Application of Neural Network in Analysis of Stock Market Prediction

NEELIMA BUDHANI
Lecturer, Department of Computer Science
Amrapali Institute, Haldwani (Nainital) (Uttarakhand) India.
E-mail: neelimap28@gmail.com

Dr. C. K. JHA
Associate Professor, Department of Computer Science
Banasthali Vidyapeeth, Banasthali (Rajasthan) India.
E-mail: ckjha1@gmail.com

SANDEEP K. BUDHANI
Assistant Professor, Department of Computer Science & Engineering
Graphic Era Hill University, Bhimtal Campus, Bhimtal (Nainital), (Uttarakhand) India
E-mail: sandeepbudhani13@gmail.com

Abstract- Predicting the stock market is very difficult since it depends on several known and unknown factors. So many methods like Technical analysis, Fundamental analysis, Time series analysis and statistical analysis etc. are all used to attempt to predict the price in the share market but none of these methods are proved as a consistently acceptable prediction tool. Artificial neural network (ANN), a field of Artificial Intelligence (AI), is relatively new, active and promising technique on finance problem such as stock exchange index prediction, bankruptcy prediction and corporate bond classification. ANN, is a popular way to identify unknown and unseen patterns in data which is suitable for share market prediction. We used Feedforward neural network trained by Back propagation algorithm to make prediction. The amalgamation of profit and time factors with training procedure made an improvement in forecasted result for Feedforward neural network.

Keywords: Artificial Neural Network, backpropagation, Prediction, Stock Market.

I. THE STOCK MARKET

Prediction in stock market has been a hot research topic for many years. The major theories include the ***Random Walk Hypothesis*** and the ***Efficient Market Hypothesis***. The Random Walk Hypothesis states that prices on the stock occur without any influence by past prices. The Efficient Market Hypothesis states that price on the stock occur without any influence by past prices. The Efficient Market Hypothesis states that the market fully reflects all of the freely available information and prices are adjusted fully and immediately once new information becomes available. If this is true then there should not be any benefit for prediction, because the market will react and compensate for any action made from available information [1].

II. PREDICTION METHOD

The prediction of the market is without doubt an interesting task. In the literature there are a number of methods applied to accomplish this task. These methods use various approaches, ranging from highly informal ways (e.g. the study of a chart with the fluctuation of the market) to more formal ways (e.g. linear or non-linear regressions). These approaches can be categorized as follows:

A. Technical Analysis Methods

The technical analysis predicts the appropriate time to buy or sell a share. Technical analysts use charts which contain technical data like price, volume, highest and lowest prices per trading to predict future share movements. This is a very popular approach used to predict the market. But the problem of this analysis is that the extraction of trading rules from the study of charts is highly subjective, as a result different analysts

extract different trading rules studying the same charts. Alongside the patterns, statistical techniques are utilized such as the ***exponential moving average (EMA)***.

B. Fundamental Analysis Methods

Fundamental analysis is the physical study of a company in terms of its product sales, man power, quality, infrastructure etc to understand it standing in the market and thereby its profitability as an investment. The fundamental analysts believe that the market is defined 90 percent by logical and 10 percent by physiological factors [2]. Many performance ratios are created that aid the fundamental analyst with assessing the validity of a stock, such as the ***P/E ratio, Warren Buffett*** is perhaps the most famous of all Fundamental Analysts.

C. Traditional Time Series Prediction Methods

Time series analysis comprises methods for analyzing time series data in order to extract meaningful statistics and other characteristics of the data. Time series forecasting is the use of a model to predict future values based on previously observed values. Time series models will often make use of the natural one-way ordering of time so that values for a given period will be expressed as deriving in some way from past values, rather than from future values. Methods for time series analyses may be divided into two classes: ***frequency-domain methods and time-domain methods.*** Models for time series data are ***ARMA, ARIMA, ARFIMA, and GARCH*** [3]**.**

D. Machine Learning Method

Machine learning approach is attractive for artificial intelligence since it is based on the principle of learning from training and experience. Connectionist models such as ***ANNs*** are well suited for machine learning where connection weights adjusted to improve the performance of a network.

III. NEURAL NETWORKS

A. Neurons and Neural Networks

In 1943, W.S.Mcculloch and W.Pitts established Neural Network and its mathematical model, which was called MP model. Artificial neural network is a mathematical model simulating the learning and decision making processes of the human brain. ANN is the interconnection of artificial neurons working in a fashion to solve a specific problem. A neuron is the basic unit of the nervous system such as brain. Neurons connected via "dendrites" (small branches extension of nerve cells that receive signals from other cells). Biological neuron stores knowledge in a memory bank, while in an artificial neuron the data or information is distributed through the network and stored in the form of weighted interconnections.

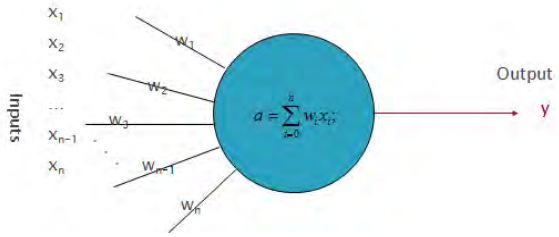

Fig.1 Graphical representation of artificial neuron

Rather than receiving electrical signals, artificial neurons receive a number from other neurons, and process these numbers accordingly. **Fig.1** shows a graphical representation of artificial neurons, where x_i represents the inputs to the neuron and w_i represents weights of the neuron. The overall input to the neuron is calculated by $a= \sum^{n}_{i=0} w_{ixi.}$

For explanatory purposes, a neuron may be broken down into three parts:

- Input connection
- Summing and activation function
- Output connection

A.1 Input connections

Unless the artificial neuron is an input neuron, a neuron is connected to other neurons and depends on them to receive the information that it processes. There is no limit to the amount of connections a neuron may receive information from.
The information that a neuron receives from others is regulated through the use of weights. When a neuron receives information from other neurons, each piece of information is multiplied by a weight with a value between -1 and 1, which allows the neuron to judge how important the information it receives from its input neurons is. These weights are integral to the way a network works and is trained: specifically, training a network means modifying all the weights regulating information flow to ensure outputs are correct [4].

A.2 Summing and Activation function

The information sent to the neuron and multiplied by corresponding weights is added together and used as a parameter within an activation function. If these signals are sufficient, the neuron will become "activated" it will send electrical signals to the neurons connected to it. An activation function is similar: the artificial neuron will output a value based on these inputs. It is almost always the case that a neuron will output a value between [0, 1] or [-1, 1], and this normalization of data occurs by using the summed inputs as a parameter to a normalizing function, called an "activation function".
Numerous activation functions exist, but within this paper, three types of activation functions are explored:

(i) Threshold Function: a simple function that compares the summed inputs to a constant, and depending on the result, may return a -1, 0, or 1. specifically, for summed input

$$f(x) = \begin{cases} 1 & if \quad x > 0 \\ 0 & if \quad x = 0 \\ -1 & if \quad x < 0 \end{cases}$$

(ii) Piecewise-Linear Function: This activation function is also called saturating linear function and can have either a binary or bipolar range for the saturation limits of the output. The mathematical model for a symmetric saturation function is described below.

$$f(x) = \begin{cases} 1 & if \quad x \geq 0 \\ 0 & if \quad -1 \geq x \geq 1 \\ -1 & if \quad x < 0 \end{cases}$$

(iii) Hyperbolic tangent Function: a continuous function with a domain of $(-\infty, \infty)$ and a range of (-1, 1). Specifically (Weisstein, 1999),

$$\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$$

A.3 Output connection

Finally, once the activation function returns a corresponding value for the summed inputs, these values are sent to the neurons that treat the current neuron as an input. The process repeats again, with the current neuron's output being summed with others, and more activation functions accepting the sum of these inputs. The only time this may be ignored is if the current neuron is an output neuron. In this case, the summed inputs and normalized sum is sent as an output and not processed again.

B. Feedforward Networks

While each neuron is, in and of itself, a computational unit, neurons may be combined into layers to create complex and efficient groups that can learn to distinguish between patterns within a set of given inputs. Indeed, by combining multiple layers of such groups, it is theoretically possible to learn any pattern.

There are many combinations of neurons that allow one to create different types of neural networks, but the simplest type is a single-layer Feedforward network. In this case, a network is composed of three parts: a layer

of input nodes, a layer of hidden neurons, and a layer of output nodes. Within this network, there is a neuron for each input variable in an input pattern, which is then propagated to the layer of hidden neurons.

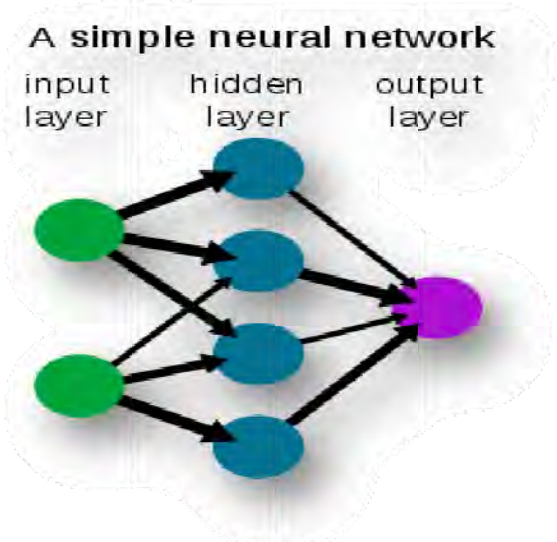

Fig.2 Feedforward Neural Network

Fig.2 shows the simple Feedforward neural network. The input layer takes input values from software interface. Neurons are connected to other neurons via synapses. Values from input layer are fed from left to right, through various hidden layers. The output values of neurons obtained from the output layer. The layered architecture is inspired directly by the design of brain, in which layers of neuron cells are arranged in an onion skin pattern**.** As mentioned earlier, each neuron in the hidden layer has the inputs multiplied by a weight, and all inputs are summed. After this is done, the value is passed to an activation function, and each neuron in the hidden layer then passes the output on to neurons in an output layer, which also multiply the values by weights and sum them.

A multilayer Feedforward network is similar to a single-layer one. The main difference is that instead of having a hidden layer pass its calculated values to an output layer, it passes them on to another hidden layer.
Both types of networks are typically implemented by fully connecting each layer's neurons with the preceding layer's neurons. Thus, if *Layer A* has *k* neurons and sends its information to *Layer B*, with *n* neurons, each neuron in *Layer A* has *n* connections for its calculated output, while each neuron in *Layer B* has *k* input connections.

Interestingly, such a network can be represented mathematically in a simple manner. Supposing there are k neurons in Layer A, let $\underline{a}$ represent a vector, where a_i is the i^{th} neuron's activation function output. Let $\underline{b}$ represent the input values to neurons in Layer B, with bj be the j^{th} neuron. Let W be a n by k matrix where w_{ji} represents the weight affecting the connection from a_i to b_j. Keeping this in mind, we can see that for a single-layer Feedforward network, we can mathematically represent the flow of information by,

$W\underline{a} = \underline{b}$

and the learning thus becomes a modification of each w_{ji} in W. A similar mathematical analogy applies to multilayer Feedforward networks, but in this case, there is a W for every layer and $\underline{b}$ is used as the value for a when moving to subsequent layers.

The most popular type of learning within a single-layer Feedforward network is the Delta Rule, while multilayer Feedforward networks implement the Backpropagation algorithm, which is a generalization of the Delta Rule.

IV. WHY NEURAL NETWORK USED

Predicting stock market return is a central issue in the field of finance, engineering, mathematics, economy and science due to its potential finance gains. Neural network may be one of the best approaches to predict the nature of stock market.

There are several distinguished features that promulgate the use of neural network as a preferred technique over other traditional models of prediction. Artificial neural networks are nonlinear in nature and where most of the natural world system are non linear. This is because the linear model is generally failed to understand data pattern and analyse when the underlying system is a non linear one. However, some parametric nonlinear model such as Autoregressive Conditional Heteroskedasticity (Engle, 1982) and General Autoregressive Conditional Heteroskedasticity have been in use for stock prediction [5, 6].

Artificial neural networks are data driven models. The novelty of the neural network lies in their ability to discover nonlinear relationship in the input data set without a priori assumption of the knowledge of relation between the input and the output (Hagen et al., 1996) the input variables are mapped to the output variables by squashing or transforming by a special function known as activation function. They independently learn the relationship inherent in the variables from a set of labeled training example and therefore in modification of the network parameters.

Neural networks can be used for prediction with various levels of success. The advantage of then includes automatic learning of dependencies only from measured data without any need to add further information (such as type of dependency like with the regression). The neural network is trained from the historical data with the hope that it will discover hidden dependencies and that it will be able to use them for predicting into future. In other words, neural network is not represented by an explicitly given model. It is more a black box that is able to learn something. Moreover, when the system under study is non stationary and dynamic in nature, the neural network can change its network parameters (synaptic weights) in real time. So, neural network suits better than other models in predicting the stock market returns.

Regarding downsides the black-box-property first spring to mind. Relating one single outcome of a network to a specific internal decision (known as credit assignment problem) is very difficult. Noisy data also reinforce the negative implication of establishing incorrect causalities, *overtraining*, which will harm generalization. Finally, a certain degree of knowledge in current subject is required as it is not trivial to assess the relevance of chosen input series.

Thus a short summary of ***benefits and drawbacks***:
+ Generalization ability and robustness
+Mapping of input/output
+ Flexibility
- Black-Box property
- Overtraining
- Training takes a lot of time

V. TRAINING A NEURAL NETWORK

The way that a network is trained is depicted by the ***Fig.3***. Each sample consists of two parts the input and the target part (supervised learning). Initially the weights of the network are assigned random values (usually within [-1 1]). Then the input part of the first sample is presented to the network. The network computes an output based on: the values of its weights, the number of its layers and the type and mass of neurons per layer.

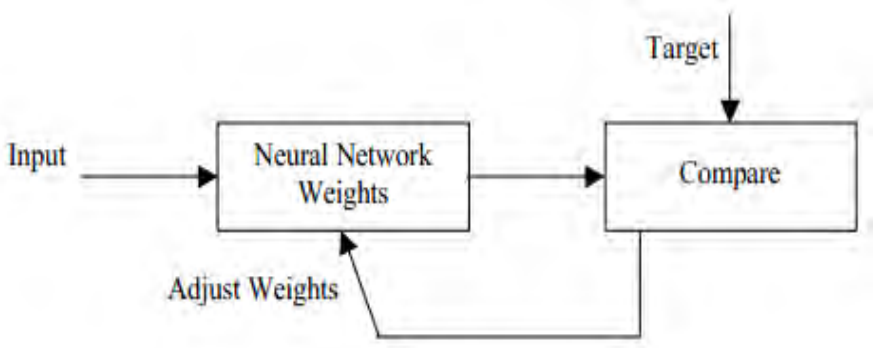

Fig.3 The training procedure of Neural Network

This output is compared with the target value of the sample and the weights of the network are adjusted in a way that a metric that describes the distance between outputs and targets is minimized.

There are two major categories of network training the incremental and the batch training. During the incremental training the weights of the network are adjusted each time that each one of the input samples are presented to the network, while in batch mode training the weights are adjusted only when all the training samples have been presented to the network The number of times that the training set will be feed to the network is called number of ***epochs.***

A. Back propagation

The objective of the training is to minimize the divergence between real data and the output of the network. This principle is referred to as ***Supervised Learning.***

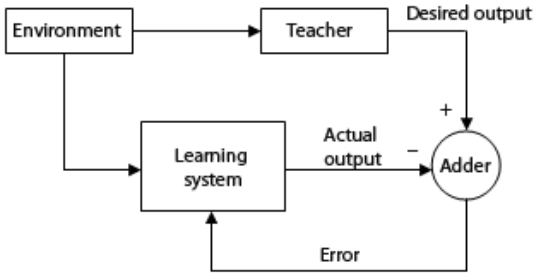

Fig.4 Backpropagation algorithm model.

In a step by step manner the error guides the network in the direction towards the target data. The back propagation algorithm belongs to this class and can be described as "an efficient way to calculate the partial derivatives of the network error function with respect to the weights". ***Fig.4*** shows the general model for backpropagation algorithm.

A.1 Learning Algorithm

The Backpropagation algorithm supplies information about the gradients. However, a learning rule that uses this information to update the weights efficiently is also needed.

A weight update from iteration k to k+1 may look like

$$w_{k+1} = w_k + \eta.d_k$$

where d_k describe the search direction and η the learning rate. Issues that have to be addressed are how to determine
(i) the search direction (ii) the learning rate and (iii) which pattern has to include.

A familiar way of determining the search direction d_k is to apply ***gradient descent.*** *The major drawback is that the learning easily is caught in local minima.* To avoid this, ***vario-eta algorithm*** can be chosen as learning rule. Basically it is a stochastic approximation of a ***Quasi-Newton method***. In the vario-eta algorithm a weight-specific factor, β, related to each weight. For an arbitrary weight, e.g. the j^{th}, β is defined as

$$\beta^j = \frac{1}{\sqrt{\sum_{t=1}^{N}\left(\frac{\partial E_t}{\partial E^j} - \bar{E}\right)^2}}$$

Where,

$$\bar{E} = \frac{1}{N}\sum_{t=1}^{N}\frac{\partial E_t}{\partial E^j}$$

Let us assume there are p weights in the network. The search direction is determined by multiplying each component of the negative gradient with its weight-specific factor as below:

$$d_k = -\begin{pmatrix} \beta^1 & 0 & 0 \\ 0 & \cdot & 0 \\ 0 & 0 & \beta^p \end{pmatrix} \cdot \nabla E$$

Above E denotes the error function and N the number of patterns. A benefit with the vario-eta rule is that weight increments η. d_k become non-static. This property implies a potentially fast learning phase.

Concerning a reasonable value of the learning rate η, there is no simple answer. The learning rate is many times determined on an ad hoc basis.

Regarding pattern selection, a stochastic procedure can be used. This simply means that the gradient ∇E^M of a subset of M of all patterns at hand are used as an approximation of the true gradient ∇E according to

$$\nabla E^M = \frac{1}{|M|} \sum_{t \in M} \nabla E_t$$

Where $|M|$ denotes the number of elements of M.

M can be composed in several ways. Once, all weights are updated, out of the remaining training pattern new subset picked up for the next iteration.

B. Cleaning

The dilemma of over fitting is deeply rooted in neural networks. One way to suppress overfitting is to assume input data not to be exact (which certainly is the case in the field of financial analysis). The total error of pattern t can be split into two components associated with the weights and the erroneous input respectively. The corrected input data, $\tilde{x}_t$, can be expressed as

$$\tilde{x}_t = x_t + \Delta x_t$$

Where x_t is the original data and Δx_t a correction vector. During training the correction vector must be updated in parallel with the weights. To this end, the output target difference i.e. the difference in output from using original and corrected input data has to be known which is only true for training data. Accordingly, the model might be optimized for training data but not for generalization data because the latter has a different noise distribution and an unknown output target difference. To work around this disadvantage the model $\tilde{x}_t$is composed according to

$$\tilde{x}_t = x_t + \Delta x_t - \delta.$$

δ is exactly one element drawn at random from $\{\Delta x_t$, i=1,…..T}.

The input modification “cleaning with noise” described above helps the network to concentrate on broader structures in data. To some extent the model is prevented from establishing false causalities.

C. Stopping Criteria

For how many ***epochs*** (an epoch is completed when all training patterns have been read in exactly one) should a network be trained? Mainly two prototypes exist, late and early stopping. Late stopping means the network is trained until a minimum error on the training set is reached i.e. network is overfitted.
In early stopping, the training set is split into a new training set and a validation set. Gradient descent is applied to the new training set. After each sweep through the new training set, the network is evaluated on the validation set. This technique is a simple but efficient hack to deal with the problem of overfitting.

D. Error Function

The error function or the cost function is used to measure the distance between the targets and the outputs of the network. The weights of the network are updated in the direction that makes the error function minimum. The most common error functions are the Mean Square Error ***(MSE)*** and the Mean Absolute Error ***(MAE)*** [7].

E. Drawbacks

Essentially, backpropagation is a learning algorithm that allows a network to find a state that minimizes the amount of error the network exhibits (Churchland & Sejnowski, 1994). It does this by modifying weights

connecting neurons using a ***gradient descent*** (Arbib, 2003), which allows the weights to be modified in a manner following a negative slope along the "error landscape" – a n-dimensional surface representing all possible errors within the network.

Foremost, backpropagation does not guarantee to find the global network minimum. While it does minimize error, there is a chance the weights will be changed to fit a local minimum in the error landscape, but the network will not be optimized

Second, the ***convergence*** obtained from backpropagation is very slow and not guaranteed.

Third, backpropagation learning requires input scaling and normalization.

VI. CONCLUSION

In this paper, we tried to sum up the application of Artificial Neural Networks (ANN) for predicting stock market. ANN have shown to be an effective, general purpose approach for pattern recognition, classification, clustering and especially time series prediction with a great degree of accuracy. Nevertheless, their performance is not always satisfactory. Backpropagation algorithm is the best algorithm to be used in Feedforward Neural network because it reduces an error between the actual output and desired output in a ***gradient descent manner.***

VII. REFERENCES

[1] Hossein Abdoh Tabrizi, Hossein Panahian," Stock Price Prediction by Artificial Neural Networks: A Study of Tehran's Stock Exchange (T.S.E)".

[2] Zabir Haider Khan, Tasnim, Md. Akter Hussain, "Price Prediction of Share Market using Artificial Neural Network (ANN)", International Journal of Computer Applications (0975 – 8887) Vol.22 No.2, (2011)

[3] Jibendu Kumar Mantri, Dr. P. Gahan and B.B.Nayak,"Artificial Neural Networks-an application to stock market volatility". International journal of Engineering Science and Technology. Vol2, 2010.

[4] Wojciech Gryc," Neural Network Predictions of Stock Price Fluctuations".

[5] Karl Nygren (2004),"Stock prediction- A Neural Netwok Approach". Ph.D thesis.

[6] Sneha Soni,"Application of ANNs in Stock Market Prediction: A Survey". IJCSET. Vol.2 Issue 3

[7] G Dutta, Neeraj Mohan, Pankaj Jha, and Laha, "Artificial Neural Network Models for Forecasting Stock Price Index in Bombay Stock Exchange".

[8] Birgul Egeli, Meltem, Bertan Badur,"Stock market prediction Using Artificial Neural Network"

[9] Dogac Senol (2008)," Prediction of Stock Price Direction by Artificial Neural Network Approach"

[10] Efstathios Kalyvas ," Using Neural Networks and Genetic Algorithms to Predict Stock Market return". Ph.D thesis. 2001.

[11] Khoa, Sakakibara, and Nishikawa," Stock Price Forecasting using Back Propagation Neural Networks with Time and Profit Based Adjusted Weight Factors", SICE-ICASE, 2006. International Joint Conference. 2006.

[12] Mizuno, Kosaka , M., Yajima , H and Komoda, Application of Neural Network to Technical Analysis of Stock Market Prediction, Studies in Information and Control , vol.7, no.3, pp.111-120. 1998.

[13] Z.Tang, P.A.Fishwick, "Backpropagation neural nets as models for time series forecasting," ORSA journal on computing, vol.5, No. 4, pp 374-384, 1993.