Design and Implementation of Artificial Neural Network System for Stock Exchange Prediction

*A. Ghezelbash
 Lecturer, Faculty of Engineering
 University of Jiroft
 Jiroft, Iran,
 E-mail: teacher.ghezelbash@yahoo.com

F. Keynia
 Energy Research Institute
 Institute of Science and High Technology and Environmental Science
 Graduate University of Advanced Technology
 Kerman, Iran,
 E-mail: fkeynia@kgut.ac.ir

*Corresponding author

ABSTRACT
Stock prediction with artificial neural network (ANN) techniques is one of the most important issues in finance being investigated by researchers across the globe. ANN techniques can be used extensively in the financial markets to help investors make qualitative decision. In this methodology a multilayer perception (M.L.P) neural network model is used to determine and explore the relationship between some variables as independent factors and the level of stock price index as a dependent element in the stock market under study over time. The results show that the neural network models can get better outcomes compared with statistical and parametric models like as multiple regression and other traditional statistical techniques. This study and test also show that useful predictions can be made without the use of extensive market data or knowledge, and in the data mining process, neural networks and some non algorithmic models can explore high level orders in complex time series which hide in the market structure and need very huge calculations in normal conditions. Our study was including of a relatively extensive range of indexes stock market prices in Iran. We've made two different predictions in Tehran Stock Exchange (TSE), and by help ANN and a new method of data mining, indexes stock market prices with about 1% error level, we predict.

Keywords - Stock Prediction, Artificial Neural Networks, Stock Market Indexes

African Journal of Computing & ICT Reference Format:

1. INTRODUCTION
Stock market forecasters focus on developing approach-es to successfully forecast/ predict index values or stock prices, aiming at high profits using well defined trading strategies. The central idea to successful stock market prediction is achieving best results using minimum required input data and the least complex stock market model. Investors in stock market primarily traded stocks based on intuition before the advent of computers. The continuous growth level of investing and trading necessitate a search for better tools to accurately predict the market in order to increase profits and reduce losses. Statistics, technical analysis, fundamental analysis, time series analysis, chaos theory and linear regression are some of the techniques that have been adopted to predict the market direction [14].

Artificial neural network (ANN) technique is one of data mining techniques that is gaining increasing acceptance in the business area due to its ability to learn and detect relationships among nonlinear variables. Also, it allows deeper analysis of large sets of data, especially those that have the tendency to fluctuate within a short of period of time. This makes ANN a candidate for stock market prediction. Many research efforts have been made to improve the predictive accuracy and computational efficiency of share values [17]. Financial forecasting is of considerable practical interest and due to artificial neural networks’ ability to mine valuable information from a mass history of data; its applications to financial forecasting have been very popular over the last few years [15, 4, and 9].
Neural networks have been criticized and their widespread successful application likely hindered because of the black box nature of their solutions, excessive training times, difficulty in obtaining and later replicating a stable solution, the danger of overfitting, tedious software, and the large number of parameters that must be experimentally selected to generate a good forecast. Table 1 lists the most common parameters that a researcher must choose when designing a neural network forecasting model. It excludes the many different proprietary features offered by neural network software vendors and ignores some more advanced topics. The large numbers of ways to organize a neural network account for its powerful function approximation capabilities.

The cost of such flexibility in modeling time series data is that the researcher must select the right combination of parameters. As a result of the large number of parameters and the relatively recent introduction of neural networks to finance, deciding on the appropriate network paradigm still involves much trial and error. Therefore, the objective of this paper is to provide an overview of a step by step methodology to design a neural network for forecasting economic time series data.

First, the architecture of a backpropagation (BP) neural network is briefly discussed. The BP network is used to illustrate the design steps since it is capable of solving a wide variety of problems and it is the most common type of neural network in time series forecasting. This is followed by an explanation of an eight-step procedure for designing a neural network including a discussion of tradeoffs in parameter selection, some common pitfalls, and points of disagreement among practitioners.

Research on Iranian studies for business excellence in TSE, like FadaiNejad [5], and Abdoh Tabrizi and Jouhari [2] and Namazi and Shoustarian [11] to be inefficient than the mature markets. In fact, even the stock price movements of U.S [6] and Japan [3] have been shown to conform only the weak from of the efficient market hypothesis.

In this paper, taking the output of the neural network, it has been implemented using MATLAB software.

Table 1. Common Parameters in Designing Back Propagation ANN

<table>
<thead>
<tr>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data preprocessing</td>
</tr>
<tr>
<td>Frequency of data-daily, weekly, monthly, quarterly type of data - technical, fundamental</td>
</tr>
<tr>
<td>Method of data sampling</td>
</tr>
<tr>
<td>Method of data scaling- maximum/minimum, mean/standard deviation</td>
</tr>
<tr>
<td>Training</td>
</tr>
<tr>
<td>Learning rate per layer momentum term training tolerance</td>
</tr>
<tr>
<td>epoch size</td>
</tr>
<tr>
<td>Learning rate limit</td>
</tr>
<tr>
<td>Maximum number of runs</td>
</tr>
<tr>
<td>Number of times to randomize weights</td>
</tr>
<tr>
<td>Size of training, testing, and validation sets</td>
</tr>
<tr>
<td>Topology</td>
</tr>
<tr>
<td>Number of input neurons number of hidden layers</td>
</tr>
<tr>
<td>Number of hidden neurons in each hidden layer number of output neurons</td>
</tr>
<tr>
<td>Transfer functions for each neuron error function</td>
</tr>
</tbody>
</table>

2. BACKPROPAGATION NEURAL NETWORKS

Backpropagation (BP) neural networks consist of a collection of inputs and processing units known as either neurons or nodes (Fig 1). The neurons in each layer are fully interconnected by connection strengths called weights which, along with the network architecture, store the knowledge of a trained network [13]. In addition to the processing neurons, there is a bias neuron connected to each processing unit in the hidden and output layers. The bias neuron has a value of positive one and serves a similar purpose as the intercept in regression models. The neurons and bias terms are arranged in layers. Input layer, one or more hidden layers and an output layer. The number of hidden layers and neurons within each layer can vary depending on the size and nature of the data set. ANNs are similar to linear and non-linear least squares regression and can be viewed as an alternative statistical approach to solving the least squares problem [10].
The number of input neurons is equal to the number of independent variables while the output neuron(s) represent the dependent variable(s). Linear regression models may be viewed as a feed-forward ANN with no hidden layers and one output neuron with a linear transfer function. The weights connecting the input neurons to the single output neuron are analogous to the coefficients in a linear least squares regression. Networks with one hidden layer resemble nonlinear regression models. The weights represent regression curve parameters.

**Figure 1: Layer Perceptron Networks With n Neurons in Each Layer**

Considering the output of $i$th neuron (in the last layer) can be shown as equation 1 [12].

$$\hat{y}_i = \sigma_{gm}(\sum_{j=1}^{n} w_{ij} x_j)$$  \hspace{1cm} (1)

Where, $O$ and $h$ respectively represent the hidden layer and output layer and $w$ is the weight of the layers. $sgm$ is also sigmoid function which is defined as [12].

$$\sigma_{gm}(x) = \frac{1}{1 + e^{-x}}$$  \hspace{1cm} (2)

For training network and weight improve to achieve a significant error, there are so many ways. One of the most famous of these methods is the error back propagation algorithm, which is described below.

### 3. ERROR BACK PROPAGATION ALGORITHM

The propagation error method is of methods with the supervisor. This means that the input samples are labeled and their expected output of each has been known before. The network output is compared with the ideal output and the network error is calculated.

BP networks are a class of feed-forward ANNs with supervised learning rules. Feed-forward refers to the direction of information flow from the input to the output layer. Inputs are passed through the neural network once to determine the output. Supervised learning is the process of comparing each of the network's forecasts with the known correct answer and adjusting the weights based on the resulting forecast error to minimize the error function.
With accuracy in relation 4 we find that weights can be changed to change the output.

As mentioned before that the purpose of the training process to achieve optimal output (or ear optimal) is. Therefore for each neuron must first define the error function, the error of the difference between the actual network output and expected output is obtained as follows:

$$E_i(\theta, w, a_i) = (a_i(\theta, w) - d_i)^2$$ (5)

4. ANN DESIGN FOR PREDICTION IN T.S.E

Action choice of index actual to predict the behavior of stock index and the designed model Prediction System is very difficult.

4.1 Application of Trained Model in an Actual Setting

The TEPIX is calculated on the basis of Iranian stocks. It is capitalization – weighted by Laspiers relation. Tehran Stock Exchange from March 1990 attempted to calculate and publish their price index name is TEPIX, this total time index included 52 companies that listed companies were. This index indicates that total market value than the base year (or 1991) is several times [11]. Main index are wide application in country macroeconomic and investment market investors and is calculated using relation 6.

$$\text{TEPIX}_t = \left( \frac{\sum P_i q_i D_i}{\sum P_i q_{io}} \right) \times 100$$ (6)

Where:
- \(P_i\) = price of \(i\)th company at time \(t\)
- \(q_i\) = number of shares outstanding \(i\)th company at time \(t\)
- \(D_i\) = the base value at time \(t\) when equal source was \(P_{io}\) Price Company \(i\)th at the time of source
- \(q_{io}\) = number of shares outstanding company \(i\)th at the time of source
- \(n\) = number of companies eligible index

4.2 Steps in Designing an ANN Forecasting Model

A method of designing an ANN forecasting model into distinct steps is used here. The eight-step design methodology presented in Table 2 draws on the steps outlined by Deboeck [7], Masters [16], Blum [1], and Nelson and Illingworth.

The procedure is usually not a single-pass one, but may require visiting previous steps especially between training and variable selection.
Table 2. Eight steps in designing an ANN forecasting models

| Step 1: Variable selection | Step 2: Data collection | Step 3: Data preprocessing | Step 4: Training, testing, and validation sets | Step 5: Neural network paradigms | number of hidden layers | number of hidden neurons | number of output neurons | transfer functions | Step 6: Evaluation criteria | Step 7: Neural network training | number of training iterations | learning rate and momentum | Step 8: Implementation |
|---------------------------|------------------------|--------------------------|-----------------------------------------------|--------------------------------|-------------------------|------------------------|------------------------|-----------------------------|-----------------------|--------------------------|-----------------------------|---------------------------|-------------|------------------------|

4.3 Variable Selection & Data Collection & Data Preprocessing

Success in designing a neural network depends on a clear understanding of the problem [10]. Knowing which input variables are important in the market being forecasted is critical. This is easier said than done because the very reason for relying on a neural network is for its powerful ability to detect complex nonlinear relationships among a number of different variables. However, economic theory can help in choosing variables which are likely important predictors. At this point in the design process, the concern is about the raw data from which a variety of indicators will be developed. These indicators will form the actual inputs to the neural network. When using basic data as an input in an ANN four issues must be kept in mind:

First, the method of calculating the fundamental indicator should be consistent over the time series. Second, the data should not have been retroactively revised after its initial publication as is commonly done in databases since the revised numbers are not available in actual forecasting. Third, the data must be appropriately logged as inputs in the neural network since fundamental information is not available as quickly as market quotations. Fourth, the researcher should be confident that the source will continue to publish the particular fundamental indicator or other similar sources are available. The daily data from 2009 to 2012 are used for the first trial. The Figure shows the graph of the TEPIX represented as logarithmic return in \((Te_{i+1}/Te_i)\) for the defined period, where \(Te\) is the index value noisy which markets forecasting very difficult. The inputs to the neural network models are:

- Gold coin average change 2 weeks ago.
- Gold coin average change 1 week ago.
- U.S. Dollar exchange 2 weeks ago.
- U.S. Dollar exchange 1 week ago.
- Euro exchange 2 weeks ago.
- Euro exchange 1 week ago.
- Market value of Iran Mercantile Exchange (IME) 2 weeks ago.
- Market Value of Iran Mercantile Exchange (IME) 1 week ago.
- T.S.E volume change 2 weeks ago.
- T.S.E volume change 1 week ago.
- Moving average of TEPIX 2 weeks ago.
- Moving average of TEPIX 1 week ago.

The output of the neural network is stock price index, which shows the price level of the market. We've done two types of forecasts in the Tehran Stock Exchange. In the first phase, we have predicted Main index during the recent 100 days, Due to data 550 days before that. In the second phase, this was stage original and useful of our project, each of the eight sub-indexes. We forecast for the next business day in the stock market. We estimate Main index (the main and most important stock index) model based on eight indexes the past 500 days:

1. First market index, Due to data 550 days before that
2. Second market index, Due to data 550 days before that
3. Financial index, Due to data 550 days before that
4. Industry index, Due to data 550 days before that
5. Weighted average of the top fifty companies, Due to data 550 days before that
6. Simple average of the top fifty companies, Due to data 550 days before that
7. Price index and cash returns, Due to data 550 days before that
8. Free float index, Due to data 550 days before that.

In general, the stock price data have bias due to difference in name and spans. Normalization can be used to reduce the range of the data set to values appropriate for input to the activation function being used. The normalization and scaling formula 7 is:

\[
y = \frac{x - \text{min}}{\text{max} - \text{min}}
\]

(7)

Where
\(x\) is the data before normalizing.
\(y\) is the data after normalizing.

Stock price index is normalized in the same scale. The outputs of the neural network will be rescaled back to the original value according to the same formula.
4.4 Training, Testing And Validation Sets & Neural Network Paradigms

Common practice is to divide the time series into three distinct sets called the training, testing, and validation (out-of-sample) sets. The training set is the largest set and is used by the neural network to learn the patterns present in the data. The testing set, ranging in size from 10% to 30% of the training set, is used to evaluate the generalization ability of a supposedly trained network. The researcher would select the network(s) which perform best on the testing set. A final check on the performance of the trained network is made using the validation set. The size of the validation set chosen must strike a balance between obtaining a sufficient sample size to evaluate a trained network and having enough remaining observations for both training and testing. The validation set should consist of the most recent contiguous observations. Care must be taken not to use the validation set as a testing set by repeatedly performing a series of train, test and validation steps and adjusting the input variables based on the network's performance on the validation set.

4.5 Architectures Of ANN

Determine the number of layers, the number of inputs, outputs and the number of hidden layers and number of neurons in each layer of an ANN is the most important design issues. There are several approaches or rules of thumb for choosing the number of neurons in hidden layer(s) while designing an ANN topology. The ones retrieved from literature are:

\[
\frac{i+o}{2}, \text{ as defined by Man-Chung et al (2000)}
\]

\[
\frac{2i}{s+o}, \text{ as defined by Azoff (1994)}
\]

\[
\left(\frac{i+o}{s}\right), \text{ as defined in the Neural Ware Software Manual}
\]

\[
\frac{1}{p+q}, \text{ as defined in the Neural Ware Software Manual}
\]

\[
\frac{3}{2}, \text{ as defined by Heaton (2005)}
\]

\[
k, \text{ as defined by Freisleben (1992)}
\]

\[
\frac{1}{k}, \text{ as defined by Freisleben (1992)}
\]

Where \(i\) the number of inputs, \(o\) is the number of outputs and \(k\) is the number of hidden layer. We used these relations in the construction of neural networks predictor.

4.6 Evaluation Criteria & Neural Network Training & Implementation

The most common error function minimized in neural networks is the sum of squared errors. Other error functions offered by software vendors include least absolute deviations, least fourth powers, asymmetric least squares, and percentage differences. These error functions may not be the final evaluation criteria since other common forecasting evaluation methods such as the mean absolute percent-age error (MAPE) are typically not minimized in neural networks.

Training a neural network to learn patterns in the data involves iteratively presenting it with examples of the correct known answers. The objective of training is to find the set of weights between the neurons that determine the global minimum of the error function. Unless the model is overfitted, this set of weights should provide good generalization. The BP network uses a gradient descent training algorithm which adjusts the weights to move down the steepest slope of the error surface. One method to determine a reasonable value for the maximum number of runs is to plot the mean correlation, sum of squared errors, or other appropriate error measure for each iteration or at predetermined intervals up to the point where improvement is negligible (usually up to a maximum of 10,000 iterations).

Each iteration can be easily plotted if the neural network software creates a statistics file or, if this is not the case, the mean correlation can be recorded at intervals of 100 or 200 from the computer monitor. After plotting the mean correlation for a number of randomly selected starting weights, the researcher can choose the maximum number of runs based on the point where the mean correlation stops increasing quickly and flattens. The implementation step is listed as the last one, but in fact requires careful consideration prior to collecting data. Data availability, evaluation criteria, and training times are all shaped by the environment in which the neural network will be deployed. Most neural network software vendors provide the means by which trained networks can be implemented either in the neural network program itself or as an executable file. If not, a trained network can be easily created in a spreadsheet by knowing its architecture, transfer functions, and weights. Care should be taken that all data transformations, scaling, and other parameters remain the same from testing to actual use.

An advantage of neural networks is their ability to adapt to changing market conditions through periodic retraining. Once deployed, a neural network's performance will degrade over time unless retraining takes place. However, even with periodic retraining, there is no guarantee that network performance can be maintained as the independent variables selected may have become less important.

5. RESULTS

Statistics characteristics of TEPIX series are analyzed first before applying it to neural network models. Table 3 shows means, maximum, minimum, Variance, standard deviation, skewness, and kurtosis.
Table 3. Statistics results of TEPIX

<table>
<thead>
<tr>
<th></th>
<th>min</th>
<th>mean</th>
<th>max</th>
<th>stdev</th>
<th>var</th>
<th>skew</th>
<th>kurt</th>
</tr>
</thead>
<tbody>
<tr>
<td>min</td>
<td>170</td>
<td>1983.27</td>
<td>1980.5</td>
<td>180.002</td>
<td>29653.07</td>
<td>1.00159</td>
<td>1.0283</td>
</tr>
</tbody>
</table>

Accordance table 4, we found that the most ideal ANN architecture, for our issue, structure 8-7-1 is, with fault \( \alpha =0.5802 \) ability to extract each of indexes based on other indexes for every day. This neural network has nearest reply to actual data with minimum error after 171 replications. We used the training functions and transfer function for this ANN), following the: TRAINGDM\(^2\), TRAINGDA\(^3\), TRAINGDX\(^4\) and transfer function LOGSIG\(^5\).

Fig. 2 illustrates the training phase of the model 7 for Index II. That shows this model has a high performance, because almost all 3 graphs train, validation and test, movements are quite smoothly and at epoch 171th collide together. This means that minimize the prediction error. Fig 3 shows regression over 99%. In this case, we can trust to the output values of the neural network.

5. CONCLUSION

We forecasts values for each indexes from 04/27/2013 to 08/09/2013, were compared with actual values on the same day in the Table 4 and have brought them each rate errors. Our review shows that this issue is the in months year that symmetrical is the religious months of Muharram and Ramazan, TSE indexes, compared to other months, the movements are more different. According to Table IV, it should be emphasized that the ANN models showed significant performance in predicting the direction of stock price movement. Thus, we can say that the ANN is useful prediction tools for this topic.

---

1. Gradient Descent with Momentum back-propagation
2. Gradient Descent with Adaptive learning back-propagation
3. Gradient Descent with momentum and adaptive learning back-propagation
4. Log-Sigmoid transfer function
5. Log-Sigmoid transfer function
Table 4. ANNs Is Built For Prediction 8 Indexes

<table>
<thead>
<tr>
<th>Neurons of hidden layer</th>
<th>Index I</th>
<th>Index II</th>
<th>Index III</th>
<th>Index IV</th>
<th>Index V</th>
<th>Index VI</th>
<th>Index VII</th>
<th>Index VIII</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>3.2562</td>
<td>3.2580</td>
<td>1.2303</td>
<td>2.1489</td>
<td>1.28091</td>
<td>1.9630</td>
<td>2.0051</td>
<td>1.7419</td>
</tr>
<tr>
<td>6</td>
<td>2.2078</td>
<td>2.2614</td>
<td>3.6520</td>
<td>2.1050</td>
<td>1.9073</td>
<td>1.8124</td>
<td>1.7966</td>
<td>2.0400</td>
</tr>
<tr>
<td>7</td>
<td>1.2519</td>
<td>0.5802</td>
<td>2.5712</td>
<td>3.4268</td>
<td>3.1057</td>
<td>2.1140</td>
<td>2.8409</td>
<td>4.8620</td>
</tr>
<tr>
<td>17</td>
<td>2.5901</td>
<td>1.8960</td>
<td>2.2501</td>
<td>1.9637</td>
<td>2.3088</td>
<td>1.9543</td>
<td>1.1982</td>
<td>3.0012</td>
</tr>
</tbody>
</table>

REFERENCES


