PROJECT REPORT
on

STOCK MARKET PREDICTION USING AN ENSEMBLE OF NEURAL NETWORKS

submitted by

Name	Identity No.	SAP No.
1) Bhagya Parekh		60002100061
2) Naineel Shah		60002100082
3) Chintan Shah		60002100086

under the guidance of

Prof. Vishakha Kelkar

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING
in
ELECTRONICS AND TELECOMMUNICATION ENGINEERING
at

Shri Vile Parle Kelavani Mandal's

Dwarkadas J. Sanghvi College of Engineering

Plot no. U-15, JVPD Scheme, Bhaktivedanta Swami Marg,
Vile Parle (W), Mumbai – 400 056

YEAR 2013 – 2014

Shri Vile Parle Kelavani Mandal's

Dwarkadas J Sanghvi College of Engineering

Plot No. U – 15, JVPD Scheme, Bhaktivedanta Swami Marg,
Vile Parle (W), Mumbai – 400 056

Department of Electronics and Telecommunication Engineering

This is to certify that the Project Report Stage – II entitled

"Stock market Prediction using an Ensemble of Neural Networks"

Submitted by:

1. Bhagya Parekh
2. Naineel Shah
3. Chintan Shah

Students of Electronics and Telecommunication Engineering have successfully completed their Project Stage – II required for the partial fulfillment of SEM VIII as per the norms prescribed by the University of Mumbai during the First half of the year 2014. The project report has been assessed and found to be satisfactory.

_____	_____
Internal Guide	**External Guide**
_____	_____
Head of Department	**Principal**
_____	_____
Internal Examiner	**External Examiner**

Table of Contents

CHAPTER 1. INTRODUCTION

1.1 Neural Networks

1.1.1 What is a Neural Network?

n Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of ANNs as well.

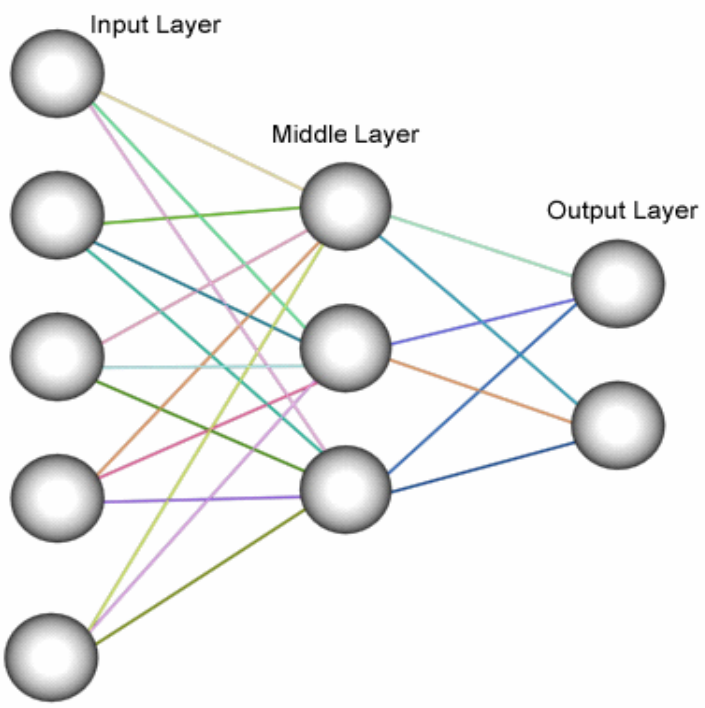


Fig 1-Neural Network

1.1.2 Historical background

Neural network simulations appear to be a recent development. However, this field was established before the advent of computers, and has survived at least one major setback and several eras.

Many important advances have been boosted by the use of inexpensive computer emulations. During this period when funding and professional support was minimal, important advances were made by relatively few researchers. These pioneers were able to develop convincing technology which surpassed the limitations identified by Minsky and Papert. Minsky and Papert, published a book (in 1969) in which they summed up a general feeling of frustration (against neural networks) among researchers, and was thus accepted by most without further analysis. Currently, the neural network field enjoys a resurgence of interest and a corresponding increase in funding. The first artificial neuron was produced in 1943 by the neurophysiologist Warren McCulloch and the logician Walter Pits. But the technology available at that time did not allow them to do too much.

1.1.3 Why are neural networks used?

Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyse. This expert can then be used to provide projections given new situations of interest and answer "what if" questions. Other advantages include:

1. Adaptive learning: An ability to learn how to do tasks based on the data given for training or initial experience.
2. Self-Organisation: An ANN can create its own organisation or representation of the information it receives during learning time.
3. Real Time Operation: ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability

1.2 Ensemble of Neural Networks

1.2.1 What is an ensemble of neural network?

Ensemble of artificial neural networks is now being used widely by neural network practitioners due to its wide range of applications. Ensemble is a method in which we combine number of neural networks to enhance the output. Most of the neural networks are designed to produce linearity or non linearity in the data but there arises model wherein we are supposed to track the linearity and non linearity in the data and produce desirable outputs. In such cases ensemble neural networks play a major role as it combines various neural networks to form a single model. In our model for prediction we have used one Back propagation network, one linear regression network and two Radial basis function. The purpose for using two radial basis function will be understood when we analyze the output tables. There are many different ensemble techniques but have used weighted average technique.

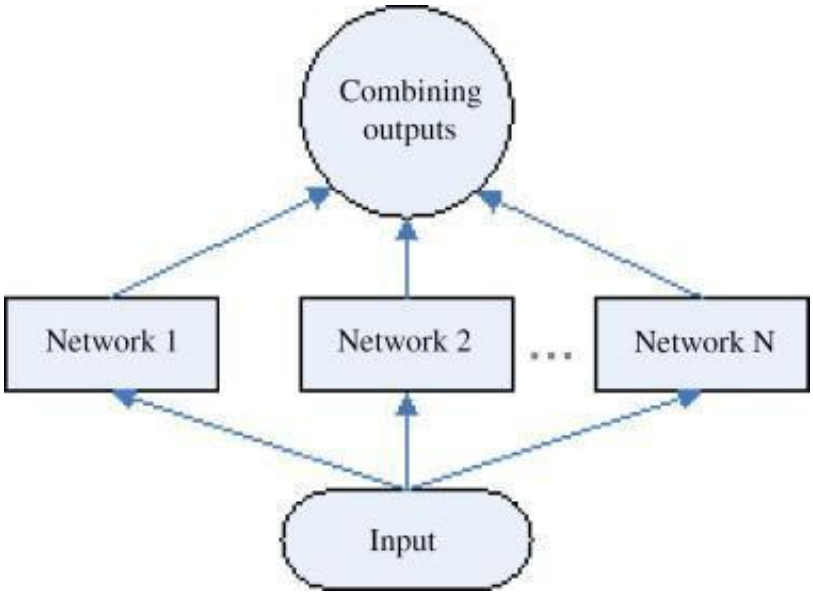


Fig 2- A classifier ensemble of neural networks.

1.2.2 Ensemble learning methods

Bootstrap aggregating (bagging)

Bootstrap aggregating, often abbreviated as bagging, involves having each model in the ensemble vote with equal weight. In order to promote model variance, bagging trains each model in the ensemble using a randomly drawn subset of the training set. As an example, the random forest algorithm combines random decision trees with bagging to achieve very high classification accuracy.

Boosting

Boosting involves incrementally building an ensemble by training each new model instance to emphasize the training instances that previous models mis-classified. In some cases, boosting has been shown to yield better accuracy than bagging, but it also tends to be more likely to over-fit the training data. By far, the most common implementation of Boosting is Adaboost, although some newer algorithms are reported to achieve better results.

1.3 Financial Forecasting

1.3.1 What is financial forecasting?

Forecasting is the process of making projections about future performance based on existing historic data. An accurate forecast aids in decision-making and planning for the future. Forecasts empower people to modify current variables in the present to predict the future to result in a favorable scenario.

The selection and implementation of a proper forecasting methodology has always been an important planning and control issue for most firms and agencies. The organizational and financial stability of an organization depends on the accuracy of the forecast since such information will most likely be used to make key decisions in the areas of human resources, purchasing, marketing, advertising and capital financing.

When a variable is to be predicted, the difficulty of forecasting depends on various factors. Most importantly, the historic pattern of the variable and the underlying input factors that affect a variable may increase the complexity of the forecasting task. A volatile historic pattern may suggest that the factor to be forecast has a number of underlying factors, the effects of each dynamically changing over time. Some of these contributing factors may not be identifiable and hence require an amount of expert knowledge gained over time to be built into the forecasting model. Forecasting a financial scenario such as the stock market would be a very important step when assessing the prudence of an investment. This step is very difficult due to complexity and presence of a multitude of factors that may affect the value of a certain financial instrument[3].

1.3.2 Forecasting Problems

The following are the major challenges that are identified in the field of forecasting:

1. In certain cases, it may be difficult to ascertain a future scenario during forecasting. Regardless of the techniques that may be used, it is always assumed that there will be a variable measure of uncertainty.

2. Forecasting variables for which there are no existing paradigms or historical data is often prone to errors, primarily due to the lack of ability to understand underlying factors which could affect the forecast.
3. Selection and implementation of a proper forecasting methodology is very important due to the fact that different forecasting problems must to be addressed using different tailor made models to suit each.

1.3.3 Prediction Methods

- **Fundamental Analysis**

Fundamental analysis is the physical study of a company in terms of its product sales, manpower, 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. But, this analysis is not suitable for our study because the data it uses to determine the intrinsic value of an asset does not change on daily basis and therefore is not suitable for short-term basis. However, this analysis is suitable for predicting the share market only in long-term basis.

- **Technical Analysis**

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. Price charts are used to recognize trends. These trends are understood by supply and demand issues that often have cyclical or some sort of noticeable patterns. To understand a company and its profitability through its share prices in the market, some parameters can guide an investor towards making a careful decision. These parameters are termed Indicators and Oscillators. 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. This analysis can be used to predict the market price on daily basis but we will not use this approach because of its subjective nature[4].

1.4 Stock market

1.4.1 What is a stock market?

A stock market or equity market is the aggregation of buyers and sellers (a loose network of economic transactions, not a physical facility or discrete entity) of stocks (shares); these are securities listed on a stock exchange as well as those only traded privately.

1.4.2 Functions

The stock market is one of the most important sources for companies to raise money. This allows businesses to be publicly traded, or raise additional financial capital for expansion by selling shares of ownership of the company in a public market. The liquidity that an exchange affords the investors gives them the ability to quickly and easily sell securities. This is an attractive feature of investing in stocks, compared to other less liquid investments. Some companies actively increase liquidity by trading in their own shares.

History has shown that the price of shares and other assets is an important part of the dynamics of economic activity, and can influence or be an indicator of social mood. An economy where the stock market is on the rise is considered to be an up-and-coming economy. In fact, the stock market is often considered the primary indicator of a country's economic strength and development.

Rising share prices, for instance, tend to be associated with increased business investment and vice versa. Share prices also affect the wealth of households and their consumption. Therefore, central banks tend to keep an eye on the control and behavior of the stock market and, in general, on the smooth operation of financial system functions. Financial stability is the raison d'être of central banks.

Exchanges also act as the clearinghouse for each transaction, meaning that they collect and deliver the shares, and guarantee payment to the seller of a security. This eliminates the risk to an individual buyer or seller that the counterparty could default on the transaction.

1.5 Why use neural networks in stock market prediction?

- NN can approximate any LINEAR and NONLINEAR function to any desired degree of accuracy
  - Can learn linear time series patterns
  - Can learn nonlinear time series patterns
  - Can extrapolate linear & nonlinear patterns
- NN are nonparametric
- NN model (learn) linear and nonlinear process directly from data
- Approximate underlying data generating process

CHAPTER 2. THEORY

2.1 Stock Market Indicators

2.1.1 INTRODUCTION

An indicator may be defined as a series of data points that are derived from the price data of a security by applying a basic formula. Price data is a combination of open, close, high, or low over a period of time. For example, the average of 3 closing prices is one data point ((41+43+43)/3=42.33). However, one data point does not offer much information and does not make an indicator. A series of data points over a period of time is required to create valid reference points to enable analysis. By creating a time series of data points, a comparison can be made between present and past levels. An indicator offers a different perspective from which to analyze the price action.

The function of indicators may be classified into three broad categories: to alert, to confirm, and to predict. An indicator can act as an alert to study price action a little more closely. If momentum is waning, it may be a signal to watch for a break of support. Or, if there is a large positive divergence building, it may serve as an alert to watch for a resistance break-out.

Indicators can be used to confirm other technical analysis tools. If there is a break-out on the price chart, a corresponding moving average crossover could serve to confirm the break-out. Or, if a stock breaks support, a corresponding low in the OnBalance-Volume (OBV) could serve to confirm the weakness.

2.1.2 CLASSIFICATION OF INDICATORS

Indicators are mathematical/statistical functions that are applied over stock properties such as close, high, low and volume. These indicators are broadly classified into the following important categories:

- Market Momentum Indicators
- Market Volatility Indicators
- Market Trend Indicators
- Broad Market Indicators
- General Momentum Indicators

Analysts generally use at least one indicator from each of these categories for their forecasts. The indicator is generally chosen by evaluating the accuracy of the model. Most often many indicators are overlooked and a good model may never be discovered for that particular stock. A common misconception with those new to Neural Networks is that the more inputs you give a Neural Network, the more information it has, so the resulting model will be better. If the input data is not relevant to the desired output, the network will have a more difficult time learning the associations between the relevant inputs and the desired output. The first step is to use a set of indicators commonly used in traditional technical analysis. Another approach is to use the Correlation Analysis.

Correlation Analysis is useful for searching for linear relationships between the desired output and a set of proposed inputs. Specifically, correlation analysis determines if an input and the desired output move in the same direction by similar amounts. Using inputs with high (positive or negative) correlation values with the output often produce good models[6].

2.1.3 BASIC INDICATORS

2.1.3.1 Simple Moving average:

A simple moving average is formed by computing the average price of a security over a specific number of periods**.** Most moving averages are based on closing prices. A 5-day simple moving average is the five day sum of closing prices divided by five. As its name implies, a moving average is an average that moves. Old data is dropped as new data comes available. This causes the average to move along the time scale. Below is an example of a 5-day moving average evolving over three days.

Calculation

Daily Closing Prices: 11,12,13,14,15,16,17

First day of 5-day SMA: (11 + 12 + 13 + 14 + 15) / 5 = 13

Second day of 5-day SMA: (12 + 13 + 14 + 15 + 16) / 5 = 14

Third day of 5-day SMA: (13 + 14 + 15 + 16 + 17) / 5 = 15

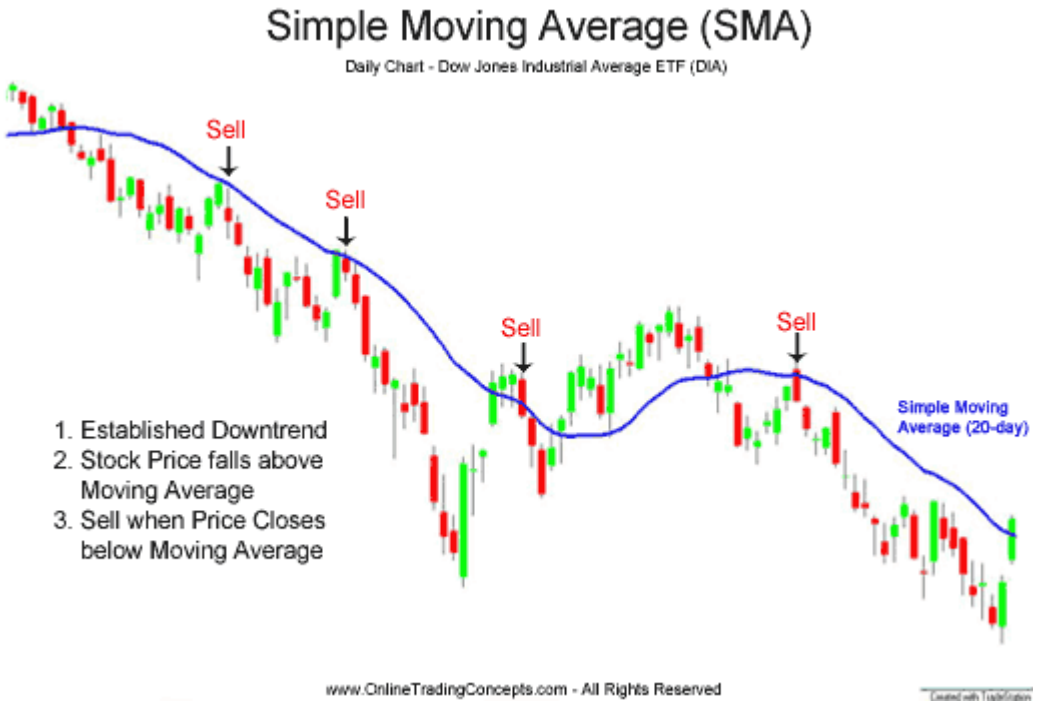


Fig 3

2.1.3.2 Exponential Moving Average:

Exponential moving averages reduce the lag by applying more weight to recent prices. The weighting applied to the most recent price depends on the number of periods in the moving average. There are three steps to calculating an exponential moving average. First, calculate the simple moving average. An exponential moving average (EMA) has to start somewhere so a simple moving average is used as the previous period's EMA in the first calculation. Second, calculate the weighting multiplier. Third, calculate the exponential moving average. The formula below is for a 10-day EMA.

Calculation

SMA: 10 period sum / 10

Multiplier: (2 / (Time periods + 1)) = (2 / (10 + 1)) = 0.1818 (18.18%)

EMA: {Close - EMA(previous day)} x multiplier + EMA(previous day).

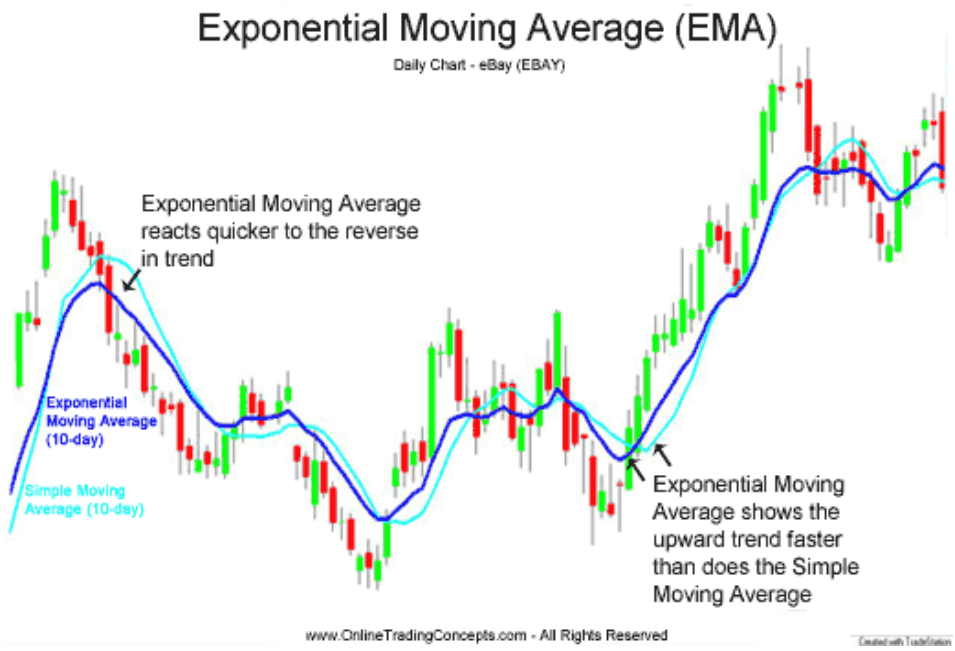


Fig 5

2.1.3.3 Relative Strength Index:

Relative Strength Index is a measure of the strength that is intrinsic in a field and is calculated using the amount of upward and downward price changes over a given period of time. It has a range of 0 to 100 with values typically remaining between 30 and 70. Overbought conditions are indicated by higher values of the Relative Strength Index while lower values indicate oversold conditions. The formula for computing the Relative Strength Index is as follows.

RSI = 100- [100 / (1+RS)]

Where RSI = Relative Strength Index.

RS = Average of x days' up closes Average of x days' down closes[8].

Fig 6

2.1.3.4 Stochastic Oscillator (SO)

The stochastic oscillator may be defined as a measure of the difference between the current closing price of a security and its lowest low price, relative to its highest high price for a given period of time. The formula for this computation is as follows.

$$\%K = \frac{C - LX}{HX - LX} \times 100$$

Where,

- C is Recent closing price
- LX is Lowest low price during the period

- HX is Highest high price during the period
- %K is Stochastic Oscillator

The value is a percent rating for the closing price, relative to the trading range between its recent highest and lowest prices. A value of zero indicates that the security had a closing price at its lowest recent low. A value of 100 indicates it that the security had a closing price at its highest recent high. The value is often smoothed using a slowing period to eliminate noise in the trend graph[9].

What is the difference between fast and slow stochastics in technical analysis?

The main difference between fast and slow stochastics is summed up in one word: sensitivity. The fast stochastic is more sensitive than the slow stochastic to changes in the price of the underlying security and will likely result in many transaction signals.

Fig 7

2.1.3.5 Moving Average Convergence/divergence (MACD)

The MACD is the difference between the short and the long term moving averages for a field. The MACD is generally a specific instance of a Value Oscillator and is mostly used on the closing price of a security to detect price trends. When the MACD is on an increasing trend, prices are trending higher. If the MACD is on a decreasing trend, prices are trending lower.

$$EMA = [\alpha \times Today's\ Close] + [(1-\alpha) \times Yesterday's\ EMA]$$
$$MACD = [0.075\ EMA\ of\ Closing\ Prices] - [0.15\ EMA\ of\ Closing\ Prices]$$
$$Signal\ Line = 0.20\ EMA\ of\ MACD$$

Where,

- EMA is Exponential Moving Average
- α is the period multiplier

Moving Average Convergence/divergence indicator is traditionally traded against a 9-day exponential average of its value, called its signal line. When the Moving Average Convergence/divergence indicator increases above its signal line, a buy signal is generated. When the Moving Average Convergence/divergence indicator decreases below its signal line, a sell signal is generated[10].

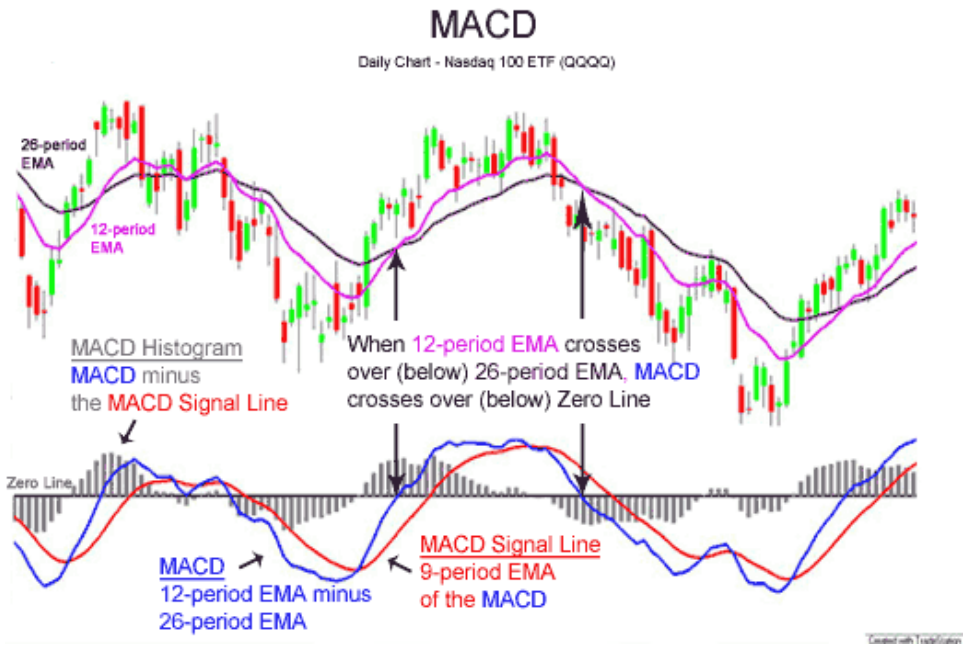


Fig 8

2.1.3.6 Chaikin Oscillator

It is an oscillator which measures the accumulation distribution line of the MACD. The Chaikin Oscillator is calculated by subtracting a 10-day EMA from a 3-day EMA of the accumulation distribution line, and outlines the momentum implied by the accumulation distribution line.
This indicator is named after its creator, Marc Chaikin. The goal of the Chaikin line is to recognize moving momentum levels within the MACD, more specifically the accumulation distribution line, in hopes of being able to act on said momentum. By being able to recognize momentum, technical traders hope that the momentum is the first step in the development of a trend that they can capitalize upon.

Calculation

1. Money Flow Multiplier = [(Close - Low) - (High - Close)] /(High - Low)

2. Money Flow Volume = Money Flow Multiplier x Volume for the Period

3. ADL = Previous ADL + Current Period's Money Flow Volume

4.Chaikin Oscillator = (3-day EMA of ADL) - (10-day EMA of ADL)

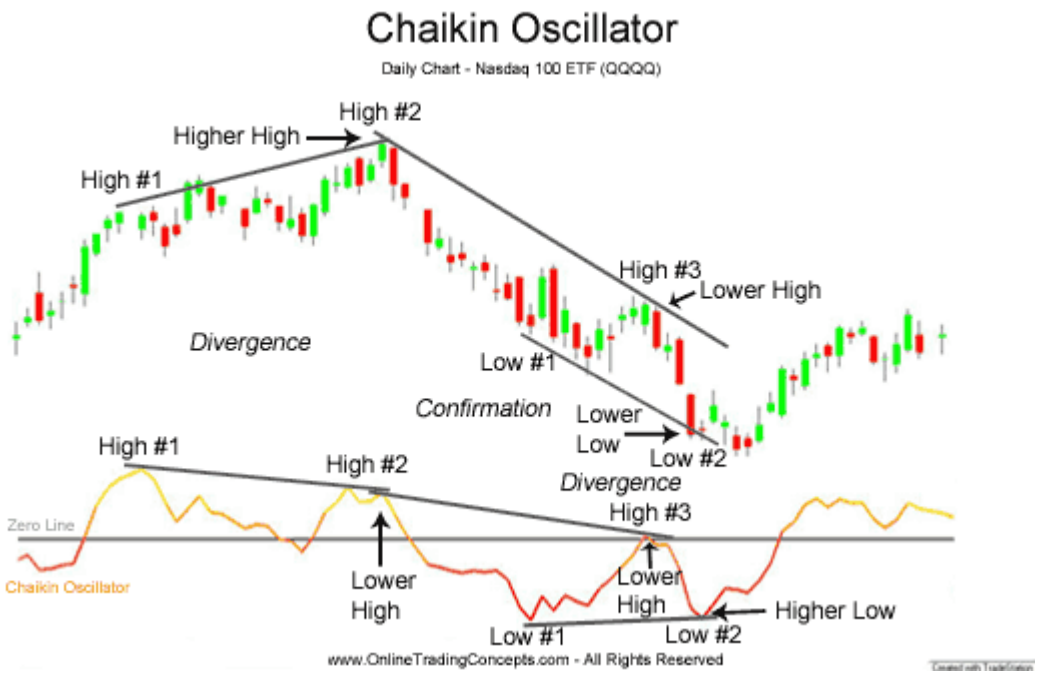


Fig 9

2.1.3.7 Bollinger Bands

Developed by John Bollinger, Bollinger Bands are volatility bands placed above and below a moving average. Volatility is based on the standard deviation, which changes as volatility increases and decreases. The bands automatically widen when volatility increases and narrow when volatility decreases. This dynamic nature of Bollinger Bands also means they can be used on different securities with the standard settings. For signals, Bollinger Bands can be used to identify M-Tops and W-Bottoms or to determine the strength of the trend.

Investopedia explains 'Bollinger Band'

Because standard deviation is a measure of volatility, Bollinger Bands® adjust themselves to the market conditions. When the markets become more volatile, the bands widen (move further away from the average), and during less volatile periods, the bands contract (move closer to the average). The tightening of the bands is often used by technical traders as an early indication that the volatility is about to increase sharply.

This is one of the most popular technical analysis techniques. The closer the prices move to the

upper band, the more overbought the market, and the closer the prices move to the lower band, the more oversold the market[7].

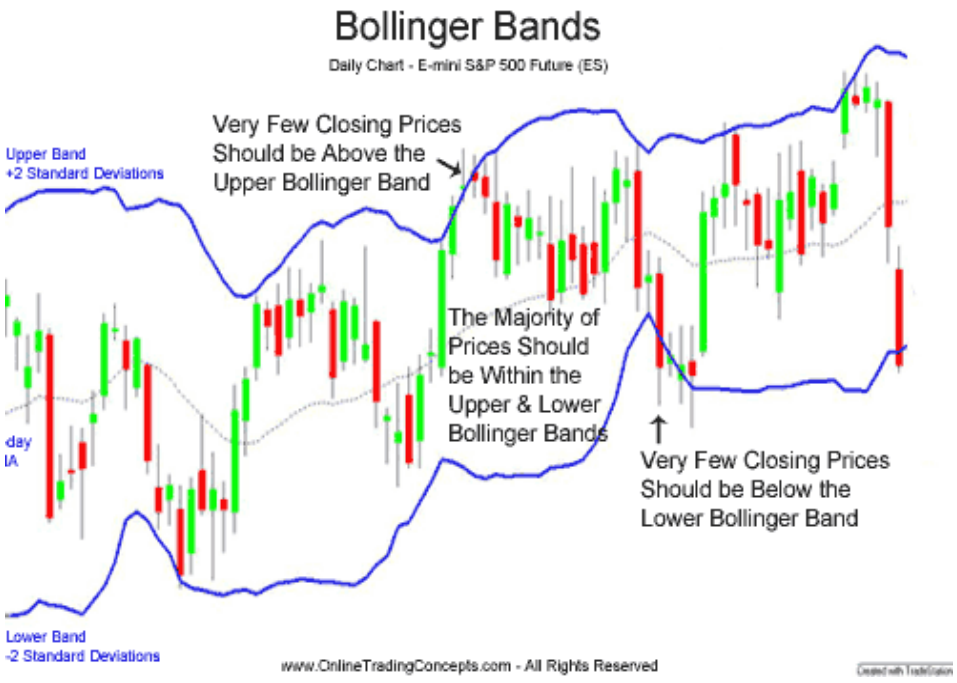


Fig 10

2.2 Neural Networks

2.2.1 Basics

We are going to discuss the basic terminologies and concepts of neural networks that will help us in understanding the forthcoming sections easily.

H. Demuth states that "Neural networks are composed of simple elements operating in Parallel." These elements are inspired by biological nervous systems. As in nature, the network function is determined largely by the connections between elements. You can train a neural network to perform a particular function by adjusting the values of the connections (weights) between elements."

Each such 'single element' is termed as a neuron.

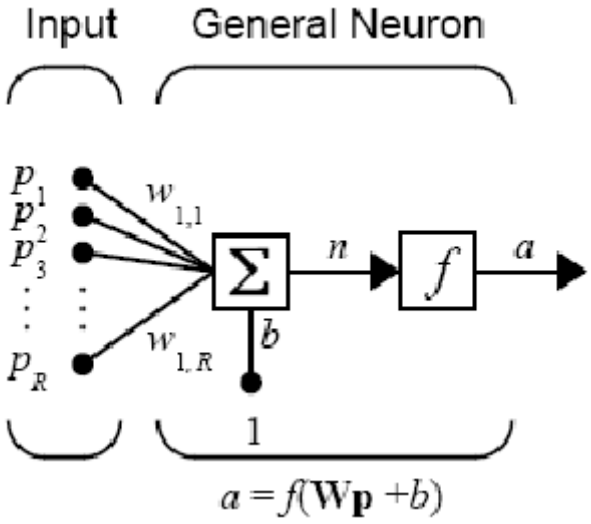


Fig 11

In figure :

f: transfer function of the neuron

R: number of inputs

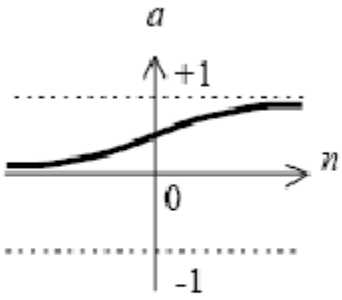
W: weight

b: bias

The output of the neuron is compared with the ideal output and changes are made to the weight and in such a way that the ideal output would be achieved.

The neuron receives inputs from one or more inputs. The output of this neuron depends upon the 'activation function' or 'transfer function' of the neuron.

The basic transfer functions that we will be talking about in the coming chapters are the sigmoid (or the log-sigmoid transfer function) and the tan based (tansigmoid) transfer function. They are described as follows:



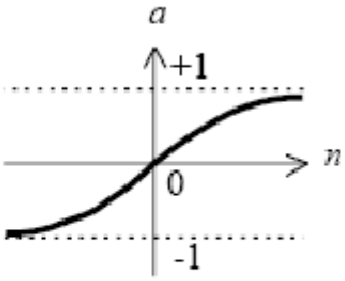
$a = logsig(n)$

Log-Sigmoid Transfer Function

Fig 12

Log sigmoid transfer function:

This network can receive inputs from negative infinity to positive infinity and always generates an output between 0 and 1



$a = tansig(n)$

Tan-Sigmoid Transfer Function

Fig 13

This transfer function receives inputs from negative infinity to positive infinity and gives an output between -1 and 1.

The arrangement of these neurons leads to different types of neural networks. Broadly they can be classified as static and dynamic. Static networks do not have any feedback loops (outputs of a neuron fed back to some previous neuron) or taps (delay lines that feed the network with past values of inputs). Dynamic networks may have one of these two. Dynamic networks are preferable for time series as they have memory in the form of loops or delay lines.

The tapped delay line (TDL):

The taps or the delay line is used to feed the network with the past values of inputs. In figure 4 we see that input enters from the left and goes through N-1 delay elements to generate a vector of N outputs.

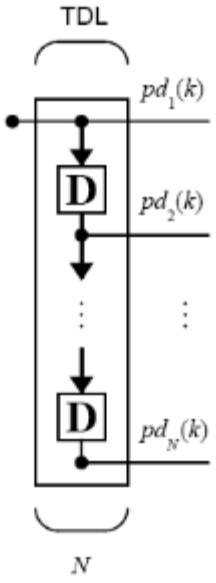


Fig 14

We will now discuss in brief about the NARX network that we propose to use in this research. The NARX network uses the past values of the actual time series to be predicted and past values of other inputs (like currencies of other nations and technical indicators in our case) to make predictions about the future value of the target series. These networks are again classified as series and parallel architecture.

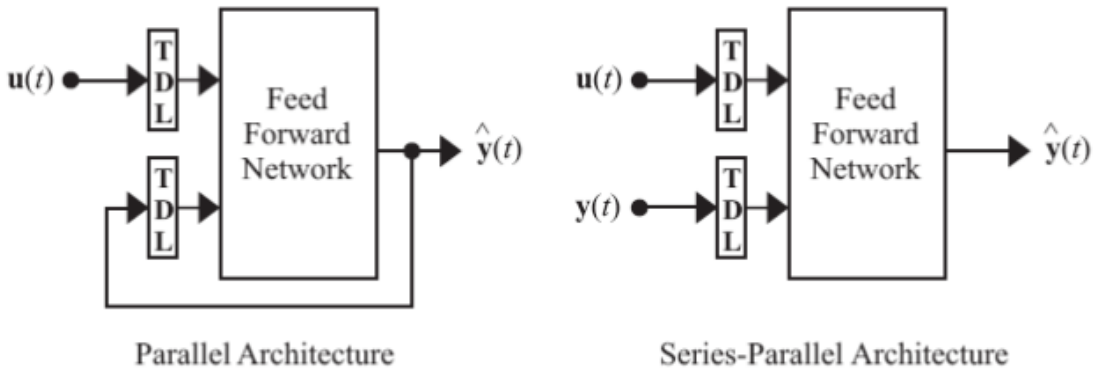


Fig 15

In figure, u(t) represents the past exogenous values (currencies of other nations and technical indicators in our case) y(t) represents the past values of the actual series to be predicted. .(t) indicates the predicted values. If past values of actual series are not being recorded, they will not be available to the system. In such situations the networks uses its past predicted values. In our case we will have the actual past values; hence we prefer to use them instead of our predictions.
Thus we are able to base the model on actual values which are more reliable than our predictions.

2.2.2 Selection of neural network type

Neural networks can in general be divided into two categories – static and dynamic. Static networks have no feedback elements and no delays. The output is calculated directly from the current inputs. Such networks assume that the data is concurrent and no sense of time can be encoded. These networks can thus lead to instantaneous behavior.

Dynamic networks may be difficult to train but are more powerful than static networks. As they have memory in form of delays or recurrent loops, they can be trained to learn sequential or time varying patterns. This makes them networks of choice for various applications like financial predictions, channel equalization, sorting, speech recognition, fault detection etc.
Since we are dealing with a time series it is necessary to use dynamic networks.
Dynamic networks can be of two types ones with feed forward connections and taps and those with feedback or recurrent networks.

At this stage two breeds of networks were considered the NARX (Nonlinear Autoregressive Neural Network) and recurrent networks. NARX networks use taps to set up delays across the inputs and also incorporates the past values of the output. Recurrent networks have loops within intermediate layers and incorporate memory via these loops.

Both the networks have been employed in dynamic applications .The difficulties and the time required in training the recurrent networks is a known problem. To evaluate the resources requires by each of these networks, a batch of 40 Networks (1 neuron to 40 neurons) was trained under the same conditions for same datasets and training parameters (MATLAB® Neural Network Toolbox). It was observed that recurrent networks took almost 17 times more time to complete training and simulations as compared to the NARX batch.

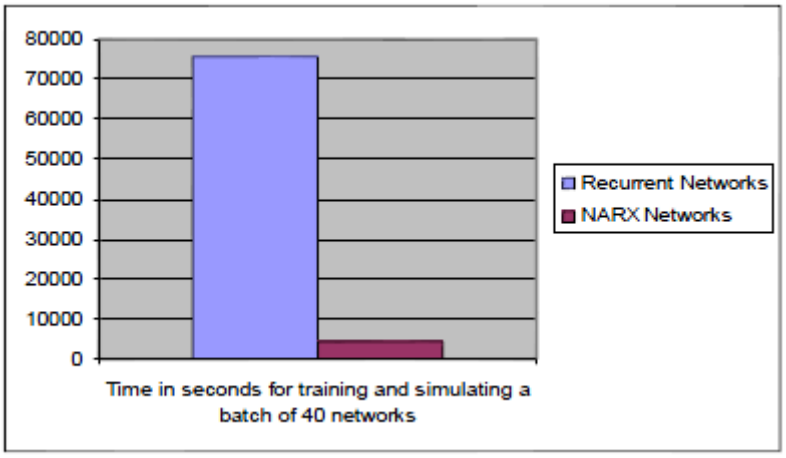


Fig 16

Our approach is that of simulating several networks, we will have to simulate several networks ranging up to a few hundred neurons for each time frame. This ratio of required time would further be aggravated when we would train networks with hundreds of neurons.

Dunis and Williams who have evaluated neural network application to financial predictions have stated the problems in using recurrent networks. They state that there is no theoretical proof or reason why outputs of layers should looped back to the layers additionally the difference in the performance of recurrent and simpler models is marginal. The neuron structure of the NARX is similar to simple feed forward network making it a simple model.

Our limited computational resources do not allow us to train and simulate such networks. However for applications where these gains outweigh the cost of computation, recurrent networks should be tried out.

Due to the above mentioned finding and the limited computing resources available for this research it was decided that NARX networks will be used for our prediction system.

The basic NARX network is used for multi step predictions. It is assumed that actual past values of target are not available and the predictions themselves are fed back to the network. Since we will have access to the actual past values we will provide those values instead of our past predictions. This helps the system train on actual values rather than predictions. This is achieved by using the series-parallel version of the NARX network which is described in following section.

Thus a series parallel NARX dynamic network will be used as a basis of our system.

2.2.3 Network architecture to be considered

On the basis of our earlier findings we have decided to use NARX networks for our application. Now we determine the other parameters of the network architecture.

The NARX networks will have a linear input layer of neurons (default by MATLAB®) for the hidden and the output layers we will use the Tansig neurons.

Tansig neurons and sigmoid neurons have been extensively used in most of the neural network applications. Both of them are similar in behavior and have a similar looking transfer function except for the range of outputs that they can generate. A sigmoid layer has a range form 0 to 1 whereas the range for Tansig transfer function ranges from -1 to 1. Klimasauskas suggests that sigmoid neurons be preferred to determine average behavior and Tan based layers to find deviations from normal. The application at hand also prompts us to use the Tansig layer. Additionally our development environment the MATLAB® Neural Network Toolbox also recommends Tansig layers for pattern recognition problems and provides it as the default layer. The Tansig activation function has been described before.

2.2.4 Number of layers

We have a linear layer of linear neurons at the input; the number of neurons in this layer will be equal to the number of inputs that we have provided to the networks. Using linear neurons at the input is a standard practice and is used merely as an interface between the inputs and the hidden layers. In fact MATLAB® Neural Network Toolbox does not count it as an independent layer.

There is no certain figure for the number of hidden layers to be used. Cybenko Hornik et al. show how a single layer of hidden neurons is capable of adapting to complex functions. Survey papers on this field also reveal how a single hidden layer of neurons is the most preferred option. Using additional layers adds up complexities to the model and increases the time required for training and simulation.

The framework described earlier has capabilities to generate, simulate networks with multiple hidden layers. It can be seen from the code that the framework can generate the number of layers specified by the user, and use user specified number of layers for each layer. However the limited processing resources available did not permit us to perform tests with several layers, hence we too decided to use single layer of hidden neurons for this research.

The inputs were mapped in the input range of -1 to 1 by our massager. The outputs were also to be mapped within this range. Naturally Tansig was the output neuron of choice since it maps the inputs to this range. A single output neuron was thus used at the output.

2.2.5 Number of taps and hidden layers

The tapped delay line of the NARX network allows passing of past values to the network. They make the data sequential unlike the original concurrent dataset.

Thus these taps set up a sense of time and correlation of past values.

The numbers of neurons are supposed to be related to the complexity of the application at hand as each neuron in the hidden layer contributes weights and flexibility to the network. There is no standard method to determine the number of neurons to be used and several thumb rules are used.

Regarding the numerosity of neurons to be used in a network different thumb rules have different opinion. Klimasauskas suggests that there at least be 5 examples per weight. For our application

of using 10 inputs and upto 4 taps with 240 training points – we have 240 examples hence we will be allowed upto 48 total weights and biases. The number of weights and biases in our structure can be calculated as

Number of weights and biases = (I + 1) * H + (H + 1)* O

Where,

I = number of inputs (10 to 40)

H = Number of hidden neurons (to be determined)

O= outputs (1 i.e. daily prediction)

This would limit us to a range of 1 to 4 neurons.

The rule mentioned by Gallo would suggest using twice the number as inputs i.e. upto 80 neurons.

Bayesian regularization is a method that uses the LM algorithm and tries to optimize the error and number of neurons both. This method is not optimized to minimize error alone, also is more suited to function approximation applications.

Different researchers have used 5 or 20 neurons in their applications.

Some have used extremely large number of neurons in their application Some researchers have used some kind of sensitivity analysis to determine the number of neurons. These researchers use errors or analyze behavior on the test set to settle for a fixed architecture. We are going to follow an adaptive approach in determining the number of taps and neurons in this research[8].

2.3 NARX Model

2.3.1 Introduction

. The nonlinear autoregressive network with exogenous inputs (NARX) is a recurrent dynamic network, with feedback connections enclosing several layers of the network. The NARX model is based on the linear ARX model, which is commonly used in time-series modeling.

The defining equation for the NARX model is

$$y(t) = f(y(t-1), y(t-2), \ldots, y(t-n_y), u(t-1), u(t-2), \ldots, u(t-n_u))$$

where the next value of the dependent output signal $y(t)$ is regressed on previous values of the output signal and previous values of an independent (exogenous) input signal. You can implement the NARX model by using a feedforward neural network to approximate the function f. A diagram of the resulting network is shown below, where a two-layer feedforward network is used for the approximation. This implementation also allows for a vector ARX model, where the input and output can be multidimensional.

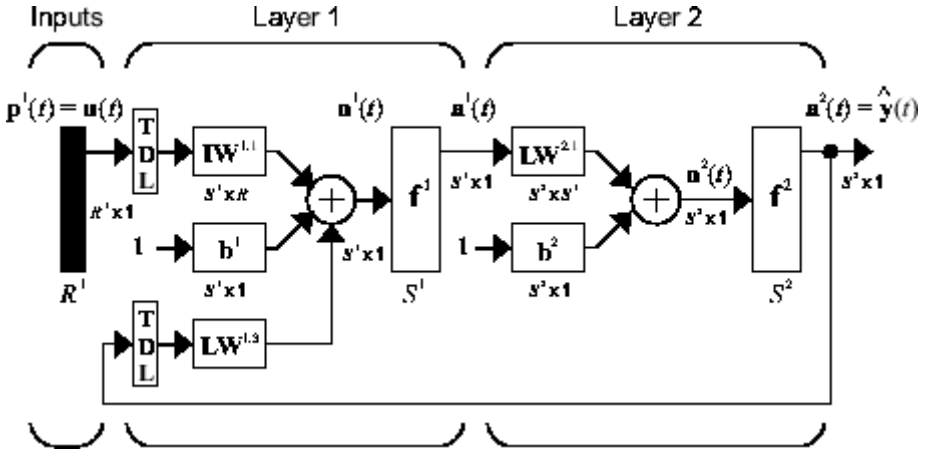


Fig 17

There are many applications for the NARX network. It can be used as a predictor, to predict the next value of the input signal. It can also be used for nonlinear filtering, in which the target

output is a noise-free version of the input signal. The use of the NARX network is shown in another important application, the modeling of nonlinear dynamic systems.

Before showing the training of the NARX network, an important configuration that is useful in training needs explanation. You can consider the output of the NARX network to be an estimate of the output of some nonlinear dynamic system that you are trying to model. The output is fed back to the input of the feedforward neural network as part of the standard NARX architecture, as shown in the left figure below. Because the true output is available during the training of the network, you could create a series-parallel architecture in which the true output is used instead of feeding back the estimated output, as shown in the right figure below. This has two advantages. The first is that the input to the feedforward network is more accurate. The second is that the resulting network has a purely feedforward architecture, and static backpropagation can be used for training.

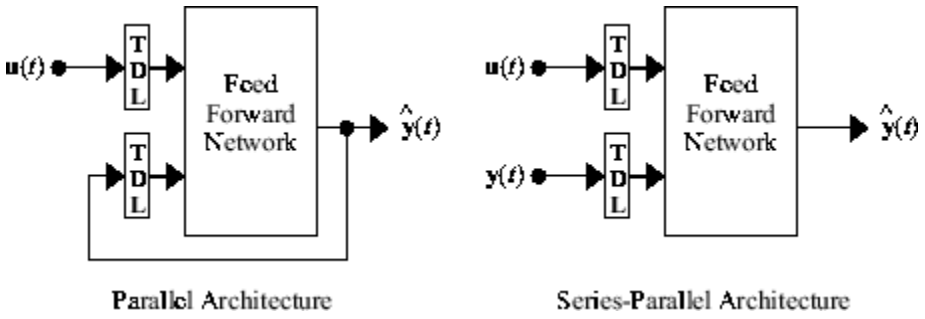


Fig 18

Series-Parallel Mode(SP):

In this case, the output's regressor is formed only by actual values of the system's output:

yb(n + 1)=fb[ysp(n); u(n)];

=fb[y(n); : : : ; y(n - dy + 1); u(n); u(n - 1); : : : ; u(n - du + 1)] :

Parallel Mode:

In this case, estimated outputs are fed back and included in the output's regressor:

yb(n + 1)=fb[yp(n); u(n)]; (4)

=fb[yb(n); : : : ; yb(n - dy + 1); u(n); u(n-1); : : : ; u(n - du + 1)]:

2.3.2 Nonlinear time series prediction using NARX Model:

The state of a deterministic dynamical system is the information necessary to determine the evolution of the system in time. In discrete time, this evolution can be described by the following system of difference equations:

$$x(n + 1) = F[x(n)] ;$$

where x(n) Є Rd is the state of the system at time step n, and F[.] is a nonlinear vector valued function. A time series is a time-ordered set of measures fx(n)g, n = 1; : : : ;N, of a scalar quantity observed at the output of the system. This observable quantity is defined in terms of the state x(n) of the underlying system as follows:

$$x(n) = h[x(n)] + \varepsilon(t) \quad (7);$$

Where h(.) is a nonlinear scalar-valued function, " is a random variable which accounts for modeling uncertainties and/or measurement noise. It is commonly assumed that "(t) is drawn from a Gaussian white noise process. It can be inferred immediately from Equation (7) that the observations fx(n)g can be seen as a projection of the multivariate state space of the system onto the one-dimensional space. Equations (6) and (7) describe together the state-space behavior of the dynamical system.

In order to perform prediction, one needs to reconstruct (estimate) as well as possible the state space of the system using the information provided by fx(n)g only. In [35], Takens has shown that, under very general conditions, the state of a deterministic dynamic system can be accurately reconstructed by a time window of finite length sliding over the observed time series as follows:

$$x1(n) = [x(n); x(n- T); : : : ; x(n - (dE - 1)T)] ;$$

where x(n) is the sample value of the time series at time n, dE is the embedding dimension and _ is the embedding delay. Equation implements the delay embedding theorem. According to this theorem, a collection of time-lagged values in a dE-dimensional vector space should provide sufficient information to reconstruct the states of an observable dynamical system. By doing this, we are indeed trying to unfold the projection back to a multivariate state space whose topological properties are equivalent to those of the state space that actually generated the observable time series, provided the embedding dimension dE is large enough.

The embedding theorem also provides a theoretical framework for nonlinear time series prediction, where the predictive relationship between the current state x1(t) and the next value of the time series is given by the following equation:

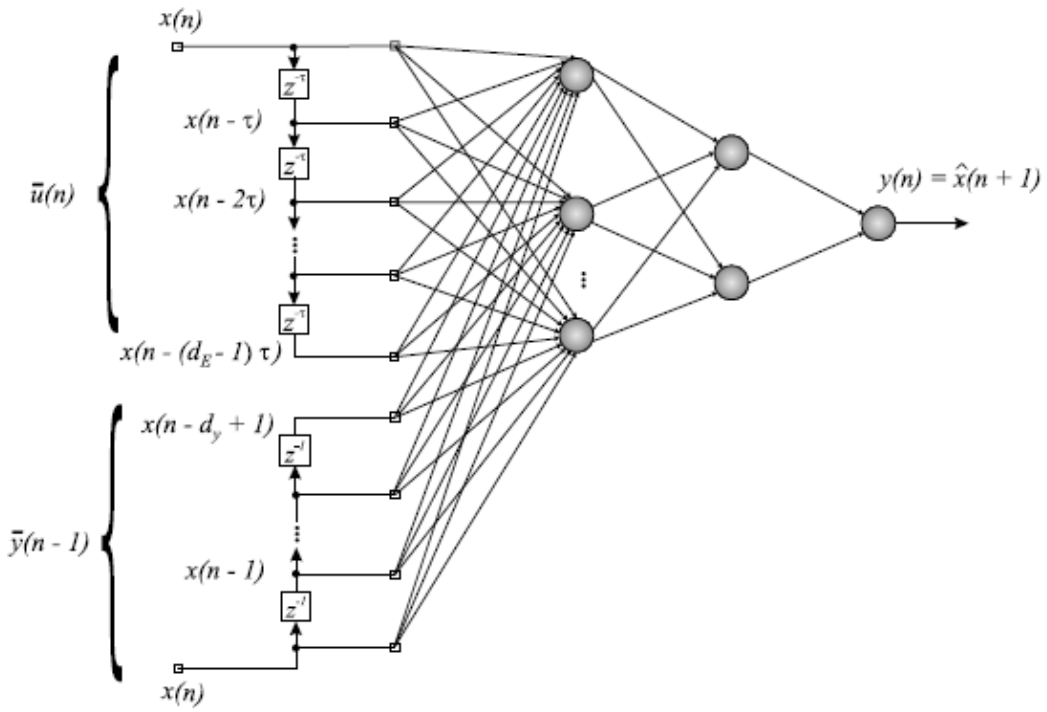
$$x(n + 1) = g[x1(n)]$$

Once the embedding dimension dE and delay T are chosen, one remaining task is to approximate the mapping function g(.). It has been shown that a feedforward neural network with enough neurons is capable of approximating any nonlinear function to an arbitrary degree of accuracy. Thus, it can provide a good approximation to the function g(.) by implementing the following mapping:

$$X(n + 1) = g[x1(n)];$$

Where X(n + 1) is an estimate of x(n + 1) and g(.) is the corresponding approximation of g(.). The estimation error, e(n + 1) = x(n + 1)- X(n + 1), is commonly used to evaluate the quality of the approximation.

If we set u(n) = x1(n) and y(n + 1) = x(n + 1) in Equation (5), then it leads to an intuitive interpretation of the nonlinear state-space reconstruction procedure as equivalent to the time series prediction problem whose the goal is to compute an estimate of x(n + 1). Thus, the only thing we have to do is to train a TDNN model. Once training is completed, the TDNN can be used for predicting the next samples of the time series.



Architecture of the NARX network during training in the SP-mode.

Fig 19

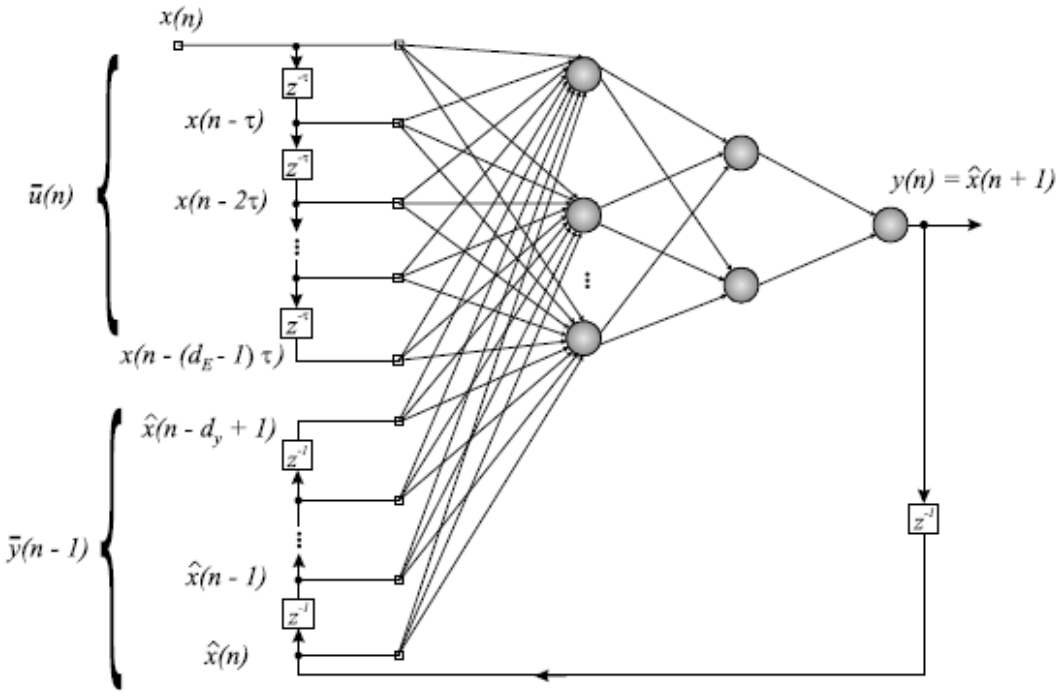
Despite the correctness of the TDNN approach, recall that it is derived from a simplified version of the NARX network by eliminating the output memory. In order to use the full computational abilities of the NARX network for nonlinear time series prediction, we repose novel definitions for its input and output regressors. Firstly, the input signal regressor, denoted by u(n), is defined by the delay embedding coordinates of Equation :

u(n) = x1(n) = [x(n); x(n - T); : : : ; x(n - (dE - 1)T)];

Where we set du = dE. In words, the input signal regressor u(n) is composed of dE actual values of the observed time series, separated from each other of _ time steps.

Secondly, since the NARX network can be trained in two different modes, the output signal regressor y(n) can be written accordingly as:

ysp(n) = [x(n); : : : ; x(n- dy + 1)];



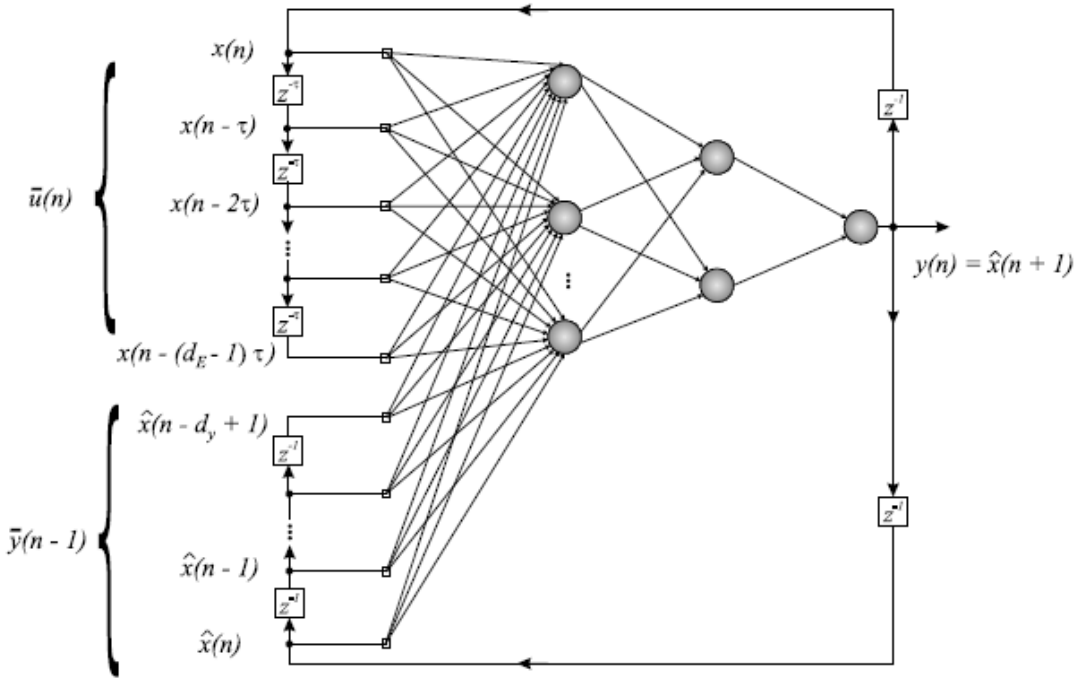
Architecture of the NARX network during training in the P-mode.

Fig 20

yp(n) = [bx(n); : : : ;bx(n - dy + 1)];

Note that the output regressor for the SP-mode shown in Equation (12) contains dy past values of the actual time series, while the output regressor for the P-mode shown in Equation (13) contains dy past values of the estimated time series. For a suitably trained network, no matter under which training mode, these outputs are estimates of previous values of x(n+1). Henceforth, NARX networks trained using the regression pairs fysp(n); x1(n)g and fyp(n); x1(n)g are denoted by NARX-SP and NARX-P networks, respectively. These NARX networks implement following predictive mappings, can be visualized in Figure:

x(n + 1)= f[ysp(n); u(n)] = f[ysp(n); x1(n)];
x(n + 1)= f[yp(n); u(n)] = f[yp(n); x1(n)];
where the nonlinear function f(.) be readily implemented through a MLP trained with plain back propagation algorithm.



Common architecture for the NARX-P and NARX-SP networks during the (prediction) phase.

Fig 21

It is worth noting that preceding Figures correspond to the different ways the NARX network is trained; that is, in SP-mode or in P-mode, respectively. During the testing phase, however, since long-term predictions are required, the predicted values should be fed back to both, the input regressor u(n) and the output regressor ysp(n) (or yp(n)), simultaneously. Thus, the resulting predictive model has two feedback loops, one for the input regressor and another for the output regressor, as illustrated in Figure above.

Thus, unlike the TDNN-based approach for the nonlinear time series prediction problem, the proposed approach makes full use of the output feedback loop. Equations are valid only for one-step-ahead prediction tasks. Again, if one is interested in multi-step-ahead or recursive prediction tasks, the estimates x should also be inserted into both regressors in a recursive fashion.

One may argue that, in addition to the parameters dE and T, the proposed approach introduces one more to be determined, namely, dy. However, this parameter can be eliminated if we recall that, as pointed out in, the delay embedding of Equation has an alternative form given by:

x2(n) , [x(n); x(n - 1); : : : ; x(n - m + 1)]

where m is an integer defined as m >T. dE. By comparing Equations (12) and (16), we find that a suitable choice is given by dy >T. dE, which also also satisfies the necessary condition dy > du. However, we have found by experimentation that a value chosen from the interval dE < dy <T. dE is sufficient for achieving a prediction performance better than those achieved by conventional neural based time series predictors, such as the TDNN and Elman architectures. Finally, the proposed approach is summarized as follows. A NARX network is defined so that its input regressor u(n) contains samples of the measured variable x(n) separated T> 0 time steps from each other, while the output regressor y(n) contains actual or estimated values of the same variable, but sampled at consecutive time steps. As training proceeds, these estimates should become more and more similar to the actual values of the time series, indicating convergence of the training process. Thus, it is interesting to note that the input regressor supplies medium- to long-term information about the dynamical behavior of the time series, since the delay T is usually larger than unity, while the output regressor, once the network has converged, supplies short-term information about the same time series[9][10].

2.4 Ensemble of Neural Networks

Ensemble learning algorithms are an important topic of interest in the research community because of their capability of improving the classification accuracy of any single classifier.

Reasons for choosing Ensemble based systems:

1. Statistical Reasons: Good performance on training data does not predict good generalization performance. A set of classifiers with similar training performances may have different generalization performances. Also classifiers with similar generalization performances may perform differently in the field , particularly if the data set used to determine the generalization performances is not sufficiently representative of the future field data. In such cases, combining the outputs of several classifiers by averaging may reduce the risk of an unfortunate selection of a poor selection. The averaging may or may not beat the performance of the best classifier in the ensemble ,but it takes care to reduce the overall risk of making a particularly poor selection. This is precisely the reason we consult the opinions of other experts: having several doctors agree on a diagnosis ,or several users agree on the quality of a product , reduces the risk of following the advice of a single doctor whose specific experience may be significantly different than those of others.

2.Large Volumes of Data: In some applications, in which data which is required to be analyzed can be too large to be effectively handled by a single classifier. For example, inspection of gas transmission pipelines using magnetic flux leakage techniques generate 10GB of data every 100km of pipeline of which there are 2 million km. Training a classifier with such a vast amount of data is usually not practical. So partitioning the data into smaller subsets and then training by different classifiers by different partitions of data is done. Further combining their outputs using an intelligent combination rule often provides to be a more efficient approach[11].

Bagging and Boosting

When applied to ANNs, bagging and boosting ensemble techniques can produce dramatic improvements in generalization performance. The underlying idea of these techniques is to generate multiple versions of a predictor, which when combined, will provide "smoother" more stable predictions.

Bagging (Bootstrap Algorithm)

Bootstrap aggregating (bagging) is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used instatistical classification and regression. It also reduces variance and helps to avoid overfitting. Although it is usually applied to decision tree methods, it can be used with any type of method. Bagging is a special case of the model averaging approach.

Given a standard training set D of size n, bagging generates m new training sets D_i, each of size n′ < n, by sampling from D uniformly and with replacement. By sampling with replacement, some observations may be repeated in each D_i. If n′=n, then for large n the set D_i is expected to have the fraction (1 - 1/e) (≈63.2%) of the unique examples of D, the rest being duplicates. This kind of sample is known as a bootstrap sample. The m models are fitted using the above m bootstrap samples and combined by averaging the output (for regression) or voting (for classification).
Bagging leads to "improvements for unstable procedures" (Breiman, 1996), which include, for example, neural nets, classification and regression trees, and subset selection inlinear regression (Breiman, 1994). An interesting application of bagging showing improvement in preimage learning is provided here. On the other hand, it can mildly degrade the performance of stable methods such as K-nearest neighbors (Breiman, 1996)[2].

Given a training set, bagging generates multiple boot-strapped training sets and calls the base model learning algorithm with each of them to yield a set of base models. Given a training set of size t, bootstrapping generates a new training set by repeatedly (t times) selecting one of the t examples at random, where all of them have equal probability of being selected. Some training examples may not be selected at all and others may be selected multiple times. A bagged

ensemble classifies a new example by having each of its base models classify the example and returning the class that receives the maximum number of votes. The hope is that the base models generated from the different bootstrapped training sets disagree often enough that the ensemble performs better than the base models.

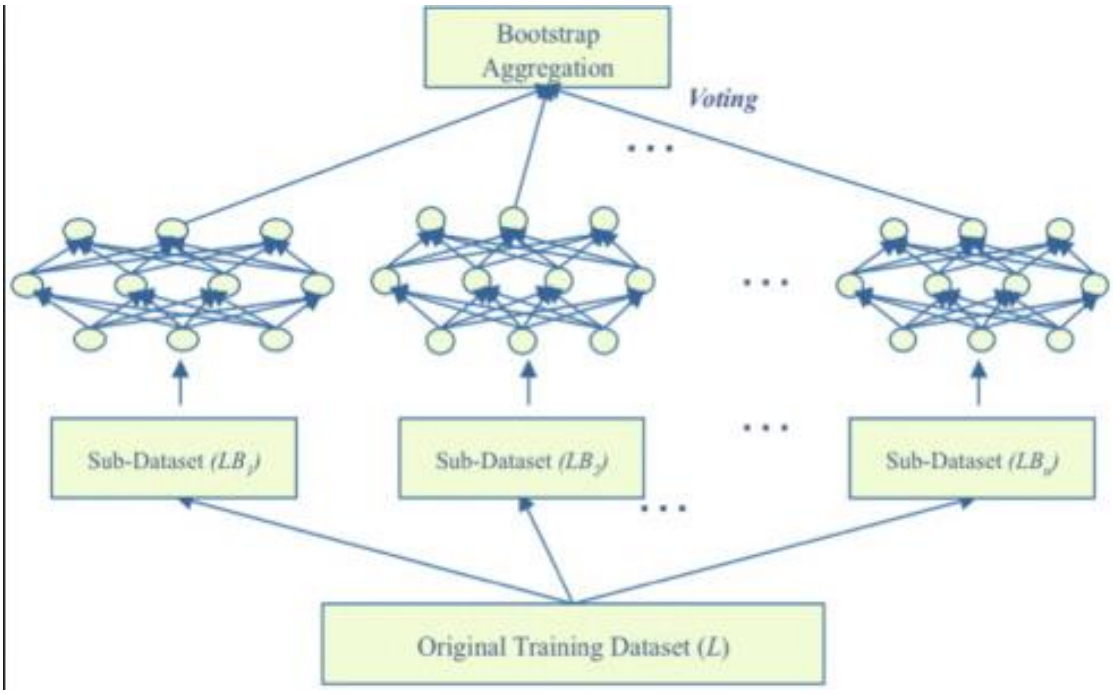


Fig 22-Bagging Technique

Boosting

Boosting is a machine learning meta-algorithm for reducing bias in supervised learning. Boosting is based on the question posed by Kearns: Can a set of weak learners create a single strong learner? A weak learner is defined to be a classifier which is only slightly correlated with the true classification (it can label examples better than random guessing). In contrast, a strong learner is a classifier that is arbitrarily well-correlated with the true classification.

Schapire's affirmative answer to Kearns' question has had significant ramifications in machine learning and statistics, most notably leading to the development of boosting.

When first introduced, the hypothesis boosting problem simply referred to the process of turning a weak learner into a strong learner. "Informally, [the hypothesis boosting] problem asks whether an efficient learning algorithm that outputs a hypothesis whose performance is only slightly better than random guessing [i.e. a weak learner] implies the existence of an efficient algorithm that outputs an hypothesis of arbitrary accuracy [i.e. a strong learner]. Algorithms that achieve hypothesis boosting quickly became simply known as "boosting".

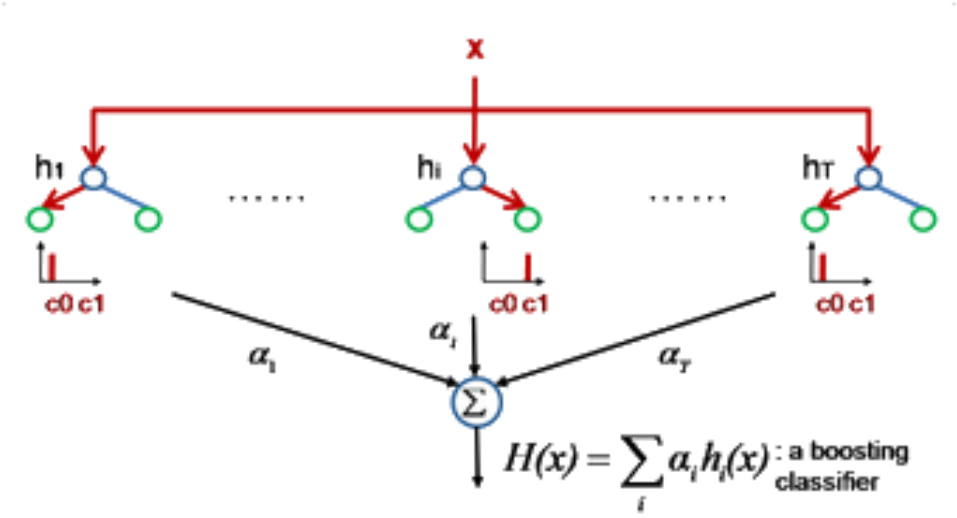


Fig 23-Boosting Technique

Boosting algorithms

While boosting is not algorithmically constrained, most boosting algorithms consist of iteratively learning weak classifiers with respect to a distribution and adding them to a final strong classifier. When they are added, they are typically weighted in some way that is usually related to the weak learners' accuracy. After a weak learner is added, the data is reweighted: examples that are misclassified gain weight and examples that are classified correctly lose weight (some boosting algorithms actually decrease the weight of repeatedly misclassified examples, e.g., boost by majority and BrownBoost). Thus, future weak learners focus more on the examples that previous weak learners misclassified.

There are many boosting algorithms. The original ones, proposed by Robert Schapire (a recursive majority gate formulation) and Yoav Freund (boost by majority), were not adaptive and could not take full advantage of the weak learners. They won the prestigious Gödel Prize for their work on AdaBoost, a boosting algorithm that they developed together. Only algorithms that are provable boosting algorithms in the probably approximately correct learning formulation are called boosting algorithms. Other algorithms that are similar in spirit to boosting algorithms are sometimes called "leveraging algorithms", although they are also sometimes incorrectly called boosting algorithms

Another method that uses different subsets of training data with a single learning method is the boosting approach. It assigns weights to the training instances, and these weight values are changed depending upon how well the associated training instance is learned by the classifier;

the weights for misclassified instances are increased. Thus, re-sampling occurs based on how well the training samples are classified by the previous model. Since the training set for one model depends on the previous model, boosting requires sequential runs and thus is not readily adapted to a parallel environment. After several cycles, the prediction is performed by taking a weighted vote of the predictions of each classifier, with the weights being proportional to each classifiers accuracy on its training set.

LP Boost

Linear Programming Boosting (LPBoost) is a supervised classifier from the boosting family of classifiers. LPBoost maximizes a margin between training samples of different classes and hence also belongs to the class of margin-maximizing supervised classification algorithms. Consider a classification function

$$f : \mathcal{X} \to \{-1, 1\},$$

which classifies samples from a space $\mathcal{X}$ into one of two classes, labelled 1 and -1, respectively. LPBoost is an algorithm to learn such a classification function given a set of training examples with known class labels. LPBoost is a machine learning technique and especially suited for applications of joint classification and feature selection in structured domains.

Algorithm

- Input:
  - Training set $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\ell\}, \boldsymbol{x}_i \in \mathcal{X}$
  - Training labels $Y = \{y_1, \ldots, y_\ell\}$, $y_i \in \{-1, 1\}$
  - Convergence threshold $\theta \geq 0$
- Output:
  - Classification function $f : \mathcal{X} \to \{-1, 1\}$

1. Initialization
   1. Weights, uniform $\lambda_n \leftarrow \frac{1}{\ell}, \quad n = 1, \ldots, \ell$
   2. Edge $\gamma \leftarrow 0$
   3. Hypothesis count $J \leftarrow 1$

2. Iterate
    1. $\hat{h} \leftarrow \underset{\omega \in \Omega}{\operatorname{argmax}} \sum_{n=1}^{\ell} y_n h(\boldsymbol{x}_n; \omega) \lambda_n$
    2. if $\sum_{n=1}^{\ell} y_n \hat{h}(\boldsymbol{x}_n) \lambda_n + \gamma \leq \theta$ then
        1. break
    3. $h_J \leftarrow \hat{h}$
    4. $J \leftarrow J + 1$
    5. $(\boldsymbol{\lambda}, \gamma) \leftarrow$ Solution of the LPBoost dual
    6. $\boldsymbol{\alpha} \leftarrow$ Lagrangian multipliers of solution to LPBoost dual problem
3. $f(\boldsymbol{x}) := \operatorname{sign}\left(\sum_{j=1}^{J} \alpha_j h_j(\boldsymbol{x})\right)$

Note that if the convergence threshold is set to $\theta = 0$ the solution obtained is the global optimal solution of the above linear program. In practice, θ is set to a small positive value in order obtain a good solution quickly[12].

Learning Method used

We implemented our model using Bagging and LP Boosting. Higher accuracy was achieved using Bagging. We have compared the output of Bagging and LP Boosting in the results section.

CHAPTER 3: IMPLEMENTATIONS AND RESULTS

3.1 METHODOLOGY

This chapter describes the implementation of a Neural Network model in a stock market scenario and the application of indicator pruning techniques and NARX model. The primary objective of this research is to generate a one-day forecast of the closing price using the techniques mentioned.

3.1.1 Stock Market Data

The stock market data used in this research was derived from various sources on the World Wide Web, most notably, the financial websites maintained by Yahoo Inc.

We can import data from Yahoo Inc. for any company for any desire time period. That we take as an input for the final project. We take - 'Adjacent Close' 'Close' 'High' 'Low' 'Open' 'Volume' for the company selected.

The following are the broad economic sectors that are taken into consideration by typical stock market investors.

1. Basic Materials [BSC]
2. Consumer, Cyclical [CYC]
3. Energy [ENE]
4. Financial [FIN]
5. Healthcare [HCR]
6. Industrial [IDU]
7. Investment Products [IVP]
8. Consumer, Non-Cyclical [NCY]
9. Technology [TEC]
10. Telecommunications [TLS]
11. Utilities [UTI]

According to these classifications we can take other parameters as the inputs to get a better prediction. For example: If we are dealing with TATASTEEEL.NS we can take BSE.METALS as the benchmark to compare the input and provide a general trend in the specific industry.

Also we can provide different set of inputs for different economic sectors. These we can predefine and keep to help the input field.

3.1.2 Selection of Indicators

The initial indicators used are as follows:

- Relative Strength Index
- Stochastic Oscillator
- Bollinger Bands
- Simple Moving Average
- Exponential Moving Average
- MACD

3.2 IMPLEMENTATION

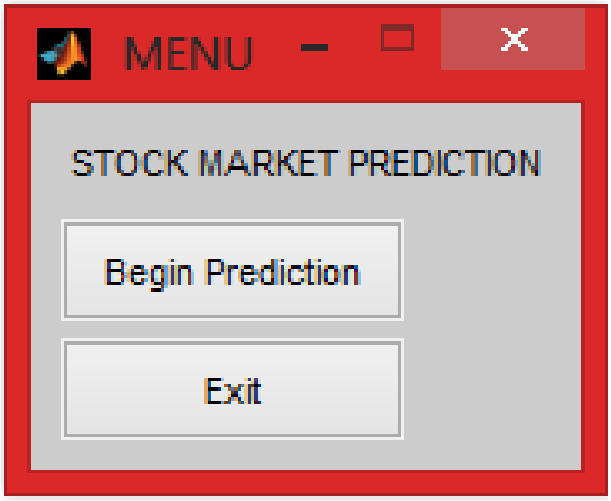


Fig 24

We have created a GUI to implement stock market prediction. It has two buttons : Begin Prediction and Exit.

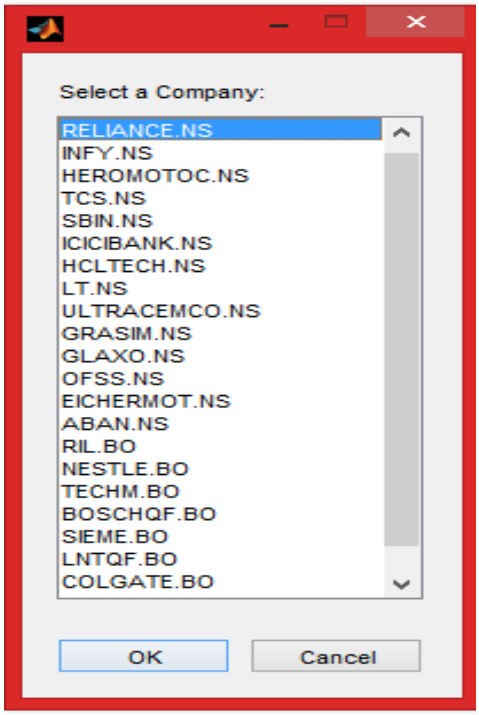


Fig 25

Once you click the Begin prediction button a list of companies are shown. We have used the following companies:

- RELIANCE
- INFOSYS
- TCS
- SBI
- ICICI
- HCL TECH
- GRASIM
- GLAXO
- ORACLE FINANCIAL SERVICES
- NESTLE
- COLGATE
- GLENMARK
- DR.REDDY

All companies listed in the NSE and BSE can be added. Once you select any one of the companies , you will be asked to select a date for the prediction.

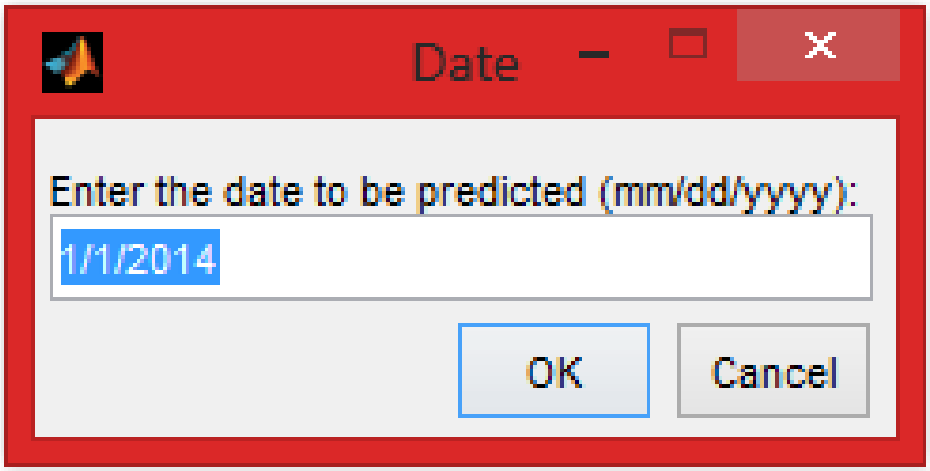


Fig 26

Once you enter the date, the neural network starts training and accordingly the closing price of the day that is input by the user is predicted.

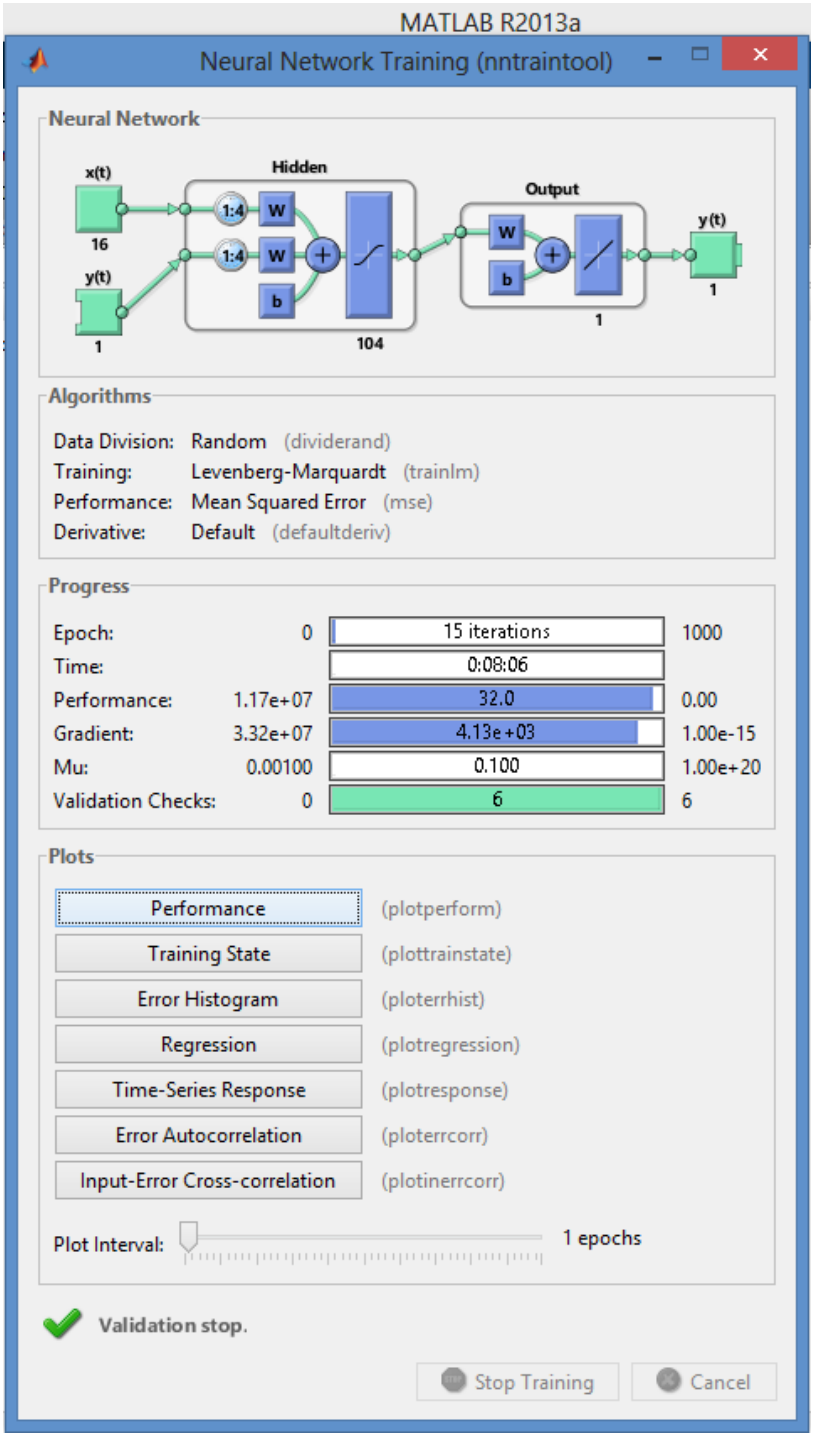


Fig 27

The above window is displayed once training is complete. The predictions are then made and an appropriate response is plotted. An epoch is a measure of the number of times all of the training vectors are used once to update the weights. The performance shows Guidelines for improving generated code performance. Gradient is the numerical gradient. Regression is for Linear Regression. Mu means to Input matrix of means `mu`.

`plotresponse(t,y)` takes a target time series `t` and an output time series `y`, and plots them on the same axis showing the errors between them.

`ploterrhist(e)` plots a histogram of error values `e`.

`ploterrhist(e1,'name1',e2,'name2',...)` takes any number of errors and names and plots each pair.

`plotinerrcorr(x,e)` takes an input time series `x` and an error time series `e`, and plots the autocorrelation of inputs to errors across varying lags.

`plotregression(targets,outputs)` plots the linear regression of `targets` relative to `outputs`.`

`plotperform(TR)` plots the training, validation, and test performances given the training record `TR` returned by the function `train`.

`plottrainstate(tr)` plots the training state from a training record `tr` returned by `train`.

`ploterrcorr(error)` takes an error time series and plots the autocorrelation of errors across varying lags.

3.3 RESULTS

3.3.1 Different company predictions on the same day

Sr.No	Company	Date MM/DD/YYYY	Target Price	Predicted Price (without ensemble)	Error (without ensemble)	Predicted Price (with ensemble)	Error (with ensemble)
1.	GLENMARK-BSE	01/01/2014	530.0	551.28	21.28	530.4	0.4
2.	ICICI BANK-NSE	01/01/2014	1097.7	1073.2	24.5	1099.3	1.6
3.	LNT-NSE	01/01/2014	1068.6	1076.4	7.8	1073.3	4.7
4.	RELIANCE-BSE	01/01/2014	888.7	893.4	4.7	889.6	0.9
5.	TECH MAHINDRA-NSE	01/01/2014	1827.3	1798.9	28.4	1824.7	2.6
6.	SIEMENS-BSE	01/01/2014	663.9	667.9	4.0	664	0.1
7.	ABAN-NSE	01/01/2014	386.5	406.4	19.9	386.8	0.3

We have computed the prices of the above seven companies for 01/01/2014.As seen using ensemble of neural networks has reduced the error by a great extent.

We have illustrated the improved accuracy of few of the above companies in the graphs shown below:

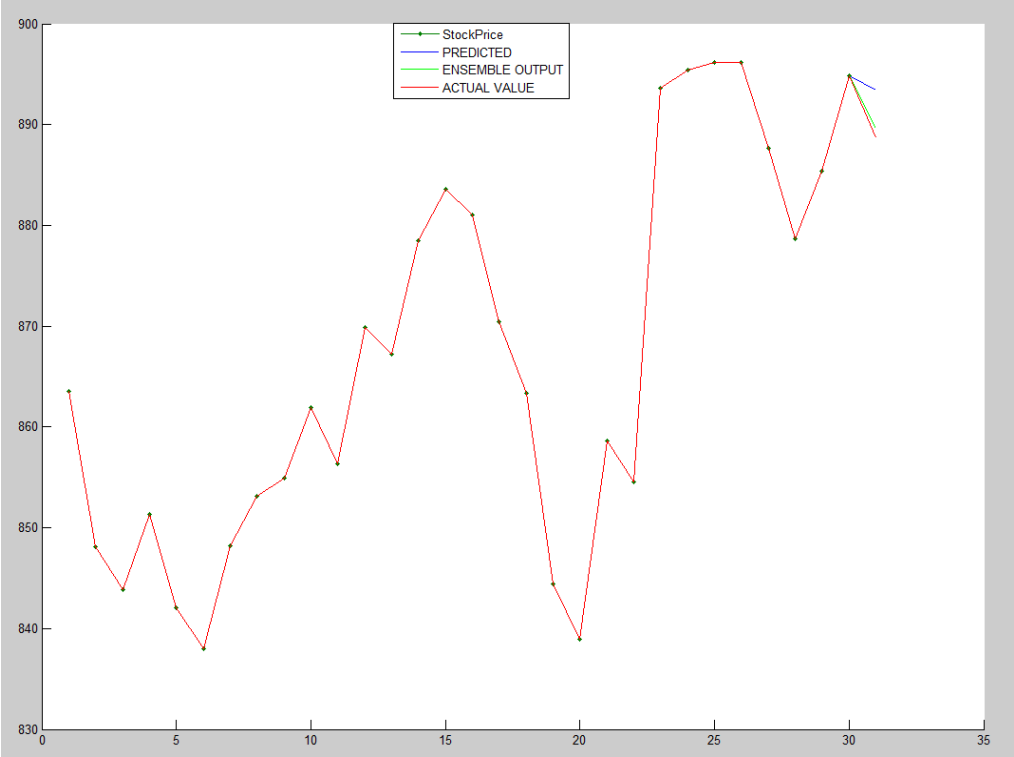


Fig 28:Reliance-BSE

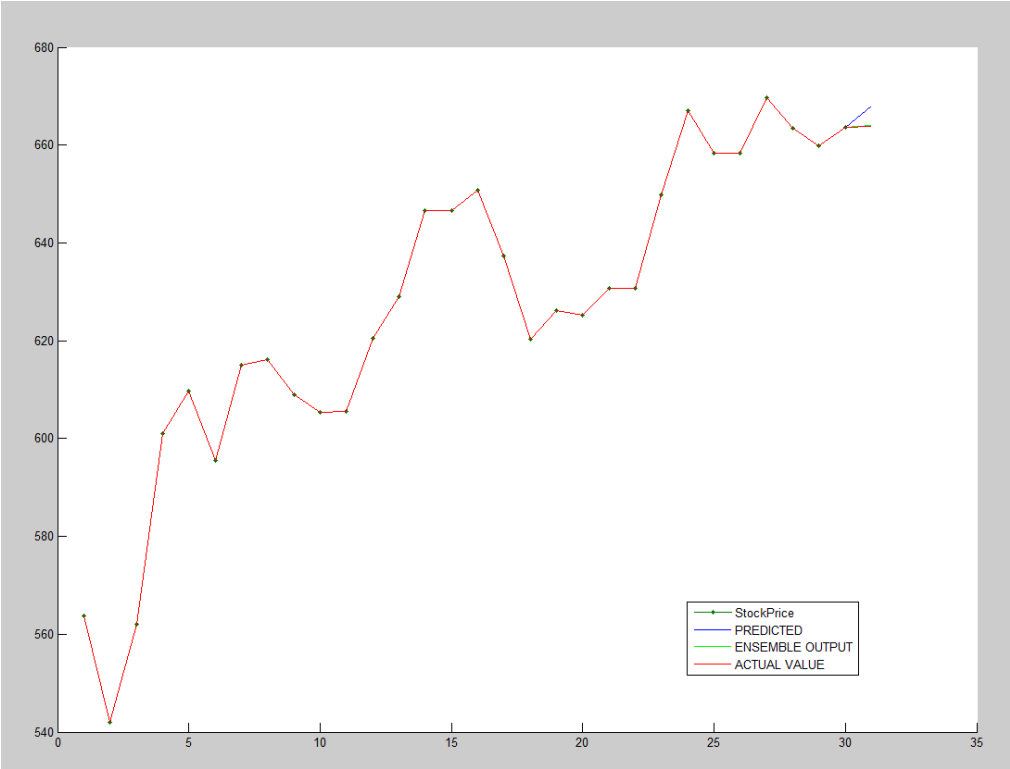


Fig 29:Siemens- BSE

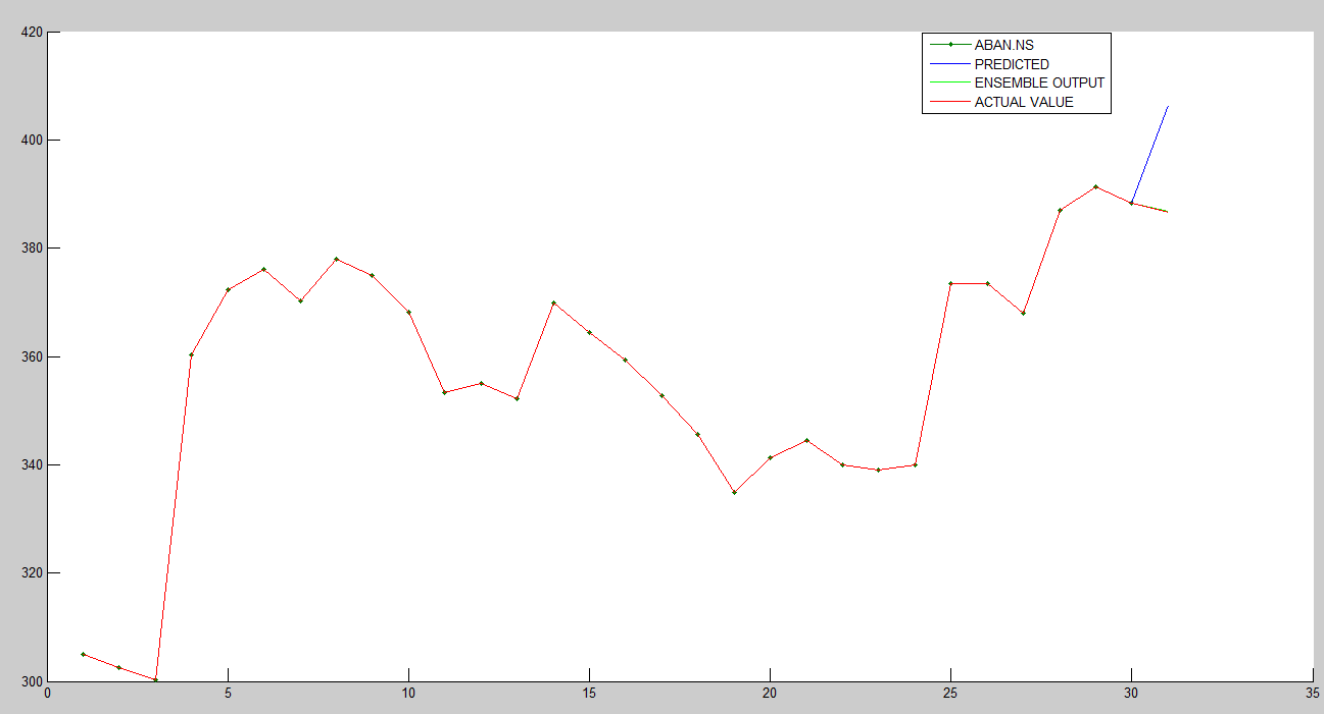


Fig 30-Aban.NS

3.3.2 Same company prediction on 10 consecutive days

Date	Target Stock	Predicted Price (without ensemble)	Error (without ensemble)	Predicted Price (with ensemble)	Error (with ensemble)
01/04/2014	942.6	932.87	9.7284	942.62	0.022
02/04/2014	952.65	940.81	11.8367	952.77	0.129
03/04/2014	958.2	936.8726	21.3274	958.0803	0.117
04/04/2014	944.25	940.231	4.0190	945.3922	1.1422
05/04/2014	944.25	940.25	3.99	945.39	1.142
06/04/2014	944.25	940.2577	3.9923	945.3922	1.1422
07/04/2014	944.15	945.93	1.78	947.2	3.05
08/04/2014	944.15	937.06	7.0868	944.1778	0.0278
09/04/2014	962.15	939.5568	22.5932	962.2073	0.0573
10/04/2014	969.05	948.2712	20.7788	968.8073	0.2427

In the table given above ,we have predicted the output with and without ensemble learning for different dates of 'RELIANCE.NS'. We have also calculated the error. We have found that ensemble learning increases the accuracy by a very big margin.

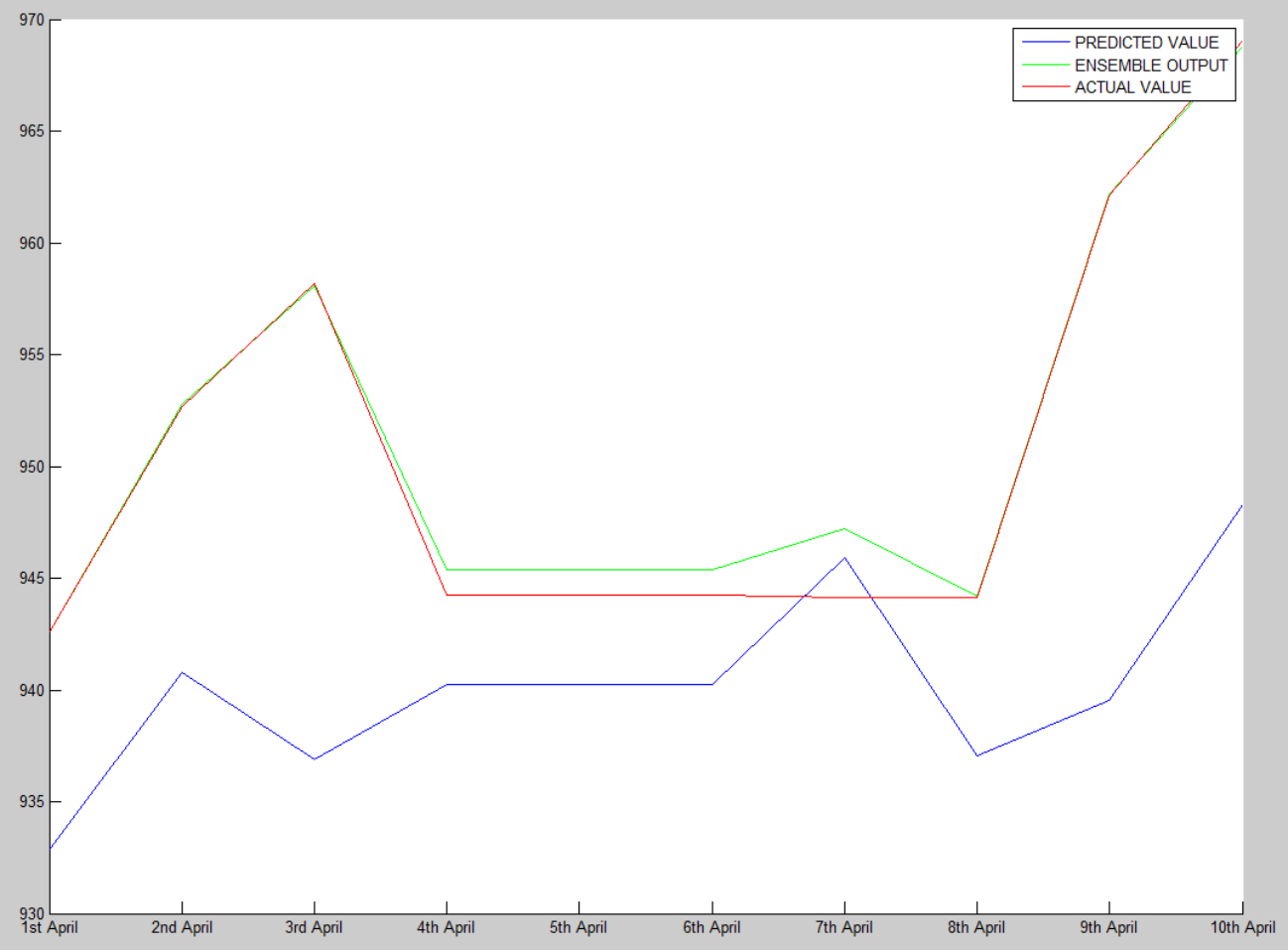


Fig 30

The graph above shows the closing price of Reliance from 1st April 2014 to 10th April 2014.The blue line shows the predicted value without the ensemble learning. The green line shows the predicted value with the ensemble learning. It can be seen that using an ensemble of neural networks gives a higher accuracy.

3.3.3 Different Company Prediction With and Without Bollinger Bands

Company	Date	Target Price	Without Bollinger				With Bollinger			
			Predicted Price (without ensemble)	Error (without ensemble)	Predicted Price (with ensemble)	Error (with ensemble)	Predicted Price (without ensemble)	Error (without ensemble)	Predicted Price (with ensemble)	Error (with ensemble)
SIEMENS.NS	15/03/2014	682.15	651.5	30.64	678.37	3.77	675.9	6.16	682.18	0.04
ICICIBANK.NS	15/07/2013	1061.1	1035	26.01	1061.3	0.12	1047.1	13.95	1060.7	0.30
OFSS.NS	20/05/2013	2438.7	2498.5	59.75	2431	7.814	2462.4	23.73	2437.5	1.18
LT.NS	25/12/2013	1076	1016.3	58.68	1074	2	1099	23	`1075.72	0.28
LNTQF.BO	01/10/2013	800	853.64	53.64	806.56	6.56	825.69	25.69	800.61	0.61

In the table given above we have selected random companies and predicted the output for different dates with and without Bollinger bands. It can be proved that by adding indicators accuracy of the model can be increased.

3.3.4 Future Prediction

Company	Prediction for 04/23/2014		Actual closing price on 04/23/2014	Error(without ensemble)	Error(with ensemble)
	Without ensemble	With ensemble			
ICICIBANK-NSE	1289.9	1299.0	1299.55	9.65	0.55
RELIANCE-NSE	965	968.72	967.15	2.15	1.57
INFOSYS-NSE	3239.8	3177.8	3172.65	67.15	5.15
L&T-NSE	1313.4	1376.5	1376.35	62.95	0.15
TECHMAHINDRA-BSE	1722.4	1766.6	1765.95	43.55	0.65

We have tried to predict the future value and then we compared it with the actual closing price and saw that the ensemble output generated a very low error.

CHAPTER 4. APPLICATIONS

Innovativeness and usefulness: With the advent of the digital computer, stock market prediction has since moved into the technological realm. The most prominent and beneficial technique as compared to other traditional techniques (like mathematical analysis) involves the use of artificial neural networks (ANNs) and Genetic Algorithms

Stock market prediction is the act of trying to determine the future value of a company stock or other financial instrument traded on a financial exchange. The successful prediction of a stock's future price could yield significant profit. Some believe that stock price movements are governed by the random walk hypothesis and thus are unpredictable. Others disagree and those with this viewpoint possess a myriad of methods and technologies which purportedly allow them to gain future price information.

Market potential and competitive advantage: Currently very few software (only as research) are available in the market and there is huge potential for this kind of prediction which employs Neural Networks in future on a commercial basis.

CHAPTER 5. CONSTRAINTS

1. Our model doesn’t take into account the following situations
:

- If the CEO of a company resigns
- Natural disasters such as earthquakes,tsunami,drought,etc
- Terrorist attacks

2. The computation time is too long. Other neural networks can be used to lessen the computation time but the results wouldn’t be as accurate.

3. The Indian stock market highly depends on the USA stock market and the Hong kong stock market. Fluctuations in either will cause a fluctuation in the Indian stock market. Such parameters haven’t been taken into consideration.

CHAPTER 6. FUTURE SCOPE

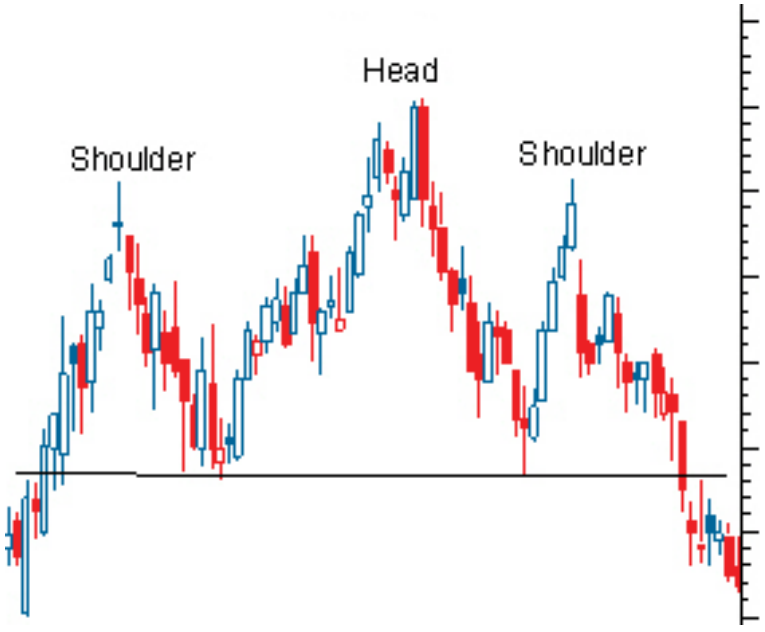
1. Stock Market prediction using Patterns

Head And Shoulders

The head-and-shoulders pattern is one of the most popular and reliable chart patterns in technical analysis. And as one might imagine from the name, the pattern looks like a head with two shoulders.

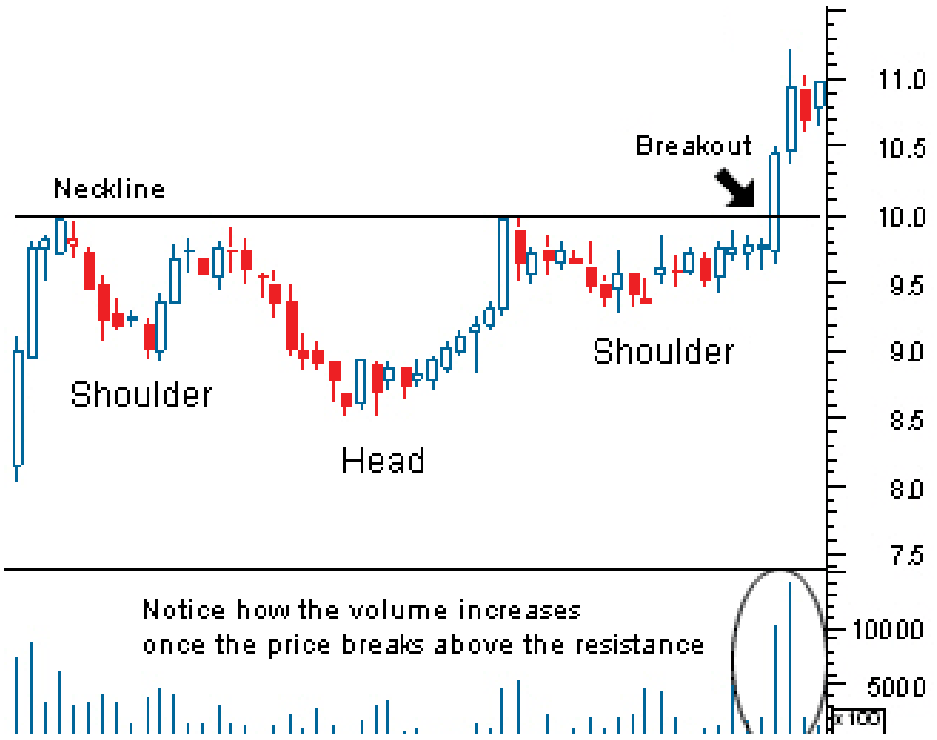
Head and shoulders is a reversal pattern that, when formed, signals the security is likely to move against the previous trend. There are two versions of the head-and-shoulders pattern. The head-and-shoulders top is a signal that a security's price is set to fall, once the pattern is complete, and is usually formed at the peak of an upward trend. The second version, the head-and-shoulders bottom (also known as inverse head and shoulders), signals that a security's price is set to rise and usually forms during a downward trend.

Both of these head and shoulders have a similar construction in that there are four main parts to the head-and-shoulder chart pattern: two shoulders, a head and a neckline. The patterns are confirmed when the neckline is broken, after the formation of the second shoulder.



Inverse Head and Shoulders (Head-and-Shoulders Bottom)

The inverse head-and-shoulders pattern is the exact opposite of the head-and-shoulders top, as it signals that the security is set to make an upward move. Often coming at the end of a downtrend, the inverse head and shoulders is considered to be a reversal pattern, as the security typically heads higher after the completion of the pattern.



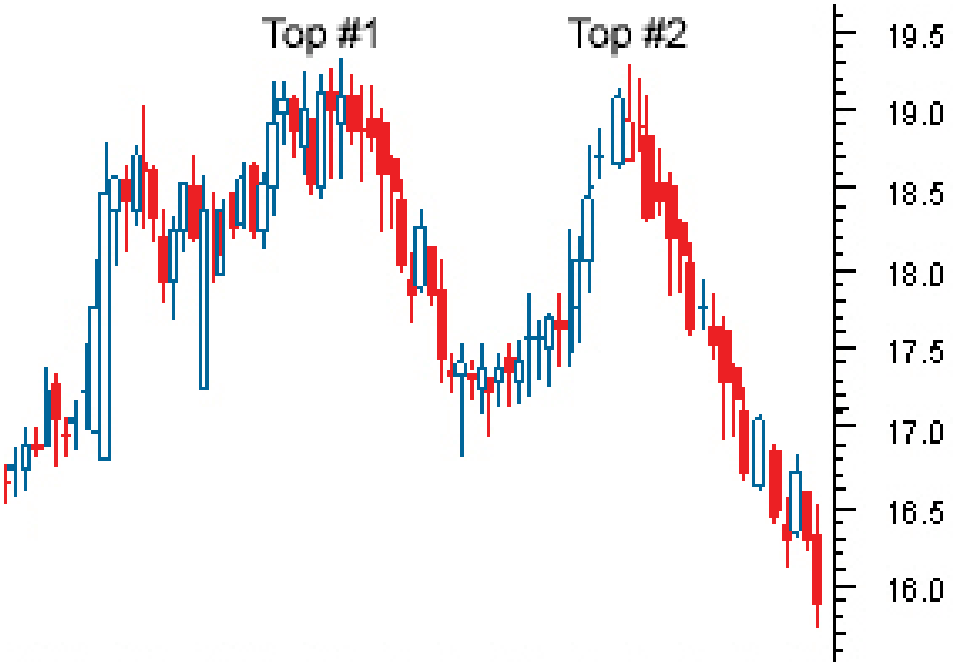
Inverse Head and Shoulder Pattern

Again, there are four steps to this pattern, starting with the formation of the left shoulder, which occurs when the price falls to a new low and rallies to a high. The formation of the head, which is the second step, occurs when the price moves to a low that is below the previous low, followed by a return to the previous high. This move back to the previous high creates the neckline for this chart pattern. The third step is the formation of the right shoulder, which sees a sell-off, but to a low that is higher than the previous one, followed by a return to the neckline. The pattern is complete when the price breaks above the neckline.

Double Top

The double-top pattern is found at the peaks of an upward trend and is a clear signal that the preceding upward trend is weakening and that buyers are losing interest. Upon completion of this pattern, the trend is considered to be reversed and the security is expected to move lower.

The first stage of this pattern is the creation of a new high during the upward trend, which, after peaking, faces resistance and sells off to a level of support. The next stage of this pattern will see the price start to move back towards the level of resistance found in the previous run-up, which again sells off back to the support level. The pattern is completed when the security falls below (orbreaks down) the support level that had backstopped each move the security made, thus marking the beginnings of a downward trend.



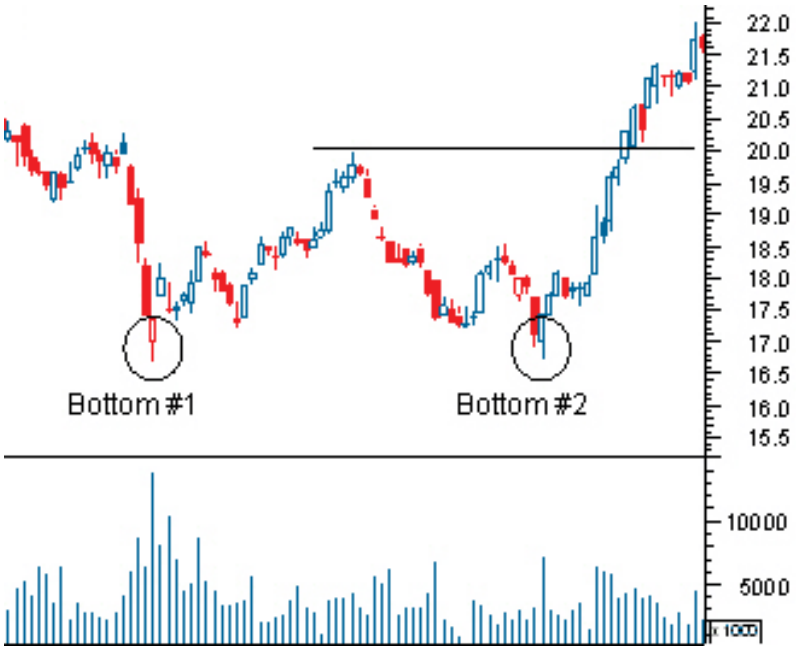
Double Pattern

It's important to note that the price does not need to touch the level of resistance but should be close to the prior peak. Also, when using this chart pattern one should wait for the price to break below the key level of support before entering. Trading before the signal is formed can yield disastrous results, as the pattern is only setting up the possibility for the trend reversal and could trade within this banded range for some time without falling through.

This pattern is a clear illustration of a battle between buyers and sellers. The buyers are attempting to push the security but are facing resistance, which prevents the continuation of the upward trend. After this goes on a couple of times, the buyers in the market start to give up or dry up, and the sellers start to take a stranglehold of the security, sending it down into a new downtrend.

Double Bottom

This is the opposite chart pattern of the double top as it signals a reversal of the downtrend into an uptrend. This pattern will closely resemble the shape of a "W".



Double-bottom pattern

The double bottom is formed when a downtrend sets a new low in the price movement. This downward move will find support, which prevents the security from moving lower. Upon finding support, the security will rally to a new high, which forms the security's resistance point. The next stage of this pattern is another sell-off that takes the security down to the previous low. These two support tests form the two bottoms in the chart pattern. But again, the security finds

support and heads back up. The pattern is confirmed when the price moves above the resistance the security faced on the prior move up[7].

2 This model can also be used as a base to make a model on portfolio management. Portfolio Management can be used by portfolio management companies, hedge fund managers and financial managers in order to optimize an investment for maximum returns using the minimum risk possible

3 Multiple day testing: There are two types of traders. Firstly ,those who buy and sell stocks on the same day and secondly those who sell after a long period say after 20 days or even a month. A model can be made to predict the closing price of a company after 20 days. So this would be helpful to those who invest for long periods so as to get a better return.

4 We have implemented the stock market prediction model using NARX neural network.Many other neural networks can be used to improve the accuracy and reduce the running time

CHAPTER 7. CONCLUSION

Stock markets are highly volatile. If there could be any model or software that could predict the fluctuations in the stock market ,then it would be in high demand. We have used neural networks to implement our stock market prediction model. The reason being that neural networks help to approximate any linear or non linear function to a high degree of desired accuracy. After a lot of research we decided to use NARX(Non Linear Auto Regressive) Neural network because it is useful in financial time series forecasting. The architecture of NARX includes delays which makes future prediction possible.

We have implemented an ensemble of neural networks by which we were able to significantly increase our accuracy of prediction. We have implemented those stock market indicators which gave a reasonable accuracy in predicting an uptrend or downtrend of stock prices.

By increasing(decreasing) the number of oscillators in our model the accuracy increases(decreases). We have tried to predict the future closing price which gave us decent accuracy.

With an increased database and more processing power, the accuracy can further be increased.

Currently very few software (only as research) are available in the market and there is huge potential for this kind of prediction which employs Neural Networks in future on a commercial basis.

CHAPTER 8. REFERENCES – BIBLIOGRAPHY / LITERATURE

[1] Neural Networks by Christos Stergiou and Dimitrios Siganos.

[2] Boosting and Bagging of Neural Networks with applications to financial Time series by Zhou Zheng, August 4, 2006.

[3] Sven F. Crone - Forecasting with Artificial Neural Networks (Centre for Forecasting

Department of Management Science)

[4] Sriram Lakshminarayanan - "AN INTEGRATED STOCK MARKET FORECASTING MODEL USING NEURAL NETWORKS"

the Department of Industrial and Manufacturing Systems Engineering and the Russ College of Engineering and Technology by

[5]http://en.wikipedia.org/wiki/Stock_market

[6] Weckman, G.R., and Agarwala, R., "Identifying Relative Contribution of Selected Technical Indicators in Stock Market Prediction", Proceedings of the IIE Annual Conference 2002, May 18-21, Portland, Oregon, 2002.

[7] http://www.investopedia.com

[8]--Simon Haykin, "Neural Network A Comprehensive Foundation", second edition, Prentice Hall, 1998, page 161 – 173.

[9]--The use of NARX Neural Networks to predict Chaotic Time Series - EUGEN DIACONESCU, PhD, Electronics, Communications and Computer Science Faculty

University of Pitesti

[10] Long-Term Time Series Prediction with the NARX Network: An Empirical Evaluation - Jose Maria P. Junior and Guilherme A. Barreto ,Department of Teleinformatics engineering Federal University of Ceara, Av. Mister Hull, S/N,CP 6005, CEP 60455-760, Fortaleza-CE, Brazil

[11] Robi Polikar , "Ensemble based systems in decision making",IEEE circuits and systems magazine, 3rd quarter 2006,pp 21-45.

[12] Boosting the minimum margin: LPBoost vs. AdaBoost by Hanxi Li, Chunhua Shen.