Hybridization of evolutionary Levenberg–Marquardt neural networks and data pre-processing for stock market prediction

Shahrokh Asadi a, Esmaiel Hadavandi a,*, Farhad Mehmanpazir b, Mohammad Masoud Nakhhostin b

a Department of Industrial Engineering, Amirkabir University of Technology (Polytechnic of Tehran), P.O. Box 15875-4413, Tehran, Iran
b School of Industrial Engineering, Islamic Azad University (South Branch of Tehran), P.O. Box 11365-9466, Tehran, Iran

A R T I C L E   I N F O
Article history:
Received 13 October 2010
Received in revised form 8 May 2012
Accepted 9 May 2012
Available online 18 May 2012

Keywords:
Stock price prediction
Genetic algorithms
Evolutionary Neural Networks
Levenberg–Marquardt algorithm
Data pre-processing

A B S T R A C T
Artificial Intelligence models (AI) which computerize human reasoning has found a challenging test bed for various paradigms in many areas including financial time series prediction. Extensive researches have resulted in numerous financial applications using AI models. Since stock investment is a major investment activity, Lack of accurate information and comprehensive knowledge would result in some certain loss of investment. Hence, stock market prediction has always been a subject of interest for most investors and professional analysts. Stock market prediction is a challenging problem because uncertainties are always involved in the market movements. This paper proposes a hybrid intelligent model for stock exchange index 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 and input variables selection for improving the accuracy of the model. The capability of the proposed method is tested by applying it for predicting some stock exchange indices used in the literature. The results show that the proposed approach is able to cope with the fluctuations of stock market values and also yields good prediction accuracy. So it can be used to model complex relationships between inputs and outputs or to find data patterns while performing financial prediction.

© 2012 Elsevier B.V. All rights reserved.

1. Introduction and literature review

Prediction is the process of making projection about future performance based on existing historical data. Accurate prediction aids in decision-making and planning for the future. Prediction empowers people to modify current variables to make a prediction of the future to result in favorable scenario. Selection and implementation of a proper prediction methodology has always been an important planning and control issue for firms and agencies. Financial stability of an organization depends on the accuracy of the predictions been made, since such information will most likely be used to make key decisions in important areas such as human resources, purchasing, marketing, advertising and capital financing. The complexity of the prediction process used for predicting a variable depends on various factors. Most importantly, the historical pattern of variable changes and the underlying input factors which affect a variable may increase the complexity of the prediction process.

There are two hypotheses to be considered while predicting stock price. First one is, The Efficient Market Hypothesis (EMH) stating that at any time, the price of a stock fully captures all known information about the stock. Since all known information is used optimally by market participants, price variations are random, as new information occurs randomly. Thus, stock prices perform a “random walk”, meaning that all future prices do not follow any trends or patterns and are random departure from the previous prices and it is not possible for an investor to predict the market. This hypothesis interprets fluctuations as the result of delayed or incomplete information that influence stock market prices. External phenomena affect subsequent stock market prices. Accurate predictions are difficult because it is not possible to model, quantify, or even know a priori such external phenomena [54].

There has been a lot of debate about the validity of the EMH and random walk (RW) theory. However, with the advent of computational and intelligent finance, and behavioral finance, economists have tried to establish another hypothesis which may be collectively called as the Inefficient Market Hypothesis (IMH). IMH states that financial markets are at least not always efficient, the market...
is not always in a RW, and inefficiencies exists [47]. Many of these studies used intelligent systems such as neural networks to justify their claims. The fact that many market participants can consistently beat the market is an indication that the EMH may not be true in practice [22].

Predicting a change in market prices and making correct decisions based on that, is one of the most important requirements for anyone who has something to do with economical environments. For many years, how to make a prediction of stock market has been a prevalent research topic. Financial gain might be considered as the most basic issue in stock market prediction. If a system is able to consistently pick winners and losers in the dynamic market place, then it will be able to provide the owner of the system with more wealth. Thus, many individuals including researchers, investment professionals, and average investors are continually looking for this superior system which will yield them high returns.

There are several different approaches to time series modeling. Traditional statistical models including moving average, exponential smoothing, and ARIMA are linear in that predictions of the future values. These models use piecewise linear function as basic element of prediction model [24,26,37,7,20,49,46]. The functional form for the problem has to be specified by the user. It could take a lot of time to experiment with different possible function relations and algorithms to obtain proper models. Moreover, many researchers claim that the stock market is a chaos system. Chaos is a non-linear deterministic system which only appears random because of its irregular fluctuations. So in finance theory, non-linear models are very important. For example, almost all of real business cycle models are highly non-linear. Superiority of non-linear models in finance is not supposed to be conflicting with the use of linear models by a practitioner. Linear models can be considered as the reasonable approximations of the non-linear phenomenon of interest and non-linear models can be applied in prediction. It is not surprising that non-linear models, from regime-switching models to neural networks and genetic algorithms are receiving a great deal of attention in the literature [16,48].

The RW model says that not only all historical information are summarized in the current value but also those increments—positive or negative—that are uncorrelated and balanced with an expected value equal to zero. In other words, the positive fluctuations are as many as negatives in long term. The nature of this model makes it difficult to perform accurate predictions. Hence, prediction of the stock market seems to be a daunting task. Mainly, since stock time series have a close to random-walk behavior due to the fact that stock markets have numerous underlying factors which most of them are currently not fully understood, therefore a non-linear model can be beneficial. A large set of interacting input series are often required to explain a specific stock. If we are able to predict stock market time series more accurately, we can allocate society resources to the right place so that national resources will not be wasted. The Intelligent Systems are capable of learning such non-linear chaotic systems because they make very few assumptions about the functional form of the underlying dynamic dependencies and their initial conditions. This may eventually put a question on the traditional financial theory of efficient market. Intelligent Systems are those which learn from their past experiences and use this knowledge in current and future decision making [22].

Artificial Intelligence (AI) is one of the intelligent systems that computerizes human reasoning has found a challenging test bed for various paradigms in many areas including financial time series prediction. Extensive research has resulted in numerous prediction applications using artificial neural networks (ANN), fuzzy logic and genetic algorithms (GA) and other techniques [12,39,58,44,56,32,33,53,34,25]. Most progress to date in AI has been made in the areas of problem solving; concepts and methods for building programs that reason about problems rather than calculate a solution.

A number of studies have compared the capability of AI techniques over conventional techniques such as ARIMA, Regression etc. in prediction problems, and they have found that AI-based systems result in more accurate outputs than conventional approaches such as ARIMA and Regression [30,31,28].

Nowadays, increasing number of efforts have been focused on AI models for stock market prediction, while using AI models or a combining of several models has become a common practice to improve prediction accuracy. Hence, the literature on this topic has expanded dramatically [1].

Havadandi et al. [30] presented an integrated approach based on genetic fuzzy systems (GFS) and artificial neural networks for constructing a stock price prediction expert system. They used stepwise regression analysis to determine factors which have most impact on stock prices, then divided the raw data into k clusters using self-organizing map (SOM) neural networks. Finally, all clusters formed into independent GFS models with the ability of rule base extraction and data base tuning. Results showed that their approach outperforms other methods such as ANN and ARIMA.

Chang and Liu [11] used a Takagi–Sugeno–Kang (TSK) type Fuzzy Rule Based System (FRBS) for stock price prediction. They used simulated annealing (SA) for training the best parameters of fuzzy systems. They found that the forecasted results from TSK fuzzy rule based model were much better than those of back propagation network (BPNN) or multiple regressions.

Esfahanipour and Aghamiri [21] used Neuro-Fuzzy Inference System adopted on a TSK type Fuzzy Rule Based System for stock price prediction. The TSK fuzzy model applies the technical index as the input variables and uses Fuzzy C-Mean clustering for identifying number of rules. They tested the proposed model on Tehran Stock Exchange Indices (TPEIX) and Taiwan stock exchange index (TSE) data (the same data set that used by Chang and Liu [11]). Results showed that the proposed model can effectively improve the prediction performance and outperforms other models.

Shen et al. [53] introduced the artificial fish swarm algorithm to optimize RBF. In this paper, to improve prediction efficiency, a K-means clustering algorithm is optimized by artificial fish swarm algorithm during the learning process of RBF. To confirm the usefulness of their algorithm, they compared the prediction results of RBF optimized by three methods: artificial fish swarm algorithm, genetic algorithms, and particle swarm optimization as well as prediction results of ARIMA, BP and support vector machine. Results demonstrated that the proposed model performs better than other models.

De and Araujo [18] introduced a class of hybrid morphological perceptrons, called dilation–erosion perceptron to overcome the RW dilemma in the time series forecasting problem. A gradient steepest descent method was presented to design the proposed dilation–erosion perceptron, using the back propagation (BP) algorithm and a systematic approach to overcome the problem of non-differentiability of morphological operators.

Cho [15] used a new architecture for financial information systems. The developed prototype was entitled as the Multi-level and Interactive Stock Market Investment System. It is specially designed for investors to build their financial models to forecast stock price and index.

Chang Chien and Chen [14] focused on stock trading data with many numerical technical indicators, and the classification problem is finding sell and buy signals from the technical indicators. This study proposed a GA-based algorithm to build an associative classifier that can discover trading rules from these numerical indicators. The experiment results revealed that the proposed approach is an effective classification technique with high prediction accuracy. It is also highly competitive when compared with the data distribution method.
ANNs is one of the strongest AI models which can learn the complex nature of the relationship between inputs and outputs. The technique is rooted in and inspired by the biological network of neurons in the human brain that learns from external experience, handles imprecise information, stores the essential characteristics of the external input, and generalizes previous experience.

Atsalakis and Valavanis [2] pointed out about 60% of the surveyed articles in recent years, used feed forward neural networks (FFNN) and recurrent networks for stock market prediction.

When developing a feed forward neural network model for prediction purposes, specifying its architecture in terms of the number of input, hidden, and output neurons and weight training 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 over all training data. Iteratively adjusting connection weights. Most training algorithms, such as back propagation and conjugate gradient are based on gradient descent [6].

Among the literatures regarding using the ANNs as the prediction tool, most of them focus on back-propagation Neural Network. 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 introduced to the neural net research because it provided methods to accelerate the training and convergence of the algorithm. It utilizes the BP procedures in which derivatives are processed from the last layer of the network to the first [35].

Yet, there are two shortcomings of LMBP: firstly, despite that LMBP has shown to be successful in some areas, the algorithm often gets trapped in a local minimum of the error function and is incapable of finding a global minimum if the error function is multimodal and/or non-differentiable [59], secondly, it has been found that it does not perform well with networks with more than two or three hidden layers [5]. The mentioned issues beside other problems have guided the researches toward employing evolutionary techniques to find the best set of network weights.

Evolutionary techniques have several obvious advantages over BP: genetic algorithms and other evolutionary approaches are able to find global minima in complex, multimodal spaces, they do not require a differentiable error function and they are more flexible, allowing the fitness evaluation to be changed to take into account extra factors that are not easy to incorporate in the BP algorithm [41]. Many researchers have used GA for learning of neural networks and they've found that GAs have better performance compared to BPs [13,43,52].

In spite of the good performance of Evolutionary Neural Networks (ENN), unfortunately a few researches can be found which verify applying of ENNs for dealing with stock market prediction problems. Wittkemper and Steiner [57] discussed different methods for predicting a stock's systematic risk. They showed that the most precise forecasts are given by neural networks, whose topology has been optimized by a genetic algorithm. Versace et al. [55] used a genetic algorithm to find the best mixture of neural networks, the topology of neural networks, and to determine the features set. They found the application of genetic algorithm on neural network architectures shows promise for prediction of stock market time series.

Due to nature of LMBP to converge locally, it can be demonstrated that solutions are highly dependent upon the initial random draw of weights. If these initial weights are located on a local grade, which is probable, the BP algorithm will likely become trapped in a local solution that may or may not be the global solution. This local convergence could present serious problems when using NNs for real-world applications. So using global search techniques is needed for Network Training. Considering the points mentioned, in this paper the powerful combination of positive aspects of genetic algorithms and LMBP algorithm is presented to predict stock market indices. 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 the beginning, genetic algorithm is used with abroad search to find weights of artificial neural network. After searching space becomes 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.

Data quality is a key issue in prediction concepts. To increase the accuracy of the prediction, we may perform data pre-processing techniques such as input selection and data transformation. Atsalakis and Valavanis [2] pointed out that input data pre-processing may impact prediction performance. The process of choosing indicators as inputs through sensitivity analysis may help eliminate redundant inputs. In many cases input data has a large range of values reducing effectiveness of training procedures. One solution to this impediment is applying data transformation techniques such as data scaling.

This paper presents a hybrid intelligent model for stock exchange index prediction. The proposed model which is 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 seven stock exchange indices used by Chang and Liu [11]. Esfahanipour and Aghamiri [21], Ferreira et al. [22] as a case study.

2. Methodology
A time series is a time dependent sequence of points, generally equidistant, such as: \( y_t = \{ y_t \in \mathbb{R}^t = 1, 2, 3, \ldots, M \} \), 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 [22]. A time series is considered as nonlinear function of several input variable as \( y_t = f(x_1, x_2, \ldots, x_n) \), where \( f \) is a nonlinear function determined by the neural network, \( x_{\text{norm}} \) is normalized value of \( x_{\text{data}} \) (input variable) and \( n \) is integer.

This paper presents four-stage architecture to develop a hybrid intelligent model for stock exchange index time series prediction. The first stage is data pre-processing. In this stage, we apply a data transformation technique to scale data then we employ stepwise regression analysis (SRA) for variable selection to choose the key variables that are to be considered in the model. 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 [4]. 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 study, for example; Azadeh et al. [3], Oliveira [45], and Niska et al. [29] applied this method as a part of their approach to estimate time series functions using heuristic approaches and they obtained quite promising results.

There are different normalization algorithms, such as Min–Max normalization, Z-score normalization and sigmoid normalization. In this paper we use Min–Max normalization which is a common approach in this field. The Min–Max normalization scales the numbers in a data set to improve the accuracy of the subsequent
numic computations. If $X_{old}, X_{max}, X_{min}$ are the original, maximum and minimum values of the raw data, respectively and $X'_{max}, X'_{min}$ are the maximum and minimum of the normalized data, respectively, then the normalization of $X_{old}$ called $X'_{raw}$, can be obtained by the following transformation function:

$$X'_{raw} = \frac{(X_{old} - X_{min})}{X_{max} - X_{min}}(X'_{max} - X'_{min}) + X'_{min} \quad (1)$$

2.1.2. Variable selection by stepwise regression analysis

Variable selection is the problem of selecting input variables that are most predictive of a given output. It identifies a small subset of variables so that the prediction model constructed with the selected variables minimizes the error and the selected variables also better explain the data. The variable selection gives a better generalization error. Typically, only a small number of variables of $X$ give sufficient information for the prediction. The variable selection problem finds the small subset of variables that are relevant to the target concept. A small subset of relevant variables gives more discriminating power than using more variables. This is counter-intuitive since more variables give more information and therefore should give more discriminating power. But if a variable is irrelevant, then that variable does not affect the target concept. If a variable is redundant, then that variable does not add anything new to the target concept [42].

In recent years some researcher have used SRA for input variable selection in the field of stock market prediction and they’ve obtained very promising results [21,30]. So, in this paper we adopt stepwise regression to analyze and select variables, and as a result improve the prediction accuracy of the system. Stepwise regression method determines a set of independent factors that most closely determine the dependent variable. This task is carried out using the repetition of a variable selection. At each of these steps, a single variable is either entered or removed from the model. For each step, simple regression is performed using the previously included independent variables and one of the excluded variables.

To illustrate the procedure [40], assume that we have $K$ candidate variables $x_1, x_2, \ldots, x_K$ and a single response variable $y$. In classification the candidate variables correspond to the polynomial-expanded elements of the feature vectors and the response variable corresponds to the class label. Note that with the intercept term $\beta_0$ we end up with $K + 1$ variables. In the procedure the polynomial weights (or the regression model) are iteratively found by adding or removing variables at each step. The procedure starts by building a one variable regression model using the variable that has the highest correlation with the response variable $y$. This variable will also generate the largest partial $F$-statistic. In the second step, the remaining $K - 1$ variables are examined. The variable that generates the maximum partial $F$-statistic is added to the model provided that the partial $F$-statistic is larger than the value of the $F$-random variable for adding a variable to the model, such an F-random variable is referred to as $f_{fin}$. Formally the partial $F$-statistic for the second variable is computed by:

$$f_k = \frac{SS_k(\beta_2 | \beta_1, \beta_0)}{MSE(s(x_1, x_2))} \quad (2)$$

where $MSE(x_1, x_2)$ denotes the mean square error for the model containing both $x_1$ and $x_2$. $SS_k(\beta_k | \beta_1, \beta_0)$ is the regression sum of squares due to $\beta_k$ given that $\beta_1, \beta_0$ are already in the model. In general the partial $F$-statistic for variable $j$ is computed by:

$$f_j = \frac{SS_j(\beta_0, \beta_1, \ldots, \beta_{j-1}, \beta_{j+1}, \ldots, \beta_K)}{MSE} \quad (3)$$

If variable $x_3$ is added to the model then the procedure determines whether the variable $x_1$ should be removed. This is determined by computing the $F$-statistic.

$$f_1 = \frac{SS_1(\beta_2, \beta_1, \beta_0)}{MSE(s(x_2, x_3))} \quad (4)$$

If $f_1$ is less than the value of the $F$-random variable for removing variables from the model, such an $F$-random variable is referred to as $f_{old}$. The procedure examines the remaining variables and stops when no other variable can be added or removed from the model. It is also worth mentioning that one cannot arrive to the conclusion that all of the regressors that are important for predicting the response variable have been retained in the stepwise procedure. This is so because such a procedure retains regressors based on the use of sample estimates of the true model weights. It is understood that there is a probability of making errors in retaining regressors.

2.2. 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 consists of an inter-connection of a number of neurons. There are many varieties of connections under study; however, here we will discuss only one type of network which is called multilayer perceptron (MLP). In this network the data flows forward to the output continuously without any feedback. We have used a typical four-layer feed forward model for prediction stock exchange. The input nodes are the technical indices, while the output provides the prediction for the stock exchange index. Hidden nodes with appropriate nonlinear transfer functions are used to process the information received by the input nodes. The model can be written as Eq. (5).

$$y_i = \gamma_0 + g\left(\sum_{k=1}^m \beta_k f\left(\sum_{j=1}^n \gamma_{jk} x_j + \beta_{0j}\right) + \beta_{00}\right) + \epsilon_i \quad (5)$$

where $m$ is the number of input nodes, $n$ is the number of hidden nodes in 1st hidden layer and $s$ is the number of hidden nodes in 2st hidden layer, $f$ is a sigmoid transfer function such as the logistic, $f(x) = \frac{1}{1 + e^{-x}}$, and $g$ is hyperbolic tangent (tanh), $g(x) = \frac{e^{xp(x)} - e^{-xp(x)}}{e^{xp(x)} + e^{-xp(x)}}$. $\gamma_{jk} = 0.1, \ldots, s$, $\beta_{0j} = 0, \ldots, n$ is a vector of weights from the 2nd hidden layer to output nodes and 1st hidden layer to 2nd hidden layer nodes respectively. $\beta_{00}, i = 1, 2, \ldots, m; j = 0, 1, \ldots, n$ are weights from the input to 1st hidden layer nodes, $\gamma_0, \beta_{00}, \beta_{0j}$ are weights of arcs leading from the bias terms that have values always equal to 1.

The attraction of MLP has been explained by the ability of the network to learn complex relationships between input and output patterns, which would be difficult to model with conventional algorithmic methods. The disadvantage of ANN is that because the network finds out how to solve the problem by itself, its operation can be unpredictable. In this paper the effort is made to identify the best-fitted network for the desired model according to the characteristics of the problem and ANN features. In Fig. 2, a MLP neural network in shown.

Although many different approaches exist in order to find the optimal architecture of an ANN, these methods are usually quite complex in nature and are difficult to implement [60]. Furthermore, none of these methods can guarantee the optimal solution for all real prediction problems. To date, there is no simple clear-cut method for determination of these parameters and the usual procedure is to test numerous networks with varying numbers of
input and hidden units \((p, q)\), estimate generalization error for each and select the network with the lowest generalization error \([36]\). The parameters are estimated such that the cost function of neural network is minimized. Cost function is an overall accuracy criterion such as the following mean squared error:

\[
E = \frac{1}{N} \sum_{i=1}^{N} (e_i)^2
\]

where \(N\) is the number of error terms. This minimization is done with some efficient nonlinear optimization algorithms other than the basic backpropagation training algorithm \([49]\), in which the parameters of the neural network, \(W\) are changed by an amount \(\delta W\), according to the following formula:

\[
\Delta W = -\eta \frac{\partial E}{\partial W}
\]

where the parameter \(\eta\) is the learning rate and \(\frac{\partial E}{\partial W}\) is the partial derivative of the function \(E\) with respect to the weights 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 gives a greater than average chance of promoting the information they contain to the next generation.

Genetic algorithms are inspired by biological systems’ improved fitness through evolution \([38]\). Using the GAs, each individual in the population needs to be described in a chromosome representation. A chromosome is made up of a sequence of genes from a certain alphabet. An alphabet could consist of binary digits, continuous values, integers, symbols, matrices, etc. The representation method determines how the problem is structured in the GA and determines the genetic operators to be used. In this work, a chromosome is represented by a vector of continuous values, as it has been shown that natural representations are more efficient and produce better solutions. In this case, the chromosome length is the vector length of the solution which is coefficients of our model.

In GA, searching starts with an initial set of random solutions known as population. Each chromosome of population is evaluated using some measure of fitness function which represents a measure of the success of the chromosome. Based on the value of the fitness functions, a set of chromosomes is selected for breeding. In order to simulate a new generation, genetic operators such as crossover and mutation are applied. According to the fitness value, parents and offsprings are selected, while rejecting some of them so that the population size is kept constant for new generation. The cycle of evaluation–selection–reproduction is continued until an optimal or a near-optimal solution is found.

Selection attempts to apply pressure upon the population in a manner similar to that of natural selection found in biological systems. Poorer performing individuals (evaluated by a fitness function) are weeded out and better performing, or fitter, individuals have a greater than average chance of promoting the information they contain to the next generation.
Crossover allows solutions to exchange information in a way similar to that used by a natural organism undergoing reproduction. This operator randomly chooses a locus and exchanges the subsequences before and after that locus between two chromosomes to create two offspring.

Mutation is used to randomly change (flip) the value of single bits within individual strings to keep the diversity of a population and help a genetic algorithm to get out of a local optimum. It is typically used sparingly [17]. Principal structure of genetic algorithm is shown in Fig. 3.

GAs may deal successfully with a wide range of problem areas. The main reasons for this success are: (1) GAs can solve hard problems quickly and reliably, (2) GAs are easy to interface to existing simulations and models, (3) GAs are extensible and (4) GAs are easy to hybridize. All these reasons may be summed up in only one: GAs are robust. GAs are more powerful in difficult environments where the space is usually large, discontinuous, complex and poorly understood. They are not guaranteed to find the global optimum solution to a problem, but they are generally good at finding acceptably good solutions to problems acceptably quickly. These reasons have been behind the fact that, during the last few years, GA applications have grown enormously in many fields.

2.4. Evolutionary Neural Networks (ENN)

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:

Step 1. Encoding

Each gene presents the weight between two neurons in different layers. A chromosome is constructed from a series of genes as shown in Fig. 4. In this figure, for a normal feed forward neural network that has 3 neurons in input layer, 2 neurons in hidden layer and 1 neuron in output layer, the first gene in the chromosome is the weight between neuron 1 and neuron 4, i.e. $W_{14}$, the second gene is the weight between neuron 1 and neuron 5, i.e. $W_{15}$ and so on. We use real number form to represent the connection weights.

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 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:

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (Y_i - P_i)^2}$$  (11)

where $Y_i$ is the actual value and $P_i$ is the output value of $i$th training data obtained from the neural network using the weights coded in $j$th 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 to selection of parents for generating new offsprings by use of genetic operators. In binary tournament selection, two members of the population are selected at random and their fitness 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 [27] 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.

2.5. Data post-processing and prediction

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. Case study

We implement PELMNN model by using Taiwan Stock Exchange index (TSE), Tehran Stock Exchange indices (TEPIX) data and other indices of Tehran Stock Exchange such as top 50 Companies index, Industry index and Financial Group index (the same dataset which were used by Chang and Liu [11], Esfahanipour and Aghamiri [21], Dow Jones Industrial Average Index (DJIA) and the Nasdaq Index.

TEPIX has evolved into an exciting and growing marketplace where individuals and institutional investors trade securities of over 420 companies. TEPIX is a weighted market value all share prices appearing on Tehran Stock Exchange price board. TEPIX calculation method is as follows:
where $P_i$ and $P_b$ represent share price of company $i$, respectively at time $t$ and at the close of trading on March 21st, 1990; and $C$ shows the total number of shares.

The TSE index dataset is divided into two different data groups: training data and test data. The TSE index dataset contains total 620 records collected from July 18, 2003 to December 31, 2005. TEPIX and other indices dataset are also divided in the same way as TSE index dataset. These indices contain total 620 records collected from 10 April 2006 to 30 January 2009. The DJIA investigated corresponds to daily observations of the DJIA series from 18th July 2001 to 30 December 2003, constituting a data set of 620 points. The Nasdaq series correspond to daily observations of the Nasdaq index from 10 April 2002 to 28 January 2004 totally 620 records. The first 500 records are included as training data and the rest (i.e. 120 records) as test data. The details about data sets are shown in Table 1 and you can also find the demonstration of datasets plots in Fig 5.

3.2. Input variables

Technical indices are calculated through the variation of stock price, trading volumes and time following a set of formula to reflect the current tendency of the stock price fluctuations. These indices can be applied to make decisions for evaluating the phenomena of oversold or overbought in the stock market. We use technical indices as input variables. The technical indices can be classified as movement or particular stock price variations, such as KD, RSI, MACD, MA, BIAS [21]. We use these seven technical indices as input variables because they are among the most interesting and reliable technical indices in that they integrate features of trading system, are quite simple and respond to a wide variety of possible market situations adequately. These indices are defined in Table 2 [11,21,51].

3.3. Evaluation metrics

To carry out the comparison we use several common evaluation metrics.

3.3.1. MAPE METRIC

The first metric is called mean absolute percentage error (MAPE). Calculation of this metric includes two steps. First, take the absolute deviation between the actual value and the predicted value. Second, compute the total ratio of deviation value by dividing it to its actual value.

$$\text{MAPE} = \frac{100 \times \frac{1}{N} \sum_{i=1}^{N} \frac{|Y_i - P_i|}{Y_i}}{}$$

(13)

where $Y_i$ is the actual value and $P_i$ is the predicted value of $i$th test data obtained from the models and $N$ is the number of test data.

All of the references with which we compared our results have used MAPE metric, so the improvement metric can only be compared and evaluated for MAPE. The ideal value for MAPE is zero, so being closer to zero for the metric indicates that prediction accuracy is higher. The lower the value of MAPE more desired results from the prediction method. Definition of improvement for MAPE metric (%) is:

$$\text{Improvement} = 100 \times \frac{|\text{MAPE Metric of PELMNN} - \text{MAPE Metric of the other model}|}{\text{MAPE Metric of the other model}}$$

(14)
3.3.2. POCID METRIC

A second interesting measure is prediction of change in direction (POCID) in Eq. (14):

$$POCID = 100 \times \frac{\sum_{i=1}^{N} D_i}{N}$$

where

$$D_i = \begin{cases} 1, & \text{if} \left( Y_i - Y_{i-1} \right) \left( P_i - P_{i-1} \right) > 0 \\ 0, & \text{otherwise} \end{cases}$$

This measure allows for an account of the number of correct decisions when predicting whether the value of the series will go up or down during the next time steps. The assigned values by POCID supposed to be between 0 and 100, the closer the values to 100 the more accurate the prediction model. This measure is more important when is applied to the stock market, because a correct prediction on the direction of the series of the stock quotation have a direct impact on financial gains and losses of the investment [23,10].

3.3.3. U of Theil Metric

A third measure is the U of Theil statistics, given by:

$$U_{\text{of Theil}} = \frac{\sum_{i=1}^{N} (Y_i - P_i)^2}{\sum_{i=1}^{N} (Y_i - Y_{i+1})^2}$$

This measure associates the model performance with a RW model. If the U of Theil statistics is greater than 1, then the predictor has a worse performance in comparison to RW model. If the U of Theil statistics is equal to 1, then the predictor has the same performance of the RW model. If the U of Theil statistics is less than 1, then the predictor is better than a RW model. So, the predictor is usable if its U of Theil statistics tends to the perfect model if the U of Theil statistics tends to zero [23].

3.3.4. ARV METRIC

A fourth evaluation measure is the Average Relative Variance (ARV), which is given by:

$$ARV = \frac{\sum_{i=1}^{N} (P_i - Y_i)^2}{\sum_{i=1}^{N} (P_i - \bar{Y})^2}$$

Table 2
Definition of technical indices.

<table>
<thead>
<tr>
<th>Technical index</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Six days moving average (MA6)</td>
<td>Moving averages are used to emphasize the direction of a trend and smooth out price and volume fluctuations that can confuse interpretation</td>
</tr>
<tr>
<td>Six days bias (BIAS6)</td>
<td>The difference between the closing value and moving average line, which uses the stock price nature of returning back to average price to analyze the stock market</td>
</tr>
<tr>
<td>Six days relative strength index (RSI)</td>
<td>RSI compares the magnitude of recent gains to recent losses in an attempt to determine overbought and oversold conditions of an asset</td>
</tr>
<tr>
<td>Nine days stochastic line (K,D)</td>
<td>The stochastic line K and line D are used to determine the signals of over-purchasing, overselling, or deviation</td>
</tr>
<tr>
<td>Moving average convergence and divergence (MACD)</td>
<td>MACD shows the difference between a fast and slow exponential moving average (EMA) of closing prices. Fast means a short-period average, and slow means a long period one</td>
</tr>
<tr>
<td>13 days psychological line (PSY)</td>
<td>PSY is the ratio of the number of rising periods over the total number of periods. It reflects the buying power in relation to the selling power</td>
</tr>
<tr>
<td>Volume</td>
<td>Volume is a basic yet very important element of market timing strategy; volume provides clues as to the intensity of a given price move</td>
</tr>
</tbody>
</table>
If the ARV value is greater than 1, then the predictor is worse than simply taking the mean, and, if the ARV is less than 1, then the predictor is better than considering the mean as the prediction. If the ARV value is equal to 1, the predictor has the same performance as calculating the mean over the series. Hence, we benefit using the predictor if the value of ARV is less than 1, and it becomes even better when the ARV tends to zero. Therefore, the model is practical if the value of ARV is less than 1, and the closer the value to 0, it means that the predictor tends to be perfect [19,10].

3.4. Implementing PELMNN for stock index prediction

In the first stage, we normalize data over range \([-0.9, 0.9]\). This 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 because either a sigmoid function or a hyperbolic tangent (tanh) is calculated and consequently these can only be performed over a limited range of values. A neuron only produces output whose absolute value is less than 1, since the transfer function has asymptotes at \(f(x) = 1\) and \(-1\) (the exception to this is when a linear transformation is used). If the data used with an ANN are not scaled to an appropriate range, then the network will not converge on training, or otherwise will not produce meaningful results [3].

Then we use stepwise regression [9] 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 \( F_e \) (F-to-enter) and \( F_r \) (F-to-remove) have to be determined first. Then the partial F value of each step has to be calculated and compared to \( F_e \) and \( F_r \); if \( F > F_e \), it is considered to add variables to the model; otherwise, if \( F < F_r \), the variables are removed from model \[9\].

The statistical software SPSS 17.0 was used to applying stepwise regression analysis in this research considering \( F_e = 3.84 \) and \( F_r = 2.71 \). The outcomes of this stage are MA6 and BIAS6 for all indices.

In the second stage, we use GA to evolve initial weights of neural networks by using ENN. Then by using LMBP algorithm we tune obtained weights in previous stage. To meet the best network architecture with 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 ENN and LMBP stages of PELMNN after tuning process are detailed in Tables 3 and 4.

<table>
<thead>
<tr>
<th>Optimum features</th>
<th>Parameters</th>
<th>TSE</th>
<th>TEPIX</th>
<th>Top 50 companies</th>
<th>Industry index</th>
<th>Financial group</th>
<th>DJIA</th>
<th>Nasdaq</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>Learning Rate</td>
<td>0.15</td>
<td>0.1</td>
<td>0.15</td>
<td>0.15</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td># iterations</td>
<td>1000</td>
<td>700</td>
<td>900</td>
<td>1000</td>
<td>800</td>
<td>800</td>
<td>800</td>
</tr>
</tbody>
</table>

Fig. 7. Residuals of predicted values by PELMNN.
respectively. At the last stage stock index prediction is done by means of test data and outcomes are post-processed (returned to original scale).

### 3.5. Performance analysis of PELMNN

For the purpose of evaluating prediction accuracy of the PELMNN, we will compare its outcomes with BPNN, PENN, PEBPNN, Esfahanipour and Aghamiri's method [21], Chang and Liu's method [11] and Ferreira et al. method [23] (it should be noted that all of these methods are applied to the same datasets with the same portion of training and test data).

The prediction results of PELMNN for test data sets and residuals for stock indices are shown in Figs. 6 and 7 respectively. Comparison of the performance of the proposed model with other models and percentage improvement of the PELMNN in comparison with other prediction models are shown in Table 5 and 6 respectively.

#### Table 5

Comparison of the performance of the proposed model with other models.

<table>
<thead>
<tr>
<th>Stock name</th>
<th>Method</th>
<th>MAPE (%)</th>
<th>Improvement (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Taiwan Stock Exchange index (TSE)</td>
<td>Hybrid of fuzzy clustering and TSK fuzzy system [21]</td>
<td>1.3</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>TSK fuzzy system tuned by Simulated annealing [11]</td>
<td>2.4</td>
<td>78</td>
</tr>
<tr>
<td></td>
<td>ANN trained with back-propagation (BPNN)</td>
<td>0.78</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>Pre-processing Evolutionary Neural Networks (PENN)</td>
<td>0.67</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>Pre-processing Evolutionary Neural Networks back propagation (PEBPNN)</td>
<td>0.52</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td><strong>PELMNN (Proposed model)</strong></td>
<td><strong>0.51</strong></td>
<td></td>
</tr>
<tr>
<td>Tehran Stock Exchange Prices Index (TEPIX)</td>
<td>Hybrid of fuzzy clustering and TSK fuzzy system [21]</td>
<td>2.4</td>
<td>78</td>
</tr>
<tr>
<td></td>
<td>ANN trained with back-propagation (BPNN)</td>
<td>0.97</td>
<td>47</td>
</tr>
<tr>
<td></td>
<td>Pre-processing Evolutionary Neural Networks (PENN)</td>
<td>0.64</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Pre-processing Evolutionary Neural Networks back propagation (PEBPNN)</td>
<td>0.61</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td><strong>PELMNN (Proposed model)</strong></td>
<td><strong>0.50</strong></td>
<td></td>
</tr>
<tr>
<td>Index of top 50 Companies</td>
<td>Hybrid of fuzzy clustering and TSK fuzzy system [21]</td>
<td>1.85</td>
<td>58</td>
</tr>
<tr>
<td></td>
<td>ANN trained with back-propagation (BPNN)</td>
<td>1.4</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>Pre-processing Evolutionary Neural Networks (PENN)</td>
<td>1.12</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>Pre-processing Evolutionary Neural Networks back propagation (PEBPNN)</td>
<td>0.83</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td><strong>PELMNN (Proposed model)</strong></td>
<td><strong>0.76</strong></td>
<td></td>
</tr>
<tr>
<td>Industry index</td>
<td>Hybrid of fuzzy clustering and TSK fuzzy system [21]</td>
<td>2.02</td>
<td>54</td>
</tr>
<tr>
<td></td>
<td>ANN trained with back-propagation (BPNN)</td>
<td>1.73</td>
<td>47</td>
</tr>
<tr>
<td></td>
<td>Pre-processing Evolutionary Neural Networks (PENN)</td>
<td>1.3</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>Pre-processing Evolutionary Neural Networks back propagation (PEBPNN)</td>
<td>0.98</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td><strong>PELMNN (Proposed model)</strong></td>
<td><strong>0.89</strong></td>
<td></td>
</tr>
<tr>
<td>Index of Financial Group</td>
<td>Hybrid of fuzzy clustering and TSK fuzzy system [21]</td>
<td>1.03</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>ANN trained with back-propagation (BPNN)</td>
<td>0.94</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>Pre-processing Evolutionary Neural Networks (PENN)</td>
<td>0.79</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>Pre-processing Evolutionary Neural Networks back propagation (PEBPNN)</td>
<td>0.69</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td><strong>PELMNN (Proposed model)</strong></td>
<td><strong>0.66</strong></td>
<td></td>
</tr>
<tr>
<td>Dow Jones Industrial Average Index Series</td>
<td>ARIMA</td>
<td>10.23</td>
<td>76</td>
</tr>
<tr>
<td></td>
<td>ANN (LM)</td>
<td>3.9</td>
<td>93</td>
</tr>
<tr>
<td></td>
<td>TAEF[23]</td>
<td>1.13</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ANN trained with back-propagation (BPNN)</td>
<td>3.5</td>
<td>93</td>
</tr>
<tr>
<td></td>
<td>Pre-processing Evolutionary Neural Networks (PENN)</td>
<td>2.4</td>
<td>82</td>
</tr>
<tr>
<td></td>
<td>Pre-processing Evolutionary Neural Networks back propagation (PEBPNN)</td>
<td>2</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td><strong>PELMNN (Proposed model)</strong></td>
<td><strong>1.4</strong></td>
<td></td>
</tr>
<tr>
<td>Nasdaq Index Series</td>
<td>ARIMA</td>
<td>0.57</td>
<td>77</td>
</tr>
<tr>
<td></td>
<td>ANN (LM)</td>
<td>0.27</td>
<td>51</td>
</tr>
<tr>
<td></td>
<td>TAEF[23]</td>
<td>0.16</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>ANN trained with back-propagation (BPNN)</td>
<td>0.3</td>
<td>56</td>
</tr>
<tr>
<td></td>
<td>Pre-processing Evolutionary Neural Networks (PENN)</td>
<td>0.2</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>Pre-processing Evolutionary Neural Networks back propagation (PEBPNN)</td>
<td>0.15</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td><strong>PELMNN (Proposed model)</strong></td>
<td><strong>0.13</strong></td>
<td></td>
</tr>
</tbody>
</table>

#### Table 6

Evaluation measures for all indices.

<table>
<thead>
<tr>
<th>Stock name</th>
<th>MAPE</th>
<th>U of Theil</th>
<th>POCID</th>
<th>ARV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Taiwan Stock Exchange index (TSE)</td>
<td>0.51</td>
<td>0.48</td>
<td>85</td>
<td>0.024</td>
</tr>
<tr>
<td>Tehran Stock Exchange Prices Index (TEPIX)</td>
<td>0.50</td>
<td>0.72</td>
<td>60</td>
<td>0.0012</td>
</tr>
<tr>
<td>Index of top 50 companies</td>
<td>0.76</td>
<td>0.87</td>
<td>57.5</td>
<td>0.0036</td>
</tr>
<tr>
<td>Industry index</td>
<td>0.89</td>
<td>0.09</td>
<td>71.5</td>
<td>0.0108</td>
</tr>
<tr>
<td>Index of Financial Group</td>
<td>0.66</td>
<td>0.048</td>
<td>66.6</td>
<td>0.012</td>
</tr>
<tr>
<td>Dow Jones Industrial Average Index Series</td>
<td>1.41</td>
<td>0.96</td>
<td>58.3</td>
<td>0.084</td>
</tr>
<tr>
<td>Nasdaq Index Series</td>
<td>0.13</td>
<td>0.018</td>
<td>94.16</td>
<td>0.0024</td>
</tr>
</tbody>
</table>
for all cases. The ARV for each case is less than 1; it means that the PELMNN is rather perfect model. POCID metric for...and high degree of accuracy. The results indicate that the ANN model and the predicted values were generated. The advantages of PELMNN and its other models and can be considered as a promising alternative for stock market prediction problems. In the PELMNN, the GA has been employed to escape the local optimal and in the following, LM 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 stock market.

4. Conclusions

Both theoretical and empirical findings suggest that combining different methods can be an effective and efficient way to improve prediction results. This paper presented a pre-processed evolution-ary Levenberg–Marquardt neural networks (PELMNN) model for stock market 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 means of data transformation technique and also stepwise regression was used for input selection to filter out the unrelated variables and keep only those variables, which have significant effects to the stock index. 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 local search, because of LMPP is capable in local search. Finally in post-processed stage, the output data were returned to its original scale and the predicted values were generated. The advantages of PELMNN are fast training ability, good adaptation of non-linear problem and high degree of accuracy. The results indicate that the ANN generated by the PELMNN method is not a RW model and it has been shown that PELMNN model for all cases is better than RW model. Moreover, experimental results showed that the PELMNN is able to cope with the fluctuations of stock market values and it also yields good prediction accuracy.

Acknowledgments

The authors wish to express their gratitude to two anonymous referees for their helpful comments, which greatly helped us to improve our paper.

References