.

In the second stage, we employ genetic algorithm as a global search method to evolve artificial neural networks' initial weights by using the proposed ENN. In the third stage, we tune the obtained weights in previous stage using LMBP algorithm. In the last stage, we post-process the outcomes and generate predicted values. General framework of PELMNN is shown in Fig. 1. Each stage is detailed in the following.

2.1. Data pre-processing

2.1.1. Data transformation

Using transformed data is more useful in most heuristic methods especially when dealing with forecasting problems [63]. A pre-processing method should contain the capability of transforming pre-processed data into its original scale (called post-processing). One of the most useful data transformation techniques is data normalization which is used in different forecasting studies. In this paper the time series data before going through the network are normalized between 0.05 and 1. If X, X_{max}, X_{min} are the original, maximum and minimum values of the raw data, respectively then the normalization of X called X^*, can be obtained by the following transformation function:

$$X^* = 0.05 + 0.95\frac{X - X_{min}}{X_{max} - X_{min}} \quad (1)$$

2.1.2. Variable selection by stepwise regression analysis

Variable selection is the process of selecting an optimum subset of input variables from the set of potentially useful variables which may be available in a given problem. Different researchers have applied variety of feature selection methods such as genetic algorithm [64] principal component analysis [65] and stepwise regression analysis (SRA) to select key factors in their forecasting systems. Among them, in recent years researchers have used SRA for input variable selection in the field of forecasting and they have obtained very promising results [47]. So, in this paper we adopt stepwise regression to analyze and select variables, and its consequence improves the forecasting accuracy of the system. Stepwise regression method determines the set of independent factors that most closely determine the dependent variable. This task is carried out by the means of the repetition of a variable selection. At each of these steps, a single variable is either entered to or removed from the model. For each step, simple regression is performed using the previously included independent variables and one of the excluded variables. Detail of SRA is shown in Fig. 1.

2.1.3. K-means clustering model

Clustering is finding groups of objects in a way that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups. A good clustering method will produce high-quality clusters with high intra-class similarity and low inter-class similarity. In this research, there is a heterogonous population of body size that must be grouped into a more homogenous population.

Clustering algorithms are classified into two groups: the agglomerative hierarchical algorithms [66] such as the centroid and Ward methods and the nonhierarchical clustering [67], such as K-means and self-organizing map (SOM) neural networks. Each of these algorithms has their own advantages and disadvantages. Dependent on the application, a particular type of clustering method should be chosen. Among clustering algorithms, because of simplicity and fast convergence of K-means, it has been used in a wide range of applications [15,68–70]. The K-means method [66] is a technique employed for partitioning a set of objects into k groups such that each group is homogeneous with respect to certain attributes based on the specific criterion. We use K-means data clustering to: divide the data into sub-populations and reduce the complexity of the whole data space, decompose complicated patterns to a set of simple patterns and decrease noise effects by focusing on similar data in each cluster and consequently have higher accuracy. The procedures of the k-means method can be summarized as Fig. 1.

2.1.4. Artificial neural networks

ANNs are flexible computing frameworks for modeling a broad range of nonlinear problems. One significant advantage of the ANN models over other classes of nonlinear model is that ANNs are universal approximators which can approximate a large class of functions with a high degree of accuracy. Their power comes from the parallel processing of the information from the data. No prior assumption of the model form is required in the model building process. Instead, the network model is largely determined by the characteristics of the data.

ANNs consist of an inter-connection of the number of neurons. There are many varieties of connections being studied; however, here we discuss only one type of network which is called multi-layer perceptron (MLP). In this network the data flow forward to the output continuously without any feedback. We have used a typical three-layer feed forward model. The input nodes are the multi-scale time series of rainfall and runoff data, while the one-day-ahead runoff prediction is the network output. Hidden nodes with appropriate nonlinear transfer functions are used to process the information received by the input nodes. The model can be

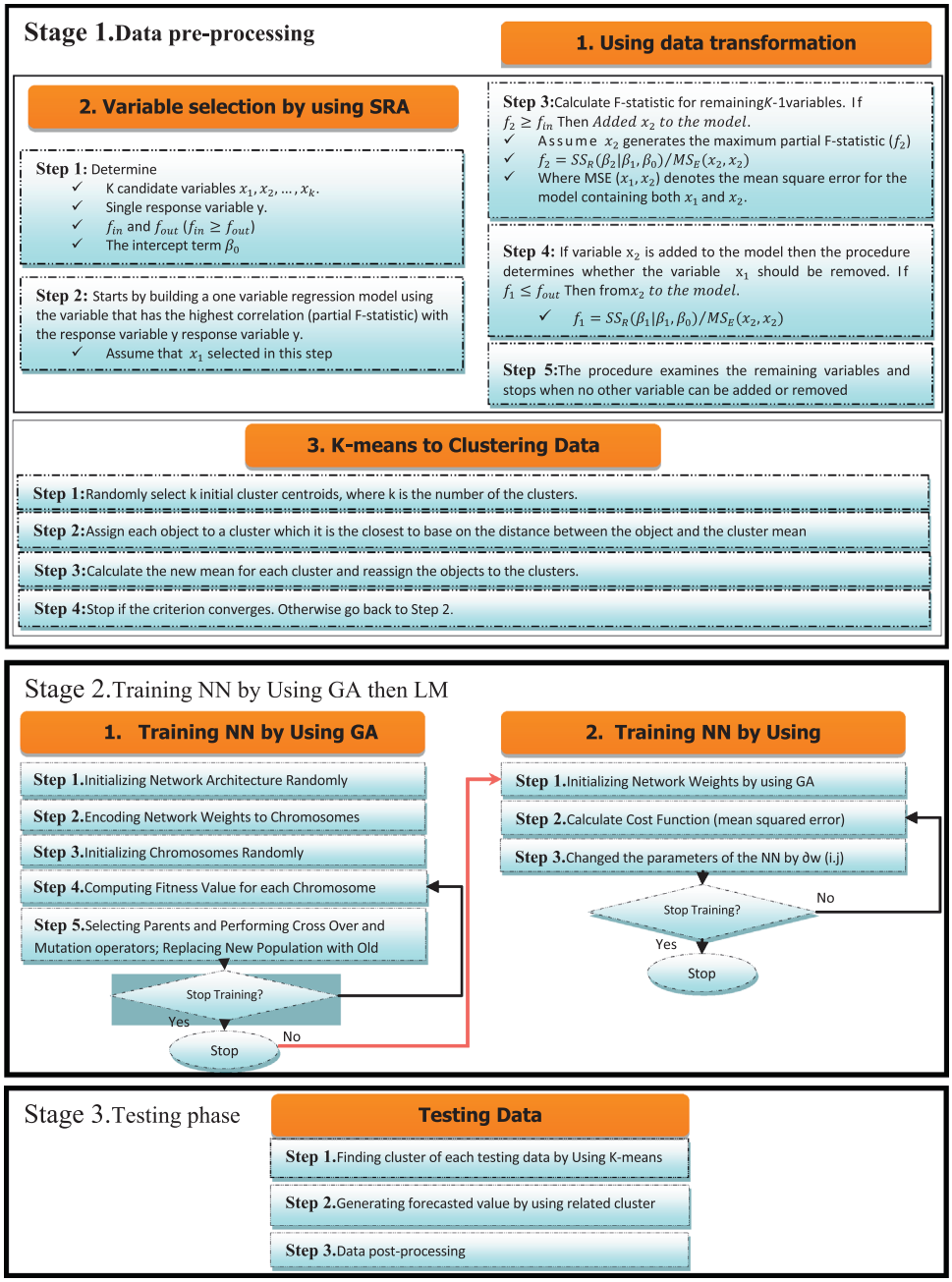

Fig. 1. Framework of PELMNN.

written as Eq. (2).

$$Y_t = P_t + \varepsilon_t = w_0 + \sum_{j=1}^{q} w_j . g\left(\sum_{i=1}^{p} w_{ij} y_i + w_{0j} \right) + \varepsilon_t \tag{2}$$

where w_{ij} $(i = 0, 1, 2, \cdots p;\ j = 1, 2, \cdots q)$ and w_j $(j = 0, 1, 2, \cdots, q)$ are model parameters often called connection weights; p is the number of nodes; and q is the number of hidden nodes; $P_t$is predicted value. Activation function can take several forms. The type of activation function is indicated by situation of the neuron within the network. In the majority of case input layer neurons there is no activation function, as their role is to transfer the inputs to the hidden layer. The most widely used activation function is non-linear activation that may introduce distortion to the predicted output. The logistic, $f(x) = 1/1 + \exp(-x)$, and g is hyperbolic tangent (tanh), $g(x) = \exp(x) - \exp(-x)/\exp(x) + \exp(-x)$, and hyperbolic function are often used as the hidden layer transfer function.

Cost function is an overall accuracy criterion such as the following mean squared error:

$$E = \frac{1}{N} \sum_{n=1}^{N} (e_i)^2 = \frac{1}{N} \sum_{n=1}^{N} (Y_t - P_t)^2$$

$$= \frac{1}{N} \sum_{n=1}^{N} \left(y_t - \left(w_0 + \sum_{j=1}^{q} w_{jg} (w_{0j} + \sum_{j=1}^{p} w_{i,j} y_{t-j}) \right) \right)^2 \tag{3}$$

where, N is the number of error terms. This minimization is done with some efficient nonlinear optimization algorithms rather than the basic back propagation training algorithm [71], in which the parameters of the neural network, W are changed by an amount ∂W,

according to the following formula:

$$\Delta W = -\eta \frac{\partial E}{\partial W} \tag{4}$$

where, the parameter η is the learning rate and $\partial E/\partial W$ is the partial derivative of the function E with respect to the weights of matrix W. This derivative is commonly computed in two passes. In the forward pass, an input vector from the training set is applied to the input units of the network and is propagated through the network, layer by layer, producing the final output. During the backward pass, the output of the network is compared with the desired output and the resulting error is then propagated backward through the network, adjusting the weights accordingly.

The Levenberg–Marquardt algorithm is the most widely used optimization algorithm. It outperforms simple gradient descent and other conjugate gradient methods in a wide variety of problems. LM algorithm is a second order algorithm that many times is overlooked by those attempting to train neural networks possibly because it is more complex to implement than Error Back Propagation (EBP). However, it definitely makes up for this in superior performance. LM is similar to EBP in that it requires the calculation of the gradient vector, but in addition, LM also computes the Jacobian. The gradient vector is represented as

$$g = \begin{pmatrix} \frac{\partial E}{\partial W_1} \\ \frac{\partial E}{\partial W_2} \\ . \\ . \\ . \\ \frac{\partial E}{\partial W_\Omega} \end{pmatrix} \tag{5}$$

where E the total error of the network for that pattern and W is refers to the weights. The Jacobian is essentially every gradient for every training pattern and network output. The Jacobian is shown below.

$$J = \left[J_{ij}\right] = \left[\frac{\partial E_j}{\partial W_i}\right] = \begin{bmatrix} \frac{\partial E_1}{\partial W_1} & \frac{\partial E_1}{\partial W_2} & \cdots & \frac{\partial E_1}{W_\Omega} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial E_2}{\partial W_1} & \frac{\partial E_2}{\partial W_2} & \cdots & \frac{\partial E_2}{\partial W_\Omega} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial E_N}{\partial W_1} & \frac{\partial E_N}{\partial W_2} & \cdots & \frac{\partial E_N}{\partial W_\Omega} \end{bmatrix}; \quad i = 1:\Omega;\ j = 1:N \tag{6}$$

where Ω and N are the number of weights and the number of patterns, respectively. In other words the Jacobian will have as many columns as the number of weights, and the number of rows will be equal to the product of Ω and N. Once the Jacobian is calculated, the LM algorithm can be represented by the following:

$$W_{k+1} = W_k - (J_k^T J_k + \mu I)^{-1} J_k^T E \tag{7}$$

I is the identity matrix, and μ is a learning parameter. The learning parameter μ is then adjusted several times in each iteration and the result with the greatest reduction of error is selected. Where J is the Jacobian matrix, μ is the learning rate which is to be updated using β depending on the outcome. In particular, μ is multiplied by decay rate β) μ is divided by β, $0 < \beta < 1$) When μ value is very large the LM algorithm becomes the steepest decent or BP, and when μ is equal to zero it is the Newton method. The entire process is then repeated until the error is reduced to the required value [72,73].

This second order algorithm is significantly faster than BP. For small networks with few training patterns this is not a major issue, but for networks with many training patterns it is very computationally intensive. This inversion will cause each training iteration for LM to take longer than iteration for BP. The required training time will still be far less than that BP, because the LM will require such few iterations [36].

2.1.5. Genetic algorithm

GAs are search methods based on principles of natural selection and genetics. GAs encode the decision variables of a search problem into finite-length strings of alphabets of certain cardinality. The strings which are candidate solutions to the search problem are referred to as chromosomes, the alphabets are referred to as genes and the values of genes are called alleles. For example, in a problem such as traveling salesman problem, a chromosome represents a route, and a gene may represent a city. In contrast to traditional optimization techniques, GAs work with parameters coding, rather than the parameters themselves. To evolve good solutions and to implement natural selection, we need a measure for distinguishing between good and bad solutions. The measure could be an objective function that is a mathematical model or a computer simulation, or it can be a subjective function where humans choose better solutions rather than worse ones. In essence, the fitness measure must determine a candidate solution relative fitness, which will be subsequently used by GA to guide the evolution of good solutions [74,75].

2.1.6. Evolutionary neural networks (ENN)

Areas where GAs have met with great success are in the training and optimization of Neural Networks. GAs have been used in both construction and training of neural networks [76–78] Janson and Frenzel's [77] research showed that GA significantly outperformed BP in training a neural network to predict the optimum transistor width in a complementary metal-oxide semiconductor switch.

In this section, we apply GAs to evolve the weights between neurons in different layers in the neural network. The steps needed for evolving connection weights are described below [76–78]:

Step 1—Encoding

Each gene represents the weight between two neurons in different layers. A chromosome is constructed from a series of genes as shown in Fig. 2. In this figure, for a normal feed forward neural network that has three neurons in input layer, two neurons in hidden layer and one neuron in output layer, the first gene in the chromosome is the weight between neuron one and neuron four, i.e. W_{14}, the Second gene is the weight between neuron one and neuron five, i.e. W_{15} and so on. We use real number form to represent the connection weights.

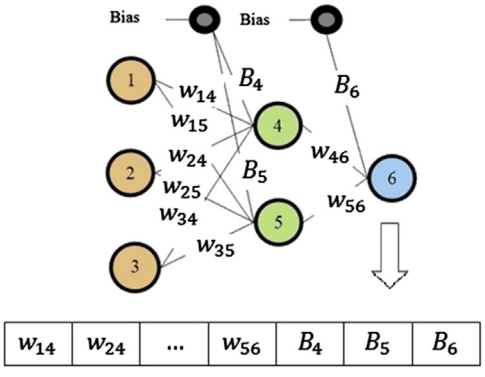

Fig. 2. Chromosome encoding.

Step 2—Generating the initial population
The initial population (N_{POP}) is generated randomly. Each of initial weights is randomly generated between −1 and 1.
Step 3—Calculating the fitness values
As regards to the fitness function, it is based on the root mean squared error (RMSE) over a training data set, which is represented by the following expression:

$$\mathrm{RMSE}(C_j)=\sqrt{\frac{1}{N}\sum_{i=1}^{N}(Y_i-P_i)^2} \tag{8}$$

where Y_i is the actual value and P_i is the output value of ith training data obtained from the neural network using the weights coded in jth chromosome (C_j) and N is the number of training data.
Step 4—Selection mechanism
We use truncation selection scheme for selection procedure. In truncation selection, individuals are sorted according to their fitness. Only the best individuals are selected for parents. The truncation threshold indicates the proportion of the population to be selected as parents. Then we use binary tournament selection scheme for selection of parents for generating new offsprings by using genetic operators. In binary tournament selection, two members of the population are selected at random and their fitness are compared and the best one according to fitness value will be chosen as one parent. Also another parent is selected with the same procedure.
Step 5—Genetic operators
We use two-point crossover and one point mutation [79] for genetic operators.
Step 6—Replacement
The current population is replaced by the newly generated offsprings, which forms the next generation.
Step 7—Stopping criteria
If the number of generations equals to the maximum generation number, then stop; otherwise go to step 3.

2.1.7. Testing phase

This stage is quite simpler. In this stage output dataset is post-processed (returned to its original scale) and the predicted values are generated by the trained network.

3. Experimental results

3.1. Study area

The data used in this paper is from Aghchai watershed, located in northwest Iran at Azerbaijan province (between 38°40′ and 39°30′ North latitude and 44°10′and 44°57′ East longitude). The watershed area is 1440 km^2 (Fig. 3) and its main channel is a sub-branch of the Ajichai River which discharges to Urmieh Lake. Watershed Elevation varies between 1168 m and about 3280 m above sea level and its longest waterway has 64.88 km length. Watershed topography is quite steep, the average slope is 25%. The time series data for 12 years (from September 1995 to October 2007) were used in the modeling process (the first nine years for training and the rest three years for verification). The statistic characteristics of rainfall and runoff for Aghchai watershed in daily time scale is tabulated in Table 1.

Table 1
Statistical characteristics of rainfall and runoff data Aghchai watershed.

Case study	Rainfall time series (mm)				Runoff time series (m^3/s)			
Daily data collection	Max	Min	Mean	Standard deviation	Max	Min	Mean	Standard deviation
Calibration data set	29	0	0.69	6.75	85.40	0.12	4.03	33.81
Verification data set	30	0	0.91	8.96	68.30	0.1	4.72	33.54

3.2. Input variables

In general, the causal variables involved in rainfall–runoff relations are those associated with rainfall (precipitation), previous flows, evaporation, temperature, etc. According to the available data, the input variables tend to be varied in previous studies. Most studies employed rainfall and previous flow (or water level) as inputs [80]. Thus, the best input combination is selected according to the stepwise regression method in which both pervious rainfall and runoff time series that are more related to $Q_{(t+1)}$, are considered as input for proposed model.

3.3. Efficiency criteria

We use the following evaluation metrics to measure the performance of the proposed model:

$$R^2=1-\frac{\sum_{i=1}^{N}(Y_i-P_i)^2}{\sum_{i=1}^{N}(Y_i-\overline{Y}_1)^2} \tag{9}$$

$$RMSE=\sqrt{\frac{1}{N}\sum_{i=1}^{N}(Y_i-P_i)^2} \tag{10}$$

where R^2, $RMSE$, N, Y_i, P_i and $\overline{Y}_i$ are determination coefficient, Root Mean Squared Error, number of observations, observed data, computed values and mean of observed data, respectively. Likewise, Legates and McCabe [81] indicated that a hydrological model can be sufficiently evaluated by these two statistics. The RMSE is used to measure forecasting accuracy, which produces a positive value by squaring the errors. The RMSE increases from zero for perfect forecasts to large positive values as the discrepancies between forecasts and observations become increasingly large. Obviously high value for R^2 (up to one) and small value for RMSE indicate high efficiency of the model. Moreover, due to upmost importance of the extreme values in the rainfall–runoff modeling, Eq. (11) can be used to compare the ability of different models in capturing the peak values in runoff time series as similar as Eq. (9) for the total data.

$$R^2_{Peak}=1-\frac{\sum_{i=1}^{n}(Q_{PC_i}-Q_{PO_i})^2}{\sum_{i=1}^{n}(Q_{PO_i}-\overline{Q}_{PO})} \tag{11}$$

where R^2_{Peak} determination coefficient for peak values, n number of peak values, Q_{PO_i}, Q_{PC_i} and $\overline{Q}_{PO}$ are observed data, computed values and mean of observed data for peak values, respectively.

Similar to the above mentioned formula, the coefficient of persistence can also be used in order to evaluate the capability of the model in capturing the peak values in runoff time series [82] (not used in this paper). These measures of evaluation are more appropriate for analyzing the flood forecasting performance.

3.4. Implementing PELMNN for modeling rainfall–runoff process

Data normalization is necessary for two reasons: first, all entries need to have the same weight. If the inputs of two neurons lie in different ranges, then the neuron with the larger absolute scale will be favored during training. Second, because of the neurons' transfer functions a sigmoid function is calculated and consequently these can only be performed over a limited range of values [83].

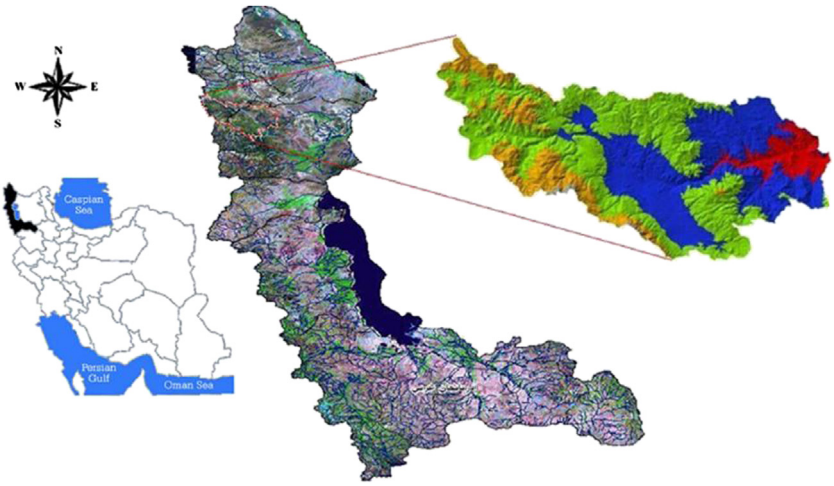

Fig. 3. Aghchai watershed.

Then the SRA [84–89] is used in order to eliminate low impact factors and choose the most influential ones out of mentioned factors. The criterion for adding or removing is determined by *F*-test statistic value and decreasing the sum of squared error. After the entrance of first variable to the model, the variable number is increased step by step; once it is removed from this model, it will never enter the model again. Before selecting variables, the critical point, level of significant and the values of Fe (F-to-enter) and Fr (F-to-remove) have to be determined first. Then the partial *F*-value of each step has to be calculated and compared to Fe and Fr; If $F > F_e$, it is considered to add variables to the model; otherwise, if $F < F_r$, the variables are removed from model [64].The statistical software SPSS 17.0 was used to apply stepwise regression analysis in this research considering $F_e = 3.84$ and $F_r = 2.71$. Hence, runoff at time steps $t-1(Q_{t-1})$, $t-2$ (Q_{t-2}) and rainfall at time step $t-1(I_{t-1})$ are considered as inputs to the model. After SRA, lags are inputted into the K-means model and different clusters are generated, where each cluster contains a portion of the training data. The try-and-error process is used for determining the number of clusters. After examination of the different number of clusters such as 2 and 3, it was obtained that in our case the suitable number of clusters (k) that provides minimum R^2 value for our case is 2, Moreover, Fig. 4 visualizes the two clusters of data.

Afterwards, we use GA to evolve initial weights of neural networks by using ENN. Then the weights which were obtained in previous stage are tuned by using LMBP algorithm. Finely tuning the parameters of a meta-heuristic algorithm is almost always a difficult task and the parameter values may have a very strong effect on the results of the meta-heuristic for each problem [85]. We start with an initial set of parameters; it is recommended to start with the values which are known to be good for a number of numeric test problems, accordingly we improve the architecture of the model by sampling in parameter space, considering a stream of instances, sequentially evaluating candidates by comparing number of hit rates, discarding statistically worse candidates and selecting the winner architecture. In order to identify the best network architecture with the least error, different feature of parameters such as transfer function types, number of hidden layers, number of nodes for each layer and suitable features of genetic algorithm has been examined. Best obtained features of PELMNN after tuning process for both clusters are added up in Table 2 and also in tuning features of LMBP part of PELMNN the learning rate and iterations as optimum features are obtained 0.2 and 3000, respectively. As the last stage of modeling process, runoff prediction is done by the means of test data and outcomes are post-processed (returned to original scale). The architecture of the designed network is shown in Fig. 5.

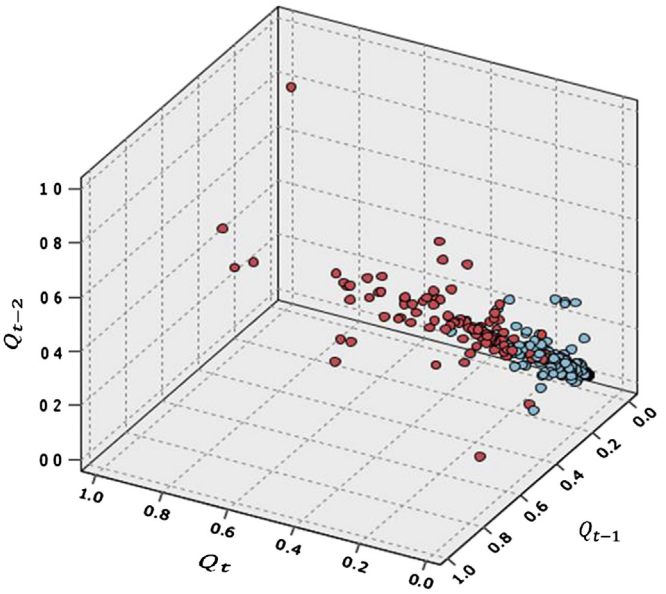

Fig. 4. K-means clustering results.

3.5. Performance analysis of PELMNN

Finally, in order to evaluate the proposed model performance (PELMNN model with clustering and without clustering), the obtained results were compared with results from ANN (BPNN), WANN, ANFIS and WANFIS models which were presented by Nourani et al [7] (it should be noted that this model is applied to the same datasets with the same portion of training and test data) and also with results of Pre-processing-ENN (PENN) and Pre-processing-Evolutionary back propagation neural network (PEBPNN) methods. A comparison of observed versus simulated daily runoff for a testing period is shown in Fig. 6. Comparisons of the performance of the proposed model with other models are shown in Table 3. In addition to the results presented in Table 3, the scatter plot of the verification stage is shown in Fig. 6.

Whereas multi-step ahead forecasting is quite essential to show that the proposed model is properly tested or not, the potential of PELMNN model (with clustering and without clustering) for 3 and 7 days ahead flow forecasting was investigated in the current research for the Aghchai watershed. The PELMNN model was

Table 2
Tuned features of ENN part of PELMNN.

Optimum features	Parameters	Cluster_1	Cluster_1
Training	Population size	100	120
	Crossover rate	0.85	0.8
	Mutation rate	0.05	0.05
	Truncation threshold	0.2	0.2
	# Iterations	3000	3300
Network architecture (input-hidden-output)	3-4-4-1		
Transfer function	Sigmoid–Sigmoid–Tanh		

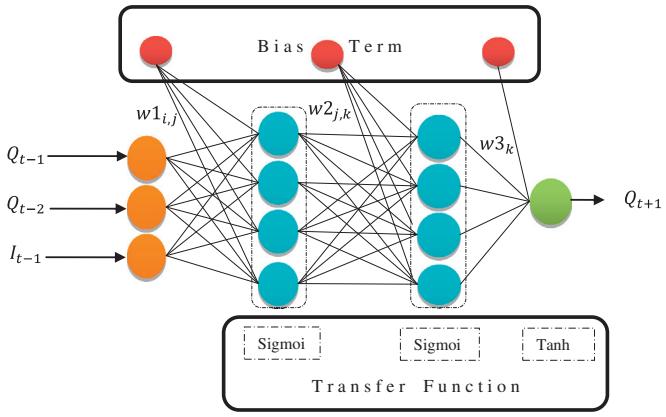

Fig. 5. Architecture of the best fitted PELMNN model.

Table 3
Comparison of different rainfall–runoff models.

Models	Calibration		Verification	
	R^2	RMSE	R^2	RMSE
ANN	0.80	0.039	0.71	0.043
WANN[a]	0.90	0.025	0.88	0.034
ANFIS[a]	0.89	0.026	0.86	0.036
WANFIS[a]	0.93	0.020	0.92	0.020
PENN	0.85	0.036	0.81	0.038
PEBPNN	0.88	0.029	0.85	0.036
PELMNN (without clustering)	0.926	0.020	0.913	0.022
PELMNN (with clustering)	0.95	0.017	0.94	0.018

[a] The reference for ANFIS, WANN and WANFIS model is Nourani et al. [7].

compared to ANN model (BPNN) for 3 and 7 days ahead flow forecasting. It was determined that for both 3 and 7 days lead time forecasting, the PELMNN model provided more accurate results than the ANN model (BPNN). Table 4 shows the ANN (BPNN) and PELMNN model performance statistics (R^2 and RMSE) for the best ANN (BPNN) and PELMNN models for both 3 and 7 days ahead flow forecasting.

According to the obtained results, it is clear that the PELMNN model is more efficient than the auto regressive (i.e., ANN, PENN, PEBPNN and ANFIS) and seasonal models (i.e., WANN and WANFIS) in forecasting watershed runoff. The reason for this may be related to the fact that the GA has been employed to escape the local optimal and in the following, LMBP′ algorithm is used for finding the best local optimal. In this way the proposed model can find the appropriate weights of the network under the complex and chaotic conditions of the rainfall–runoff process.

We will choose the best fit model out of PELMNN and WANFIS [7] by hypothesis testing. To meet this purpose following hypothesis is proposed:

H_0: There is no difference between prediction accuracy of PELMNN and WANFIS
H1: There is a difference between the prediction accuracy of the two models.

Since the data used for prediction in both models are the same, we carry out paired t-test (two samples for mean) on prediction accuracy (relative error percentage) to test the hypothesis. The results of paired t-tests are shown in Table 5.

As it is shown, since P–value < 0.002 so H_0 is rejected in level of confidence $\alpha = 0.002$. The evidence indicates that the average prediction error of PELMNN is significantly lower than that of WANFIS. Then, PELMNN will be considered as the preferred model for constructing watershed's runoff response. Also we may conclude that data clustering has remarkably improved accuracy of forecasts.

Another key point in the rainfall–runoff modeling which makes sure that our model is most promising when comparing with different models is the capability of the proposed model in estimating peak values. Whereas the estimation of peak values is usually the most important part of the flood mitigation program, as final step of the modeling process, PELMNN model should be able to capture the peak values. Threshold of the top 5% of the data from the original runoff time series are considered contractually as peak values. The performances of the various models for this modeling were evaluated using Eq. (11) and are presented in Table 6. By comparing the results, it is found out that the capability of the PELMNN model for predicting extreme values is more than ANN, ANFIS, PENN, PEBPNN, WANN and WANFIS models. Also, the efficiency of PELMNN model (with clustering) is 0.96 (and without clustering is 0.94) as against 0.95 for WANFIS. There is an identical high capability of two models for prediction of peak flows. But, as mentioned above, PELMNN has promising results, especially in the low-flow context. Therefore, not only the proposed model is appropriate in monitoring peak values, but also it can be considered as a most powerful tool for streamflow forecasting which is necessary in the water resources systems management, where it is directly influenced by streamflow forecasting.

4. Conclusions

It can be realized from both theoretical and empirical findings that combining different methods is the effective and efficient way to improve prediction results. This paper presented a pre-processed evolutionary Levenberg–Marquardt neural networks (PELMNN) model for runoff prediction by combining genetic algorithms and feed forward neural networks along with employing data pre-processing and post-processing concepts as reinforcement equipments. In the pre-processing stage, input data were scaled by the means of data transformation technique and also

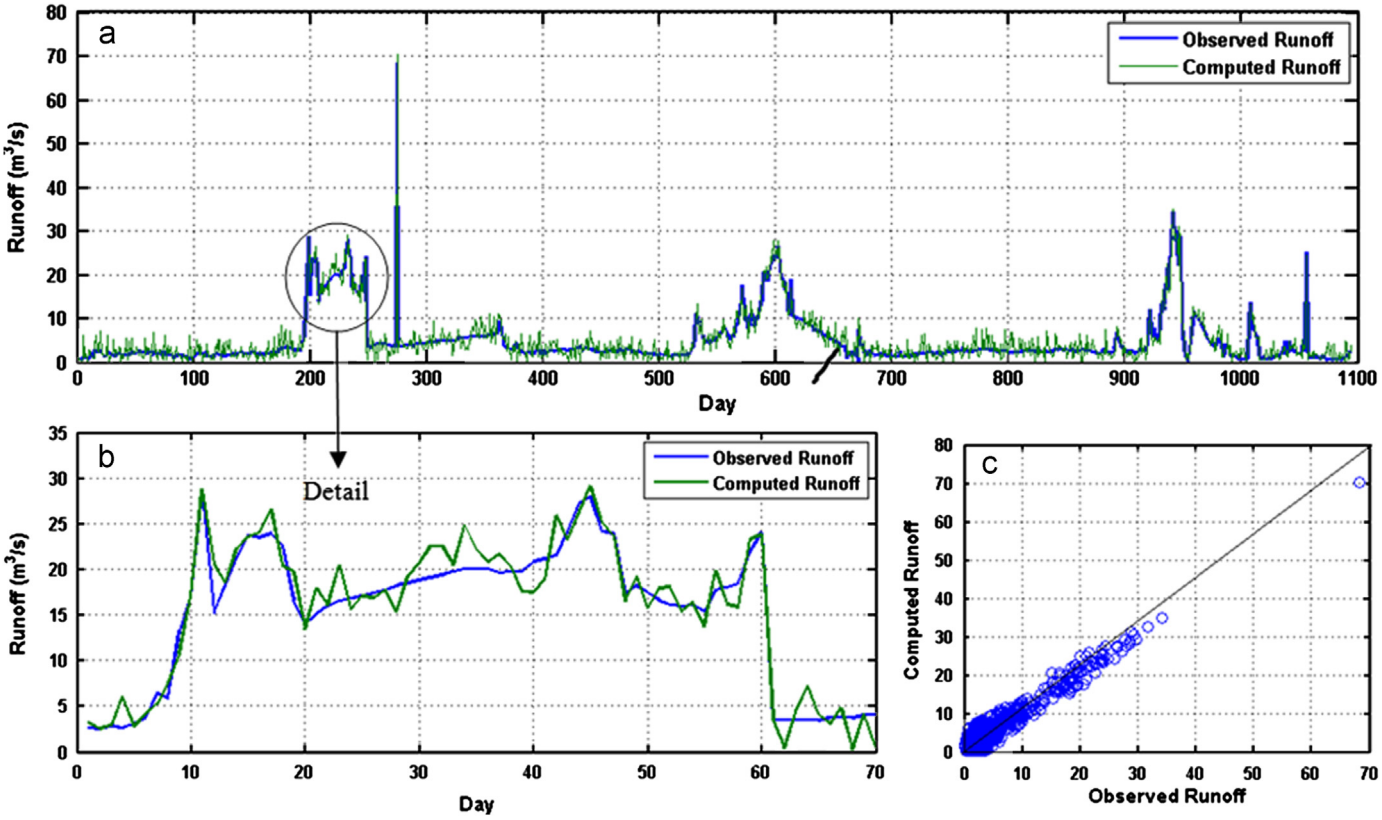

Fig. 6. The result of PELMNN model during the test period: (a) the ability of proposed model in capturing peak values, (b) detail and (c) scatter plot.

Table 4
Results of multi-step ahead forecasting for ANN and PELMNN models.

Models	R^2 (t−3)	RMSE (t−3)	R^2 (t−7)	RMSE (t−7)
ANN				
Calibration	0.76	0.041	0.64	0.048
Verification	0.67	0.045	0.60	0.051
PELMNN (without clustering)				
Calibration	0.90	0.026	0.84	0.036
Verification	0.88	0.029	0.81	0.038
PELMNN (with clustering)				
Calibration	0.92	0.020	0.88	0.029
Verification	0.89	0.028	0.85	0.035

Table 5
The results of paired t-test.

Mean of deviation	Std. deviation	t-Stat	P-value	df	Conclusion
−2.65	4.935	−3.442	0.0013	40	$\mu_{WANFIS} > \mu_{PELMNN}$

Table 6
The ability of different models in capturing peak values.

Models	Determination coefficient for peak values (R^2_{peak})
ANN	0.47
WANN[a]	0.78
ANFIS[a]	0.72
WANFIS[a]	0.95
PENN	0.63
PEBPNN	0.70
PELMNN (without clustering)	0.94
PELMNN (with clustering)	0.96

[a] The reference for ANFIS, WANN and WANFIS model is Nourani et al. [7].

stepwise regression was used for input selection to filter out the unrelated variables and keep only those variables, which have a significant role in predicting runoff at t+1 ($Q_{(t+1)}$) and using clustering method divides the data into sub-populations and reduces the complexity of the whole data space to something more homogeneous. In the next stage genetic algorithm was adopted as a global search method to evolve neural networks' initial weights for tuning with LM algorithm. Obtained weights are used as initial weights for the Levenberg–Marquardt BP algorithm to search locally, because LMBP is capable in local search. Finally in post-processed stage, the output data were returned to their original scale and the predicted values were generated. The advantages of PELMNN are fast training capability, good adaptation of non-linear problem and high degree of accuracy. Results showed that prediction accuracy of PELMNN model is better than ANN, PENN, PEBPNN, ANFIS, WANN and WANFIS models.

Acknowledgment

The authors would like to acknowledge the valuable help and assistance given by Mr. Esmaeil Hadavandi, Ph.D. candidate of Industrial Engineering in Amirkabir University of Technology (Tehran Poly technique).

References

[1] D.P. Solomatine, K.N. Dulal, Model trees as an alternative to neural networks in rainfall–runoff modeling, Hydrol. Sci. J. 48 (2003) 3399–3411.
[2] K.P. Sudheer, P.H. Gowda, I. Chaubey, T.A. Howell, Artificial neural network approach for mapping contrasting tillage practices, Remote Sensing 2 (2010) 579–590.
[3] P.C. Nayak, K.P. Sudheer, S.K. Jain, Rainfall–runoff modeling through hybrid intelligent system, Water Resour. Res. 43 (2007) W07415, http://dx.doi.org/10.1029/2006WR004930.
[4] C.W.J. Granger, Combining forecasts—twenty years later, J. Forecast. 8 (1989) 167–173.
[5] I. Ginzburg, D. Horn, Combined neural networks for time series analysis, Adv. Neural Inf. Process. Syst. 6 (1994) 224–231.
[6] B. Krishna, Y.R. Satyaji Rao, P.C. Nayak, Time series modeling of river flow using wavelet neural networks, J. Water Resour. Prot. 3 (2011) 50–59.
[7] V. Nourani, O. Kisi, K. Komasi, Two hybrid Artificial Intelligence approaches for modeling rainfall–runoff process, J. Hydrol. 402 (2011) 41–59.
[8] D. Jeong, Y.O. Kim, Rainfall–runoff models using artificial neural Networks for ensemble stream flow prediction, Hydrol. Processes 19 (2005) 3819–3835.
[9] O. Kisi, M.E. Karahan, Z. Sen, River suspended sediment modeling using fuzzy logic approach, Hydrol. Processes 20 (2006) 204351–204362.
[10] P.C. Chang, C.H. Liu, C.Y. Fan, Data clustering and fuzzy neural network for sales forecasting: a case study in printed circuit board industry, Knowl. Based Syst. 22 (2009) 344–355.
[11] A. Keles, M. Kolcak, A. Keles, The adaptive neuro-fuzzy model for forecasting the domestic debt, Knowl. Based Syst. 21 (2008) 951–957.
[12] L. Yang, C. Dawson, M. Brown, M. Gell, Neural network and GA approaches for dwelling fire occurrence prediction, Knowl. Based Syst. 19 (2006) 213–219.
[13] N. O'Connor, M.G. Madden, A neural network approach to predicting stock exchange movements using external factors, Knowl. Based Syst. 19 (2006) 371–378.
[14] I. Wilson, S. Paris, J. Ware, D. Jenkins, Residential property price time series forecasting with neural networks, Knowl. Based Syst. 15 (2002) 335–341.
[15] E. Hadavandi, H. Shavandi, A. Ghanbari, An improved sales forecasting approach by the integration of genetic fuzzy systems and data clustering: case study of printed circuit board, Expert Syst. Appl. 38 (2011) 9392–9399.
[16] E. Hadavandi, H. Shavandi, A. Ghanbari, S. Abbasian, Developing a hybrid artificial intelligence model for outpatient visit forecasting in hospitals, Appl. Soft Comput. 12 (2) (2012) 700–711.
[17] W. Shen, X. Guo, C. Wu, D. Wu, Forecasting stock indices using radial basis function neural networks optimized by artificial fish swarm algorithm, Knowl. Based Syst. 24 (2011) 378–385.
[18] E. Hadavandi, A. Ghanbari, K. Shahanaghi, S. Abbasian, Tourist arrival forecasting by evolutionary fuzzy systems, Tourism Manage. 32 (2011) 203–1196.
[19] Y. El Hamzaoui, J.A. Hernandez, A.G. Roman, J.A. Ramirez, ANN and ANFIS models for COP prediction of a water purification process integrated heat to a transformer with energy recycling, Chem. Product Process Model. 7 (2012) 1–25.
[20] Y. El Hamzaoui, J.A. Hernandez, S. Silva-Martinez, A. Bassam, A. Alvarez, C. Lizama-Bahena, Optimal performance of COD removal during aqueous treatment of alazine and gesaprim commercial herbicides by direct and inverse neural network, Desalination 277 (2011) 325–337.
[21] M.R. Amiryousefi, M. Mohebbi, F. Khodaiyan, S. Asadi, An empowered adaptive neuro-fuzzy inference system using self-organizing map clustering to predict mass transfer kinetics in deep-fat frying of ostrich meat plates, Comput. Electron. Agric. 76 (1) (2011) 89–95.
[22] A. Elshorbagy, G. Corzo, S. Srinivasulu, D.P. Solomatine, Experimental investigation of the predictive capabilities of data driven modeling techniques in hydrology—Part 2: application, Hydrol. Earth Syst. Sci. 14 (1943–1961).
[23] Y. El Hamzaoui, J.A. Hernandez, A.G.Ramírez Roman, Search for optimum operating conditions for a water purification process integrated to a heat transformer with energy recycling using artificial neural network inverse solved by genetic and particle swarm algorithms, Chem. Product Process Model. 7 (2012) 1–25.
[24] S. Ahmad, P.S. Simonovic, An artificial neural network model for generating hydrograph from hydro-meteorological parameters, J. Hydrol. 315 (2005) 236–251.
[25] P. Nilsson, C.B. Uvo, R. Berndtsson, Monthly runoff simulation: comparing and combining conceptual and neural networks models, J. Hydrol. 321 (2006) 344–363.
[26] M.A. Antar, I. Elassiouti, M.N. Alam, Rainfall–runoff modeling using artificial neural networks technique: a Blue Nile catchment case study, Hydrol. Processes 20 (2006) 51201–51216.
[27] A. Jain, K.P. Sudheer, S. Srinivasulu, Identification of physical processes inherent in artificial neural network rainfall–runoff models, Hydrol. Processes 18 (2004) 571–581.
[28] A.R. Senthil Kumar, K.P. Sudheer, S.K. Jain, P.K. Agarwal, Rainfall–runoff modeling using artificial neural network: comparison of networks types, Hydrol. Processes 19 (2004) 61277–61291.
[29] P. Leahy, G. Kiely, G. Corcoran, Structural optimisation and input selection of an artificial neural network for river level prediction, J. Hydrol. 355 (192–1) (2008) 201.
[30] C.W. Dawson, L.M. See, R. Abrahart, A.J. Heppenstall, Symbiotic adaptive neuro-evolution applied to rainfall–runoff modeling in northern England, Neural Netw. 19 (2006) 236–247.
[31] Y. Wang, H. Wang, X. Lei, Y. Jiang, X. Song, Flood simulation using parallel genetic algorithm integrated wavelet neural networks, Neurocomputing 74 (2011) 172734–172744.
[32] K.P. Sudheer, P.C. Nayak, K.S. Ramasastri, Improving peak flow estimates in artificial neural network river flow models, Hydrol. Processes 17 (2003) 3677–3686.
[33] K. Hornik, Multilayer feed-forward networks are universal approximators, Neural Netw. 2 (1988) 5359–5366.
[34] ASCE Task Committee on Application of Artificial Neural Networks in Hydrology, Artificial Neural Networks in hydrology 1: hydrology application, J. Hydrol. Eng. 5 (2000) 2124–2137.
[35] I.A. Basheer, M. Hajmeer, Artificial neural networks: fundamentals, computing, design, and applications, J. Microbiol. Methods 43 (2000) 3–31.
[36] M. Hagan, M. Menhaj, Training feedforward networks with the marquardt algorithm, IEEE TNN 5 (1994) 6989–6993.
[37] X. Yao, Evolving artificial neural networks, Proc. IEEE 87 (1999).
[38] P. Bartlett, T. Downs, Training a Neural Network With a Genetic Algorithm, Dept. of Electrical Engineering, University of Queensland, 1990, Technical Report.
[39] S. Asadi, E. Hadavandi, F. Mehmanpazir, M.M. Nakhostin, Hybridization of evolutionary Levenberg–Marquardt neural networks and data pre-processing for stock market prediction, Knowl. Based Syst. 35 (2012) 245–258.
[40] A. Fiszelew, P. Britos, A. Ochoa, H. Merlino, Finding optimal neural network architecture using genetic algorithms, Res. Comput. Sci. 27 (2007) 15–24.
[41] R. Sexton, S. McMurtrey, D. Cleavenger, Knowledge discovery using a neural network simultaneous optimization algorithm on a real world, Eur. J. Oper. Res. 168 (2006) 1009–1018.
[42] R. Sexton, S. McMurtrey, J.Smith Michalopoulos, Employee turnover: a neural network solution, Comput. Oper. Res. 32 (2005) 2635–2651.
[43] R. Sexton, R. Sriram, H. Etheridge, Improving decision effectiveness of artificial neural networks: a modified genetic algorithm, Decis. Sci. 34 (2003) 421–442.
[44] R. Sexton, R. Dorsey, J. Johnson, Toward global optimization of neural networks: a comparison of the genetic algorithm, Decis. Support Syst. 22 (1998) 171–185.
[45] R. Dorsey, W. Mayer, Genetic algorithms for estimation problems with multiple optima, nondifferentiability, and other irregular features, J. Bus. Econ. Stat. 1 (1995) 53–66.
[46] J.D. Knowles, D.W. Corne, Evolving neural networks for cancer radiotherapy, The Practical Handbook of Genetic Algorithms Applications, Chapman & Hall/CRC, London, 2001.
[47] P.C. Chang, Y.W. Wang, C.Y. Tsai, Evolving neural network for printed circuit board sales forecasting, Expert Syst. Appl. 29 (2005) 83–92.
[48] R.J. Kuo, J.A. Chen, A decision support system for order selection in electronic commerce based on fuzzy neural network supported by real-coded genetic algorithm, Expert Syst. Appl. 26 (2004) 141–154.
[49] R.S. Sexton, J.N. Gupta, Comparative evaluation of genetic algorithm and backpropagation for training neural networks, Inf. Sci. 129 (2000) 45–59.
[50] Y.H. Chen, F.J. Chang, Evolutionary artificial neural networks for hydrological systems forecasting, J. Hydrol. 367 (2009) 125–137.
[51] A.P. Piotrowski, J.J. Napiorkowski, Optimizing neural networks for river flow forecasting—evolutionary computation methods versus Levenberg–Marquardt approach, J. Hydrol. 407 (2011) 12–27.
[52] C.T. Cheng, C.P. Ou, K.W. Chau, Combining a fuzzy optimal model with a genetic algorithm to solve multi-objective rainfall–runoff model calibration, J. Hydrol. 268 (2002) 72–86.
[53] A. Jain, S. Srinivasulu, Development of effective and efficient rainfall–runoff models using integration of deterministic, real-coded genetic algorithms, and artificial neural network techniques, Water Resour. 40 (4) (2004) W04302, http://dx.doi.org/10.1029/2003WR00.
[54] A. Sedki, D. Ouazar, E. El Mazoudi, Evolving neural network using real coded genetic algorithm for daily rainfall–runoff forecasting, Expert Syst. Appl. 36 (2009) 4523–4527.
[55] J.V. Hansen, J.B. McDonald, R.D. Nelson, Time series prediction with genetic algorithm designed neural networks: an experimental comparison with modern statistical models, Comput. Intell. 15 (1999) 3171–3184.
[56] P.S. Heckerling, B.S. Gerber, T.G. Tape, R.S. Wigton, Use of genetic algorithm for neural network to predict community-acquired pneumonia, Artif. Intell. Med. 30 (2004) 71–84.
[57] G.S. Atsalakis, K.P. Valavanis, Surveying stock market forecasting techniques—Part II: soft computing methods, Expert Syst. Appl. 36 (2009) 5932–5941.
[58] G. Auda, M. Kamel, Modular neural networks: a survey, Int. J. Neural Syst. 9 (1999) 129–151.
[59] Amanda J.C. Sharkey, Combining Artificial Neural Networks, Springer, 1999.

[60] M.H. FazelZarandi, Esmaeil Hadavandi, I.B. Turksen, A hybrid fuzzy intelligent agent-based system for stock price prediction, Int. J. Intell. Syst. (2012) 947–969.
[61] E.F. Fama, Efficient capital markets: a review of theory and empirical work, J. Finance 25 (1970) 383–417.
[62] S. Asadi, A. Tavakoli, S.R. Hejazi, A new hybrid for improvement of auto-regressive integrated moving average models applying particle swarm optimization, Expert Syst. Appl. 39 (2012) 5332–5337.
[63] A. Azadeh, M. Saberi, S. Ghaderi, A. Gitiforouz, V. Ebrahimipour, Improved estimation of electricity demand function by integration of fuzzy system and data mining approach, Energy Convers. Manage. 49 (2008) 2165–2177.
[64] M.E. ElAlami, A filter model for feature subset selection based on genetic algorithm, Knowl. Based Syst. 22 (2009) 356–362.
[65] Y.X. Zhang, Artificial neural networks based on principal component analysis input selection for clinical pattern recognition analysis, Talanta 73 (2007) 68–75.
[66] M.R. Anderberg, Cluster Analysis for Applications, Academic Press, New York, 1973.
[67] J.Valente de Oliveira, W. Pedrycz, Advances in Fuzzy Clustering and its Applications, John Wiley & Sons, New York, 2007.
[68] C.T. Yiakopoulos, K.C. Gryllias, I.A. Antoniadis, Rolling element bearing fault detection in industrial environments based on a K-means clustering approach, Expert Syst. Appl. 38 (3) (2011) 2888–2911.
[69] Y.P. Pandit, Classification of Indian power coals using K-means clustering and Self Organizing Map neural network, Fuel 90 (2010) 339–347.
[70] T. Niknam, B. Amiri, An efficient hybrid approach based on PSO, ACO and k-means for cluster analysis, Appl. Soft Comput. 10 (2010) 183–197.
[71] D.E. Rumelhart, J.L. McCleland, Parallel Distributed Processing: Explorations in the Micro Structure of Cognition, Foundations, MIT Press, Cambridge, MA, 1986.
[72] A.E.B. Ruano, D.I. Jones, P.J. Fleming, A New Formulation of the Learning Problem for a Neural Network Controller. in: Proceedings of the 30th IEEE Conference on Decision and Control, Brighton, UK, 1991.
[73] M. Hagan, M. Menhaj, Training feedforward networks with the marquardt algorithm, IEEE Transactions on Neural Networks 5 (1994) 989–993.
[74] H.J. Bremermann, The evolution of intelligence. The nervous system as a model of its environment, Department ofMathematics, University of Washington, Seattle, WA, 1958, Technical Report no. 1.
[75] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, 1975.
[76] L. Chambers, Practical Handbook of Genetic Algorithms. New Frontiers, vol. II, CRC Press, New York448.
[77] D. Janson, J. Frenzel, Training product unit neural networks with genetic algorithms, IEEE Expert 8 (1993) 26–33.
[78] G. Miller, P. Todd, S. Hegde, Designing neural networks using genetic algorithms, in: Proceedings of the third International Conference on Genetic Algorithms, 1989.
[79] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, The Netherlands, 1989.
[80] C.L. Wu, K.W. Chau, Rainfall–runoff modelling using artificial neural network coupled with singular spectrum analysis, J. Hydrol. 399 (2011) 394–409.
[81] D.R. Legates, G.J. McCabe, Evaluating the use of goodness of fit measures in hydrologic and hydroclimatic model validation, Water Resour. Res. 35 (1999) 233–241.
[82] S. Barbetta, T. Moramarco, M. Franchini, F. Melone, L. Brocca, V.P. Singh, Case study: improving real-time stage forecasting Muskingum model by incorporating the Rating Curve Model, J. Hydrol. Eng. 16 (2011) 540–557.
[83] A. Azadeh, S.M. Asadzadeh, A. Ghanbari, An adaptivenetwork-based fuzzy inference system for short-term natural gas demand estimation: uncertainand complex environments, Energy Policy 38 (2010) 1529–1536.
[84] T.J. Burkholder, R.L. Lieber, Stepwise regression is an alternative to splines for fitting noisy data, J. Biomech. 29 (1996) 235–238.
[85] A. Eiben, R. Hinterding, Z. Michalewicz, Parameter control in evolutionary algorithms, IEEE Trans. Evol. Comput. 3 (1999) 124–141.
[86] J. Shahrabi, E. Hadavandi, S. Asadi, Developing a hybrid intelligent model for forecasting problems: case study of tourism demand time series, Knowl. Based Syst. 43 (2013) 112–122.
[87] M.R. Amiryousefi, M. Mohebbi, F. Khodaiyan, S. Asadi, An empowered adaptive neuro-fuzzy inference system using self-organizing map clustering to predict mass transfer kinetics in deep-fat frying of ostrich meat plates, Comput. Electron. Agric. 76 (2011) 89–95.
[88] P. Soltani, J. Shahrabi, S. Asadi, E. Hadavandi, M.S. Johari, A study on siro, solo, compact, and conventional ring-spun yarns. Part III: modeling fiber migration using modular adaptive neuro-fuzzy inference system, J. Text. Inst. (2013) 1–11.
[89] S.M.R. Kazemi, E. Hadavandi, F. Mehmanpazir, M.M. Nakhostin, A hybrid intelligent approach for modeling brand choice and constructing a market response simulator, Knowl. Based Syst. 40 (2012) 101–110.

Shahrokh Asadi received the master degree in Industrial Engineering from Isfahan University of Technology (IUT). He is now a Ph.D. candidate in Amirkabir University of Technology (Tehran Polytechnic) in Tehran. He is the author and co-author of about 20 scientific papers in international journals and international conferences. His research is mainly in the field of Data Mining (DM), Business Intelligence (BI), Customer Relationship Management (CRM) and Strategic Management (SM).

Jamal shahrabi received the Ph.D. degree in Industrial Engineering from Dalhousie University; Halifax, Canada in 2004. He is a full time faculty member of Amirkabir University of Technology (Tehran Polytechnic). He is the member of two main Canadian research organization and group; Geoide and MARIN. He is the author or co-author of lots of scientific papers in international journals and international conferences. He is the author of 11 books in the field of Data Mining in Persian language those are the main text books of Data Mining courses in most universities. His research is mainly in the field of Hybrid Soft Computing models and Data Mining.

Peyman Abbaszadeh holds Master of Water Engineering from the Amirkabir University of Technology (Tehran Polytechnic) in 2013, and B.Sc. in Civil Engineering from the University of Tabriz in 2011. His core research interests and experiences include Hydro-meteorological forecasting (e.g., water demand, flood, groundwater level, drought, etc.) using wavelet transforms and artificial intelligence methods. He is the author or co-author of several scientific papers in international journals and international conferences.

Shabnam Tabanmehr holds B.Sc. of Industrial and Systems Engineering from Amirkabir University of Technology (Tehran Polytechnic). Her core research interests and experiences include Business Intelligence (BI), Data Mining (DM), Customer Relationship Management (CRM), Strategic Management (SM) and Entrepreneurship.