Modeling and Forecasting Exchange Rate and Stock Returns in India using Artificial Neural Network

A Thesis submitted for the Degree of
Doctor of Philosophy
in Economics

By

Chakradhara Panda

Department of Economics
School of Social Sciences
University of Hyderabad
Hyderabad - 500 046
India
November 2003

Dedicated

To

My Parents

Department of Economics,
School of Social Sciences,
University of Hyderabad,
Hyderabad-500 046.

12th November 2003

This is to certify that I, Chakradhara Panda have carried the research embodied in the present thesis entitled **"Modeling and Forecasting Exchange Rate and Stock Returns in India using Artificial Neural Network"** within the full period prescribed under Ph.D. ordinances of the University of Hyderabad.

I declare to the best of my knowledge that no part of the thesis was earlier submitted for the award of research degree of any University or Institute.

V. Narasimhan

(Signature of the Supervisor)

V. Narasimhan

[illegible]

(Signature of the Candidate)

Name: Chakradhara Panda

Reg.No:2KSEPH01.

[illegible]

Head of the Department

12/11/03

[illegible]

Dean of the School

HEAD
Department of Economics
University of Hyderabad
Hyderabad - 500 046.

Acknowledgements

This thesis is borne out of frustration. A number of people are associated with this project who I feet should be duty acknowledged (First and foremost, (Dr. V. Narasimhan, my supervisor, deserves my heartfelt gratitude. Her endless patience, unflagging persistence, and academic motivation have provided me with the necessary fillip to make this work a reality. I am indebted to her for her openmindedness in our stimulating discussion and for the freedom she rendered me to carry out this particular work. Working with her has been a unique and illuminating experience.

I gratefully acknowledge Prof. K.N. Murty, Head, Department of' Economics for creating the conducive academic atmosphere and for the helpful discussions when I was writing the thesis. I thank Prof. D.N. Reddy, (Dean, School of Social Sciences and supporting staff of the department in particular Mr. Venkateswara Rao for their cooperation.

My sincere thanks to Mr. Prabir Kumar Mohanty and (Pratap Chandra Biswal for teaching me the basics of research and more particularly for introducing me to this field of research. Without their help, this thesis would never have assumed its present form. Apart from their alt around academic help, I remain grateful to them for their understanding of my situation.

I have been fortunate enough to be associated with Mr. Sujit Kumar Mishra and Mr. K.S. Sujit. It is Mr. Sujit Kumar Mishra who stood by me at every juncture and gave me moral, emotional and academic support and helped me attain my goats.

A very special thanks to Mr. Aruna Kumar Dash who has always been a pillar of strength to me in my research career. I will always remember his unbridled and unselfish support which he has showered on me during my stay in the campus.

A friend in deed is a friend indeed My friends Mr. Jhadeswar Murmu, Mr. Alok Kumar Mishra, Mr. Badri Narayan Rath and Mr. B.K. Srinivas have made me realize this. I would be failing in my duty if I do not acknowledge them who showed keen interest in the progress of my work and more importantly had enough patience to listen to me when I was talking some nonsense on my topic. It is also my pleasure to acknowledge the friendly cooperation of Mr. Bhaskar Rao Chinthapalli and Mr. D. Sudhakara Rao. Friends, all of you realty made my stay in the campus a memorable experience. My sincere thanks to all of you for sharing my sweet and bitter moments.

I wish to record here my thanks to Mrs. Sanghamitra Sahu and Ms. Pragyan Mishra for their inspiration and their interest in my career. I am also thankful to Mr. Mirza Allim Baig and Mr. Amiya Kumar Shaoo for Shaoo for their help and encouragement

I owe my everything to my family. They all deserve my deep sense of gratitude, especially my father who left no stone unturned to shape my career and my mother whose enormous confidence and faith in me made me what I am today. I wish to acknowledge the lovely

Rids in the family, Bapu, Arun, Sipu and Sumi whose charming faces and innocent smiles made it possible for me to overcome the lean phase of my research career.

At this juncture, when I have successfully completed my research, it would Se worth my white, to mention the contributions made by Prof. Gianluigi Rech, Stockholm School of Economics, Prof. R. Glen Donaldson, University of British Columbia, Canada, and (Prof Mark Kamstra, Simon Fraser University, Canada through sending good number of research articles and invariable suggestions. At the same time, I owe a great debt of thankfulness to Dr. Amit Mitra, IIT, Bombay, Dr. M. Sonar, IIT, Bombay, and Mr. Sanjib Bordoloi, RBI, (Bombay for their constructive suggestions which led to the significant improvements of this work.

I thank University authorities for providing the infrastructure facilities, Financial support from University Grants Commission, New Delhi in the form of research fellowship is gratefully acknowledged

My sincere apologies to all those whose names J would like to very much mention here but am unable to do so because of lack of space. I wish to convince one and all that the contribution made by those whose names I am unable to mention is as valuable and significant as the ones I did, if not more.

Thank you all for making my dream come true

Chakradhara Panda

CONTENTS

List of Tables

List of Figures

Chapter I

Introduction, Background and Objectives of the Study

I am not interested in developing a powerful brain
All I'm after is just a mediocre brain...

1.0 Introduction

The question "what is the need of forecasts at all?" has engrossed many economists and managers for many years. Forecasts are essential for two basic reasons: the future is uncertain and the full impact of many decisions taken now is not felt until later. If a person is told that he or she is to relocate to a new city, he or she might very well be interested in knowing what life will be like there. It is for the same reason that every one should be highly interested in the future - for it is where they are going to spend the rest of their lives and it is likely to be quite different from the past or present. However, for economists and managers, knowing the future is not merely a matter of academic interest. Economics and Management are decision sciences: they are concerned with sensible decision making and the effects of decisions. Any decision has to take a viewpoint about what will occur in the future and thus comes the idea of forecasts. But forecasting is generally not easy for the reason that the methods that can be used to forecast can vary greatly and will depend upon data availability, the quality of models available, and the kinds of assumptions made, amongst other things. This is one reason why the topic is so interesting.

Two aspects of forecasting that are of particular importance are the degree of predictability, and forecasting accuracy. The degree of predictability of variables is one important aspect of forecasting. An astronomer can predict just where every star will be at half past eleven tonight; he can make no such prediction about his daughter. Even this simple avowal brings out an important point about forecasting. It states that variables to be forecast vary greatly in their degree of predictability. An astronomer can have much greater confidence in forecasting the movement of a star than the movement of his own daughter. The reason is that the movement of the star is more predictable than the movement of his own daughter. To generalize, it can be said that some variables can be predicted to a considerable degree of certainty, and others are almost entirely unpredictable.

The other aspect, forecasting accuracy is of major interest to anyone concerned with the future. On one side, increasing forecasting accuracy could facilitate the saving of millions of dollars and becomes a major motivation for using formal forecasting methods. On the other, forecasting errors are inevitable, because forecasting is not a substitute for prophecy. Forecasters unfortunately do not possess crystal balls enabling them to look into the future. What is extremely important here, from a practical point of view, is to be able realistically to assess the advantages and limitations of forecasting methods and to take them into account while utilizing prediction for some purposes. A forecasting method with less error and greater accuracy should be adopted and appreciated.

Keeping in view the importance of forecasting, a study dealing with the prediction of exchange rates and stock returns is of obvious interest as both practitioners and policy makers always keep a close watch on the movement of exchange rates and stock prices. However, forecasting exchange rates and stock returns is rather a hazardous operation as these variables are notorious for their unpredictability, and are often characterized by high volatility, complexity, noise, nonstationarity, non-linearity and chaos. The view of their unpredictability has been further reinforced by the efficient market hypothesis (EMH), which found broad acceptance in the financial community.

According to efficient market hypothesis, a market is said to be efficient if price at any point of time contains all available information about the market. In its strong form, it says the current price of an asset reflects all publicly available information, while in its weak form it says the current price reflects all of the information gleaned from past prices of asset. Thus there is no arbitrage possibility, and the movement of a price is unpredictable. The EMH is based on the assumption that all news is promptly incorporated in prices; since news is unpredictable by definition, prices are unpredictable. The best prediction for a price is the current price and the actual prices follow what is called a random walk. In other words, it claims that exchange rates and stock prices follow a random walk and hence can not be predicted from their past prices. If a market is efficient, price will show a lower degree of volatility.

However, whether the financial markets are truly efficient and follow a random walk has always been a matter of empirical investigation. Much effort has been expended in trying to prove or disprove the EMH. Current opinion is that the theory has been disproved (Taylor, 1994 and Ingber, 1996) and much evidence suggests that financial markets are not efficient (Malkiel, 1996 and Lo *et al* 1999). Many financial theorists believe that the death-knell for the random walk model (for financial markets) occurred in 1987, October 19, when the collapse (in tandem) of the world equity markets occurred. Now, all the academics who were grounded in an efficient market model felt that there must be something drastically wrong with the theory if there could be a 500-point move in the Dow in one day. These markets could not be efficient if they reflected all available information on Friday, October 16. This ignited enthusiasm among financial economists to make an attempt to forecast exchange rates and stock prices. Various alternative linear models such as linear regression, vector autoregression (VAR), and ARIMA models were then used for forecasting these variables. But unfortunately these linear models failed to improve upon the simple random walk model in out-of-sample forecasting of both stock prices and exchange rate prices. The reason could be that all the models investigated were linear and it was natural to conjecture that exchange rate and stock price data contain non-linearities, which may not be accounted for or approximated well by linear models. This inevitably called for the emergence of a non-linear model, which could capture the non-linearities in financial variables.

Following this need, a variety of parametric non-linear models such as autoregressive conditional heteroskedasticity (ARCH), general autoregressive conditional heteroskedasticity (GARCH), and self-exciting threshold autoregression models have been proposed and applied to financial forecasting. While these models may be good for a specific data series, they do not have general appeal for other applications. Because there are too many possible nonlinear patterns, the pre-specification of the model form restricts the usefulness of these parametric nonlinear models.

Attempts to do away with these deficiencies gave birth to the use of **a** powerful nonparametric non-linear model called artificial neural network (ANN). Neural networks for forecasting exchange rates and stock prices have been proposed and examined in recent years. Broadly speaking, artificial neural network is a nonlinear, nonparametric and data driven modeling approach. It allows one to fully utilize the data and let the data determine the structure and parameters of a model without any restrictive parametric modeling assumptions. ANN is appealing in financial area because of the abundance of high quality financial data and the paucity of testable financial models. As the speed of computers increases and the cost of computing declines exponentially, this computer intensive method becomes attractive.

Neural network has some advantages over other linear and nonlinear models which make it attractive in financial modeling. First, neural network has flexible nonlinear function mapping capability which can approximate any continuous measurable function with arbitrarily desired accuracy (Hornik *et al,* 1989; and Cybenko, 1989), whereas most of the commonly used non-linear time series models do not have this property. Second, being a nonparametric and data-driven model, neural network imposes few prior assumptions on the underlying process from which data are generated. Because of this property, neural network is less susceptible to model misspecification problem than most parametric nonlinear methods. This is an important advantage since exchange rates and stock returns do not exhibit a specific nonlinear pattern. Third, neural network is adaptive in nature. The adaptivity implies that the network's generalization capabilities remain accurate and robust in a nonstationary environment whose characteristics may change over time. Fourth, **neural** network model uses only linearly many parameters, whereas traditional polynomial, spline, and trigonometric expansions use exponentially many parameters to achieve the same approximation rate (Barron, 1991).

1.1 Some Interesting Aspects of Exchange Rate and Stock Return

Since 1973, with the abandonment of the fixed exchange rates and the implementation of the floating exchange rate system by industrialized countries, researchers have been striving to explain the movement of exchange rates. Foreign exchange rates are affected by many highly correlated factors. These factors could be economic, political and even psychological factors. The interaction of these factors is in a very complex form. Consequently, to forecast the changes of foreign exchange rates is generally very difficult. Again, the interaction between economic systems of different countries and the consequences of globalization phenomenon in economies, increase the dimensionality of factors affecting exchange rates, which makes the accurate prediction of the exchange rate difficult.

Before going to the prediction of foreign exchange rate one should be aware of the general properties it possesses. The first property is nonlinearity. A number of empirical studies (e.g. Hsieh, 1989; De Grauwe *et al*, 1993; Brooks, 1996; and Drunat *et al*, 1996) have uncovered significant nonlinearities in nominal bilateral exchange rates and some authors have estimated nonlinear models for these nominal rates. Economic theory offers a number of potential explanations for the presence of non-linearities and cycles in exchange rates. Heterogeneity of participants in the foreign exchange market is often cited as the major source of nonlinearities in the exchange rate process. Market imperfections such as taxes, transaction costs and the timing of the information reaction introduce nonlinearities in the foreign exchange market. This means market participants do not trade every time news arrives in the market, rather they trade whenever it is economically feasible, leading to clustering of price changes. Furthermore, nonlinearities are observed when announcement of important factors are made less often than the sampling frequency. For example, weekly money supply announcements will cause nonlinearities in daily but not in monthly data. The other sources of nonlinearities are discrete regime shifts, time varying coefficients and data generating process that is inherently nonlinear.

Second, the exchange rates are found to be unconditionally leptokurtic. Leptokurtosis in exchange rates has been appreciated since Mandelbrot (1963). Similarly, Westerfield (1977), Boothe *et al* (1987), Hsieh (1988), and Diebold and Nerlove (1989) all show that exchange rates are leptokurtic. A number of explanations have been advanced for the leptokurtosis and volatility clustering. To cite one, Clark (1973) in his seminal work shows that subordination to an i.i.d. information-arrival process produces leptokurtic returns.

Third, it is widely agreed that exchange rate returns are conditionally heteroskedastic. Diebold (1988), for example, examines seven nominal dollar spot rates and finds little linear predictability but conditional heteroscedasticity implied by strong ARCH effects in all of them. Cumby and Obstfeld (1984), Domowitz and Hakkio (1985), Diebold (1988), Hsieh (1989) and Engel *et al* (1990) also find conditional heteroscedasticity in the residuals of exchange rates. The evidence of nonlinear dependencies, leptokurtosis and heteroscedasticity goes against the random walk hypothesis and further indicates the predictability of exchange rates.

Given these general properties, obviously benchmark models such as linear regression, ARIMA and random walk models are unlikely to give good predictions of the exchange rate. However, using a nonlinear model or introducing nonlinearities in exchange rate models does not necessarily improve the forecasting of exchange rates as these effects operate through even ordered moments. For example, using non-parametric kernel regression, Diebold and Nason (1990) and Meese and Rose (1990) were not able to improve upon a simple random walk in the out-of sample prediction of 10 major dollar spot rates in the post -1973 float period. Meese and Rose (1991) examine five structural exchange rate models to account for potential nonlinearities. Their results indicate that although nonlinear effects may be important in the even moments of the exchange rate dynamics, the incorporation of nonlinearities into the structural models of exchange rates does not improve our ability to understand exchange rate movements.

A number of explanations have been offered for this happening. One, of course, is that the nonlinearity may be present in even ordered conditional moments and therefore are not useful for point prediction. Only nonlinearity in conditional mean can improve the point prediction. Second, in-sample nonlinearities such as outliers and structural shifts may be present, and may cause rejection in various linearity tests while nevertheless being of no use for Out-of-sample forecasting. Third, very slight conditional-mean nonlinearities might be truly present and be detectable with large data sets, while nevertheless guiding negligible ex ante forecast improvement. Finally, even if conditional nonlinearities are present and are important, the overwhelming variety of plausible candidate nonlinear models makes determination of a good approximation to data generating process (DGP) a difficult task. The seemingly large variety of parametric nonlinear models that have received attention lately (e.g. bilinear, threshold and exponential autoregressive) is in fact a very small subset of the class of plausible nonlinear DGPs.

Similarly, stock returns are also well described as nonlinear, heteroscedastic and leptokurtic. Forecasting stock returns is also a difficult job for the same reasons cited in the case of exchange rates. A great deal of effort has been devoted to developing systems for predicting stock returns in the capital markets. Limited success has been achieved. It is believed that the main reason for this is that the structural relationship between an asset price and its determinants changes over time. However, using linear models Chen *et al* (1986), Campbell (1987), Fama and French (1988, 1989), Whitelaw (1994) find that stock returns are predictable by publicly available information such as time series data on financial economic variables. Their findings go against the efficient market hypothesis and random walk model, which advocate for the unpredictability of asset returns. More importantly, Hinich and Patterson (1985), Abhyankar *et al* (1997) have indicated the presence of structural nonlinear dynamics. Pesaran and Timmermann (1994) find significant nonlinear effects in quarterly and monthly regression of excess stock returns on economic variables and some non-linear terms such as the lagged values of the square of returns.

1.2 Need for the Present Study

We can summarize the above discussion to say that random walk hypothesis in financial community has been refuted but there also appears to be little linear predictability of exchange rates and stock returns. Both exchange rates and stock returns are seen to contain nonlinearities, which may not be accounted for or approximated well by linear models. However, it is difficult to zero in on a particular data generating process, given the large class of plausible nonlinear data generating processes. It is in this context that the use of artificial neural network, which does not presume any specific form of nonlinearity, gains importance. It would be of interest to see whether the predictability of exchange rates and stock returns can be improved upon by using artificial neural network, a nonlinear model. A number of studies have been carried out on expediency of neural network in predicting exchange rate and stock returns overseas, but very few of them are done in the context of India. The only studies that come to mind are Nag and Mitra (2002), Bordolai (2001) and Kamath (2001). There is a paucity in the number of studies, and even those studies that exist are not extensive enough to prove the credibility of neural network as an alternative model of forecasting financial variables in India. It is for this reason that a study exploring the possibility of artificial neural network in predicting exchange rates and stock returns assumes immense importance. The present work proposes to do just this.

1.3 Objectives of the Study

In the light of the above discussion, the chief objectives of the study are set as follows

To use artificial neural network in forecasting daily and weekly exchange rate returns and to compare its predictive ability with benchmark models such as linear autoregressive and random walk models.

- To use artificial neural network in forecasting daily and weekly stock returns and to compare its predictive ability with linear autoregressive and random walk models.
- To improve the generalization or Out-of-sample performance of neural network through Bayesian regularization and early stopping technique in forecasting exchange rate and stock returns.

1.4 Nature and Sources of Data

The study has made use of daily and weekly data on closing spot rates of Indian rupee/US dollar. We have used Indian rupee/US dollar exchange rate for our study keeping in view the fact that US is the major trading partner of India. And of course, the US dollar is regarded as the best hedging currency in the world. The data are collected from Pacific FX database. The daily data consisting of 2396 observations covers the period from January 4, 1993 to July 12, 2002. The weekly data consisting of 497 observations covers the period of January 6, 1993 to July 10, 2002. The study period has been chosen purely on the basis of availability of data, and keeping in mind that neural network estimation requires a long time series data. The exchange rate returns are calculated as the log first difference of the levels.

As far as stock market data is concerned, the study has taken BSE Sensitive Index (Sensex) as the representative. This comprises 30 companies from specified group in Bombay Stock Exchange constituting 150 shares. The selection is made on the basis of liquidity, depth, floating stock adjusted depth and industry representation. The compilation of the index is based on the weighted aggregate method. The data on daily and weekly closing values of BSE Sensitive Index are collected from web pages of Bombay stock exchange (BSE). The daily stock price data consisting of 2553 observations covers the period from January 2, 1991 to December 31, 2001. The weekly stock price data consisting of 558 observations covers the period from January 3, 1992 to November 8, 2002. Here too, the study period has been chosen on the basis of availability of a long time series data. The stock returns are calculated by taking the logarithmic differences between successive trading days.

1.5 Organization of the Thesis

The remainder of the thesis is organized as follows. In chapter II, alternative forecasting techniques are discussed. This includes a detailed discussion on artificial neural network and a brief discussion on linear autoregressive and random walk models. In this chapter, we have also reviewed some select works on the expediency of neural network in forecasting exchange rates and stock returns.

Chapter III presents our empirical findings on forecasting daily and weekly exchange rates by neural network, linear autoregressive and random walk models. The in-sample and Out-of-sample performances of three studied models are compared. Besides, the effects of forecast horizon on the in-sample and Out-of-sample performances of neural network, linear autoregressive and random walk models are carried out in this chapter. This chapter also tests the statistical significance of the forecasting results of all three studied models by using forecast encompassing test.

In chapter IV, we have compared in-sample and Out-of-sample forecasting of daily and weekly stock returns by neural network, linear autoregressive and random walk models. Forecast horizon effects and forecast encompassing tests are also performed in this chapter.

Chapter V is devoted to increase the Out-of-sample performance of neural network in forecasting exchange rate and stock returns by using Bayesian regularization and early stopping technique. Finally, chapter VI concludes the thesis with a summary of major findings and the implications of the study. The scope for further research is also discussed, together with the limitations of the present study.

Chapter II

Alternative Forecasting Techniques and a Review of Select Works

Training is everything. The peach was once a bitter almond; cauliflower is nothing but cabbage with a college education.

2.0 Introduction

It is widely agreed that foreign exchange rate markets and stock markets are characterized by high volatility, complexity, noise, nonstationarity, nonlinearity and chaos. As a result, it has been an uphill task for researchers to predict exchange rates and stock returns into the future. Indeed, some researchers believe that the prediction of exchange rates and stock returns is not feasible as these financial markets are efficient and the prices in these markets do follow random walk model. However, the random walk model has been disproved in the course of time, more particularly, after the collapse of the world equity markets which occurred on October 19, 1987. Following this, various linear and parametric nonlinear models have been proposed for the problem of forecasting exchange rates and stock returns. Again, the inefficiencies of these models led to the emergence of a nonlinear and nonparametric model called artificial neural network in financial modeling. This chapter delves into the applicability of neural network in exchange rates and stock returns prediction by discussing the underlying mechanism in neural network model and by reviewing some select works on its applicability.

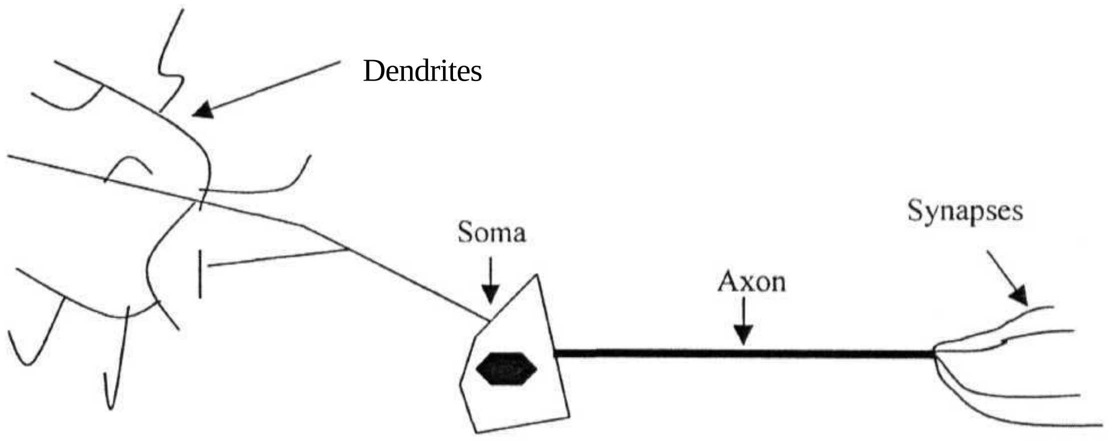
The rest of the chapter proceeds as follows. In section 2.1, we discuss the neural network model in detail. Sections 2.2 and 2.3 discuss, in brief, benchmark forecasting models such as linear autoregressive and random walk models respectively. In section 2.4, we present a review of select works on the application of neural network in exchange rates and stock returns prediction. Section 2.5 concludes the chapter.

2.1 Artificial Neural Network Model

A good analyst is not someone who is always right, but someone who is better on average, someone who has a higher efficiency than his colleagues. In the last few years it has become clear that artificial neural networks (ANNs) have become part of this class of analysts. Neural networks are programmes that are based on the geometry of the human brain. The theory was developed in 1943, when the first computers were not even produced. ANN has attracted the attention of researchers from a diverse field of applications including signal processing, medical imaging, and economic and financial modeling. Meanwhile researchers from cognitive science, neuro-science, psychology, biology, computer science, mathematics, physics and statistics have contributed to the structural and methodological developments of ANNs. This section is devoted to a discussion of the ANN and its operation. In subsection 2.1.1, we describe the artificial neural network. In subsection 2.1.2, we discuss how a neural network is trained. Subsection 2.1.3 discusses the generalization in neural network and finally subsection 2.1.4 is devoted to a discussion of the practical points to be kept in mind while designing an optimal network.

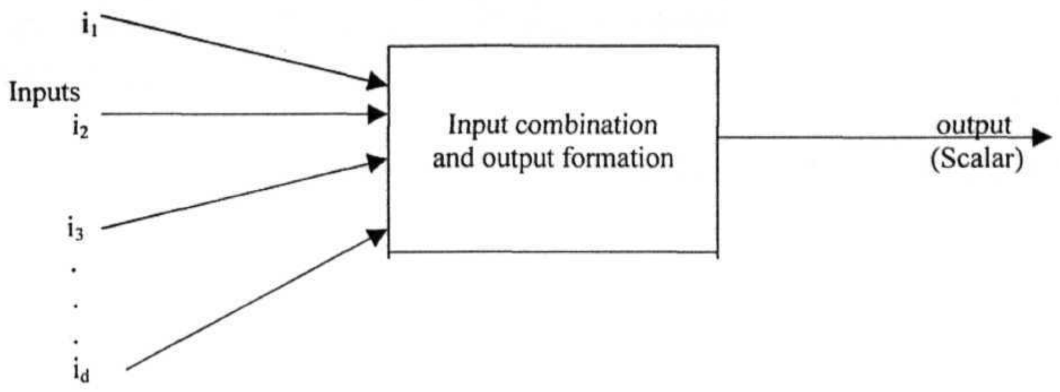
2.1.1 Description of Artificial Neural Network

ANN is nothing but a brain like computational system. The basic computing element in the human brain is the *neuron. A* neuron is a small cell that receives electrochemical stimuli from multiple sources and responds by generating electrical impulses that are transmitted to other neurons or effector cells. There is something in the order of 10^{10} to 10^{12} neurons in the human nervous system and each is capable of storing several bits of information. A neuron in the human brain consists of four parts: *dendrites, soma, axon,* and *synapses* (see Figure 2.1). The dendrites accept inputs, the soma processes the inputs, and the axon transfers processed data into outputs/stimuli. Synapses are electrochemical contacts between the different neurons that help in the transfer of stimuli. The neuron collects signals via its dendrites, which is then processed by the soma. The axon then transfers the signals into an inhibitory or exhibitory stimulus. A neuron is believed to carry out a simple *threshold* function. If the neuron is excited (i.e. the combined signal strength exceeds a certain threshold) it passes the stimuli on to its neighbors, if inhibited (i.e. the combined signal strength is less than a certain threshold) it kills the signal. Each passing of a stimulus produces a certain reaction. Over time, the reactions can be strengthened or weakened through repeated stimulation. A key feature of neurons in the brain is that they do not appear to regenerate. Due to this fact, neurons can retain prior experiences, and apply them to new situations. The neurons, dendrites, synapses, and soma in the human brain are analogous to units, inputs, interconnection weights, and transfer function (will be explained in detail later) respectively in the artificial neural network model.



(Figure 2.1 A biological neuron)

Artificial neural network model is inspired by three features of the way that biological neural networks process information. The first feature is massive parallelism. Second is nonlinear neural unit response to the neural unit input and the third feature is processing by multiple layers of neural units. In spite of getting the original inspiration from biological neural network, ANN is a mathematical model, which has little resemblance to biological neural systems. From an econometric perspective, it is a particular type of input-output model. Given an input vector *x*, the network produces an output vector *y*, say $y = g(x)$. The function *g* is determined by the network architecture. The resemblance of ANN with input-output model is shown in the Figure 2.2.



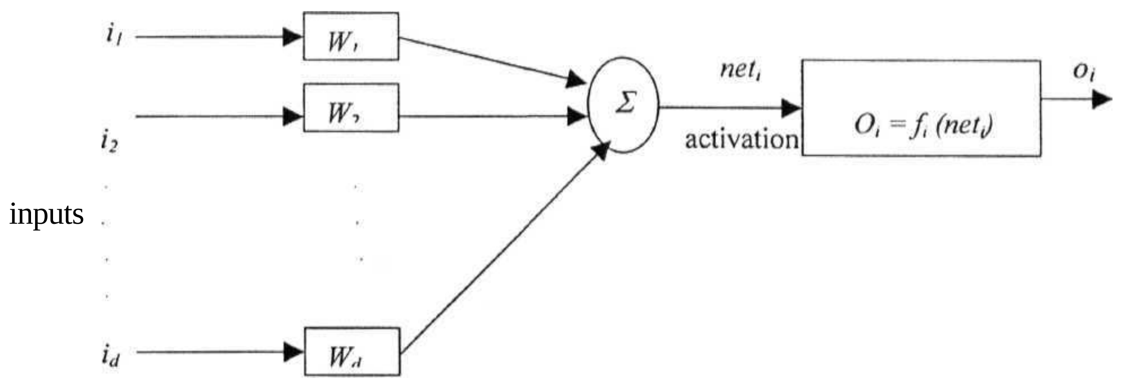
(Figure 2.2 Generic combination of inputs to produce corresponding artificial neuron output)

Many different networks have been developed for different purposes. One mode of classifying neural networks is according to the flow of information. This flow is either feedforward or feedbackward, which is called bidirectional or recurrent. Among these, feedforward networks have a wide applicability in forecasting problem and are most popular in financial application. We too use feedforward neural network for exchange rates and stock returns prediction. So, we spell this out in some detail.

The feedforward network is a hierarchical network, where the processing units are organized in a series of two or more mutually exclusive sets of neurons or layers. The first layer that contains the input units of the network is called *input layer*. The network receives its data in the input layer. The role of this layer is to feed input patterns into the rest of the network. The number of nodes (or neurons) in this layer depends on the number of inputs to a model and each input requires one neuron. Similarly, the layer where the output units of the network are located is called *output layer*. The output layer is the point at which the overall mapping of the network input is available. The output is an explicit function of the input: the input is propagated through the network and produces the output right away. The layers between the input layer and output layer are called *hidden layers*. There can be many hidden layers. They are analogous to the brain's inter-neurons, a place where the hidden correlations of the input and output data are captured. This allows the network to learn, adjust, and generalize from the previous learned facts to the new input. As each input-output set is presented to the network, the internal mapping is recorded in the hidden layer.

The feedforward network, as the name suggests, propagates signals only in the forward direction from input layer to the output layer through hidden layers. In the network each and every neuron in one layer (e.g. input layer) is connected with each and every neuron in the succeeding layer (e.g. hidden layer). More importantly, each connection has a numerical *weight*, which modifies the signals that pass through it.

A further concept we would like to explain is that of a *transfer or activation Junction.* Each unit in the hidden and output layers has a transfer function, which transfers the signals it receives to the following layers. That is, the transfer function is applied to the net activation unit, which is the weighted sum of all the inputs that a layer accepts from its previous layer. This can better be explained with the following figure where we consider a network without any hidden layer.



(Figure 2. 3 Linear activation formation)

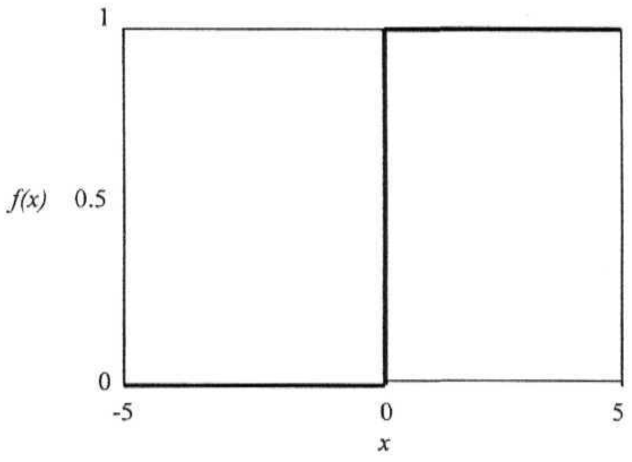
In the above figure, i_1, i_2, ...,i_d are inputs and w_1, w_2,.....w_d are network weights. net_i - $\sum_{i=1}^{d} w_i i_i$ is the net activation unit in the network. O_i is the network output whereas f_i is the transfer or net activation function in the network. Transfer function characterizes the mapping from net unit activation to output. In other words, it helps in converting the net unit activation into the output.

There are a number of different transfer functions such as threshold, piecewise linear, sigmoidal, signum, and linear function used in neural networks. Each transfer function may be suited for a particular type of problem at hand. Different transfer functions that are commonly used in neural network are:

Threshold Transfer Function

The threshold transfer function (see Figure 2.4) is written as

$$f(x) = \begin{cases} 1 & if\ x \geq 0 \\ 0 & if\ x < 0 \end{cases} \quad .... \qquad (2.1)$$



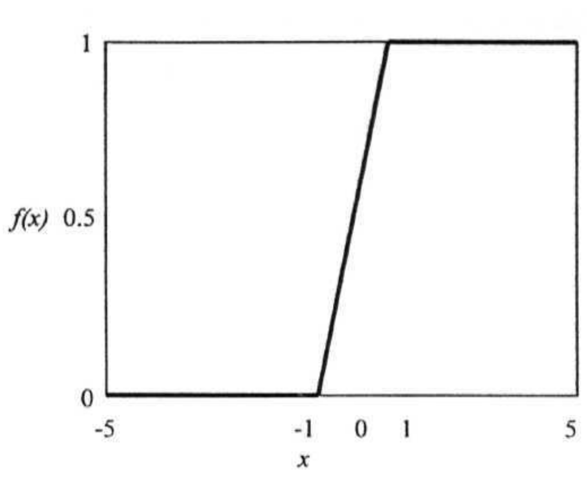
(Figure 2.4 The threshold transfer function)

Piecewise Linear Transfer function

The piecewise linear transfer function is written as

$$f(x) = \begin{cases} 1 & if\ x \geq a \\ \dfrac{x-b}{a-b} & if\ a > x > b \\ 0 & if\ x \leq b, \end{cases} \quad .... \qquad (2.2)$$

where *a* and *b* are threshold parameters. The piece wise linear transfer function is shown in the following figure.



(Figure 2.5 An example of a piecewise linear transfer function with a =1, b = -1, and $f(x)$ = 0.5 + 0.5x)

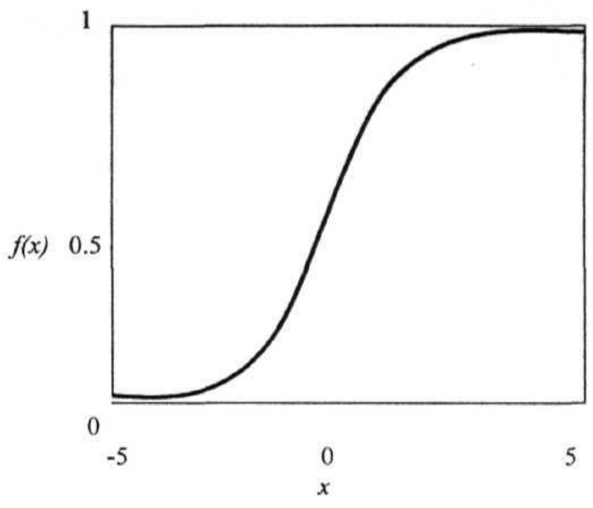
Sigmoidal Transfer Function

The sigmoidal transfer function, also called as the squashing function is written as follows.

$$f(x) = \frac{1}{1+e^{-ax-b}}, \quad \ldots \qquad (2.3)$$

where a and b are slope and location parameters respectively. As the slope parameter approaches infinity, the sigmoidal function becomes a threshold function. The sigmoidal function is also called a logistic function. The sigmoid function is the most commonly used transfer function in the neural network, because it is nonlinear and thus able to capture the nonlinearity in data. It is also continuous and differentiable everywhere. The advantage of this function is that its smoothness makes it easy to devise learning algorithms and understand the behaviour of large networks whose nodes compute such function[1].

Experimental observations of biological neurons demonstrate that the neuronal firing rate is roughly sigmoidal, when plotted against the net input to a neuron.

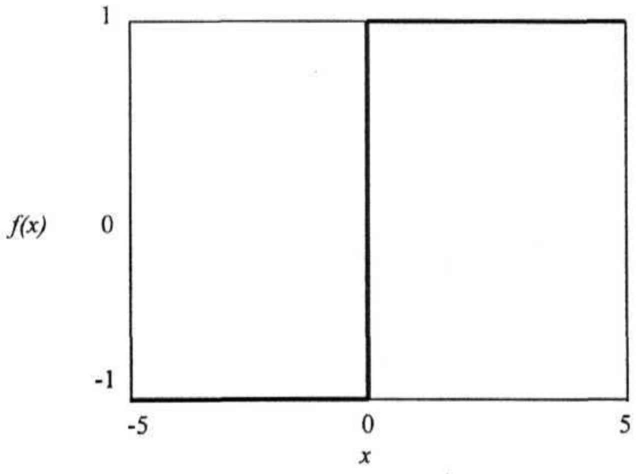


(Figure 2.6 The sigmoidal (logistic) transfer function)

Signum Transfer Function

For certain applications, it is desirable to have a transfer function that ranges from -1 to +1. This transfer function is called a signum function and it is defined by

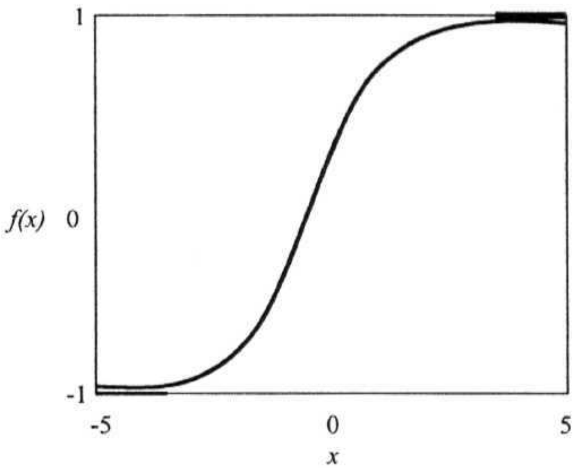
$$f(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0, \end{cases} \quad .... \qquad (2.4)$$



(Figure 2.7 The signum transfer function)

The continuous version of the signum function is defined by the hyperbolic tangent function, which is

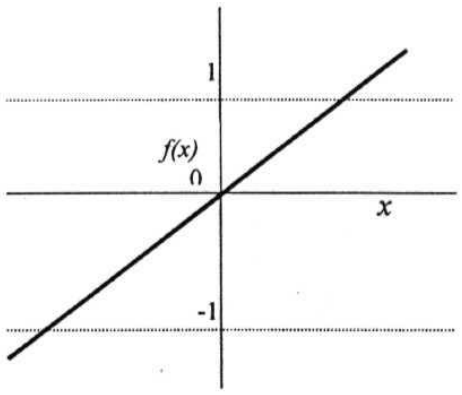
$$f(x) = \tanh(x) \quad \ldots.. \tag{2.5}$$



(Figure 2.8 The hyperbolic tangent transfer function)

Linear Transfer function

The simplest transfer function is the linear function, which simply outputs a value proportional to the summed inputs. The proportion is determined by the slope, 6. The transfer function is written *as f(x) = 8x.* However, a special case of linear transfer is the identity, which has output equals the input (see Figure 2.9). Unlike nonlinear functions, linear function does not have any restriction keeping the output between fixed bounds.



(Figure 2.9 The linear transfer function)

The feedforward network maps inputs into outputs with signals flowing in one direction only from the input layer to the hidden layer and then the output layer. Each unit in the hidden and output layers has a transfer function, which transfers the signal it receives. The input layer units do not have a transfer function but they are used to distribute input signals to the network.

The feedforward network could be of many layers. However, to explain the mechanism that is involved in feedforward network, a three-layer feedforward network with a single output unit, k hidden layer units and n input units is shown in Figure 2.10. The input layer is represented by the vector $X = (x_1, x_2, ..., x_n)'$, the hidden layer by a vector $M = (m_1, m_2, ..., m_k)'$ and y is the output. Each unit in the hidden layer receives the weighted sum of all inputs and a constant term called as bias term (denoted by x_0, which always equals one) to form the total incoming signals and applies a function ψ to this total incoming signal to construct the output of the unit.

$$m_j = \psi(\sum_{i=0}^{n} \beta_{ij} x_i) = \psi(X'\beta_j), \quad j = 1,2,........,k \tag{2.6}$$

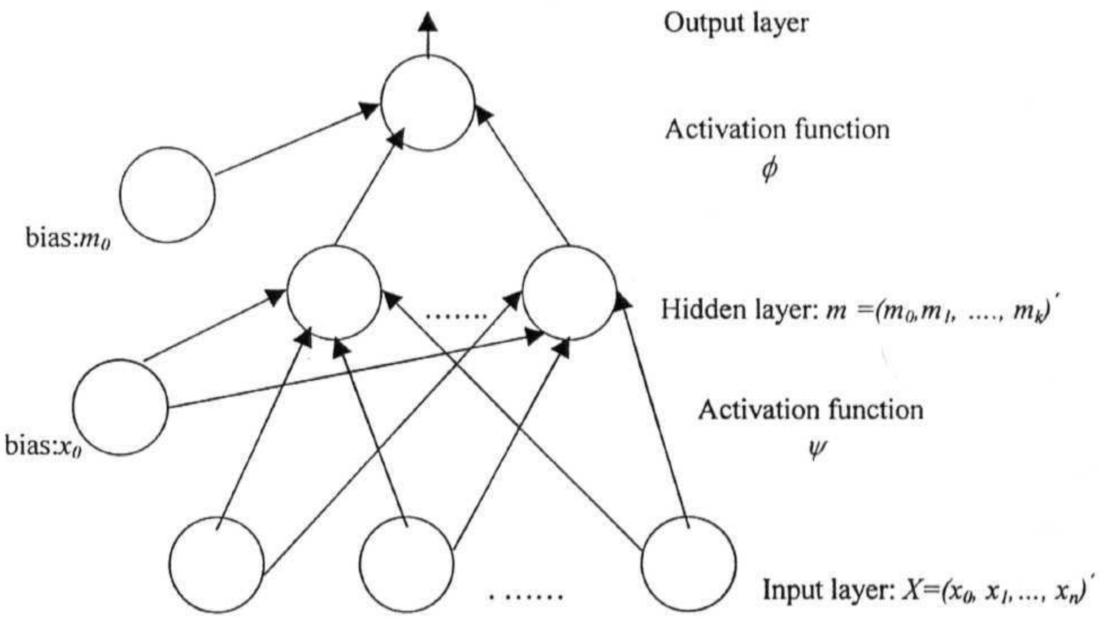
where ψ is the transfer function, x_i is the i^{th} input signal, and β_{ij} is the weight of the connection from i^{th} input unit to the j^{th} hidden layer unit. In the same way, the output unit receives the weighted sum of the output signals of the hidden layer units, and applies a function ϕ to produce a signal

$$y = \phi(\sum_{j=0}^{k} \alpha_j m_j) \tag{2.7}$$

where ϕ is the output layer unit transfer function, α_j is the weight of the connection from the j^{th} hidden layer unit to the output unit, and $j = 0$ indexes a bias unit m_0 which always equals one. Substituting (2.6) into (2.7), we get

$$y = \phi(\alpha_0 + \sum_{j=1}^{k} \alpha_j \psi \ (\sum_{i=0}^{n} \beta_{ij} x_i)) = f(X, \theta) \tag{2.8}$$

where X is the vector of inputs, and $\theta = (\alpha_0, \alpha_1, \alpha_2 .... \alpha_k, \beta_{01}, \beta_{02}, ... \beta_{0k}, \beta_{11}, \beta_{12} .... \beta_{1k}, \beta_{n1}, \beta_{n2}, ... \beta_{nk})$ is the vector of network weights. The equation 2.8 can be considered as nonlinear regression. The transfer functions ψ and ϕ can take several functional forms as we have discussed earlier. However, Hornik *et al* (1988) proved that single hidden-layer feedforward network with ϕ the identity map and ψ an arbitrary sigmoid function can approximate any measurable function to any accuracy, given sufficiently many hidden units. Thus the neural network has the ability to provide a flexible mapping between inputs and outputs, consistent with any underlying relationship, whatever its true form. This eliminates the need for *a priori* restrictions, which are used in conventional econometric modeling. It is for this reason that a neural network may be viewed as a 'universal approximator'.


Output layer
Activation function
ϕ
bias:m_0
Hidden layer: $m = (m_0, m_1, \ldots, m_k)'$
Activation function
ψ
bias:x_0
Input layer: $X = (x_0, x_1, \ldots, x_n)'$

(Figure 2.10 A three-layer feedforward network)

2.1.2 Training in Neural Network

Neural networks differ from conventional programs in that they learn to solve problems. The fundamental feature of neural network is that it is trained, not programmed. Having discussed the ANN structure, we now move on to discuss the training or learning in neural network. Training or learning in neural network takes through place the correction of the error based on a measurement of the difference between desired and actual feedforward network output. And the corrections are made through the adjustment of the network weights. Through the iterative training process, connection weights rearrange their values and reveal a data pattern.

A neural network learns with each new datum (an input-output combination) introduced into it during the training. Every processing element responds to its inputs, adjusting its behavior. The network calculates the output in accordance with the elements' transfer function. The only way to adjust to the correct response is to modify the values of the input connections. The network learns by adjusting the input weights. The equation that explains this change is called the learning law.

There are two different learning modes: supervised and unsupervised. The supervised learning mode presents input-output data combinations to the network. Consequently, the connection weights, initially randomly distributed adjust their values to produce output that is as close as possible to the actual output. With each subsequent cycle or iteration, the error between the desired and the actual output will be lower. Eventually, the result is a minimized error between the network and actual output, as well as the internal network structure, which represents the general input-output dependence. In a one-layered network, it is easy to control each individual neuron and observe the input-output pattern. In multi-layered networks, supervised learning becomes difficult. It is harder to monitor and correct neurons in hidden layers. Supervised learning is frequently used for network decision, memorization and generalization problems.

The unsupervised learning mode is independent of the external influences to adjust weights. It is self-organization oriented. There are no concrete data to correct the neural networks' pattern identification. Therefore, the unsupervised learning mode looks for the trend in inputs and adapts to the network function. Comprehensive reviews of these procedures and ways of converting one kind of learning procedure into another can be found in Hinton (1987) and Lippmann (1987). However, here, we will only concentrate on the most commonly used procedure in the problem of financial variable prediction: namely supervised learning by error backpropagation.

Error Backpropagation Algorithm

The most popular learning or training algorithm is the backpropagation-based generalized delta rule (GDR). Back propagation is a recursive gradient descent method in which weights in the network are iteratively modified to minimize the overall mean square error between the desired and actual output values for all output units over all input patterns. More specifically, network weight vector 9 in (2.8) is chosen to minimize the loss function. That is,

$$\hat{\theta} = \arg\min E \tag{2.9}$$

where E is the loss function and is given by

$$E = 1/N \sum_{t=1}^{N} (y_t - \hat{y}_t)^2 \tag{2.10}$$

where N is the sample size, y_t is the desired output and y_t is the calculated output value,

$$\hat{y}_t = f(X_t, \hat{\theta}) = \phi(\hat{\alpha}_0 + \sum_{j=1}^{k} \hat{\alpha}_j \psi(\sum_{i=0}^{n} \hat{\beta}_{ij} x_t)) \quad \ldots.$$

GDR is a weight correction procedure that adjusts the weights in proportion to a reduction in the error relative to the changes in the weights. The weights will be changed on each successive pattern presentation such that pattern errors are iteratively reduced from their previous values. This can be accomplished if the weights are adjusted in proportion to the negative of the error gradient. This can be written as

$$\Delta\theta = -\eta \frac{\partial E}{\partial \theta} \quad \ldots. \tag{2.12}$$

where E is the error function, defined in (2.10). By applying the chain rule we can write $\frac{dE}{\partial\theta}$ as:

$$\frac{\partial E}{\partial \theta} = \frac{\partial E}{\partial (\sum_{j=0}^{k} \alpha_j m_j)} \frac{\partial (\sum_{j=0}^{k} \alpha_j m_j)}{\partial \theta} \quad .... \tag{2.13}$$

where $\sum \alpha_j m_j$ (see (2.7)) is the weighted sum of inputs to the output units from the hidden layer units. Now, using the chain rule again, we can write

$$\frac{\partial E}{\partial (\sum_{j=0}^{k} \alpha_j m_j)} = \frac{\partial E}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial (\sum_{j=0}^{k} \alpha_j m_j)} \tag{2.14}$$

where y_t is the calculated output value (see (2.11)). But we have

$$\frac{\partial E}{\partial \hat{y}_t} = -(y_t - \hat{y}_t) \quad ..... \tag{2.15}$$

Now, substituting (2.15) into (2.14) and then (2.14) into (2.13), we get

$$\frac{\partial E}{\partial \theta} = -(y_t - \hat{y}_t) \frac{\partial \hat{y}_t}{\partial (\sum_{j=0}^{k} \alpha_j m_j)} \frac{\partial (\sum_{j=0}^{k} \alpha_j m_j)}{\partial \theta} \quad .... \tag{2.16}$$

Using (2.16) in (2.12) and rewriting the equation, finally, we get

$$\Delta \hat{\theta} = \eta \nabla f(X_t, \hat{\theta})(y_t - f(X_t, \hat{\theta})) \quad .... \tag{2.17}$$

where 77 is the learning rate, *∇f(X$_t$, 6)* is the gradient of *f(X$_t$, 6)* with respect to θ and the chain rule is used to calculate *∇ f(X$_t$, 6)*. The most important parameter in the equation (2.17) is the learning rate 77. Learning rate determines what amount of the calculated error sensitivity to weight change will be used for the weight correction. The best value of the learning rate depends on the characteristics of the error surface i.e. a plot of error *E* versus the weight θ. For a network with linear output units and no hidden units, the error surface always forms a bowl whose horizontal cross-sections are ellipses and whose vertical cross-sections are parabolas. Since the bowl has only one minimum (perhaps a complete subspace but nevertheless only one), gradient descent on the error surface is guaranteed to find it. If the output unit has a nonlinear but monotonic transfer function, the bowl is deformed but still has only one minimum. So gradient descent still works. However, with hidden units, the error surface may contain many local minima, so it is possible that steepest descent in weight space will be trapped in poor local minima. As a result, training the network often requires experimentation with different starting weights, adjusting the learning rate or adding a momentum term to avoid getting stuck in local optima or slow convergence.

Adding a momentum term λ in equation (2.17), we get

$$\Delta\hat{\theta} = \eta \nabla f(X_t,\hat{\theta})(y_t - f(X_t,\hat{\theta})) + \lambda\left(\Delta\hat{\theta}(old)\right) \qquad \textbf{(2.18)}$$

where $\Delta\theta(old)$ represents the weight change in the previous training iteration. The momentum coefficient encourages the movement of weights in the same direction on successive steps. By doing so, it speeds up the learning process and ensures a faster convergence. Learning is complete and the weights in the neural network are no longer modified when either the pre-specified maximum number of iterations or the convergence has been reached.

2.1.3 Generalization in Neural Network

The ultimate performance of the network depends upon its ability to generalize the regularities in the training set to unseen patterns. Generalization is a process of describing the whole from some of the parts, reasoning from the specific to the general case, or defining a class of objects from a knowledge of one or more instances. ANN generalizes when it computes or recalls full patterns from partial or noisy input patterns, when it recognizes or classifies objects not previously trained on, or when it predicts new outcomes from past behaviors. The ability to classify objects not previously trained on is a form of interpolation between trained patterns. The ability to predict from past observations is a form of extrapolation. Both of these types of mappings are a form of generalization.

Network learnability relates to the ability of a learning algorithm to find a set of weights, which give the accuracy needed for a good mapping approximation. In other words, network training is the process of finding weights through sample training, which, when applied to unseen, outside training set (i.e.testing set) data will produce the desired output; i.e. will produce good generalization. Such realization depends on the network architecture, the learning algorithm, the training set and the training process. Good generalization is possible only if the training set contains enough information relative to the concept class being learned, and if the learning algorithm is capable of extracting the information and formulating the desired function within a reasonable time for the given network complexity. Although they are closely related, generalization depends more on information content, whereas learnability relates more to network and computation complexities.

A good generalization indicates the network works well on the unseen, outside the training set, called as testing set. A good generalization indicates a good performance and a poor generalization indicates a poor performance of network. The possible source of poor generalization is *overfilling.* Overfitting refers to a situation where the neural network is overtrained or has too high a complexity. A network is said to be overtrained if it is trained for more than the necessary amount of iterations. If a network is overtrained, then instead of recognizing the patterns in the training sets, it starts memorizing the training sets. The neural network starts fitting properties in the training data that are not general but rather to be considered as noise and consequently displays poor generalization. A high complexity means the network has too many hidden layers and hidden nodes. If the network has too high a complexity then it draws too many characteristics from the data set at hand in the training process, which, when applied to the outside training set data, decreases the performance of network on that set. In both the situation of overtraining and high network complexity, the error on the training set is driven to a very small value, but when new data is presented to the network the error is large. As a result, the network produces poor generalization.

Generalization and Overfitting are two concepts in neural network that are directly related to bias-variance dilemma in statistics. Although the trade-off between bias and variance is well known in statistics, its relevance for neural network has been explicitly pointed out only recently (Geman et al., 1992). A network that overfits the data sample has low bias and high variance. Consequently, the network displays bad generalization behaviour. The other reason for poor generalization could be that the patterns or informations contained in the training set may not be present in the testing set. But our interest is in a neural network that generalizes well. So, when trying with neural network, one has to be careful to get a good generalization. Different techniques such as early stopping, cross-validation, weight elimination, weight decaying, and Bayesian regularization have been developed to prevent Overfitting and to ensure a good generalization.

2.1.4 Designing an Optimal Network in Practice

In view of the preceding discussion, it can be said that a neural network's performance greatly depends on the neural network design such as network architecture (i.e. number of hidden layers, hidden units and input units), weight initialization, transfer functions, cost function and setting of values for learning rate and momentum coefficient. Designing a neural network has got some practical problems because, unfortunately, there is no single method to choose the optimum network structure that applies to all cases. Failures of neural network in applications are sometimes due to suboptimal structure. To develop the optimal neural network in any financial application, we need to (i) identify the relevant inputs and outputs (ii) choose appropriate number of hidden layers and hidden units (iii) select a proper weight initialization method (iv) select appropriate transfer function in the hidden layers and output layer and finally (v) choose appropriate learning and momentum coefficient. We discuss these points one by one.

Inputs and Outputs

The choice of network inputs and outputs and the quality of data are critical to the success of neural network applications. The choice depends heavily on the type of task that neural network is expected to perform. While the input choice is dictated by theoretical considerations, it is more or less subjective to the modeler's discretion and of course the scope of the study. It is common practice to use independent variables as network's inputs and use dependent variables as network outputs in a model. The quality of data and how well the data in the training set represents the population is of obvious importance. This is as true of neural network as any statistical modeling. To train and test a neural network, it is also important to have a sufficiently large data set.

Number of Hidden Layers and Hidden Nodes

A network, which consists simply of an input layer and an output layer, with no hidden layer, is known as a linear perceptron or often just a perceptron. Perceptrons are only capable of modeling linear functions and are consequently rarely mentioned. For nonlinear modeling, we require at least one hidden layer. However, choosing the appropriate number of hidden layers again is a difficult task. The number of hidden layers is determined by a trade-off between network intuitive ability and efficiency. *A priori,* the optimal number of hidden layers is not clear. With too many hidden layers, an over correcting problem arises: a network is "overfitted", which prevents it from learning a general solution. On the other hand, too few layers will inhibit the learning of the input-output pattern. Typically, the number of hidden layers and nodes inside the network is determined through experimentation. However, Kurkova (1991) uses Kolmogorov's theorem to show that any function can be approximated by at most four layers. Hecht-Neilson (1987) shows the maximum to be three but stales that in reality the use of more layers could greatly reduce the total number of hidden units required in the hidden layers. Cybenko (1989) proves that neural network with at least two hidden layers can approximate a particular set of functions with arbitrary accuracy given enough units per layer. It has also been proved that only one hidden layer is enough to approximate any continuous function with arbitrarily desired accuracy provided it has sufficient hidden units (Hornik *et al* 1989; and Cybenko, 1989). Many empirical studies such as Collins *et al* (1988), and Dutta and Shekhar (1988) have confirmed this. The correctness of these results, however, hinge on the appropriate number of hidden nodes.

With too few hidden nodes, neural network may not have enough power to approximate a function at desired accuracy. With a large number of nodes, neural network may overfit and thus tend to perform poorly on new test samples. Therefore the choice of number of hidden nodes represents a compromise or a trade-off between network intuitive ability and efficiency. *A priori,* the optimal number of hidden nodes is not clear. Trial-and-error is a common method to determine the number of hidden nodes. Cross validation (White, 1990) and the predictive stochastic complexity (Kuan and Liu, 1995) are useful methods to select the number of hidden units. Different constructive techniques such as cascade correlation (Fahlman and Lebiere, 1990), tiling algorithm (Mezard and Nadal, 1989), neural decision tree (Gallant, 1986) and the CLS procedure (Refenes and Vithlani, 1991) construct the hidden units in layers one by one as they arc needed. Neural network pruning (Sietsma and Dow, 1991) sometimes is also used for this job. Recently, genetic algorithm has come into increasing use to find the optimum number of hidden nodes. Besides these techniques, several rules of thumb have also been cited, for instance, the number of connections should be less than 0.1N and the number of hidden units is of the order of (N-l) or $\log_2 N$, where N is the sample size.

Weight Initialization

Before starting any training process, the weights of a neural network should be set to an initial value. The choice of values selected plays an important role in neural network as it affects the rate of convergence. A choice of initial values of weights near the optimum weight values can result in rapid convergence. On the other hand, a poorly chosen assignment can result in tens of thousands of iterations before convergence or even failure to converge. For example, initially setting all weights to the same value will result in failure because all hidden units connected to output units will receive identical error update values since the back propagated error is proportional to the weights. Because weight adjustments depend on the backpropagated error signals, all adjustments made to the systems will be the same and the system will remain stuck at a point that keeps the weights from changing. Similarly, many other choices of initial weights may also cause the learning process to get stuck in a flat region on the error-weight surface.

A classical approach for weight initialization is to choose small random values, say between -1 to 1, which is an effective way to avoid shallow troughs and possible entrapment at the start of the training process and also ensures that no sigmoid units will be saturated (resulting in very low value for the derivative). If for instance a sigmoidal unit has its highest derivative in the range 1-3, 3], an appropriate choice for initial weight would be to draw a random number in the range [-3/sqrt (T), 3/sqrt (T)] where T is number of inputs to the neuron. Furthermore, a small initial weight value will result in relatively small weight changes since the absolute value of the weight is also taken into account. Consequently, there must be a trade-off between small absolute values and a good value range among the sigmoids.

Such rules are designed to ensure that the network will start in its 'linear' mode and not on a flat part of the error-weight space. Nevertheless, there is no guarantee that this initial guess will lead to the global minimum nor lower the convergence time. Alternative methods have been developed to create an ever better initial solution in order to reduce training time, and improve robustness in respect of local minima. For instance, Denoeux and Lengelle developed a method to initialize the weights with prototypes derived from the training set (Denoeux and Lengelle, 1993). Classical advanced weight initialization uses principal component analysis information, but this requires of course less hidden units than the actual number of inputs (Wurtz and De Groot, 1992). Another technique of weight initialization is given by Nguyen and Widrow (1990), which generates initial weight values for a layer so that the active regions of the layer's neurons will be distributed roughly evenly over the input space. It has several advantages over purely random weights: (1) few neurons are wasted (since the active regions of all the neurons are in the input space), (2) training works faster (since each area of the input space has active neuron regions). The initial solution is of course a major concern while using a training algorithm such as backpropagation and is the first step required to avoid being trapped in a local minimum.

Transfer Function

Another major determinant of neural network's performance is the choice of transfer function. Different transfer functions that are used in neural network are shown in earlier section of this chapter. The transfer function is the rule for mapping the neuron's summed input to its output, and by suitable choice, is the means of introducing nonlinearity into the network design. A range of possible transfer functions may be utilized, the necessary condition being that they should be differentiable functions. In practice the functions are also chosen to be monotonic and to saturate at the two extremes of [0, 1] or [-1, 1]. Exceptions are cyclical functions and Gaussian functions.

The most widely used transfer function is the logistic or sigmoid function which squashes the input x into the range from 0 to 1. A variant of logistic function which is also used as transfer function is tanh function with the difference that it squashes the input x into the range -1 to 1. The linear transfer function is also used as transfer function in neural network. In fact Morris *et al* (1990) suggested that one of the hidden units should always be linear. This is because most problems may well have linear components and it is very hard to model a straight line by adding together a number of weighted nonlinear transfer functions. Now, the problem is to choose appropriate transfer functions for a given problem. However, in financial applications of neural network, logistic functions are very often used as transfer functions for the hidden layer units and linear functions are used as transfer functions for units in the output layer. Hornik, Stinchcombe, and White (1989) proved that single hidden-layer feedforward network with logistic transfer function for the hidden layer units and the linear transfer function for the output layer unit can approximate any measurable function to any accuracy, given sufficiently many hidden units.

Learning and Momentum Coefficient

The learning rate and momentum term that are used in the weight update rule are shown in (2.17) and (2.18). The learning rate coefficient 77 in the equation (2.17) determines the size of the weight adjustments made at each iteration and hence influences the rate of convergence. The value of 77 is important since large variations in learning rate can result with different choices of 77. A poor choice can result in failure to converge. It is also known that 77 should not be constant through out the training process for best results. This is because error surface that has broad flat shapes near minima require large coefficient values for rapid convergence. While steep narrow shapes near minima require small values of 77 to avoid overshooting of the solution. If the chosen value of 77 is too large for the error surface, the search path will oscillate about the ideal path and converge more slowly than a direct descent. It may even diverge. On the other hand, if the chosen value of 77 is too small then the descent will progress in very small steps significantly increasing the total time to converge.

The problem of finding appropriate learning rate can be viewed as a specific instance of stability-plasticity dilemma which is expressed by the problem of how a learning system can be designed to remain plastic in response to significant new events, yet also remain stable in response to irrelevant events. Learning rate adjustment continues an attempt to resolve this dilemma. In principle, there are two approaches to learning rate adjustment: first, to use one learning rate for the entire network; second and more sophisticated, to use one learning rate for each weight. In the latter case, the general heuristics of how learning rate update has to be performed can be described as follows. If consecutive changes of a weight have the same sign, then the learning rate associated with this should be incremented. If consecutive changes of a weight have different signs (that is they oscillate around the optimal change), then the associated learning rate should be decremented. A critical problem arising from the use of modifiable learning rates, one for each direction, is that overall learning may degenerate to local learning. Furthermore, using modifiable learning rates leads to the distinction between two types of adoption rule: weight adoption rules and learning rate adoption rules.

The momentum term in (2.18) is also very crucial in the training performance because it improves the rate of convergence and prevents the network from getting stuck in local minima. Back propagation leads the weights in a neural network to a local minimum of the error; possibly substantially different from the global minimum that corresponds to the best choice of weights. 7 his problem can be particularly troublesome if the error surface is highly uneven or jagged with a large number of local minima. We can prevent a network from getting stuck in local minima by adding some inertia or momentum to the gradient expression. This can be accomplished by adding a fraction of the previous weight change to the current weight change. The addition of such a term can help smooth out the descent path by preventing extreme changes in the gradient due to local anomalies. It can act as an averaging effect which smoothes the trajectory of the gradient as it moves downhill. A commonly used update rule introduced by Rumelhart *et al* (1986) includes such a momentum term, which is used in (2.18) is reproduced below.

$$\Delta\hat{\theta} = \eta\,\nabla f(X_t,\hat{\theta})\,(y_t - f(X_t,\hat{\theta})) + \lambda\left(\Delta\hat{\theta}(old)\right) \quad .... \qquad (2.19)$$

Here *A* is the momentum coefficient. It should be noted that the above equation is a difference equation, and hence, that the momentum term is effectively an exponentially weighted sum of a weight's current and past partial derivatives. This term encourages movement in the same direction on successive steps. Therefore it tends to suppress any oscillations that result from changes in the slope of the error surface.

However, the choice of appropriate value for momentum term is also a difficult task. Typical values lie in the range [0.5, 0.9], but for some problems (see Fahlman, 1988) a value of *k* = 0 was shown to be best. To be most effective, the momentum coefficient should be adjusted dynamically, as also the learning coefficient. But because the two terms are not independent, the two values should be chosen and modified as a pair. For many standard problems, the learning coefficient should start out large and gradually decrease as the error approaches a minimum. Conversely, the momentum coefficient should start out small and gradually increase as the error approaches a solution. In this way a direction for descent will be found and rapid progress made at the start of the search with a slowdown occurring near overshoot. But again it should be emphasized that the values of both parameters are problem dependent and to generalize too freely in this regard can be misleading. So we may require some trial and error before a good choice is found.

Cost Function

The goal of training a neural network is to minimize the error, which the network makes at each output unit over the entire training data set. For that reason we have to specify a proper error measure or cost function which would lead to faster convergence and a good generalization. For a backpropagation type of training method, we do require that the cost function chosen should be differentiable and tend to zero as the collective differences between the actual and desired outputs decrease over the entire training set. The most popular cost function, which is generally used in neural network, is mean square error as defined in (2.10). This cost function is often chosen because of its statistical properties and because it is better understood than other measures. It is a non-negative differentiable function that penalizes large errors more than small ones. Its popularity stems from its similarity to other statistical measures such as the least squares criteria in curve fitting (i.e. in regression). It is also similar to the sample correlation, which tends to one as the difference between the desired and actual output decreases.

The other cost functions that are used in neural network are mean absolute error, non-euclidian, cross-entropy (-1, 1), and cross entropy (0, 1). The mean absolute error is the least sensitive to the extremes in the distribution of the data usually associated with noise. The cross-entropy (-1, 1) cost function is suitable for output neuron transfer function with range (-1, 1), for example, tanh. Whereas cross-entropy (0, 1) is suitable for output neuron transfer function with range (0, 1), for example, sigmoid.

2.2 Linear Autoregressive Model

Linear autoregressive model (LAR), which is one specific form of a general multiple linear regression model uses the lagged values of the dependent variable as explanatory variables[2]. A linear autoregressive model of order *p* is defined as

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \dots\dots + \beta_p Y_{t-p} + u_t \quad \dots. \qquad (2.20)$$

where $\beta_0, \beta_1, \beta_2, \dots, \beta_p$ are real coefficients and u_t is the stochastic disturbance term with $E(u_t) = 0$, $E(u_t, u_s) = 0$ for $s \neq t$ and $E(u_t^2) = \sigma$ for all t. The main difficulty in forming such models, and then using them to forecast, is to decide the order of the model.

2.3 Random Walk Model

Another benchmark forecasting model, which we use in this study, is random walk model (RW). The random walk model is which is otherwise known as weak efficient market hypothesis talks of no change in forecasting. The best prediction value for tomorrow is the current value. A series Y_t is said to follow a random walk if

$$Y_t = Y_{t-1} + \varepsilon_t \quad \dots.$$

where e is the white noise error term with $E(\varepsilon_t) = 0$, $E(\varepsilon_t, \varepsilon_s) = 0$ for $s \neq t$ and $E(\varepsilon^2) = \sigma$ for all t. Allowance for a trend (or drift) can be made by including a constant (d)

$$Y_t = Y_{t-1} + d + \varepsilon_t \quad \dots. \qquad (2.22)$$

[2] We have not included any macro economic variables, which determine the exchange rate and stock returns in the set of explanatory variables, as the data on these variables are not available on daily and weekly basis.

Assuming no time trend (i.e. $d=0$) a forecast can be made for period $t+1$

$$\begin{aligned} E_t(Y_{t+1}) &= E(Y_t, \ldots) \\ &= Y_t + E(\varepsilon_t) \\ &= Y_t \quad \ldots. \end{aligned} \tag{2.23}$$

Similarly the forecast two periods ahead

$$\begin{aligned} E_t(Y_{t+2}) &= E(Y_{t+1}, \ldots) \\ &= E_t(Y_{t+1} + \varepsilon_{t+2}) \\ &= E_t(Y_t + \varepsilon_{t+2} + \varepsilon_{t+1}) \\ &= Y_t \quad \ldots. \end{aligned} \tag{2.24}$$

Consequently, if prices do follow a random walk then today's price provides a forecast of future prices since the price will only change in response to the receipt by the market of new information, which is, by definition, not forecastable. A slight modification is necessary if a time trend exists so that $d \neq 0$, then $E_t(Y_{t+1}) - Y_t + d$ and $E_t(Y_{t+2}) = Y_t + 2d$, etc.

2.4 Application of ANN in Exchange Rate and Stock Returns Prediction: A Review of Select Works

Given the advantages of neural network, as discussed in chapter I, over linear as well as parametric nonlinear models, it is not surprising that this model has been applied extensively in several financial areas. In this section, we review some relevant empirical studies on the usefulness of ANN in predicting the exchange rates and stock returns. First, we present the review on exchange rate prediction followed by the review on stock returns prediction.

2.4.1 Studies on Exchange Rate Returns Prediction

Refenes (1993) developed a constructive learning algorithm to find the optimum neural network configuration to predict the exchange rate between the U.S. dollar and Deutsche mark. The constructive learning method started with a fixed architecture with one hidden node. Then, hidden nodes were added to the network one by one incrementally until the performance of the network could not be improved. The hourly exchange rate data for the period of 1988 through 1989 was divided into a training set comprised of observations of the first 200 trading days and a test set with the remaining 60 trading days. Although this forecasting problem was very difficult since the training set had a general positive slope while the test set had a negative slope, neural networks performed well in both multiple-step and single-step predictions. In addition, neural networks were much better in multiple-step forecasting than exponential smoothing and autoregressive models.

Kuan and Liu (1995) investigated the Out-of-sample forecasting ability of feed forward and recurrent neural networks on five exchange rates against the US dollar, including the British pound, the Canadian dollar, the Deustche mark, the Japanese yen, and the Swiss franc. The data used were daily opening bid prices of the New York Foreign Exchange Market from 1st March 1980 to 28th January 1985, consisting of 1245 observations. A two-step procedure was used to select the suitable network. First, networks were selected based on the Predictive Stochastic Complexity criterion. Then the selected networks were estimated using both recursive Newton algorithms and nonlinear least square methods. Their empirical results showed that ANN had significant market timing ability (sign predictions) and/or significantly lower Out-of-sample mean square error relative to the random walk model for two out of the five series i.e. for the Japanese yen and British pound. For other series, network models did not exhibit superior forecasting performance. The results also showed that non-linearity in exchange rates may be exploited to improve both point and sign forecasts. This is in contrast with the conclusion of Diebold and Nason (1990) who found that nonlinearity in exchange rates cannot be exploited to improve forecasts. Their results were also different from those in Tsibouris (1993), in which neural network was found to be useful in forecasting the direction of the exchange rate change, but not the magnitude.

Wu (1995) conducted a comparative study between neural networks and AR1MA models in forecasting the Taiwan/U.S. dollar exchange rate. The monthly average exchange rates from January 1979 to December 1992 were used, with the first 162 observations for model building and the last six for testing. Both six-step-ahead and one-step-ahead predictions were employed. Neural networks produced significantly better results than the best ARIMA models in both one-step-ahead and six-step-ahead forecasting. Similarly, Abu-Mostafa (1995) reported a statistically significant performance in four major exchange markets by neural network with a simple symmetry hint.

Borisov and Pavlov (1995) applied neural networks to forecast the Russian rouble exchange rate. The data set contained 164 observations over 13 months. The first 10 months were used for training and the remaining three for testing. Two neural network models and two exponential smoothing models were used to predict the exchange rates. A backpropagation based neural network performed the best in all cases though it consumed more time to get the results. They also concluded that if the data set contained outliers, exponential smoothing was the better forecasting method. On the other hand, if the series contained less noise, then a window based neural network method should be used.

Embrechts (1995) put much emphasis on 'if at all' predictability of financial time series such as exchange rates. Using chaos analysis and Hurst analysis he showed that daily exchange rates for the Yen/Dollar were strictly as well as statistically predictable. For forecasting exchange rate fluctuations they relied on artificial neural networks. After trying several network architectures he finally used a 5:6:5:3:1 network i.e. five inputs, three hidden layers with 6,5 and 3 neurons respectively and one output neuron. He used back propagation to train the network and found that exchange rate fluctuations could be profitably predicted with artificial neural network. In addition, it was observed that neural nets with three hidden layers led consistently to a higher profitability than ANNs with just one or two hidden layers.

Episcopos and Davis (1995) investigated the problem of predicting daily returns on five Canadian exchange rates using feedforward neural networks and E-

GARCH model. The daily spot prices of US dollar, British pound, French franc, German mark and Japanese yen were collected from CANSIM data base and the sample period covered the period from January 2, 1992 to December 15, 1994, for a total of 744 observations. They considered four neural network architectures. The first type of network model was backpropagation network (BPN) 1:50:2:1, which used lagged returns to predict current returns for each of the series individually. Second, to capture possible spillovers from market to market, all five series were used in the network BPN 5:100:10:5. The third network was BPN 2:100:10:1 where forecast variance from EGARCH-M was used as an input. Finally volatility spillovers in mean from one market to other were captured with neural network BPN 10:100:10:5, where the lagged returns from all five series and the predicted volatilities from the EGARCH-M models were used as inputs. They compared the performance of both the neural network and EGARCH model with random walk models. Taking mean absolute error as an evaluation criterion, it was found that EGARCH-M outperformed random walk models. BPN outperformed EGARCH-M in three out of five exchange rates on the hold out sample but they fared worse than EGARCH-M on the training sample. However, it was found that adding volatility as an input did not improve prediction results. Overall, they concluded that both BPN and EGARCH-M model outperformed random walk and BPN was sometimes superior to EGARCH-M.

Hann and Steurer (1996) compared neural network models with linear monetary models in forecasting the U.S. dollar/Deutsch mark exchange rate. Principal component analysis was used to heuristically find the number of hidden nodes in a three-layer feedforward network. Based on the Out-of-sample results, they found that for weekly data, neural networks were much better than linear models and the naive prediction of a random walk model with regard to Theil's U measure, the hit rate, the annualized returns, and the Sharpe Ratio. However, if monthly data were used, neural networks did not show much improvement over linear models. This outcome is consistent with the results of a Brock-Dechert-Scheinkman (BDS) test (Brock, Dechert, Scheinkman, & Lebaron, 1995) for nonlinear relationships, which indicated that weekly data contain nonlinearities whereas monthly data do not.

Zhang and Hu (1998) addressed the difficulty in selecting the appropriate neural network architecture for an exchange rate forecasting problem. In this paper, they examined the effects of the number of input and hidden nodes as well as the size of the training sample on the in-sample and out-of sample performance in British pound/US dollar exchange rate. They also compared the predictive ability of neural network model with random walk model. They used weekly data for British pound/US dollar exchange rate covering the period beginning of 1976 through the end of 1993. Their findings showed that the numbers of input nodes play an important role in the prediction of exchange rate. Neural network's performance was further shown to be insensitive to the number of hidden nodes. In addition, it was found that neural network outperformed random walk model when the forecast horizon is short but did not outperform it in the long horizon. In order to assess the effect of the size of the training sample on neural networks performance, they used two training sample sizes in their study. The large sample consisted of 887 observations from 1976 to 1992 and the small one included 261 data points from 1988 to 1992. They found that performance measures for the neural network model in 1988-1992 series (the small training sample) took on larger values than the corresponding values from the 1976-92 series (the large training sample). This observation compelled them to conclude that forecasts based on long series will lead to a lesser amount of forecast error.

This is in contrast with the study by Plasmans *et al* (1998) and Verkooijen (1996). They used macro economic variables and ANN to forecast exchange rates and to test whether the underlying relationship between exchange rate and macroeconomic variables was nonlinear. They could not produce satisfactory monthly forecasts.

Hu *et al* (1999) investigated the potentials of neural network models in the prediction of weekly British pound/U.S. dollar exchange rate by employing two cross-validation schemes such as moving cross-validation and rolling cross-validation for the period beginning of 1976 through the end of 1993. Cross-validation was used to test the effects of two possible sources of sampling variation, namely, differences across time periods and the number of observations in the training sample on the Out-of-sample performance. Out-of-sample performance with three forecast horizons, 1-

month time horizon, 6-months time horizon and 12-months time horizon was evaluated using four criteria, root mean square error, mean absolute error, mean absolute percentage error, and median absolute percentage error. Their results clearly showed that the accuracy of neural networks was not very sensitive to the sampling variation. Comparing with the performance of random walk models they concluded that overall, neural network predictions were better than those obtained through the random walk model. Furthermore their study identified the conditions under which neural networks may out-perform random walk. They found that neural network predictions were particularly superior when the forecast horizon was short. As the forecast horizon increased in length, there was no dominance of one technique over the other.

Gencay (1999) compared the Out-of-sample performance of two parametric models, random walk and GARCH (1,1) and two non-parametric methods, nearest neighbors regression and feedforward network regression conditional-mean estimators. He used these models to forecast daily spot exchange rate returns for the British pound (BP), Deutsche mark (DM), French franc (FF), Japanese yen (JY), and the Swiss franc (SF) with past returns and past buy-sell signals of the moving average rules. In total, there were 4684 observations for the period of January 2, 1973 to July 7, 1992, out of which 1561 observations were kept for the Out-of-sample predictions. Cross-validation was used as a model and data selection procedure. As a measure of performance the Out-of-sample mean square prediction error and sign prediction were used. To assess the statistical significance of the Out-of-sample predictions, the Diebold-Mariano (1995) statistic was calculated for all currencies. The results with past returns used as inputs showed that sign predictions averaged around the 49% level for random walk model. For the GARCH (1,1) model, the MSPE ratios of the corresponding model to that of random walk model were close to one. The sign predictions were 50, 51, 51, 51 and 50% for BP, DM, FF, JY, and SF respectively. The MSPE ratios indicated that feedforward network model provided an average of 7.9% forecast improvement over the random walk model across all currencies. The feedforward network model also provided more accurate sign predictions relative to the random walk model. The nearest neighbors model provided significant forecast gains over both the parametric and the feedforward network models. The predictability of the current returns with the past buy-sell signals of the moving average rules were investigated with (1,50) and (1, 200) rules. By doing this, they were able to improve the forecast performance of models across all currencies. For example, GARCH (1,1) model provided a slight improvement for the BP, DM, and FF series. The performance of the feedforward network and nearest neighbors model both indicated significantly lower MSPEs relative to the random walk model and significantly higher sign predictions. The comparison of the nearest neighbors and feedforward network models indicated that the nearest neighbors models did slightly better than the feedforward network models.

Anastasakis and Mort (2000) proposed a variety of variables in the input vector and used ANN to predict the daily exchange rate between the US dollar and British pound (USD/GBP) for the period January 1991 to December 1993.They did this specifically because the use of only historical data may be irrational given the globalization phenomenon in economies, and so would be the use of linear models to capture nonlinearity within the data. Four input vectors were considered in their work. The first input vector consisted of four past values of the USD/GBP series itself. A set of three Random Walk Indexes (RWI), the most popular technical indicator, computed for one, two, and three look back periods as well as the four past values USD/GBP were included in the second input vector. Third one included fundamental variables like exchange rate between US dollar and Japanese yen, and between British pound and German mark. Finally, the fourth vector included the complete set of variables. The hidden units that were selected empirically were fixed at six for all the input vectors. Using the multi-layer perceptron (MLP) and backpropagation (BP) learning rule, they found that every time new elements were added in the first input vector, better performance was achieved. For example, root mean square errors for all the four input vectors were 0.158, 0.154, 0.1512 and 0.046 respectively. They also attempted to improve the performance of neural network by using the autoassociator MLP network, an input vector reduction technique, for each one of the four input vectors. They found that generalization and the convergence speed of network are improved with the reduced input vector.

In a similar kind of study, Yao, Poh and Jasic (2000) used neural networks with time series and technical indicators as inputs to capture the underlying 'rules' of the movement in weekly exchange rates between the US dollar and five other major currencies like Japanese yen, Deutsch mark, British pound, Swiss franc and Australian dollar. They reported that without the use of extensive market data or knowledge, useful prediction can be made and significant paper profit can be achieved with simple technical indicators. They were also successful in showing the improvement obtained by neural networks over the ARIMA model in the prediction department.

Gradojevic and Yang (2000) combined two approaches, artificial neural networks and market microstructure, to the exchange rate determination for explaining very short-run exchange rate fluctuations. A variable from the field of microstructure, order flow, was included in a set of macro economic variables such as interest rate and crude oil price to explain daily Canada/US dollar exchange rate movements. To forecast the exchange movement, they considered two models for neural network. The first model showed the change in exchange rate as a function of change in interest rate differential, change in crude oil price, aggregate order (the sum of individual order flows) and a disturbance term. The second model explained exchange rate as a function of change in interest rate differential, change in crude oil price, commercial client transactions, foreign institutional transactions, interbank transactions and a disturbance term. In their study, networks trained and tested were three-layer and four-layer backpropagation neural networks with nonlinear sigmoid and tan-sigmoid neuron activation functions in hidden layers. The number of input neurons was three for first model and five for second model, while the number of hidden neurons varied between three and five for both of the models. To avoid over training, they adapted an early stopping technique. Using RMSE, and the ability to predict the direction of exchange rate movements as two evaluation criteria they showed that the neural model never performed worse than the random walk model. The neural network model was consistently better in terms of RMSE than random walk and linear models for the various Out-of-sample experiments. More over, neural network performed on average at least 3 percent better than other models in **its** percentage of correctly predicted signs.

Bordolai (2001) attempted to design an application of the neural network technique for the prediction of daily exchange rates relating to Indian rupee/US dollar, Japanese yen/US dollar, British pound sterling/US dollar, and Euro/US dollar and then compared the Out-of-sample performance of neural network with random walk model. He used the daily data on the four series, beginning with January 1999 up to the end of August 2000. He used the backpropagation algorithm to build the network and genetic algorithm to avoid the problem of convergence to local minima. Making use of statistics such as average absolute error, mean absolute percentage error, root mean square error, mean square percentage error, and R^2, he compared the Out-of-sample forecasting performance of neural network and random walk model. The results suggested that forecasts obtained on the basis of neural network outperformed the simple random walk model in predicting daily foreign exchange rales for the four series of exchange rates.

Medeiros *et al* (2001) compared alternative models to forecast monthly exchange rate time series. The different models considered in their paper were the neuro-coefficient smooth transition autoregressive (NCSTAR) model, artificial neural networks (ANNs), linear autoregression (LAR), and the random walk (RW) formulation. To asses the practical usefulness of ANN and NCSTAR models vis-a-vis the LAR and RW models, they experimented with 14 different monthly exchange rates time series. The ANN model was estimated with Bayesian Regularization. They compared forecasts made by each estimated model using the following statistic: RMSE, MAE, mean absolute deviation and the proportion of times the sign of excess returns is correctly forecasted. Their results showed that nonlinear models like ANNs were better predictors only in the cases where nonlinearity was uniformly spread. Otherwise, there was no significant difference in the forecasts made by a concurrent linear model. For most of the series, the difference in the forecast performance between NCSTAR, ANN, LAR, and RW models were not significant according to Diebold Mariano test. Again, for one-step-ahead forecast and especially when the

MAE was used as a comparison criterion, the results were quite supportive in favor of the linear and nonlinear (e.g. ANN model) specifications against the naive RW **model.**

2.4.2 Studies on Stock Returns Prediction

Given the failure of traditional models such as the market model, capital asset pricing model (CAPM), and arbitrage pricing theory (APT) in understanding the stock price behavior, on the one hand, and the inductive, adaptive, and robust nature of neural network on the other the latter has evolved as a powerful tool for the prediction of stock returns.

Using the past historical stock prices as explanatory variables White (1989) tried to capture the undetected nonlinear regularities that may exist in IBM daily common stock returns by using neural network. The training sample covered trading days during the period 1974:11 through 1978:1 and the testing sample covered 500 days before the training period i.e. 1972:11 through 1974:1 and 500 days after the training period i.e. 1978:11 through 1980:1. His prime concern was to test the efficient market hypothesis. He asserted that empirical evidence of R^2 - 0 is consistent with either the efficient markets hypothesis or the existence of nonlinear structure. While, empirical evidence of $R^2 \neq 0$ is evidence against the simple efficient markets hypothesis. For this purpose, he used two neural network models. First, the linear network with two layers, and second, the network with three layers including one hidden layer. The input units and the hidden units both were fixed at five. Using simple linear neural network model with two layers he obtained $R^2 = 0.0079$ which suggested little evidence against the simple efficient market hypothesis. But with the three-layered network, the in-sample R^2 was surprisingly good at 0.175, which was apparently inconsistent with the efficient market hypothesis and consistent with the presence of nonlinear regularities. For the period 1978:11 - 1980:1 (testing period after 500 days of training period), he observed a correlation between the actual and predicted stock return of-0.0699 and for the period , 500 days before the training set, it was 0.0751. These results did not constitute convincing statistical evidence against the efficient market hypothesis. To summarize, he concluded that the neural network was not a money machine but was capable of capturing some of the dynamic behavior of stock returns.

Yoon and Swales (1990) compared neural networks to discriminant analysis with respect to prediction of stock price performance and found that neural network was superior to discriminant analysis in its predictions. Trippi and Desieno (1992) applied neural network system to model the trading of Standard and Poor (S&P) 500 index futures. They found that neural network system outperformed passive investment in the index. Based on empirical results, they favored the implementation of neural network systems into the main stream of financial decision making.

Chuah (1993) used the neural network to forecast stock index returns on New York Stock Exchange using data from January 1963 to December 1988 and compared the predictability and profitability of the network forecasts with those from a benchmark linear model using the same data. The predictability tests showed that the forecast errors of the network were not significantly different from those of the benchmark model and that the network had no market timing ability. The profitability test examines profits generated from a trading simulation over a five year forecast period in comparison with a benchmark buy-and-hold strategy. The nonlinear network generated a total return of 116% versus 94% from the buy-and-hold strategy, while the linear network generated only a 38% total return.

Refenes, Zapranis and Francis (1994) examined the use of neural networks as an alternative to classical statistical techniques for forecasting within the framework of the APT model for stock ranking. Their objective was to outperform multiple linear regression with respect to three performance metrics: goodness of fit in-sample (convergence), goodness of fit Out-of-sample (generalization) and the stability of results with varying network parameters and different data sets. The network architecture consisted of three inputs and one output. They experimented with several configurations of hidden units and the best configuration in terms of the trade-offs involved between convergence and generalization, which also gave a conveniently stable network, was a 3-32-16-1 configuration. Using simple learning algorithm such as the backpropagation algorithm they showed that neural network gave better model fitness in-sample by one order of magnitude and outperformed multiple linear regression in Out-of-sample forecasting. They also identified intervals for the network parameter values for which the above performance results were statistically stable. By using sensitivity analysis, they showed that neural networks could provide a reasonable explanation of their predictive behaviour and could model their environment more convincingly than regression models. This finding was against the view that neural networks are black boxes. But they never compared the performance of neural networks with random walk model, which very often is considered as the best predicting model for stock returns.

Steiner and Wittkemper (1995) used five neural network models of varying architecture in order to model future stock returns. Out of five models three were feedforward networks and the remaining two were recurrent networks with feedbacks. The sample data for the study consisted of daily prices and turnovers from the Frankfurt Stock Exchange, starting 1983 and ending 1986, and the returns of three different indexes, the DAX index, the WestLB Index and the Index der Frankfurter WertPapierborsen. They divided stock data into two groups. The first, referred as Dax-values, consists of blue-chip stocks. The second, referred as ndax-values, consists of stocks from smaller companies. Their findings were consistent with the more general findings that the neural networks could give more reliable forecasts for stock returns than the comparable statistical techniques.

In another study, Tsibouris and Zeidenberg (1995) attempted to predict the six stock prices of Citicorp (CCI), John Deere (DE), Ford (F), General Mills (GIS), GTE, and Xerox (XRX) and to show the efficacy of efficient market hypothesis. The daily observations for all the data, spanning the period from 4 January 1988 to 29 December, were collected from the Center for Research in Security Prices. Using the feedforward backpropagation networks, temporal difference models and the cascade correlation networks, they found evidence against the weak form of the EMH. Except for stock price DE, the Out-of-sample percentages of correct responses for CCI, F, GIS, GTE and XRX were found to be 60.87%, 60.08%, 53.36%, 53.36%, and 54.15%

respectively. These results provided evidence that backpropagation models were able, using previous asset returns as inputs, to provide some predictive power for future returns. Similar results of predictability of stock prices and evidence against the efficient market hypothesis were found by using temporal difference models. But they were unable to get results from cascade correlation networks as it did not get converged for their data.

In the same year, 1995, Baestaens and Van den made an attempt to unravel relationships between seventeen 'contextual variables' and the monthly return on the Datastream Amsterdam general stock index for the period from November 1979 to March 1991. They used single hidden layer feedforward neural network and traditional OLS regression for this purpose. In their study, seventeen variables which include non-equity- related return variables, macro economic indicators identified by Chenn *et al* (1986), few characteristics variable such as terms of trade and the guilder/dollar spot exchange rate, stock index return lagged by one period, and stock index level lagged by one period were taken as input variables to the network as well as to regression. In in-sample, they found that neural network appeared to outperform OLS in both magnitude and direction. In Out-of-sample, they observed that although the regression estimates more or less tracked the actual return, their direction was often erroneous and concluded that neural network beat OLS regression in terms of both magnitude and direction. In this study, they also showed that neural network was not a black box tool and was able to give more insight into the problem under study than OLS regression.

Donaldson and Kamstra (1997) constructed a semi nonparametric nonlinear GARCH model, based on the artificial neural network literature, and evaluated its ability to forecast daily stock return volatility in four international stock market indices: Standard & Poor's 500 index, New York, Japanese Nikkei, Financial Times Stock Exchange Index, London, and Toronto Stock Exchange Composite Index from January 1, 1969 to December 31, 1990. They also tested the ability of other volatility models such as GARCH, EGARCH, and GJR in forecasting the conditional stock return volatility. For each model they carried out both in-sample and Out-of-sample forecasting. To evaluate the Out-of-sample forecasting performance of each model relative to other models, they conducted a Chong and Hendry (1986) forecast encompassing test. In-sample and Out-of-sample comparisons revealed that neural network model captured volatility effects overlooked by GARCH, EGARCH, and GJR models and produced Out-of-sample volatility forecasts, which encompassed those from other models. In 1996, they investigated the use of neural network to combine time series forecasts of stock market volatility from the USA, Canada, Japan, and UK. They found that combining with nonlinear neural network generally produces forecasts which, on the basis of Out-of-sample forecast encompassing tests and mean squared comparisons, dominate forecasts from traditional linear combining procedures.

Using both Lee-White-Granger ncural-net-based (LWG) test and BDS test, Abhyankar *et al* (1997) tested for nonlinear dependence in real time returns on the world's four most important stock market indexes. The stock market indexes, used in their study were four published cash indexes namely, the FTSE-100, the S&P 500, the DAX, and the Nikkei. They found persistent nonlinear structure for all data scries. They also tested for chaos in these four indexes. Estimates of the Lyapunov exponents using the Nychka, Ellner, Gallant, and McCaffrey neural-net method and the Zeng, Pielke, and Eykholt nearest neighbor algorithm confirmed the presence of nonlinear dependence in the returns on all indexes but provided no evidence of low-dimensional chaotic processes. However, they concluded that although there might be an underlying deterministic nonlinearity but no chaotic process in the data, there is almost certainly a stochastic component whose presence cannot be ignored.

Qi and Madala (1999) used six monthly financial and economic variables to forecast excess return on S&P 500 Index. They also compared the network predictive performance to those of linear regression and random walk model. Based on the existing literature, they chose variables that included dividend yield, one month treasury bill rate, changes in short term interest rate, growth rate of industrial production, inflation rate and money growth rate to predict excess returns. Six benchmarks such as RMSE, MAE, MAPE, Pearson correlation coefficient, direction accuracy, and the proportion of times the signs of excess returns are correctly forecast were used in comparing the performance of alternative models. The whole data set i.e. from 1954(1) to 1992(12) was split into two parts. The first part contained data from 1954(1) to 1980(12) (in-sample period) and the second part from 1981(1) to 1992(12) (Out-of-sample period). In the in-sample, the neural network was found to have unanimous improvement in all six benchmarks in comparison with linear regression and random walk model. But for all three models and by almost all performance measures, the Out-of-sample results tend to be worse than the in-samplc. However, they found that by all six measures, both the neural network and linear regression Out-of-sample forecasts were still more accurate than the random walk forecasts, which provided further evidence against the efficient market hypothesis. More importantly, except for the percentage of correct signs, the neural network forecasts were better than the linear regression forecasts by the rest of the five performance measures. The market timing ability of alternative forecasts were tested using Pesaran and Timmermann (1992) non parametric test and they found that both the neural network and linear regression forecasts showed significant market timing ability and both outperformed the random walk forecasts. In fine, they concluded that neural network model could improve upon the linear regression model and random walk model in terms of predictability.

Shachmurove and Witkowska (2000) applied ordinary least squares, general linear regression, artificial neural network model, and multi-layer perceptron models in order to examine the dynamic interrelation of major world stock markets. The multi-layer perceptron models contained one hidden layer with two and five processing elements and logistic activation. The database consisted of daily stock market indices of the following countries: Canada, France, Germany, Japan, United Kingdom (UK), United States (US), and the world excluding US (World). The indices for rest of the countries were calculated by Morgan Stanley Capital International Perspective, Geneva (MSCIP). The database covered the period January 3, 1987, through November 28, 1994, with a total of 2064 observation per stock market. Based on the criteria of RMSE, maximum absolute error and the value of the objective function, the models were compared to each other. They found that neural network consisting of multi-layer perceptron models with logistic activation functions predicted stock returns better than the traditional ordinary least squares and general linear regression models. Furthermore, it was found that a multi-layer perceptron with five units in the hidden layer predicted the stock indices for USA, France, Germany, UK and World better than a neural network with two hidden elements.

Using weekly data from January 1987 to October 2000, Bautista (2001) attempted to predict the Philippine stock price index through artificial neural networks. In this work his focus was on showing how the neural network's performance differed from that of random walk in short and long lag lengths. He also tried to study whether the network size mattered in the approximation and prediction. To gauge the performance of models, the squared correlation r^2 and the Akaike Information Criterion were used as the in-sample forecast statistics; and Out-of-sample performance was gauged using several statistics like mean squared prediction error, Diebold-Mariano Statistic and success ratio. His study considered two specifications. The first was a univariate model where the only inputs were the lagged values of the output and the second included variables that are commonly used in technical and fundamental analysis other than the lagged output values as inputs. Considering a single hidden layer network with logistic activation function in the hidden layer and using the Levenberg-Marquardt algorithm as training algorithm, he found that Out-of-sample forecast performance of univariate neural network with relatively short lag lengths did not differ significantly from those of random walk as suggested by Diebold-Mariano statistic. At longer lags, univariate networks performed better than the random walk. The second specification that included fundamental and technical analysis variables even with short lag lengths gave better prediction than the random walk model. He also observed that network size mattered and hence the larger was the network the better was the approximation and the more superior was the Out-of-sample forecast performance.

2.5 Conclusion

After the down fall of random walk model, and given the inefficiencies inherent in the various linear and nonlinear parametric models, artificial neural network has been proposed in recent years for the problem of exchange rates and stock prices prediction. In this chapter, we have presented artificial neural network model in detail along with two benchmark models namely linear autoregressive and random walk models in brief. We have also reviewed the usefulness of neural network in exchange rates and stock prices prediction. Barring a few studies, neural network was found to have superior both in-sample and Out-of-sample predictive power than other traditional statistical models such as multiple regression, ARIMA, GARCH, and random walk models in exchange rates prediction. Similarly, in case of stock prices prediction, though the applications of neural network are fewer, neural network has proved to be a better predictor than other traditional forecasting techniques. These findings, in turn, indicate that the nonlinearities contained in the exchange rates and stock prices can well be exploited by neural network to improve the forecasting of these variables. The superiority of neural network over other traditional forecasting techniques in exchange rates and stock prices prediction motivates us to use it to forecast exchange rates and stock returns which we undertake in the following chapters, together with a comparison of the forecasting ability vis-a-vis linear autoregressive and random walk models.

Chapter III

Forecasting Daily and Weekly Exchange Rate Returns

The conscious mind allows itself to be trained like a parrot, but the unconscious does not.

3.0 Introduction

Artificial neural network is a nonlinear, nonparametric and data driven modeling approach. As we saw, it has flexible nonlinear function mapping capability, which can approximate any continuous measurable function with arbitrarily desired accuracy (Hornik *et al,* 1989; and Cybenko, 1989). Neural network presents a number of advantages over conventional methods of analysis. First, neural network makes no assumptions about the nature of the distribution of the data and is not therefore, biased in its analysis. Instead of making assumptions about the underlying population, neural network with at least one hidden layer uses the data to develop an internal representation of the relationship between the variables (White, 1992). Second, since time series data are dynamic in nature, it is necessary to have nonlinear tools in order to discern relationships among time series data. Neural network is best at discovering nonlinear relationships (Wasserman, 1989; Hoptroff, 1993; Gracia and Gencay, 2000). Third, neural network performs well with missing or incomplete data. Whereas traditional regression analysis is not adaptive, typically processing all older data together with new data, neural network adapts its weights as new input data becomes available (Kuo and Reitsch, 1995). Fourth, it is relatively easy to obtain a forecast in a short period of time as compared with an econometric model.

Given the advantages of neural network, this methodology has been extensively used for the problem of exchange rate prediction. We reviewed some select works on the usefulness of neural network in exchange rate prediction in section 2.4 of chapter II. The overall findings suggested that the neural network technique is superior to other conventional forecasting techniques in exchange rate prediction. This inspires us to take up the task of using neural network to carry out prediction in Indian financial markets. We begin the task in this chapter by undertaking one-step-ahead prediction of daily and weekly Indian rupee/US dollar (INR/USD) exchange rate. We also compare **the** performance of neural network with the performances of linear autoregressive and random walk models.

The remainder of the chapter is organized in the following sections. Various performance measures by which neural network's performance is to be compared with those of linear autoregressive and random walk models are given in section 3.1. In section 3.2, we carry out daily exchange rate returns prediction using neural network, linear autoregressive and random walk models. This section also compares the predictive accuracy of all three studied models. In section 3.3, we carry out weekly exchange rate returns prediction. We examine the statistical significance of in-sample and out-of-sample results by neural network, linear autoregressive and random walk models, in section 3.4, by using forecast encompassing test. Finally, section 3.5 concludes the chapter.

3.1 Forecast Performance Measures

In this section, we discuss different criteria by which neural network forecasts are evaluated against those of linear autoregressive and random walk models. There is no accord on the appropriate forecasting performance measures, which are used to assess performances of different forecasting techniques. Here, we use eight performance criteria namely root mean square error (RMSE), mean absolute error (MAE), mean absolute percentage error (MAPE), median absolute deviation (MAD), Pearson correlation coefficient (CORR), goodness of fit (R^2), direction accuracy (DA) and sign predictions (SIGN) to evaluate the predictive power of neural networks in comparison to linear autoregressive and random walk models. Let $(\hat{y}_1, \hat{y}_2, \ldots\ldots \hat{y}_N)$ denote the predicted values and $\{y\ , y_2, \ldots\ldots y_N)$ be the actual values, where N is the sample size. The eight forecasting evaluation criteria are listed as follows.

(1) RMSE: it returns root of the mean squared errors between actual and predicted values and can be written as

$$RMSE = \sqrt{1/N\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \qquad \textbf{(3.1)}$$

(2) MAE: it gives the average absolute error between actual and predicted values,

$$MAE = 1/N\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{3.2}$$

(3) MAPE: mean absolute percentage error between actual and predicted values is calculated as

$$MAPE = 1/N\sum_{i=1}^{N}|(y_i - \hat{y}_i)/y_i| \tag{3.3}$$

(4) MAD: it is a performance measure which is robust to outliers, and can be written as

$$MAD = median\left(|e_i - median(e_i)|\right) \tag{3.4}$$

where $e_i = y_i - y_i$ is the forecast error values.

(5) CORR: it measures the linear correlation between actual and predicted values,

$$CORR=\left(\sum_{i=1}^{N}(y_i - \bar{y}_i)(\hat{y}_i - \bar{\hat{y}})\right)\Big/\left(\sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2}\sqrt{\sum_{i=1}^{N}(\hat{y}_i - \bar{\hat{y}})^2}\right) \tag{3.5}$$

(6) Goodness of Fit: $R^2 = 1-\left(\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \Big/ \sum_{i=1}^{N}(y_i - \bar{y})^2\right)$ (3.6)

(7) DA: it measures the proportion of times the upward or downward movements of the series y_i are correctly predicted,

$$DA = 1/N\sum_{i=1}^{N}a_i \tag{3.7}$$

where

$$a_i = \begin{cases} 1 & if\ (y_{i+1} - y_i)(\hat{y}_{i+1} - y_i) > 0 \\ 0 & otherwise \end{cases}$$

(8) SIGN: it measures the proportion of times the signs of series y_i **are** correctly forecast.

$$SIGN = 1/N\sum_{i=1}^{N}b_i \tag{3.8}$$

where

$$b_t = \begin{cases} 1 & if \;\; y_{t+1}\, \hat{y}_{t+1} > 0 \\ 0 & otherwise \end{cases}$$

3.2 Daily Exchange Rate Returns Prediction

In this section, we discuss the predictive power of neural network, linear autoregressive and random walk models in one-step-ahead prediction of daily Indian rupee/US dollar exchange rate returns. One-step-ahead prediction is useful in evaluating the adaptive ability and robustness of a forecasting method. It uses actual values rather than forecast values for the next prediction in the forecasting horizon. One poor prediction will not have serious consequences in a one-step-ahead forecasting system since the actual value will be used to correct itself for the future forecasts. Hence, this is more useful than multi-step-ahead prediction, which uses forecast values and thus introduces forecast error in the next prediction. Furthermore, the random walk model will be used as one of benchmark for comparison. The random walk model is a one-step-ahead forecasting model since it uses the current observation to predict the next one. It is easy to see that random walk model will produce more accurate forecasts in one-step-ahead than the multi-step-ahead forecasts. In subsection 3.2.1, we present the description of daily exchange rate data. Model specifications for neural network, linear autoregressive and random walk models are presented in subsection 3.2.2 In subsection 3.2.3, we present empirical findings, which includes both in-sample and out-of-sample forecasting of neural network, linear autoregressive and random walk models.

3.2.1 Data Description

The daily spot rates of Indian rupee/US dollar (INR/USD) exchange rate are taken from Pacific FX database for the period of January 4, 1993 to July 12, 2002, for a total 2396 observations. The daily returns are calculated as the log differences of the levels. Let p_t be the exchange rate price for the period t. Then the exchange rate return at time t is calculated as $y_t = (\log p_t - \log p_{t-1})*100$. We multiply the log difference by *100* to reduce round-off errors. Both daily exchange rates at level and returns are shown in Figure 3.1. Instead of exchange rates at level, we take exchange rate returns for the whole analysis because it solves the problem of non-stationarity, trend and outlier effects and thus eliminates bias in forecasting. One more reason is that return is more normally distributed than the exchange rates at level. This is clear from Figure 3.2. In Figure 3.2, we show Quantile-Quantile plots of both daily exchange rates at level and returns against the normal distribution. If the plot lies on a straight line then it can be said that the series follows a normal distribution. The two plots in Figure 3.2 clearly indicate that the daily exchange rate returns has a distribution closer to the normal than the exchange rate at level. Table 3.1 presents the summary statistics of the data. Both the skewness and kurtosis are substantially high. The kurtosis coefficient i.e. 297.82 is much larger than that of the standard normal distribution (which is equal to 3), which in turn indicates the leptokurtosis of daily INR/USD return series. Jarque-Bera statistic also rejects the normality of exchange rate returns, which is common in high frequency financial time series data. The first 20 autocorrelations are calculated but only five autocorrelations i.e. *P1, P5, P10, P15, P20* are reported in the table. The series shows evidence of autocorrelation. The Ljung-Box (LB) statistic for the first 20 lags is 50.13, which rejects the hypothesis of identical and independent observations. The value for Augmented Dicky-Fuller (ADF) test statistic assures the stationarity of the return series.

The last row of the table presents the Hurst exponent. The Hurst exponent, after the name of H.E. Hurst, is used to detect the long memory effect in a series. The Hurst analysis, otherwise known as rescaled range analysis (*R/S* analysis), is able to distinguish a random series from a fractal series, irrespective of the distribution of the underlying process (Gaussian or non-Gaussian). R captures the maximum and minimum cumulative deviations of the observation y_t of the time series from its mean (μ) and it is a function of time (the number of observations N),

$$R_N = \max_{1 \le t \le N}[y_{t,N}] - \min_{1 \le t \le N}[y_{t,N}]$$

where $y_{t,N}$ is the cumulative deviation over N periods. The R/S ratio of R and the standard deviation S of the original time series can be estimated using the empirical law $R/S = N^H$ when observed for various N values. H describes the probability that the two consecutive events are likely to occur. The value of $H = 0$ describes the series as random, consisting of uncorrelated events. A value of H different from 0.50 indicates that the observations are not independent. When H is in between 0 and 0.5, the system is an antipersistent or ergodic series with frequent reversals and high volatility. For the case $0.5<H<1.0$, H describes a persistent or trend-reinforcing series which is characterized by long memory effects. As shown in Table 3.1, the value of Hurst exponent is higher than 0.5. This indicates that daily exchange rate returns have long memory effects. In addition to this, the value of Hurst exponent i.e. 0.5273 establishes that exchange rate returns are not statistically independent, which in turn implies that exchange rate price does not follow a random walk.

3.2.2 Model Specification

A good forecast depends to a great extent on the specification of the model, whether it be neural network, linear autoregressive or random walk model. This particular section is completely devoted to a discussion of the model specification of each of these models.

As we had pointed out earlier, neural network can approximate a continuous measurable function arbitrarily to any desired accuracy. However, a suitable network structure is to be determined so that any continuous function can be approximated well. A very simple network may not approximate the function well and an excessively complex network may overfit the data with little improvement in approximation accuracy. The problems of designing an optimal neural network have already been discussed in the previous chapter in section 2.1.3. Till date, followers of neural network have not come to an agreement regarding the way complexities of neural network should be regularized.

However, we believe that experience is the best teacher. Thus, the number of inputs and hidden units are chosen through systematic experimentation. To estimate neural network model, apart from selection of appropriate number of inputs and hidden units, which we discus later, certain steps are involved as follows. First, the whole data set is divided into a training set, which generally consists of 75-80 percent of the whole data set, and the testing set consists of the rest of the observations. In our study, out of total 2395 observations, we keep 2100 observations for training (i.e. in-sample data) and remaining 295 observations are kept for testing (i.e. out-of-sample data).

The second step is data normalization. Data normalization is a process to keep the data within a certain range, for example, 0 to 1 or -1 to 1. Normalization is generally desirable in order to remove the possibility that the network parameters get 'tuned' for a given range of input-output data and also to bring the inputs of the transfer function inside the function's normal operating region. Assuming an asymptotic sigmoid function with [0, 1] range, if its input fall well outside its normal operating region, then its output tends asymptotically to 0 or 1. In that case, the derivative of the sigmoid, and the activation value of the neuron tend to 0. This is undesirable since it can bring training to a virtual standstill (known as network paralysis). On the other hand, since the range of the sigmoid function is [0, 1], a target output which falls outside that range will constantly create large backpropagated errors and the network will be unable to learn the input-output relationship implied by the particular training pattern. Furthermore, it is undesirable to have large differences between the ranges of the network inputs, since this can diminish the importance of some otherwise useful inputs. Typically, variables are normalized in order to have zero mean and unit standard deviation. Keeping it in mind that sigmoid function will be used as transfer function in the hidden layer, we normalize the data to the value between 0 to 1. The data normalization has been done using the following equation.

$$Y_i = \frac{y_i - y_{i,\min}}{y_{i,\max} - y_{,\min}}(h_i - l_i) + l_i \quad .... \qquad (3.9)$$

where: Y_i =normalized value of the input or the output value

y_i = original input or output value

$y_{i,min}$ =minimum original input or output value.

$y_{i,max}$ = maximum original input or output value

h_i = upper bound of the normalizing interval (in our case 1)

l_i = lower bound of the normalizing interval (in our case 0)

i = 1,….,N, the number of observations.

The third step is the selection of hidden layers. A single-hidden-layer feedforward network is chosen, as we did not see any improvement in the network's performance with more than one hidden layer. This, of course, falls in line with the findings by Hornik *et al* (1989) that one hidden layer is enough to approximate any continuous function

Fourth, is to choose the transfer function. In this work, sigmoid transfer function is used in the hidden layer. It is a nonlinear function and thus captures the nonlinearity in data. Linear transfer function is used in the output layer, which is a standard choice in neural network. The linear transfer function for the output layer is taken to prevent the loss of generality.

The fifth step is to initialize the connection weights. In the current study, the weights are initialized to small values based on the technique of Nguyen and Widrow (1990). The merit of using this initialization technique is that it works wonderfully where the hidden layer transfer function is sigmoid. It also speeds up the training process.

The sixth step is to select the cost or error function. Mean square error, which is generally used as cost function in neural network, is the cost function in our study.

The final step is the selection of a training algorithm, which facilitates the network training. After deliberation, the resilienl backpropagation has been taken as the training algorithm[1]. This is because it facilitates faster training and is not very sensitive to the settings of training parameters i.e. learning and momentum coefficients. By taking the resilient backpropagation training algorithm, we at least are able to avoid the risk of the neural network's training performance being sensitive to the values of learning and momentum coefficients.

As mentioned earlier, the number of input nodes and hidden nodes to the network are selected through experimentation. The number of input nodes, in this work, corresponds to the number of lagged past observations. Thus, the resulting network is a nonlinear autoregressive model, which plays an important role in determining the autocorrelation structure of a time series. Twenty levels of the number of input nodes ranging from 1 to 20 lagged values of the dependent variable i.e. daily exchange rate returns, are used in this study. The number of hidden nodes is also crucial for a neural network for the reason that it captures nonlinear patterns and detects complex relationships in the data. Networks with too few hidden nodes may not have enough

[1] We train and test the network using software package Matlab, V. 5.3.1, Neural Networks Toolbox, The Math Works, Inc. (1999).

power to model and learn the data. But networks with too many hidden nodes may cause overfitting problems and hence lead to poor forecasting ability. However, previous researchers such as Tang and Fishwick (1993), Zhang and Hu (1998) and Hu *et al* (1999) indicated that the forecasting performance of neural networks was not as sensitive to the number of hidden nodes as to the number of input nodes. Following these findings, we only experiment with five levels of hidden nodes 4, 8, 12, 16, and 20 across each level of input node. Thus, the combination of 20 input nodes and five hidden nodes yields a total of 100 neural network architectures being considered for each in-sample training data set. To avoid the problem of getting stuck in local minima and to increase the possibility of getting the true global optima, we train each of the 100 neural network architectures 10 times by using 10 different sets of initial random weights. The set of final weights that yields the smallest training sum of the squared errors or equivalently, root mean square error (RMSE) is selected and applied to the test set.

The effects of input nodes and hidden nodes on the neural network's training performance are shown in Table 3.2. The RMSEs and MAEs, in the table, at each level of input node in combination with each of the five hidden node levels are the averages of ten runs. To conserve space, we only report the results for 2, 4, 6, 8, 10, 12, 14, 16, 18, and 20 input units across five levels of hidden units i.e. 4, 8, 12, 16, and 20. The results show that RMSE decreases as the number of hidden units increases. This pattern is observed consistently when the number of input units is in the range of 1 to 10. After that the effects of hidden units at each level of input node are inconsistent. However, as the number of input nodes increases, RMSE decreases as reflected by the average of RMSEs across the five hidden node levels at each level of the input node. The best result is achieved at input levels of 20 with smallest RMSE equal to 0.0221. It is also important to note that there is less variation among different hidden node levels within each input level than among different input node levels, suggesting neural network's model fitting performance is more sensitive to the input nodes than the hidden nodes. As far as MAE is concerned, we observe no consistent pattern for the effects of the input nodes and the hidden nodes. However, this should not bother us much as the purpose of training is to minimize the sum of squared errors or RMSE and not MAE. The input units of lags 1 through 20 along with 16 hidden nodes achieve the smallest RMSE, which is equal to 0.0203. Hence, the network architecture of 20 input nodes combined with 16 hidden nodes is used for further analysis, i.e. for in-sample and out-of-sample forecasting of daily exchange rate returns. Again, the model selection for neural network is strictly based on the in-sample or training data.

Having discussed the architecture of neural network, we now turn to discuss the model specification for linear autoregressive model and random walk models. The same training or in-sample data, i.e. 2100 observations, as in the case of neural network, has been taken for the model estimation in linear autoregressive and random walk models. This ensures that forecast comparisons are fair among all the three studied models as they all use the same in-sample data for their forecasts. In a linear autoregressive model, we use lagged values of the dependent variable as explanatory variables. To select the optimum autoregressive terms, we first regress the dependent variable y_t (daily exchange rate return) on a large group of independent variables. We take the first 20 lagged values of the dependent variable i.e. from y_{t-1} to y_{t-20} as explanatory variables. Then a small group of statistically significant variables are identified and used as explanatory variables to forecast daily exchange rate returns. The regression results are presented in Table 3.3. Among all the 20 lags, lags 1, 5, 9, 11, and 15 are found to be statistically significant. Hence, these five significant lags have been used as explanatory variables for daily exchange rate returns prediction.

As far as the model specification of random walk is concerned, we do not estimate any model. We simply take random walk without drift, which forecasts no change in the value. By virtue of this, current period's value is considered as the best predicting value for tomorrow.

3.2.3 Empirical Findings

In this section, we first present the in-sample forecasts of neural network, linear autoregressive and random walk models. In-sample forecasts are obtained from the in-sample or training data, which consists of 2100 observations. Second, out-of-sample forecasts of the three studied models are presented. Out-of-sample forecasts are obtained from the unseen or untouched data that models do not use at the time of model selection. Here, it should be noted that the results for in-sample forecasts and out-of-sample forecasts are presented in terms of exchange rate returns, not in terms of normalized returns. The predicted returns have been calculated by taking the inverse of normalized equation (3.9).

In-sample Forecasting

In-sample performance of artificial neural network, linear autoregressive model and random walk models is shown in Table 3.4. ANN and LAR are found to have better in-sample forecasting power than random walk model by all the performance measures. The values of RMSE, MAE, MAD and CORR for ANN are 0.1081, 0.0593, 0.0261,and 0.7482 respectively as compared with the values of 0.2401, 0.0992, 0.0387 and -0.0861 for random walk respectively. Between ANN and LAR, the former has smaller RMSE and MAE in comparison to the latter. The fitted values of the neural network have higher correlation (0.7482) with the actual series as compared to that of value of linear autoregressive model, which is equal to 0.1316. The most important evaluation criteria are direction accuracy and sign prediction. Because an investor is more enthusiastic to know about the directional and sign change in the exchange rate return for tomorrow rather than the exact magnitude of it. The neural network also has higher percentage of correct signs than both the linear autoregressive and random walk models. In terms of direction accuracy, LAR has slightly better performance over ANN. In addition to this, LAR has a smaller MAD value than that of ANN. Here it should be noted that the direction accuracy is zero for random walk. This is because, by definition, the random walk has absolutely no ability to predict whether the exchange rate return will go up or down since it simply takes the exchange rate return of current period as the forecast of the next period and predicts no change. We are unable to calculate the MAPE for all the three models, as there are zero values for some of the actual returns.

Figure 3.3 plots the in-sample errors of neural network, linear autoregressive and random walk models in predicting the daily INR/USD exchange rate returns. The figure shows that neural network's predicted returns have small deviations from the actual returns compared to linear autoregressive and random walk models. This is reflected by smaller fluctuations around zero in the graph for neural network. Further, it is also found that the error variance of ANN, which is equal to 0.0116, is much smaller than the error variances of LAR and RW, which are equal to 0.0261 and 0.0577 respectively. All these findings suggest that neural network produces less in-sample predicted errors and in turn gives better in-sample fit than linear autoregressive and random walk models. The figure also shows that LAR gives less errors and better in-sample fit than RW in predicting daily exchange rate returns.

From the above analysis, we conclude that both neural network and linear autoregressive model have significantly better in-sample forecasting power than random walk model. Neural network outperformed linear autoregressive model in four out of six performance measures. In addition to this, neural network is found to have better in-sample fit than linear autoregressive model and random walk model.

Out-of-sample Forecasting

The better fit of the neural network, observed from in-sample performances, is not surprising given its universal approximation property. The merit of neural network is that it can approximate any measurable function with arbitrarily desired accuracy, provided it has sufficient amount of hidden units in the hidden layers. But the problem with neural network is that it may overfit the data due to its over-parametered structure. Since in-sample fit does not guarantee superior out-of-sample performance due to overfit, it is crucial to test the validity of neural network using the unseen out-of-sample data.

Table 3.5 compares the out-of-sample performance of the neural network, linear autoregressive and random walk forecasts. For all three models and by almost all performance measures the out-of-sample results are found to be better than the in-sample results. One reason for this happening could be that the likely noise contained in the in-sample data may not be present in the out-of-sample data. The results show that neural network out-of-sample forecasts are more accurate than the random walk forecasts by all six performance measures. For example, the values of RMSE, MAE, MAD, and CORR for ANN are 0.0445, 0.0307, 0.0193, and 0.1258 respectively as compared with the values of 0.0599, 0.0405, 0.0303, and 0.0096 for random walk respectively. The neural network has also better direction accuracy and sign predictions than random walk. Similarly, except for CORR, LAR outperforms RW by all other performance measures in out-of-sample forecasting. The superiority of neural network and linear autoregressive model over random walk model in out-of-sample forecasting provides further evidence against the efficient market hypothesis. However, between ANN and LAR, there is no evidence of a "winner" model. The out-of-sample results are mixed. ANN out-of-sample forecasts are better than LAR in terms of correlation coefficient and the percentage of correct sign predictions. The correlation coefficient between fitted values of neural network and actual values is 0.1258 in comparison to negative correlation coefficient i.e. -0.0017 of linear autoregressive model. The percentage of correctly predicted signs by ANN, which is equal to 0.5418, is also higher than that of LAR (0.5090). On the other hand, LAR outperforms neural network when RMSE, MAE, and MAD performance measures are considered. The direction accuracy, which is equal to 0.71, is same for both the models. The plots of out-of-sample forecast errors of ANN, LAR, and RW, given in Figure 3.4, suggest that ANN gives less prediction error than the RW model. That means neural network gives better predicted returns, which is closer to actual returns than random walk model. We also find that variance of neural network's predicted errors, which is equal to 0.0018, is much smaller than the variance of random walk predicted errors, which is equal to 0.0036. From this we conclude that ANN gives better out-of-sample fit than RW in predicting daily exchange rate returns. However, between ANN and LAR, the plots of errors of both models look similar. Further, the error variance, which is equal to 0.0018, is same for both models. Hence, we find it difficult to get a definite conclusion about a model, between ANN and LAR, giving better out-of-sample fit than other.

The above out-of-sample results show that both neural network and linear autoregressive models outperform random walk forecasts by all the performance measures. As far as the out-of-sample forecasts of neural network and linear autoregressive models are concerned, the results are mixed and we do not find a clear "winner".

Forecast Horizon Effect

The predictive ability of a model greatly varies with length of forecast horizon. A model performing well in the short run may not perform well in the long run and vice versa. Therefore, it is worthwhile to see the forecasting ability of neural network, linear autoregressive, and random walk models in long and short forecast horizons. For our analysis, we take four forecast horizons such as 1 month, 3 months, 6 months, and 12 months in both in-sample and out-of-sample forecasting of daily exchange rate returns.

The relative performances of all three studied models under four forecast horizons are measured in terms of root mean square error and sign predictions. The results are reported in Table 3.6 and Table 3.7

Table 3.6 presents the in-sample performances of neural network, linear autoregressive and random walk models in terms of RMSE and sign predictions (SIGN) in different forecast horizons. The results show that neural network performs better than linear autoregressive and random walk models under all four forecast horizons in terms of RMSE. For example, in 1 month forecast horizon ANN has smaller RMSE, which is equal to 0.1659, than the corresponding RMSEs of LAR and RW. Similarly, in 12 months forecast horizon, neural network takes on smaller RMSE value of 0.0631 as compared to 0.2886 for linear autoregressive and 0.4550 for random walk model. It is also seen from the table that the in-sample performance of neural network becomes better as the length of the forecast horizon increases. This is reflected in falling RMSE of neural network as forecast horizon increases from 1 month to 12 months.

However, contrary to this, the in-sample performance of neural network deteriorates in terms of sign prediction as the length of forecast horizon increases. For example, in 1 month forecast horizon, neural network gives as high as 85% correct sign prediction. But as the length of forecast horizon extends, the percentage of correct sign predictions falls. To illustrate, the correct sign prediction in 3 months, 6 months, and 12 months forecast horizon are 69%, 59%, and 51% respectively. But, it can also be seen in the table that neural networks in-sample sign predictions are better than both linear autoregressive and random walk models in all forecast horizons.

The out-of-sample RMSEs and SIGNs of ANN, LAR, and RW in different forecast horizons are shown in Table 3.7. The results suggest that neural network completely beats random walk under each forecast horizon in terms of RMSE. Neural network has smaller out-of-sample RMSE than that associated with the random walk under each forecast horizon. Again, this finding clearly disproves random walk model and contradicts the findings of Meese and Rogoff (1983a, 1983b, 1988), which show that the out-of-sample performance of many structural and time series models is no better than that of a simple random walk model. However, between ANN and LAR, LAR outperforms ANN under all forecasting horizons. In the table, it also can be seen that there is no consistent pattern in RMSE of neural network under different forecast horizons. The RMSE falls from 0.0657 to 0.0413 when forecast horizon increases from 1 month to 3 months. Again it rises to 0.0477 when the forecast horizon increases to 6 months and then falls. Nevertheless, it can be said that neural network performs better in long run than in short run.

On the other hand, neural network performs better in short run than in long run when sign prediction is considered as the performance measure. This is justified on the ground of falling values of sign prediction as the length of forecast horizon increases. In 1 month forecast horizon, the sign prediction for ANN is 0.6500. The value for sign prediction falls to 0.5555 and 0.5354 respectively when the forecast horizon increases to 3 months and 6 months. Neural network and linear autoregressive models outperform random walk in terms of correct sign predictions under all forecast horizons. This finding, again, strengthens the view against random walk hypothesis. Between ANN and LAR, ANN outweighs LAR in out-of-sample sign predictions under all forecast horizons except for 3 months forecast horizon in which the valuc of sign prediction is same for both models.

Daily Exchange Rate Prediction at Level

We have so far considered the forecast of daily exchange rate returns. Keeping in view the fact that it is considered relatively more difficult to predict exchange rate returns than the exchange rate at level, in this section we look at the predictive power of all the three alternative models in predicting daily exchange rates at level. Predicted values of exchange rate at level are obtained by taking the exponent of predicted exchange rate returns. We use seven performance measures, RMSE, MAE, MAPE, MAD, R^2, CORR, and DA to assess both in-sample and out-of-sample forecasts of all the three alternative models.

The in-sample results, given in Table 3.8, show neural network's in-sample forecasts are better than random walk forecasts by all the performance measures except for MAD. Similarly, neural network is found to beat linear autoregressive model by five out of seven performance measures in in-sample forecasting. For example, neural network outperforms linear autoregressive model when performance measures such as RMSE, MAE, MAPE, R^2, and DA are considered. The values for CORR are same for both the models. LAR outperforms ANN by only one performance measure i.e. MAD. Figure 3.5 which plots the in-sample errors of three alternative models shows that ANN gives better in-sample fit than LAR and RW as reflected by less fluctuations in predicted errors and smaller error variance. The value of variance of neural network predicted errors, which is equal to 0.0116, is less than the corresponding values of linear autoregressive and random walk models, which arc equal to 0.0172 and 0.0174 respectively. These findings in turn indicate that ANN gives better in-sample fit than LAR and RW in predicting daily exchange rate at level. This is consistent with the findings of in-sample R^2 (see Table 3.8) where neural network gives higher R^2 than linear autoregressive and random walk models.

The above results show that neural network gives better in-sample fit than linear autoregressive and random walk model. Now we turn to see the out-of-sample forecasts of ANN, LAR, and RW models. The out-of-sample performance of ANN, LAR, and RW in daily exchange rate at level prediction is given in Table 3.9. The results convey that except for MAE, and MAPE, ANN has superiority over random walk in out-of-sample forecasts by all other performance measures. The RMSE, MAD, CORR, R^2, and DA values for neural network are 0.0466, 0.0189, 0.9977, 0.9954, and 0.54 respectively. The corresponding figures for random walk are 0.0474, 0.0190, 0.9976, 0.9953, and 0 respectively. The values for MAPE, which is equal to 0.0006, are same for both the models. However, the out-of-sample results are mixed as far as the performances of ANN and LAR are concerned. ANN outperforms LAR in terms of RMSE, R^2, and CORR and is outperformed by LAR in terms of MAE, MAD and DA. MAPE, which is equal to 0.0006, is same for both ANN and LAR. As a result, we do not find a clear "winner" between ANN and LAR in the out-of-sample forecasts. Like neural network, except for MAE, linear autoregressive forecasts are better than random walk forecasts in terms of RMSE, MAD, and DA. The values of all other performance measures are same for both the models.

The out-of-sample forecast errors by ANN, LAR, and RW in predicting daily exchange rate at level are shown in Figure 3.6. The figure shows that the plots for neural network, linear autoregressive and random walk predicted errors are almost all similar. For this reason, we find it difficult to conclude, by looking at the figure, if any particular model is giving better fit than its colleagues. This is further corroborated on the basis of almost equal values for error variances of ANN, LAR, and RW, which are equal to 0.0021, 0.0022, and 0.0022 respectively. From these findings, however, we can conclude that all three models give high approximation to the actual values, which is also implied by high R^2 values (see Table 3.9) of ANN, LAR, and RW models.

3.3 Weekly Exchange Rate Returns Prediction

In the foregoing section, we have presented the daily exchange rate returns prediction by ANN, LAR, and RW models. However, some of the neural network practitioners expect neural network to perform better on weekly exchange rate as it is more normally distributed than daily exchange rate (Hann and Steurer, 1996; Hu et al., 1999). Hence, the present section looks at the performances of neural network and its colleagues in forecasting weekly exchange rate returns.

3.3.1 Description of Data

The weekly spot rates of Indian rupee/US dollar (INR/USD) exchange rate data set is taken from Pacific FX database for the period of January 6, 1993 to July 10, 2002, for a total 497 observations. The weekly returns are calculated as the log differences of the levels. These are both shown in Figure 3.7. In Figure 3.8, we show Quantile-Quantile plots of both weekly exchange rates at level and return against the normal distribution. The two plots in Figure 3.8 clearly indicate that the weekly exchange rate returns has a distribution closer to the normal than the exchange rates at level. Table 3.10 presents the summary statistics of the data. Both the skewness and kurtosis are substantially high indicating non-normality and leptokurtosis of weekly exchange rate returns. Jarque-Bera statistic also rejects the normality of exchange rate. The series shows evidence of autocorrelation indicated by five calculated autocorrelations. The Ljung-Box (LB) statistic for the first 20 lags is 42.34, which rejects the hypothesis of identical and independent observations. The value for Augmented Dicky-Fuller (ADF) test statistic assures the stationary of the return series. The value of Hurst exponent i.e. 0.5258 confirms the presence of long memory effect in the weekly exchange rate return.

3.3.2 Model Specification

The model specification for neural network, linear autoregressive, and random walk models in case of weekly exchange rate returns is mostly similar to the model specification in daily exchange rate returns. We will start with the model specification for neural network. In neural network, out of total 496 observations, we keep 350 observations for training (i.e. in-sample data) and remaining 146 observations are kept for testing (i.e. out-of-sample data). The input and output values are normalized in the range [0, 1]. As far as the hidden layer is concerned, a single-hidden-layer feedforward network is used for the training. We use sigmoid transfer function in the hidden layer and linear transfer function in the output layer. The weights are initialized to small values based on the technique of Nguyen and Widrow (1990). Mean square error is taken as the cost function. Resilient backpropagation algorithm is used as the training algorithm.

The number of input nodes and hidden nodes to the network are selected through experimentation. The number of input nodes, in this work, corresponds to the number of lagged past observations. Twenty levels of the number of input nodes ranging from 1 to 20 lagged values of the dependent variable, i.e. weekly exchange rate returns, are used in this study. We experiment with five levels of hidden nodes 4, 8, 12, 16, and 20 across each level of input node. Thus, the combination of 20 input nodes and five hidden nodes yields a total of 100 neural network architectures being considered for each in-sample training data set. To avoid the problem of getting stuck in local minima and to increase the possibility of getting the true global optima, we train 100 neural network architectures 10 times by using 10 different sets of initial random weights.

The effects of input nodes and hidden nodes on neural network training performance are shown in Table 3.11. The RMSEs and MAEs, in the table, at each level of input node in combination with each of five hidden node levels are the averages of ten runs. The results show that as the number of input nodes increases, except for input nodes 6, 12, 13, and 15, RMSE decreases as reflected by the average of RMSEs across the five hidden node levels at each level of the input node. Hence, we drop the input nodes 6, 12, 13, and 15 from the vector of input nodes and use other sixteen input nodes to forecast the weekly exchange rate returns. The simple reason is that the dropped variables do not contribute anything to explain the behaviour of the exchange rate returns as reflected by the increasing average RMSEs at these input levels across five hidden nodes levels. The results also show that RMSE decreases as the number of hidden nodes increases at each level of input node except for input nodes 7, 10, 11, 15, 17, 19, and 20. However, one more point to note is that there is less variation among different hidden node levels within each input level than among different input node levels. This finding suggests that neural network's model fitting performance is more sensitive to the input nodes than the hidden nodes.

We observe a different pattern for the effects of the input nodes and the hidden nodes with MAE. The average MAE decreases as the number of input nodes increases except for the levels of input nodes 6, 7, 12, 13, and 15. However, MAE does not show a clear hidden node effect. From the above results, we find that the pattern for the effect of the input node with RMSE is different from the pattern with MAE. However, we select the number of input nodes on the basis of average RMSEs at each level of input node across the five levels of hidden nodes. This is reasonable since the purpose of our neural network training is to minimize the sum of squared errors or equivalently RMSE. We also observe that networks performance is not sensitive to the number of hidden nodes. Moreover, we do not find a clear hidden node effect. Because of these findings we could not settle down at a specific optimum hidden node. We try with more than one hidden layer combined with different hidden nodes. After trying several combinations, we find two hidden layers with three hidden units in each layer as the best combination. Hence, the optimal neural network architecture i.e. 16-3-3-1 is used for the in-sample and out-of-sample forecasting of weekly exchange rate returns.

Next, we discuss the model specification of linear autoregressive and random walk models. To ensure that forecast comparisons are fair among all the three studied models, we take the same in-sample data, i.e. 350 observations, as in the case of neural network, for the model estimation in linear autoregressive and random walk models.

To select the optimum autoregressive terms in linear autoregressive model, we first regress the dependent variable y, on a large group of independent variable. We take the first 20 lagged values of the dependent variable i.e. from y_{t-l} to y_{t-20} as explanatory variables. Then a small group of statistically significant variables are identified and used as explanatory variables to forecast weekly exchange rate returns. The regression result is presented in Table 3.12. Among all the 20 lags, lags 1, 4, 6, 7, 15, and 19 are found to be significant and are used as explanatory variables for the problem of forecasting weekly exchange rate returns.

For random walk model without drift we take current period value as the best predicting value for tomorrow.

3.3.3 Empirical Findings

After selecting the appropriate models for neural network, linear autoregressive and random walk, we next turn to see the forecasting power of all three studied models. First, in-sample forecasts are presented, followed by out-of-sample forecasts.

In-sample Forecasting

In-sample performance of neural network, linear autoregressive and random walk models is presented in Table 3.13. The results show that neural network outperforms both linear autoregressive and random walk model by all the evaluation criteria. The RMSE of ANN i.e. 0.2048 is significantly lower than the RMSEs of LAR and RW model which are equal to 0.2518 and 0.3325 respectively. Neural network has also got smaller values for MAE and MAD as compared to the values of linear autoregressive and random walk models. The neural network fitted values have higher correlation, which is equal to 0.6687, with the actual series, as compared to the values of linear autoregressive (0.4062) and random walk model (0.2720). In the Table 3.13, we can see that neural network gives better sign predictions than linear autoregressive and random walk model. It can also be seen that neural network's direction accuracy of ANN, which is equal to 0.6848, is higher than the corresponding value of LAR (0.6424). As pointed out earlier, the direction accuracy is always zero for random walk. We are once again unable to calculate the MAPE for all the three models, as there are zero values for some of the actual returns.

In Figure 3.9, we find that neural network forecast values deviate less from actual values, shown by the fact that the errors are more close to horizontal line than linear autoregressive and random walk models. The forecast error variance, which is equal to 0.0421 for ANN is much smaller than forecast error variances of LAR and RW, which are equal to 0.0636 and 0.1109 respectively. From these findings we can say that neural network gives better predicted values for weekly exchange rate returns than linear autoregressive and random walk models. This implies that ANN gives better in-sample fit than LAR and RW in forecasting weekly exchange rate returns. It also can be seen from the figure that LAR gives better in-sample fit that than RW.

Out-of-sample Forecasting

The results for out-of-sample performance are presented in Table 3.14. For all the three models and by almost all evaluation criteria, the out-of-sample results tend to be better than the in-sample. The out-of-sample forecasts of neural network and linear autoregressive models are more accurate than random walk forecasts by all criteria except for the Pearson correlation coefficient. The correlation coefficient for ANN and LAR are 0.1851 and 0.2201 respectively as compared with thc corresponding figure of random walk, which is equal to 0.2535. The superiority of ncural network and linear autoregressive over random walk in out-of-sample forecasts provides further evidence against the efficient market hypothesis. However, between neural network and linear autoregressive model, neural network has superior out-of-sample when RMSE, MAE, MAD, and DA are considered as evaluation criteria. The values of RMSE, MAE, and MAD for ANN are 0.1087, 0.0676, and 0.0318 respectively. The corresponding values for LAR are 0.1096, 0.0740, and 0.0350 respectively. The neural network has higher direction accuracy (0.7063), than linear autoregressive model (0.5793). On the other hand, linear autoregressive model outperforms neural network in terms of correlation coefficient and the percentage of correct sign prediction. Figure 3.10 shows that both ANN and LAR are better than RW in terms of smaller forecast errors as shown by less fluctuation in the graph. The forecast error variances of ANN and LAR, which are equal to 0.0118 and 0.0121, are smaller than the forecast error variance of RW, which is equal to 0.0181. Hence, it can be said that both ANN and LAR give better out-of-sample fit than RW in forecasting of weekly exchange rate returns. Between ANN and LAR, the plots of both models in the figure look alike. However, if we compare the values of forecast error variance for both models then we find that ANN has smaller value, which is equal to 0.0118 than LAR, in which the value is 0.0121. Following this observation, it can be concluded that ANN gives better out-of-sample fit than LAR in predicting weekly exchange rate returns.

The above out-of-sample results show that both neural network and linear autoregressive model outperform random walk by all the evaluation criteria, except for correlation coefficient. More over, neural network out-of-sample prediction is found to be better than linear autoregressive model by four out of six evaluation criteria.

Forecast Horizon Effect

The results for horizon effects on performances of ANN, LAR, and RW in weekly exchange rate returns prediction are given in Table 3.15 and Table 3.16. As in the case of daily exchange returns, here also, we take RMSE and sign prediction as two performance measures to evaluate the performances of neural network, linear autoregressive and random walk models under different forecast horizons.

Table 3.15 reports the in-sample RMSEs and SIGNs of ANN, LAR, and RW models under 1 month, 3 months, 6 months, and 12 months forecast horizons. The table shows that in terms of RMSE, the in-sample performance of neural network is better in the long run than in the short run. For example, the RMSE of ANN in 12 months forecast horizon, which is equal to 0.0567, is lower than RMSE value of 0.0620 in 1 month forecast horizon. ANN outperforms RW model under all forecast horizons except 6 months forecast horizon. ANN has got higher RMSE value of 0.0600 than the RMSE of RW, which is equal to 0.0589, in 6 months forecast horizon. However, it also can be seen from the table that RMSEs of neural network under all forecast horizons are lower than the corresponding RMSEs of linear autoregressive model. This suggests that neural network gives superior in-sample forecasting in terms of RMSE as compared to linear autoregressive model under short as well as long forecast horizons.

As far as the sign prediction is concerned, the in-sample performances of all three models deteriorate, in general, as the forecast horizon increases. ANN and LAR are found to give better weekly in-sample sign predictions than random walk under all forecast horizons. However, the sign predictions are equal for ANN and LAR in 1 month and 3 months forecast horizon. When the forecast horizon increases beyond 3 months, the ANN is found to perform better than LAR. In other words, we can say that neural network gives better in-sample sign prediction than linear autoregressive model in long forecast horizon.

In Table 3.16, we present weekly out-of-sample RMSEs and SIGNs of ANN, LAR, and RW models under different forecast horizons. The results suggest that the out-of-sample performance in terms of RMSE of all three studied models become worse as forecast horizon increases from 1 month to 12 months. Except for 1 month forecast horizon, neural network has got smaller RMSEs than those of random walk model. In one month forecast horizon, neural network takes on an RMSE value of 0.0333 as compared to 0.0224 for random walk. From this finding it may be inferred that random walk forecasts better than neural network in short run. But when the forecast horizon increases from 3 months to 12 months, neural network starts performing better than random walk. We also find from the table that neural network has superior out-of-sample forecasting than linear autoregressive model under two forecast horizons i.e. 1 month and 3 months. Under 1 month and 3 months forecast horizons, neural network gives RMSE values of 0.0333 and 0.0638 respectively, much smaller values than 0.0434 and 0.0655 respectively, given by linear autoregressive model. However, LAR outperforms ANN in 6 months and 12 months forecasts. This is reflected in smaller values of RMSEs for LAR as compared to RMSE values of ANN under 6 months and 12 months forecasts. This finding leads us to conclude that neural network performs better than linear autoregressive model in the short run but not in the long run. In the long run, it is linear autoregressive model which performs better in out-of-sample forecasting than neural network.

The out-of-sample sign predictions presented in Table 3.16 show that neural network gives better weekly out-of-sample sign prediction in the long run than in short run. For example, neural network gives only 40% correct sign prediction in 1 month forecast horizon whereas it gives more than 70% correct prediction in 6 months forecast horizon. Similarly, LAR and RW are also found, in general, to perform better in long forecast horizon than short forecast horizon. Neural network and linear autoregressive models are outperformed by random walk model in 1 month forecast horizon. But as the length of forecast horizon increases, ANN and LAR start outperforming random walk model. For example neural network and linear autoregressive models give better weekly out-of-sample sign predictions than random walk in 3 months, 6 months, and 12 months. Both ANN and LAR give equal sign predictions in 1 month and 3 months forecast horizon. ANN outperforms LAR in terms of sign prediction in 6 months forecast horizon and is outperformed by LAR in 12 months forecast horizon. From these findings it appears that neural networks gives better out-of-sample sign prediction than random walk in long forecast horizons. The sign predictions of ANN and LAR are equal in short forecast horizon and mixed in long forecast horizon.

Weekly Exchange Rate Prediction at Level

The results for in-sample and out-of-sample prediction of weekly exchange rate at level are presented in Table 3.17 and Table 3.18. The results in Table 3.17 show that neural network gives better in-sample forecasts than both linear autoregressive and random walk models by all the seven performance measures. The values of RMSE, MAE, MAPE, and MAD for ANN, which are equal to 0.0749, 0.0385, 0.0010, and 0.0138 respectively, arc less than the corresponding values for both LAR and RW models. Neural network has also higher goodness of fit (R^2) and correlation coefficient than both the benchmark models. The percentage of correct direction prediction, equal to 0.7757 for ANN, is greater than the corresponding figure for LAR (0.7484). For random walk it is zero. The table also shows that linear autoregressive model has superior in-sample forecasts than random walk model when all the performance measures are considered.

On merely looking at the Figure 3.11, which plots the in-sample errors of ANN, LAR, and RW, it can be immediately said that ANN has less prediction errors and greater accuracy in predicting weekly exchange rate at level than LAR and RW model. This is reflected in the smaller variation of neural network forecast errors than the forecast errors of linear autoregressive and random walk models. We find that the variance of the forecast error of ANN, which is equal to 0.0056, is smaller than the variance (0.0085) of LAR error and is much smaller than the random walk's forecast error variance, which is equal to 0.0341. From this, it can be concluded that ANN gives better in-sample fit than LAR and RW. This finding is consistent with the finding of higher R^2 of ANN in comparison to R^2 of LAR and RW models (see Table 3.17). Figure 3.11 also illustrates that LAR gives better in-sample fit than random walk as shown by less fluctuation around zero in LAR errors than RW errors.

The out-of-sample results in Table 3.18 indicate that neural network is better than random walk in out-of-sample forecasting by all performance measures. For example, ANN takes on a RMSE value of 0.0502, which is less than RMSE (0.1015) for random walk. Similarly, if we compare all other performance measures, we find that neural network has complete dominance over random walk in out-of-sample forecasting. However, between LAR and ANN, ANN is found to outperform LAR by all measures except for CORR and R^2 in out-of-sample forecasting. The values of CORR and R^2, which are equal to 0.9995 and 0.9990, are same for both the models. Figure 3.12, which shows the out-of-sample errors of ANN, LAR, and RW, demonstrate that the plots of errors of ANN and LAR look almost similar. Thus, by looking at the plots, it is not easy to conclude anything in relation to the relative fits of respective models. We also find the error variance, which is equal to 0.0025, for both models as equal. However, by looking at the graph, it can be said that both ANN and LAR have lesser fluctuations in their errors relative to the errors of RW. Thus, we can conclude that both ANN and LAR give better out-of-sample fit than RW in the forecasting of weekly exchange rate at level.

3.4 Forecast Encompassing Test

In earlier sections, we discussed the relative performances of ANN, LAR, and RW models in terms of different performance measures such as RMSE, MAE, MAPE, MAD, CORR, R^2, DA and SIGN. Another measure to evaluate the out-of-sample forecasting performance of each model relative to other models is forecast encompassing test due to Chong and Hendry (1986). Forecast comparison by RMSE and other such measures does not tell whether any one model's performance is significantly better than that of other models, in a formal statistical sense. We therefore use forecast encompassing test, which allows us to test for significantly better performance. To formalize the notion of forecast encompassing, it should be noted that forecast error from a correctly specified model has a conditional first moment of 0 given any conditioning information available to the forecaster. Thus, given two different models estimated on the same data, model *j* 's forecast error should be orthogonal to model *k's* forecast provided that models accurately fits the data. Provided that model j does accurately fit the data, conditioning on the forecast from model k should therefore not help to explain any of model j's forecast error.

Forecast encompassing test formalizes the intuition that model j should be preferred to model k if model j can explain what model k cannot explain, without model k being able to explain what model j cannot explain. As such, the test provides a useful method for ranking out-of-sample forecasts. The test is therefore based on a set of OLS regressions of the forecast error from one model on the forecast error from the other model. Thus, letting E_j be model j's forecast error and F_k be model k's forecast error, the tests for encompassing involve testing for significance of the θ_l parameter in the regression

$$E_{jt} = \theta_0 + \theta_1 F_{kt} + \eta_t \quad .... \tag{3.10}$$

Given forecasts from two models, j and k, we test the null hypothesis that neither model encompasses the other. We first regress the forecast error from model j on the forecast error from model k, as in (3.10). We call this 'regression jk' and the resulting estimate of the θ_l coefficient $\theta(jk)$. We then regress the forecast error from model k on the forecast error from model j. Again, we call this 'regression kj' and the resulting θ_l estimate $\theta(kj)$. If $\theta(jk)$ is not significant at some predetermined level, but $\theta(kj)$ is significant, then we reject the null hypothesis that neither model encompasses the other in favor of the alternative hypothesis that model j encompasses model k. Conversely, if $\theta(kj)$ is not significant, but $\theta(jk)$ is significant, then we say that model k encompasses model j. However, if both $\theta(jk)$ and $\theta(kj)$ are significant, or if both $\theta(jk)$ and $\theta(kj)$ are not significant, then we fail to reject the null hypothesis that neither model encompasses the other. Multicollinearity can lead to both estimated coefficients being insignificant, while sufficiently non-overlapping information sets can lead to both estimated coefficients being significant.

The encompassing results for both daily and weekly out-of-sample forecasts of exchange rate returns are given in Table 3.19 and Table 3.20 respectively. Since the encompassing test has an easily derivable distribution when applied to the out-of-sample data, but not when applied to the in-sample data, we present only out-of-sample encompassing results. The name of the dependent variable from (3.10) is listed down the left side of the table, while the independent variable is listed along the top. The entries in the Table 3.19 and Table 3.20 are estimated coefficients θ with the associated p-values in parenthesis. The daily out-of-sample forecast encompassing test, shown in Table 3.19, shows that all the estimated coefficients are significant as indicated by their associated p-values. For example, the estimated coefficient from the regression of forecast error from ANN on the forecast error from LAR, which is equal to 0.9506, is significant as indicated by its associated p-values. Conversely, the estimated coefficient from the regression of forecast error from LAR on the forecast error from ANN is also significant. Hence, from this finding, we can say that neither ANN nor LAR encompasses each other i.e. neither of the models can explain part of other's forecast error. Since all the coefficients are significant in Table 3.19, we fail to reject the null hypothesis that none of ANN, LAR and RW model encompasses the other. Similarly, for weekly exchange rate return, the result of out-of-sample encompassing test, which is shown in Table 3.20, reveals that all the estimated coefficients are significant. As a result, in this case also, we find that none of the three studied models encompasses the other.

3.5 Conclusion

In this chapter, we have attempted to forecast daily and weekly exchange rate returns using neural network, linear autoregressive and random walk models. In case of daily exchange rate returns, neural network is seen to give better in-sample and out-of-sample performances than random walk model. However, between neural network and linear autoregressive model, though neural network beats linear autoregressive model in in-sample forecasting, we find that the results for out-of-sample forecasting are mixed. We do not find a clear "winner" model between these two. These findings are consistent with the findings of Kuan and Liu (1995) and contradict the findings of Diebold and Nason (1990) that the nonlinearity in exchange rate cannot be exploited to improve both point and sign forecasts. Again, it is also found that neural network performs better, in terms of RMSE, in long run than in short run in daily exchange rate returns prediction However, we find that neural network's performance becomes worse as the length of forecast horizon increases when sign prediction is considered as a performance measure. In case of exchange rate prediction at level, we find that random walk model is outperformed by neural network by all the performance measures in both in-sample and out-of-sample forecasting. However, there is no clear winner model between neural network and linear autoregressive model in out-of-sample forecasting.

For weekly exchange rate returns prediction, our findings suggest that neural network has superior in-sample forecast than linear autoregressive and random walk models. As far as the out-of-sample forecasting is concerned, neural network outperforms random walk by all evaluation criteria, except for Pearson correlation coefficient. Neural network is also found to beat linear autoregressive model by four out of six evaluation criteria in out-of-sample forecasting. The study also suggests that neural network's out-of-sample performance, in terms of RMSE, decreases as the length of horizon increases. However, on the other hand, neural network's out-of-sample performance, in terms of sign prediction, increases as the length of forecast horizon increases. Again it is also found that neural network has superior out-of-sample forecasting than both random walk and linear autoregressive model in exchange rate prediction at level. Forecast encompassing test shows that no model encompasses other in both daily and weekly exchange rate returns prediction. In addition to these findings, we also find that neural network's performance is more sensitive to the input nodes than the hidden nodes, which is consistent with the findings of Tang and Fishwick (1993), Zhang and Hu (1998) and Hu *et al* (1999).

Table 3.1 Summary statistics for daily exchange rates: log first difference January 4, 1993-July 12, 2002

Description	INR/USD
Sample size	2395
Mean	0.0094
Median	0.0010
SD	0.1527
Skewness	10.3158
Kurtosis	297.8218
Maximum	4.3957
Minimum	-0.9498
Jarque-Bera	8716362
	(0.000)
ρ_1	-0.086
ρ_5	0.044
ρ_{10}	-0.013
ρ_{15}	0.051
ρ_{20}	-0.017
LB statistic (20)	50.13
	(0.000)
ADF	-21.219
Hurst exponent	0.52733

Note: The p-value for Jarque-Bera and LB statistic is given in parentheses. The critical values for ADF test are -3.96, -3.41, and -3.12 at 1%, 5%, and 10% significance level respectively.

Table 3.2 Effects of input and hidden units on the training performance of ANN in forecasting daily normalized exchange rate returns

Input	Hidden	RMSE	MAE	Input	Hidden	RMSE	MAE
2	4	0.0297	0.0102	12	4	0.0271	0.0118
2	8	0.0297	0.0115	12	8	0.0272	0.0118
2	12	0.0296	0.0118	12	12	0.0274	0.0117
2	16	0.0296	0.0116	12	16	0.0274	0.0121
2	20	0.0293	0.0116	12	20	0.0259	0.0117
	Avg	0.0295	0.0113		Avg	0.0270	0.0118
4	4	0.0296	0.0119	14	4	0.0264	0.0116
4	8	0.0294	0.0119	14	8	0.0265	0.0118
4	12	0.0291	0.0018	14	12	0.0273	0.0120
4	16	0.0291	0.0118	14	16	0.0259	0.0117
4	20	0.0286	0.0117	14	20	0.0255	0.0118
	Avg	0.0291	0.0118		Avg	0.0263	0.0117
6	4	0.0297	0.0119	16	4	0.0257	0.0117
6	8	0.0291	0.0118	16	8	0.0259	0.0119
6	12	0.0289	0.0115	16	12	0.0247	0.0116
6	16	0.0288	0.0117	16	16	0.0234	0.0117
6	20	0.0286	0.0116	16	20	0.0208	0.0108
	Avg	0.0290	0.0117		Avg	0.0239	0.0115
8	4	0.0295	0.0117	18	4	0.0271	0.0120
8	8	0.0290	0.0118	18	8	0.0215	0.0111
8	12	0.0287	0.0116	18	12	0.0239	0.0116
8	16	0.0284	0.0117	18	16	0.0243	0.0120
8	20	0.0281	0.0115	18	20	0.0226	0.0115
	Avg	0.0287	0.0116		Avg	0.0238	0.0116
10	4	0.0290	0.0120	20	4	0.0255	0.0115
10	8	0.0287	0.0120	20	8	0.0221	0.0111
10	12	0.0282	0.0117	20	12	0.0218	0.0111
10	16	0.0278	0.0119	20	16	0.0203	0.0108
10	20	0.0275	0.0118	20	20	0.0206	0.0106
	Avg	0.0282	0.0118		Avg	0.0221	0.0110

Note: The RMSEs and MAEs at each level of input node in combination with each of the five hidden node levels are the averages of ten runs.

Table 3.3 Results for the regression of daily exchange rate return y_t on its own 20 lagged values

Variable	Coefficient	Std. Error	t-Statistic	Prob.
Constant	0.008916	0.003253	2.740922	0.0062
y_{t-1}	-0.084880	0.020609	-4.118600	0.0000
y_{t-2}	0.001610	0.020674	0.077855	0.9380
y_{t-3}	0.010687	0.020671	0.517014	0.6052
y_{t-4}	-0.004841	0.020662	-0.234286	0.8148
y_{t-5}	0.044556	0.020652	2.157462	0.0311
y_{t-6}	-0.011329	0.020647	-0.548716	0.5833
y_{t-7}	0.000642	0.020648	0.031093	0.9752
y_{t-8}	-0.000171	0.020642	-0.008294	0.9934
y_{t-9}	-0.043384	0.020632	-2.102788	0.0356
y_{t-10}	-0.019125	0.020613	-0.927786	0.3536
y_{t-11}	0.060542	0.020613	2.937016	0.0033
y_{t-12}	0.033295	0.020633	1.613726	0.1067
y_{t-13}	-0.022308	0.020644	-1.080617	0.2800
y_{t-14}	-0.005767	0.020649	-0.279293	0.7800
y_{t-15}	0.049920	0.020648	2.417677	0.0157
y_{t-16}	-0.006083	0.020653	-0.294526	0.7684
y_{t-17}	0.024373	0.020654	1.180060	0.2381
y_{t-18}	0.007372	0.020659	0.356831	0.7213
y_{t-19}	0.015703	0.020659	0.760083	0.4473
y_{t-20}	-0.014237	0.020587	-0.691557	0.4893

Table 3.4 **In-sample** performance of **ANN,** LAR, and RW models on **daily** exchange rate return series for the period February **2, 1993-May 9, 2001**

	ANN	LAR	RW
RMSE	0.1081	0.1615	0.2401
MAE	**0.0593**	0.0653	0.0992
MAPE	—	…	—
MAD	**0.0261**	0.0219	0.0387
CORR	**0.7482**	0.1316	**-0.0861**
DA	0.72	0.73	0
SIGN	0.5528	0.5209	0.4081

Table 3.5 Out-of-sample performance of ANN, LAR, and RW models on daily exchange rate return series **for** the period June 7, **2001-July** 12, 2002

	ANN	LAR	RW
RMSE	0.0445	0.0431	0.0599
MAE	0.0307	0.0283	0.0405
MAPE	—	—	—
MAD	0.0193	0.0158	0.0303
CORR	**0.1258**	-0.0017	0.0096
DA	0.71	0.71	0
SIGN	**0.5418**	0.5090	0.4327

Table 3.6 In-sample performance of ANN, LAR, and RW models on daily exchange rate return series in different forecast horizons

		1 month	3 months	6 months	12 months
RMSE	**ANN**	0.1659	0.1064	**0.0794**	0.0631
	LAR	1.0065	0.5797	0.4057	0.2886
	RW	1.5900	0.9140	0.6395	0.4550
SIGN	**ANN**	0.8500	0.6935	0.5984	0.5177
	LAR	0.5000	0.5161	0.4960	0.4861
	RW	0.6000	0.4677	0.3700	0.3478

Table 3.7 Out-of-sample performance of ANN, LAR, and RW models on daily exchange rate return series in different forecast horizons

		1 month	3 months	6 months	12 months
RMSE	**ANN**	0.0657	0.0413	0.0477	0.0457
	LAR	0.0652	0.0401	0.0469	0.0441
	RW	0.1214	0.0719	0.0685	0.0612
SIGN	**ANN**	0.6500	0.5555	0.5354	0.5498
	LAR	0.6000	0.5555	**0.5118**	0.5258
	RW	0.4500	0.4285	**0.4409**	0.4302

Table 3.8 In-sample performance of ANN, LAR, RW models on daily exchange rate at level series for the period February 2, 1993-May 9, 2001

	ANN	LAR	RW
RMSE	**0.1080**	0.1314	**0.1323**
MAE	0.0509	0.0544	**0.0545**
MAPE	0.0013	0.0014	0.0014
MAD	0.0200	0.0185	0.0180
CORR	0.9997	0.9997	0.9996
R^2	0.9995	0.9994	0.9993
DA	0.56	0.52	**0**

Table 3.9 Out-of-sample performance of ANN, LAR, and RW models on daily exchange rate at level series for the period June 7, 2001-July 12, 2002

	ANN	LAR	RW
RMSE	0.0466	0.0471	0.0474
MAE	0.0303	0.0299	0.0298
MAPE	0.0006	0.0006	0.0006
MAD	0.0189	0.0182	0.0190
CORR	0.9977	0.9976	0.9976
R^2	0.9954	0.9953	0.9953
DA	0.54	0.73	**0**

Table **3.10** Summary statistics for weekly exchange rates: log first difference January **6, 1993-** July **10,** 2002

Description	INR/USD
Sample size	496
Mean	0.0455
Median	0.0052
SD	0.2806
Skewness	2.4834
Kurtosis	42.5818
Maximum	3.1904
Minimum	-1.7316
Jarque-Bera	32888
	(0.000)
ρ_1	0.131
ρ_5	-0.002
ρ_{10}	-0.008
ρ_{15}	0.080
ρ_{20}	-0.059
LB statistic (20)	42.34
	(0.000)
ADF	-9.348
Hurst exponent	0.5258

Note: The p-value for Jarque-Bera and LB statistic is given in parentheses. The MacKinnon critical values for ADF test are -3.98, -3.42, and -3.13 at 1%, 5%, and 10% significance level respectively.

Table 3.11 Effects of Inputs and hidden units on the training performance of ANN in forecasting weekly normalized exchange rate returns

Input	Hidden	RMSE	MAE	Input	Hidden	RMSE	MAE
1	4	0.0522	0.0252	8	4	0.0375	0.0216
1	8	0.0508	0.0245	8	8	0.0342	0.0196
1	12	0.0508	0.0245	8	12	0.0330	0.0196
1	16	0.0492	0.0240	8	16	0.0304	0.0174
1	20	0.0485	0.0241	8	20	0.0294	0.0167
	Avg	**0.0503**	**0.0244**		Avg	**0.0329**	**0.0189**
2	4	0.0493	0.0249	9	4	0.0377	0.0216
2	8	0.0450	0.0225	9	8	0.0326	0.0192
2	12	0.0434	0.0219	9	12	0.0296	0.0175
2	16	0.0434	0.0219	9	16	0.0290	0.0170
2	20	0.0412	0.0207	9	20	0.0292	0.0171
	Avg	**0.0446**	**0.0223**		**Avg**	**0.0316**	**0.0184**
3	4	0.0417	0.0220	10	4	0.0351	0.0201
3	8	0.0418	0.0215	10	8	0.0340	0.0198
3	12	0.0397	0.0202	10	12	0.0269	0.0161
3	16	0.0382	0.0190	10	16	0.0285	0.0170
3	20	0.0375	0.0190	10	20	0.0266	0.0155
	Avg	**0.0397**	**0.0203**		Avg	**0.0302**	**0.0177**
4	4	0.0432	0.0224	11	4	0.0355	0.0211
4	8	0.0370	0.0199	11	8	0.0287	0.0171
4	12	0.0347	0.0171	11	12	0.0283	0.0166
4	16	0.0343	0.0191	11	16	0.0290	0.0172
4	20	0.0341	0.0183	11	20	0.0274	0.0158
	Avg	**0.0366**	**0.0193**		Avg	**0.0297**	**0.0175**
5	4	0.0410	0.0218	12	4	0.0332	0.0218
5	8	0.0353	0.0203	12	8	0.0309	0.0183
5	12	0.0333	0.0186	12	12	0.0298	0.0172
5	16	0.0317	0.0178	12	16	0.0278	0.0162
5	20	0.0311	0.0176	12	20	0.0279	0.0165
	Avg	**0.0344**	**0.0192**		Avg	**0.0299**	**0.0180**
6	4	0.0422	0.0235	13	4	0.0357	0.0205
6	8	0.0343	0.0213	13	8	0.0306	0.0184
6	12	0.0340	0.0200	13	12	0.0288	0.0168
6	16	0.0314	0.0185	13	16	0.0281	0.0167
6	20	0.0309	0.0176	13	20	0.0281	0.0162
	Avg	**0.0345**	**0.0201**		**Avg**	**0.0302**	**0.0177**
7	4	0.0418	0.0232	14	4	0.0388	0.0201
7	8	0.0329	0.0195	14	8	0.0278	0.0165
7	12	0.0311	0.0185	14	12	0.0276	0.0162
7	16	0.0315	0.0187	14	16	0.0270	0.0158
7	20	0.0300	0.0173	14	20	0.0254	0.0148
	Avg	**0.0334**	**0.0194**		**Avg**	**0.0293**	**0.0166**

continued from previous page

Input	Hidden	RMSE	MAE	Input	Hidden	RMSE	MAE
15	**4**	0.0349	0.0213	18	4	0.0284	0.0161
15	8	0.0278	0.0160	18	8	0.0277	0 0163
15	**12**	0.0258	0.0146	18	**12**	0.0244	0.0141
15	**16**	0.0263	0.0153	18	**16**	0.0241	0.0138
15	20	0.0249	0.0142	18	**20**	0.0238	0.0137
	Avg	**0.0349**	**0.0160**		**Avg**	**0.0256**	**0.0148**
16	**4**	0.0292	0.0167	19	4	0.0284	0.0162
16	8	0.0270	0.0154	19	8	0.0259	0.0151
16	**12**	0.0262	0.0150	19	**12**	0.0243	0.0142
16	16	0.0260	0.0152	19	**16**	0.0237	0.0139
16	20	0.0255	0.0142	19	**20**	0.0248	0.0152
	Avg	**0.0268**	**0.0153**		**Avg**	**0.0254**	**0.0149**
17	**4**	0.0303	0.0165	20	4	0.0288	0.0170
17	8	0.0272	0.0154	20	8	0.0252	0.0147
17	**12**	0.0246	0.0138	20	**12**	0.0235	0.0135
17	16	0.0262	0.0155	20	**16**	0.0236	0.0139
17	20	0.0254	0.0148	20	**20**	0.0252	0.0147
	Avg	0.0267	0.0152		**Avg**	0.0252	**0.0147**

Note: The RMSEs and MAEs at each level of input node in combination with each of the five hidden node levels are the averages of ten runs.

Table 3.12 Results for the regression of weekly exchange rate return, y_t on its own 20 lagged values

Variable	Coefficient	Std. Error	t-Statistic	Prob.
Constant	0.034513	0.016191	2.131633	0.0338
y_{t-1}	0.299380	0.056850	5.266149	0.0000
y_{t-2}	-0.084881	0.058604	-1.448386	0.1485
y_{t-3}	-0.010714	0.058756	-0.182338	0.8554
y_{t-4}	0.158997	0.058684	2.709388	0.0071
y_{t-5}	-0.111061	0.059288	-1.873252	0.0620
y.-6	0.216202	0.058777	3.678331	0.0003
y_{t-7}	-0.165247	0.059619	-2.771722	0.0059
y_{t-8}	-0.059820	0.059635	-1.003096	0.3166
y_{t-9}	-0.061949	0.059340	-1.043975	0.2973
y_{t-10}	-0.030713	0.059369	-0.517320	0.6053
y_{t-11}	0.004471	0.059215	0.075502	0.9399
y_{t-12}	0.033359	0.055107	0.605346	0.5454
y_{t-13}	0.079005	0.044623	1.770487	0.0776
y_{t-14}	-0.005359	0.044584	-0.120196	0.9044
y_{t-15}	0.096996	0.044406	2.184296	0.0297
y_{t-16}	-0.019446	0.044322	-0.438750	0.6611
y_{t-17}	0.031068	0.044248	0.702142	**0.4831**
y_{t-18}	-0.058802	0.044044	-1.335064	0.1828
y_{t-19}	-0.123937	0.043879	-2.824517	**0.0050**
y_{t-20}	0.006972	0.044180	0.157806	0.8747

Table 3.13 In-sample performance of ANN, LAR, and RW models on weekly exchange rate return series for the period May 19, 1993-September 15, 1999

	ANN	LAR	RW
RMSE	0.2048	0.2518	**0.3325**
MAE	0.1057	0.1341	0.1577
MAPE	—	—	—
MAD	0.0379	0.0519	0.0540
CORR	0.6687	0.4062	**0.2720**
DA	0.6848	0.6424	0
SIGN	**0.5181**	0.5030	**0.5030**

Table 3.14 Out-of-sample performance ANN, LAR, and RW models on weekly exchange rate return series for the period February 2, 2000-July 10, 2002

	ANN	LAR	RW
RMSE	0.1087	0.1096	0.1342
MAE	0.0676	0.0740	0.0854
MAPE	—	—	—
MAD	0.0318	0.0350	0.0474
CORR	0.1851	0.2201	0.2535
DA	0.7063	0.5793	0
SIGN	0.5714	0.6031	0.5634

Table 3.15 In-sample performance of ANN, LAR, and RW models on weekly exchange rate return series in different forecast horizons

		1 month	3 months	6 months	12 months
RMSE	ANN	**0.0620**	**0.0551**	**0.0600**	**0.0567**
	LAR	**0.0994**	**0.1228**	0.0988	**0.0816**
	RW	0.0770	**0.0569**	**0.0589**	**0.0597**
SIGN	**ANN**	0.6000	**0.6428**	**0.5185**	**0.4150**
	LAR	0.6000	0.6428	**0.4814**	**0.3962**
	RW	0.4000	0.2857	0.4074	0.3773

Table 3.16 Out-of-sample performance of ANN, LAR, and RW models on weekly exchange rate return series in different forecast horizons

		1 month	3 months	6 months	12 months
RMSE	ANN	0.0333	0.0638	**0.1359**	**0.1356**
	LAR	0.0434	0.0655	**0.1302**	**0.1334**
	RW	0.0224	**0.0681**	**0.1404**	**0.1630**
SIGN	**ANN**	**0.4000**	**0.6153**	**0.7037**	**0.6226**
	LAR	**0.4000**	**0.6153**	**0.6666**	**0.6415**
	RW	**0.6000**	**0.5384**	**0.6296**	**0.6226**

Table 3.17 In-sample performance of ANN, LAR, and RW models on weekly exchange rate at level series for the period May 19, 1993-September 15, 1999

	ANN	LAR	RW
RMSE	0.0749	0.0920	0.1880
MAE	0.0385	0.0489	0.0892
MAPE	0.0010	0.0013	0.0024
MAD	0.0138	0.0174	0.0259
CORR	0.9998	0.9997	0.9990
R^2	0.9996	0.9995	0.9981
DA	0.7757	0.7484	0

Table 3.18 Out-of-sample performance of ANN, LAR, RW models on weekly exchange rate at level series for the period February 2, 2000-July 10, 2002

	ANN	LAR	RW
RMSE	0.0502	0.0507	0.1015
MAE	0.0314	0.0344	0.0628
MAPE	0.0006	0.0007	0.0013
MAD	0.0149	0.0167	0.0320
CORR	0.9995	0.9995	0.9983
R^2	0.9990	0.9990	0.9966
DA	0.8333	0.8174	0

Table 3.19 Daily out-of-sample forecast encompassing coefficient θ_1 (p-values in parenthesis) from $E_{jt} = \theta_0 + \theta_1 F_{kt} + \eta_t$

Forecast error E_{jt} from ↓	Forecast error F_{kt} from ↓		
	ANN	LAR	RW
ANN	-----	0.9506 (0.000)	0.4561 (0.000)
LAR	0.9372 (0.000)	------	0.4549 (0.000)
RW	0.8753 (0.000)	0.8854 (0.000)	-----

Table 3.20 Weekly out-of-sample forecast encompassing coefficient θ_1 (p-values in parenthesis) from $E_{jt} = \theta_0 + \theta_1 F_{kt} + \eta_t$

Forecast error E_{jt} from ↓	Forecast error F_{kt} from ↓		
	ANN	LAR	RW
ANN	-----	0.9409 (0.000)	0.6233 (0.000)
LAR	0.9601 (0.000)	------	0.6389 (0.000)
RW	0.9549 (0.000)	0.9591 (0.000)	-----

Figure 3.1 Plots of daily INR/USD exchange rates at level and returns for the period January 4, 1993-July 12, 2002

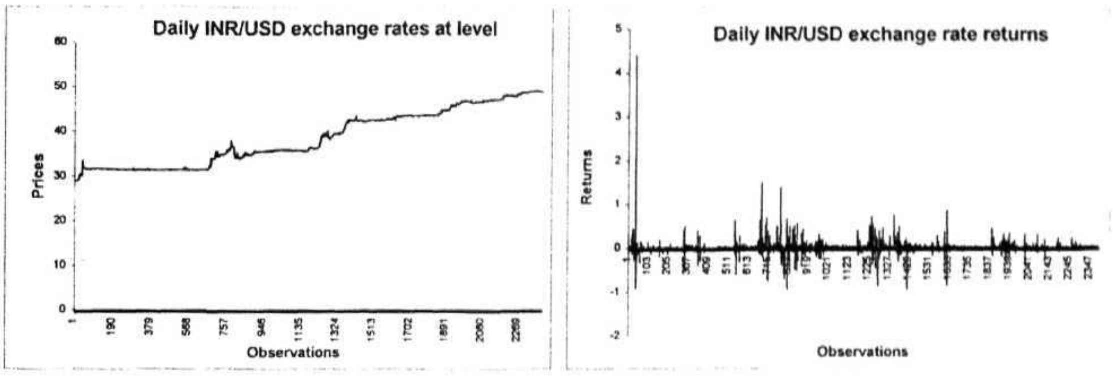


Figure 3.2 Quantile-Quantile (QQ) plots of daily exchange rates at level and returns against normal distribution

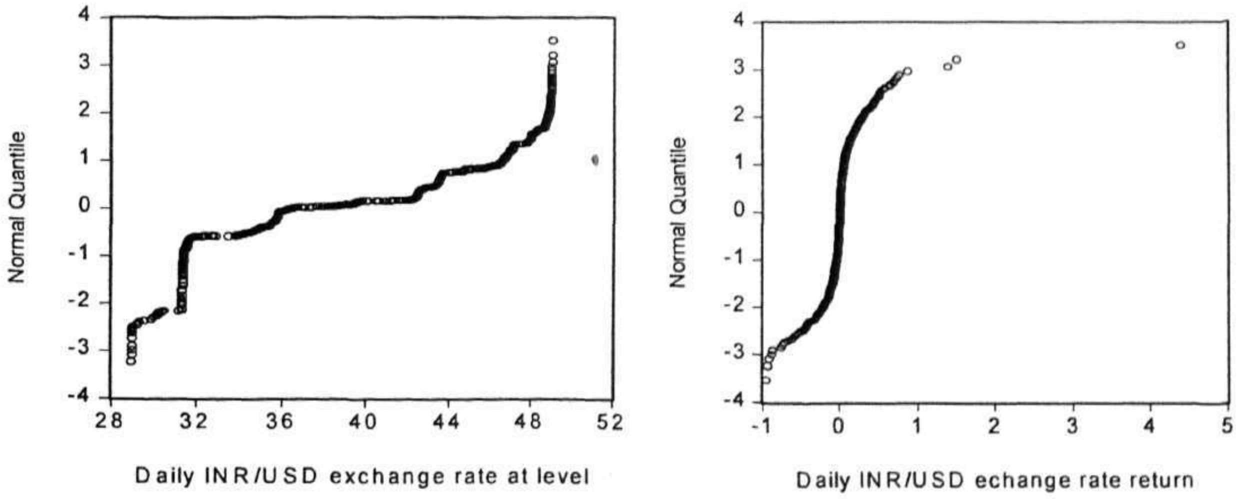


Figure 3.3 Plots of in-sample errors of ANN, LAR, and RW in predicting daily exchange rate returns for the period February 2, 1993-May 9, 2001

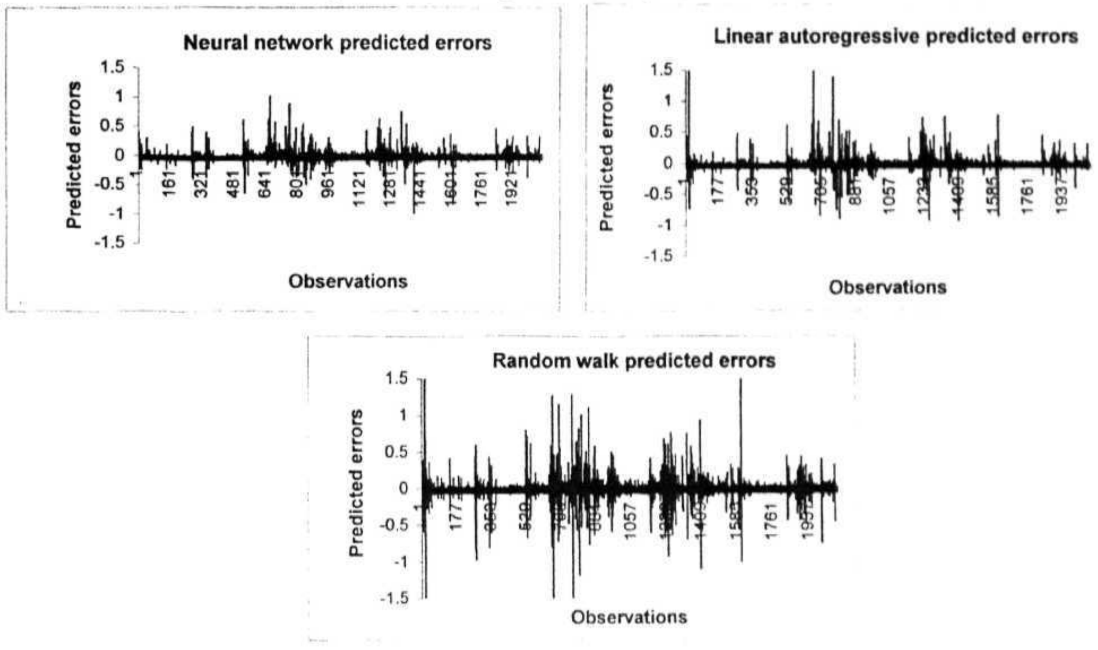


Figure 3.4 Plots of out-of-sample errors of ANN, LAR, and RW in predicting daily exchange rate returns for the period June 7, 2001-July 12,2002

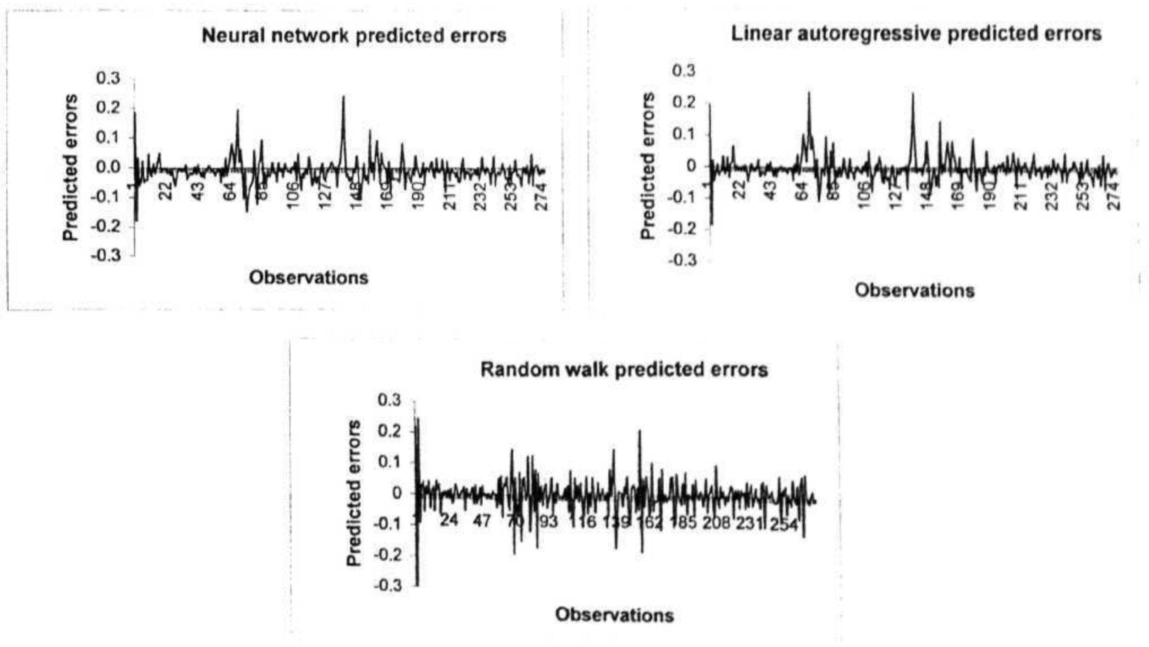


Figure 3.5 Plots of in-sample errors of ANN, LAR, and RW in predicting daily exchange rate at level for the period February 2, 1993-May 9, 2001

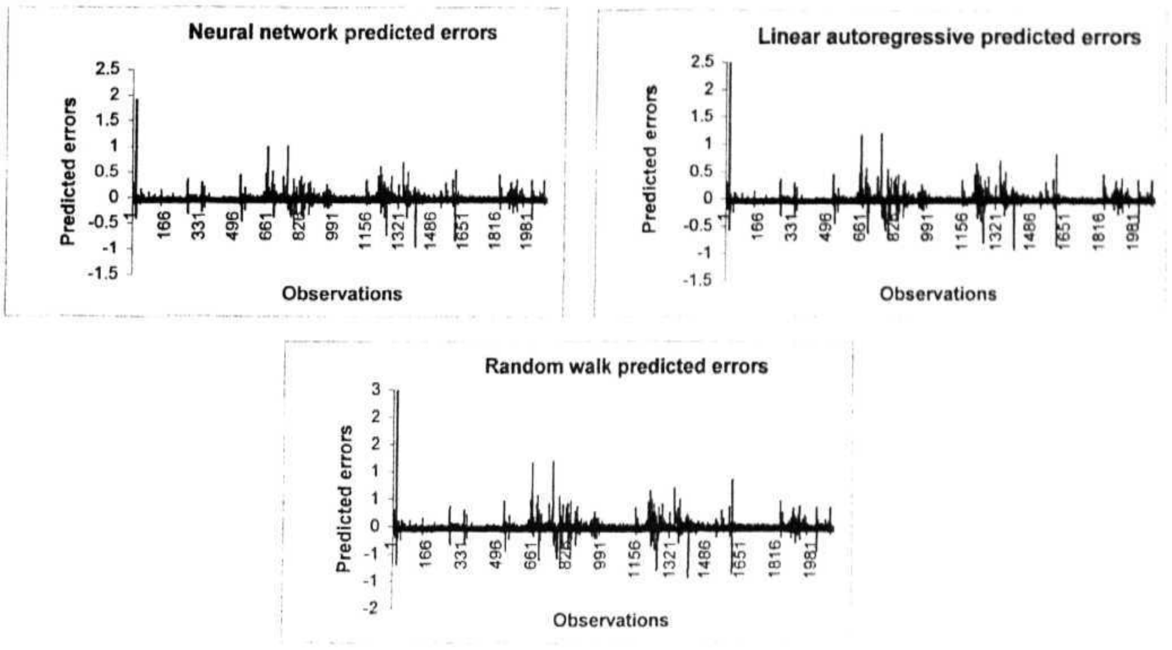


Figure 3.6 Plots of out-of-sample errors of ANN, LAR, and RW in predicting daily exchange rates at level for the period June 7, 2001-July 12,2002

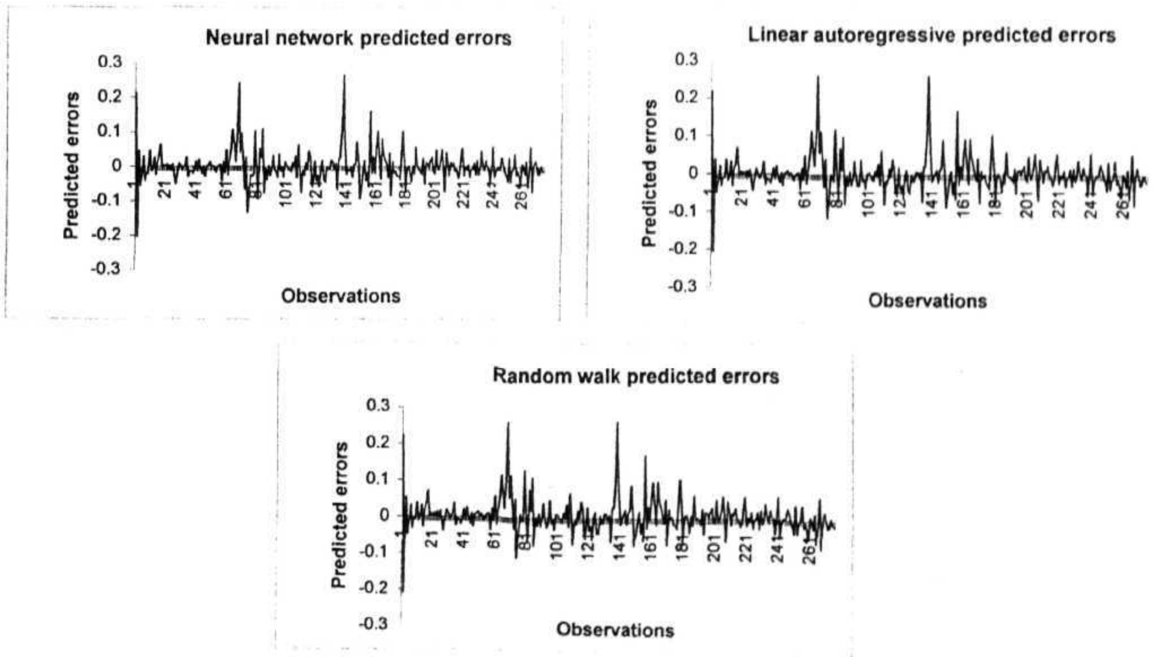


Figure 3.7 Plots of weekly INR/USD exchange rates at level and returns for the period January 6, 19993-July 10, 2002

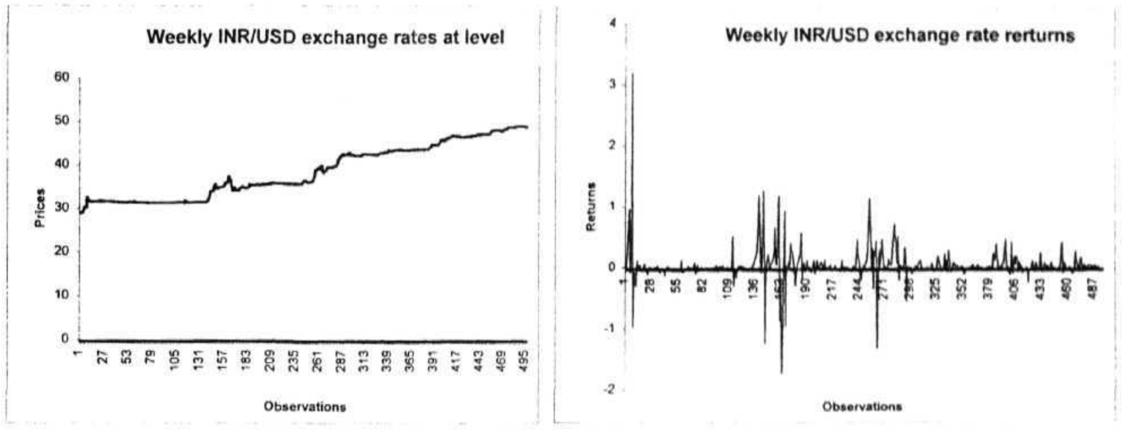


Figure 3.8 Quantile-Quantile (QQ)-plots of weekly exchange rates at level and returns against normal distribution

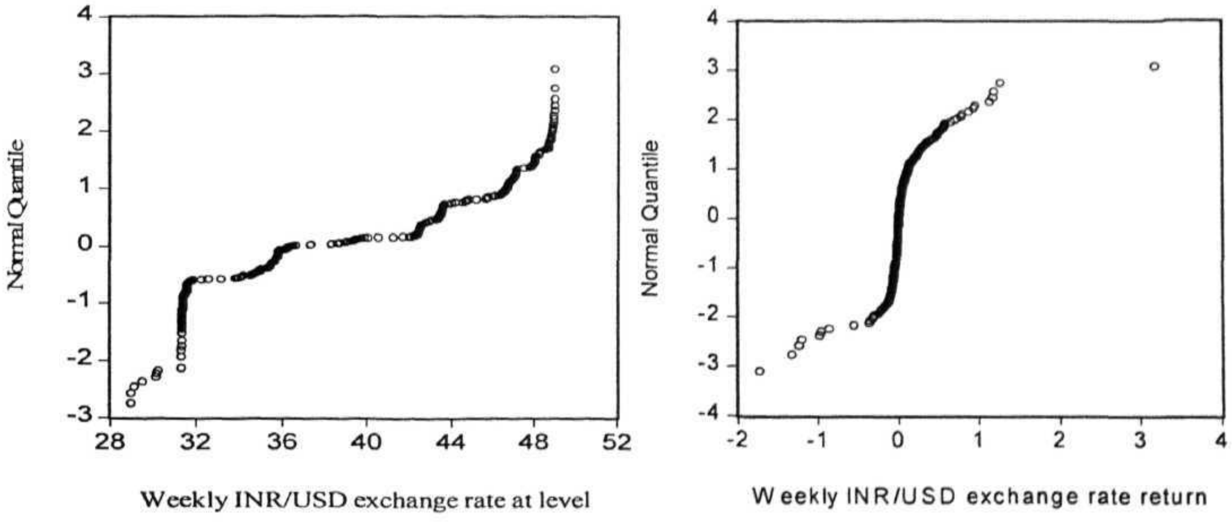


Figure 3.9 Plots of in-sample errors of ANN, LAR, and RW in predicting weekly exchange rate returns for the period May 19, 1993-September 15, 1999

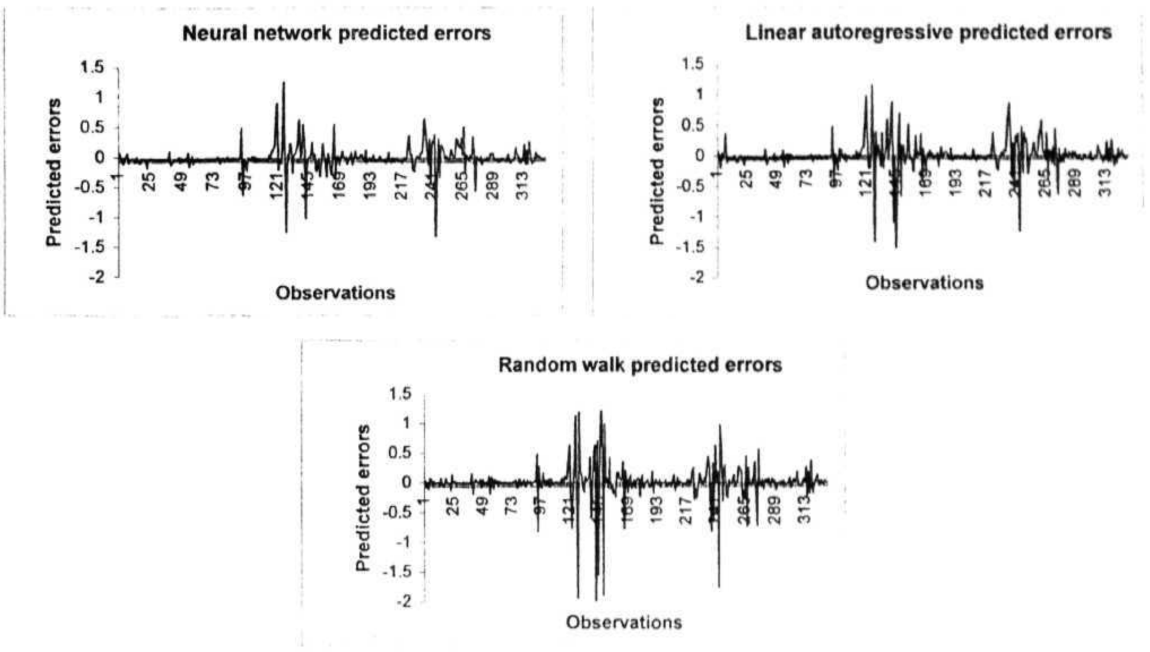


Figure **3.10** Plots of out-of-sample errors of ANN, LAR, and RW in predicting weekly exchange rate returns for the period February 2, **2000-July 10,** 2002

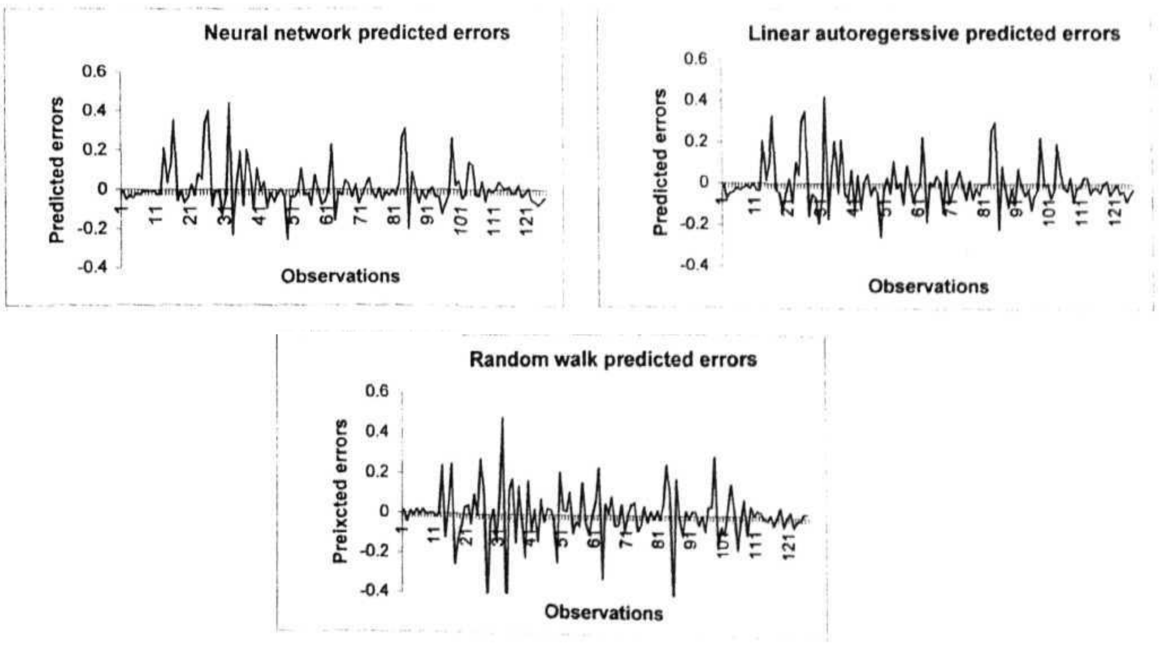


Figure 3.11 Plots of in-sample errors of ANN, LAR, and RW in predicting weekly exchange rates at level for the period May 19, 1993-September 15, 1999

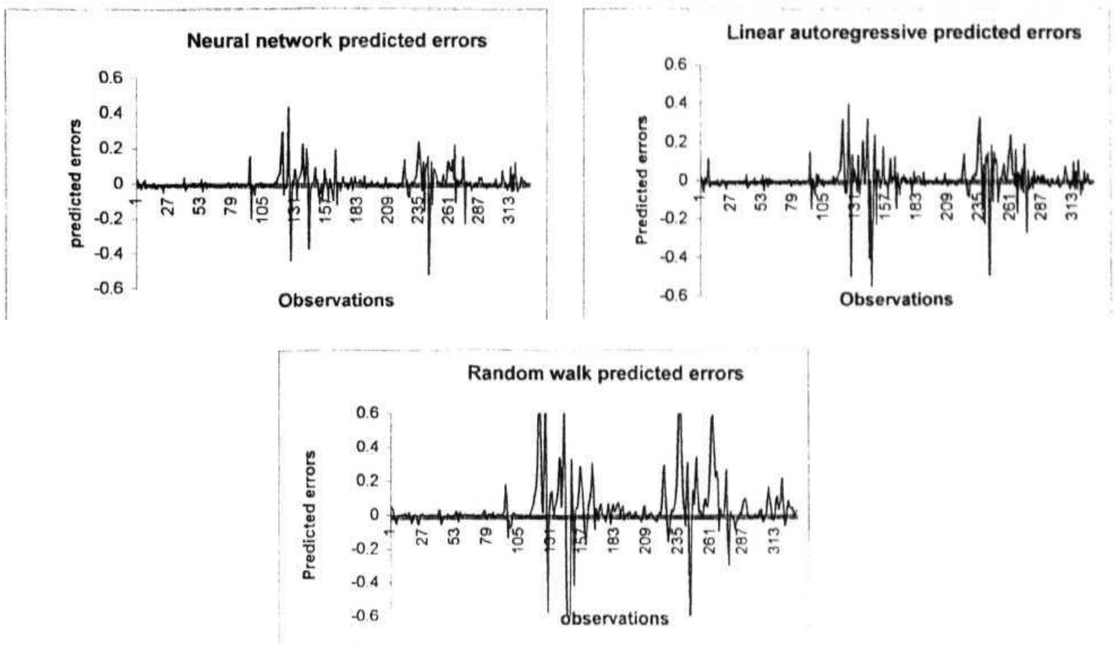
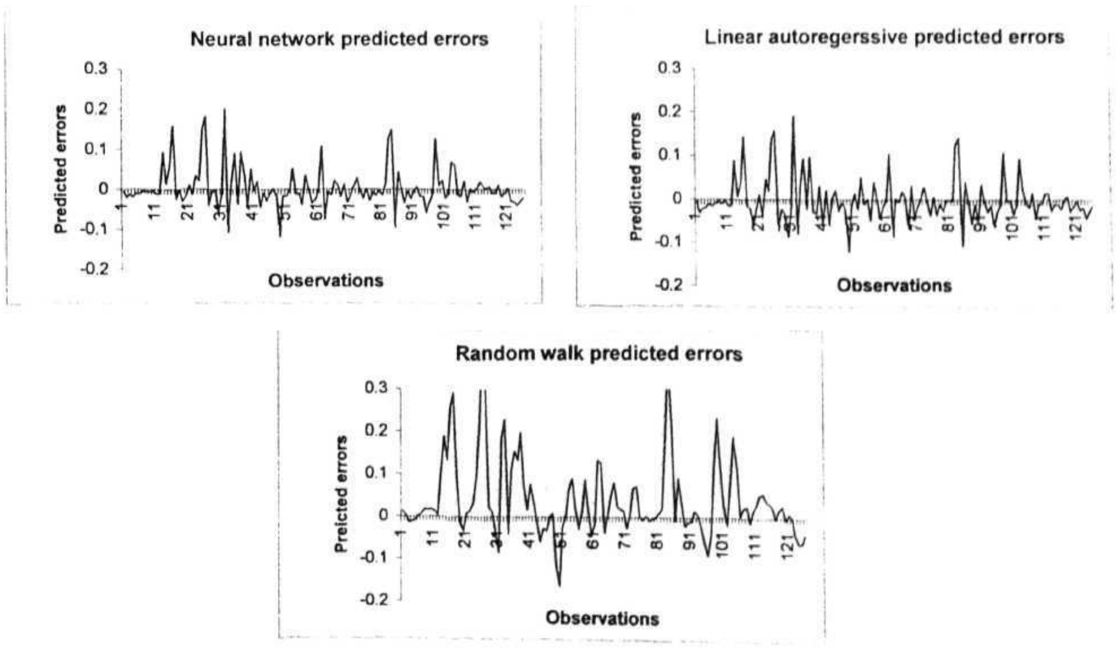


Figure 3.12 Plots of out-of-sample errors of ANN, LAR, and RW in predicting weekly exchange rates at level for the period February 2, 2000-July 10, 2002



Chapter IV

Forecasting Daily and Weekly Stock Returns

An unsophisticated forecaster uses statistics as a drunken man uses lamp-posts -for support rather than for illumination

4.0 Introduction

In the previous chapter, we made an attempt to predict daily and weekly exchange rate returns using neural network and compared its efficiency with benchmark models such as linear autoregressive and random walk models. We found neural network as giving a good account of itself in forecasting daily and weekly exchange rate returns compared to linear autoregressive and random walk models. This excited our interest to see how it performs on stock market data. This is particularly so because the literature suggests that there has been limited success of neural network in stock market prediction. Thus, the present chapter brings into play neural network to forecast daily and weekly stock returns and compares its efficiency with linear autoregressive and random walk models.

The remainder of the chapter is organized as follows. Sections 4.1 and 4.2 present empirical results of forecasting daily and weekly stock returns respectively by using neural network, linear autoregressive and random walk models. The statistical significance of in-sample and out-of-sample results of neural network, linear autoregressive and random walk models is studied in section 4.3, by using forecast encompassing test. Finally, section 4.4 concludes the chapter.

4.1 Forecasting Daily Stock Returns

In this section, we evaluate and compare the ability of neural network, linear autoregressive and random walk models in one-step-ahead forecasting of daily stock returns. Seven performance measures, root mean square error (RMSE), mean absolute error (MAE), mean absolute percentage error (MAPE), median absolute deviation (MAD), Pearson correlation coefficient (CORR), direction accuracy (DA), and percentage of correct signs predicted (SIGN) are used to assess the forecasting ability of three studied models. In subsection 4.1.1, we present the data description of daily stock returns. Subsection 4.1.2 provides model specifications for neural network, linear autoregressive and random walk models. In subsection 4.1.3, we furnish empirical findings, which include both in-sample and out-of-sample forecasting of daily stock returns by ANN, LAR, and RW.

4.1.1 Data Description

The data, consisting of daily closing values of BSE Sensitive Index, are collected from web pages of Bombay Stock Exchange (BSE). There are a total of 2,553 observations from January 2, 1991 to December 31, 2001. The daily returns are calculated by taking the logarithmic differences between successive trading days. The daily BSE 30 stock prices and returns are shown in Figure 4.1. In Figure 4.2, we show Quantilc-Quantile plots of both daily stock price and return against the normal distribution. If the plot lies on a straight line then it can be said that the series follows a normal distribution. The two plots in Figure 4.2 indicate that the stock return has a distribution closer to the normal distribution than the stock price. Table 4.1 provides summary statistics of daily stock returns data. The results in the table show that the values of skewness and kurtosis are high which are equal to 0.1045 and 6.5106 respectively. These values indicate that the series is not normally distributed. The series is positively skewed and heavy tailed i.e. leptokurtic. Jarque-Bera test of normality also confirms the non-normality of the distribution of series. Five autocorrelations reported in the table show linear dependence in the data. The Ljung-Box (LB) statistic for the first 20 lags is 61.98, which rejects the hypothesis of identical and independent observations. The stationarity of return series is confirmed by the value of ADF, which is equal to -21.839. Absolute value of the ADF statistic is found to be greater than the MacKinnon critical values at 1%, 5%, and 10% significance level. The value for Hurst exponent, which is equal to 0.5619, provides the evidence of long memory effect in the daily stock return series.

4.1.2 Model Specification

As we have previously discussed in chapter III, the model specification is very crucial for any forecasting problem. The appropriate model for neural network, linear autoregressive and random walk has been chosen in the same manner as we have done in case of exchange rate returns prediction in chapter III. First, we start with neural network's model specification. Out of total 2552 observations, we keep 2100 and 452 observations for training and testing respectively. Data has been normalized to the value between 0 and 1. A single hidden layer feedforward network is used to train the network. The sigmoid transfer function is used for hidden units in the hidden layer and the linear transfer function is used for output unit in the output layer. The weights in the network are initialized to small values based on the technique of Nguyen and Widrow (1990). Mean square error is chosen as the cost function and resilient backpropagation is taken as the training algorithm to train the network.

The relevant inputs and hidden units in the network have been chosen after experimenting with different combinations. The neural network's performances on the training set with varying combination of inputs and hidden units are presented in Table 4.2. A maximum of twenty levels of input nodes ranging from 1 to 20 lagged values of the dependent variable, i.e. daily stock return, are experimented with. The relevant inputs, which contribute to explain the behaviour of daily stock returns, are then chosen and used as explanatory variables to forecast daily stock returns. So far as the hidden units are concerned, we experiment with five levels of hidden units namely 4, 8, 12, 16, and 20 keeping in view the earlier empirical findings (for example see Tang and Fishwick (1993), Zhang and Hu (1998) and Hu *et al* (1999)) and our previous findings as well (see section 3.2.2 and section 3.3.2 in chapter III) that neural network's model fitting performance is not sensitive to the number of hidden nodes. Thus, the combination of twenty levels of input nodes and five levels of hidden nodes, yielding a total of 100 network architectures, are experimented with. We also train each of the hundred architectures ten times to avoid the possibility of getting stuck in local minima.

The RMSEs and MAEs, in Table 4.2, at each level of input in combination with each of the five levels of hidden nodes are the averages of ten runs. The results presented in Table 4.2 show that neural network's performance improves with every addition of inputs upto the input node level 8. This is reflected by the declining average RMSEs at each level of input upto 8 across five hidden node levels. But when the input node 9 is added, the average RMSE increases from its previous input level. Hence the input node 9 is considered as redundant as it does not contribute anything more to explain the behaviour of the dependent variable. Similarly, we find input levels 11, 16, 17, and 18 as unnecessary and hence drop them from the set of explanatory variables to forecast daily stock returns. In so far as the effect of hidden node is concerned, we do not find any clear hidden node effect. However, we find that the RMSE decreases as the number of hidden nodes increases at each level of input except for input levels 8, 9, 10, 11, 12, 13, 14, 16, 18, and 19. Consistent with our earlier findings in section 3.2.2 and section 3.3.2 in chapter III, we also find that neural network's performance is more sensitive to the number of inputs than the number of hidden nodes. This conclusion is based on the ground of less variation of RMSE among different hidden node levels within each input level than among different input node levels.

We find a different pattern for the effects of the input node and the hidden node with MAE. The average MAE decreases as the number of input nodes increases except for the levels of input nodes 8, 11, 16, 18, 19, and 20. Out of 20 input nodes, only at seven levels of input nodes i.e. 1, 2, 3, 5, 6, 7, and 17, the MAE falls as the number of hidden nodes increases. At other input node levels, the MAE does not show a clear hidden node effect. However, we select the optimum input nodes and hidden nodes on the basis of the pattern of RMSE and not on MAE. This is because the purpose of the training is to minimize the RMSE and not MAE. As we do not find an unambiguous hidden node effect, we could not settle down at a specific number of hidden nodes. We therefore try with more than one hidden layer in the company of varying numbers of hidden nodes and find that one hidden layer in combination with only one hidden unit achieves the best result. Hence, the optimal neural network architecture 15-1-1 is used for the in-sample and out-of-sample forecasting of daily stock returns.

As far as the specification of linear autoregressive model is concerned, we first regress the dependent variable i.e. daily stock return y_t, on a large group of independent variables, which consists of twenty lagged values of the dependent variable i.e. from y_{t-1} to y_{t-20}. Then a small group of statistically significant variables are identified and used as explanatory variables to forecast daily stock returns. The regression results of y_t on its own 20 lagged values are presented in Table 4.3. The table shows that out of 20 lags, only lags 1 and 9 are statistically significant. This is confirmed by the respective t-values and p-values. Hence, these two lags are considered as inputs or explanatory variables for the in-sample and out-of-sample forecasting of daily stock returns. For random walk, today's value is simply taken as the best predicting value for tomorrow.

4.1.3 Empirical Findings

Having discussed the model specification, we now move on to discuss the in-sample and out-of-sample forecasting of daily stock returns by neural network, linear autoregressive and random walk models.

In-sample Forecasting

The performances of neural network, linear autoregressive and random walk models in in-sample forecasting of daily stock returns are shown in Table 4.4. The performances of three alternative models are compared in terms of the seven performance measures, RMSE, MAE, MAPE, MAD, CORR, DA, and SIGN. We could not carry out MAPE because the values of some of the actual returns are found to be zero. The table shows that neural network gives better in-sample forecasts than both linear autoregressive and random walk models by all six performance measures. For example, neural network has smaller RMSE, MAE, and MAD, which are equal to 0.8101, 0.5932 and 0.4469, as compared with the RMSE, MAE, and MAD of linear autoregressive which are equal to 0.8164, 0.5979 and 0.4496 respectively. The ANN predicted values also have higher correlation with the actual series than the LAR. The direction accuracy and sign predictions, which are equal to 0.7153 and 0.5658 respectively for ANN, are also higher than the corresponding values of LAR. Similarly, neural network beats random walk model by all six performance measures in in-sample forecasting of daily stock returns. The superior performance of neural network over both linear autoregressive and random walk models in all six benchmarks shows that neural network does a better job in predicting stock returns than both LAR and RW models. Between LAR and RW model, it is also found that LAR outshines random walk model by five out of six performance measures.

Figure 4.3 plots the in-sample errors of neural network, linear autoregressive and random walk models in predicting daily stock returns. The figure suggests that neural network predicted errors are smaller than the predicted errors of linear autoregressive and random walk models. This also can be seen through the variance of errors of each model. The variance of error for ANN is 0.6567, which is smaller than the variances of LAR and RW predicted errors, which are equal to 0.6668 and 1.2492 respectively. From this, we can conclude that neural network has better in-sample fit than linear autoregressive and random walk models.

Out-of-sample Forecasting

Table 4.5 compares out-of-sample performances of neural network, linear autoregressive and random walk models in predicting daily stock returns. The results show that by all performance measures the out-of-sample results of ANN tend to be better than the in-sample results. However, in case of LAR, we find that the out-of-sample results are worse than the in-sample results by all performance measures. This observation confirms the view that ANN can generalize better than the linear models. Moreover, the improved performances of ANN in out-of-sample forecasting are overwhelming in case of two performance measures such as CORR and SIGN. The correlation coefficient for ANN in in-sample forecasting is 0.1664 whereas it is much higher, 0.6736 in case of out-of-sample forecasting. Similarly, ANN also gives higher out-of-sample sign predictions, which is equal to 0.7598 than the in-sample. These findings reinforce the view that ANN can generalize well and is more robust in nonstationary environment.

The out-of-sample results presented in Table 4.5 also convey that neural network outperforms both linear autoregressive and random walk models by all six performance measures. The neural network has smaller RMSE, MAE, and MAD than both linear autoregressive and random walk models. The values of RMSE MAE, and MAD for ANN are 0.7432, 0.5366, and 0.3885 respectively. The corresponding figures for LAR are 0.8529, 0.6317, and 0.4649 respectively and for RW these values are 1.0666, 0.7713, and 0.5667 respectively. The neural network out-of-sample fitted values have higher correlation (0.6736) with the actual values than both LAR (-0.1839) and RW (0.1615). The direction accuracy, which is equal to 0.7182, for ANN is higher than the corresponding figure of LAR (0.6882). The direction accuracy for random walk is zero. Neural network also gives better sign predictions than both linear autoregressive and random walk models. However, between LAR and RW, LAR outclasses RW by four out of six performance measures. LAR is better in out-of-sample forecasting than RW when RMSE, MAE, MAD and DA are considered as performance measures. On the other hand, RW gives better out-of-sample prediction than LAR in terms of CORR and SIGN.

Plots of out-of-sample errors of each model, shown in Figure 4.4, demonstrates that fluctuations around horizontal line in neural network predicted errors are smaller than the fluctuations in linear autoregressive and random walk predicted errors. The error variance for ANN, which is equal to 0.5487, is smaller than the error variances of LAR and RW, which are equal to 0.7257 and 1.1404 respectively. These findings in turn suggest that neural network has better out-of-sample fit than linear autoregressive and random walk models in the forecasting of daily stock returns.

We can sum up the above findings as follows. Neural network has better generalizing capability than linear autoregressive model. Neural network is also found to do better in out-of-sample forecasting than both linear autoregressive and random walk models. LAR outshines RW by four out of six performance measures in out-of-sample forecasting. These findings in turn confirm that stock market does not follow a random walk and there always exists a possibility to predict stock returns.

Forecast Horizon Effect

Here, we present both in-sample and out-of-sample performances of ANN, LAR, and RW models in predicting daily stock returns under different forecast horizons. We have taken four different forecast horizons, namely, 1 month, 3 months, 6 months, and 12 months to facilitate the comparison of the performances of ANN, LAR, and RW. Root mean square error and sign prediction are considered as performance measures for the comparison of the three alternative models in four forecast horizons. The results are presented in Table 4.6 and 4.7.

Table 4.6 presents in-sample RMSEs and SIGNs of ANN, LAR, and RW under the four forecast horizons. The results in Table 4.8 suggest that irrespective of forecast horizons, ANN performs better than LAR and RW in terms of RMSE in in-sample forecasting of daily stock returns. For example, in 1 month forecast horizon, ANN has smaller RMSE which is equal to 1.0699 than the corresponding figures of LAR and RW which are equal to 1.1247 and 2.0729 respectively. Similarly, under all other forecast horizons also, ANN takes on smaller RMSE value than both LAR and RW. LAR is also found to have smaller RMSEs than RW under all four forecast horizons. The table also shows that the in-sample performances of ANN, LAR, and RW improve as forecast horizon extends. For 1 month forecasts, ANN has a RMSE value of 1.0699. It falls to 0.7690 and 0.7671 respectively when the forecast horizon increases to 3 months and 6 months. The only exception occurs at 12 months forecast horizon when the RMSE again increases to 0.7792. Even so it can be concluded that ANN performs better in long forecast horizon than short forecast horizon when RMSE is considered as performance measure. The similar pattern is observed in case of LAR. However, RW has a consistent pattern which shows that its performance increases as the forecasts increases as shown by the falling value of RMSE throughout the forecast horizon.

As far as the effect of forecast horizon on the performances of ANN, LAR, and RW, in terms of sign prediction, is concerned, all three models are found to give better sign predictions in long forecast horizon than short forecast horizon. Neural network outperforms both linear autoregressive and random walk models under each of the four forecast horizons in in-sample forecasting of daily stock returns.

The results in Table 4.7 show daily out-of-sample RMSEs and SIGNs of ANN, LAR, and RW under 1 month, 3 months, 6 months, and 12 months forecast horizon. The Table 4.7 indicates that ANN has got a consistent pattern of falling RMSEs when the forecast horizon extends from 1 month to 12 months. For 1 month forecast, ANN has an RMSE, which is equal to 1.5021. The value of RMSE then falls to 1.0701, 0.9244, and 0.8254 respectively when the horizon increases to 3, 6, and 12 months. This finding, in turn, conveys that ANN performs better in long run than short run in terms of RMSE. Similar patterns are observed in case of both LAR and RW. The Table also conveys that ANN has superior out-of-sample performance in predicting daily stock returns than LAR and RW under all forecast horizons when RMSE is taken as the evaluation criteria. For example, neural network takes on a RMSE value of 1.5021, a much smaller value as compared to the corresponding values of linear autoregressive and random walk, which are equal to 1.6488 and 1.6927 respectively in 1 month, forecast horizon. Similarly, in other forecast horizons ANN has much smaller values of RMSE than LAR and RW. LAR also is found to do better than RW in all forecast horizons. To sum up, both neural network and linear autoregressive prevail over random walk model, in terms of RMSE, in out-of-sample forecasting under different forecast horizons. This finding further strengthens the view against efficient market hypothesis (EMH) in the capital market.

Table 4.7 also shows that neural network's performance becomes worse, in terms of correct sign prediction, as the forecast horizon increases. To illustrate, ANN gives as high as about 89% correct sign prediction in 1 month forecast horizon. The percentage of correct sign prediction falls to 80, 76, and 75 when the forecast horizon extends to 3 months, 6 months and 12 months respectively. We find the similar pattern of better performance in short run than long run for random walk model and find no consistent pattern for LAR. The results also show that ANN gives better out-of-sample sign predictions than LAR and RW in all forecast horizons.

Forecasting Daily Stock Price

Having discussed daily stock returns we now turn to see the performances of ANN, LAR, and RW models in predicting daily stock prices. The predicted values of stock price are obtained by taking exponential values of corresponding predicted stock returns. Table 4.8 presents the results for in-sample forecasting of daily stock price by neural network, linear autoregressive and random walk models. Seven performance measures, RMSE, MAE, MAPE, MAD, CORR, R^2, and DA are used to compare the performances of the three alternative forecasting techniques. The results presented in Table 4.8 show that ANN outperforms LAR by four out of seven performance measures. Neural network is found to have better in-sample forecasting ability than LAR in terms of RMSE, MAE, MAD, and DA. The performances of the two models are equal when the other three measures, namely MAPE, CORR, and R^2 are considered. However, neural network completely outweighs random walk model by all seven performance measures in in-sample forecasting of daily stock price. Similarly, LAR gives better in-sample forecasting than RW by all seven performance measures. The plots of in-sample errors of ANN, LAR, and RW, given in Figure 4.5, look similar. By looking at the plots nothing clear about the predicted error of each model can be said. This is consistent with the high and very close values of R^2 of each model (see Table 4.8).

The results for out-of-sample forecasting are presented in Table 4.7. Unlike the improved out-of-sample performance of ANN over in-sample in predicting daily stock returns, in case of stock price prediction the out-of-sample results tend to be worse than in-sample by almost all performance measures. Similarly for LAR and RW models, the out-of-sample results are worse than in-sample by all performance measures. However, by all seven measures neural network out-of-sample forecasts are better than both linear autoregressive and random walk forecasts. The neural network has smaller RMSE, MAE, MAPE, and MAD than linear autoregressive and random walk models. The values of RMSE, MAE, MAPE, and MAD for ANN are 73.22, 51.78, 0.0132, and 36.65 respectively. The corresponding values for LAR are 77.63, 55.61, 0.0142, and 40.24 respectively and for random walk, the values are 76.66, 54.71, 0.0140, and 39.33

respectively. The neural network has higher correlation coefficient and direction accuracy than both LAR and RW models. The R^2 in neural network, which is equal to 0.9843, is also higher than the R^2 in LAR (0.9823) and RW (0.9827) models. However, it is also found that random walk model outperforms LAR by six out of seven performance measures. Random walk forecasts are better than linear autoregressive forecasts in terms of RMSE, MAE, MAD, MAPE, CORR, and R^2. On the other hand LAR gives better out-of-sample forecasting in terms of only one performance measure i.e. DA. The direction accuracy for LAR is 0.4110 as compared with the value of RW, which is equal to 0. The plots of out-of-sample errors of each model, which is shown in Figure 4.6, however, look similar. Hence, an ordinary look at the plots does not enable us reach at a definite conclusion about the predicted errors of each model.

4.2 Forecasting Weekly Stock Returns

The previous section compared the performances of neural network in daily stock returns prediction with linear autoregressive and random walk models. It also presented stock price prediction in brief. The present section presents weekly stock return prediction as well as weekly stock price prediction by neural network, linear autoregressive and random walk models.

4.2.1 Data Description

The weekly data, consisting of closing values of BSE Sensitive Index, are collected from web pages of Bombay Stock Exchange (BSE). There are a total of 558 observations from January 3, 1992 to November 8, 2002. The weekly returns are calculated by taking the logarithmic differences between successive trading days. The weekly stock prices and returns are shown in Figure 4.7. The Quantile-Quantile plots are shown in Figure 4.8. The figure shows that weekly stock return is more close to normal distribution than weekly stock price. The summary statistics of weekly stock return series are presented in Table 4.10. The weekly return series is found to be volatile given the high value of its standard deviation, which is equal to 1.8395. The values of skewness and kurtosis describe weekly stock returns as skewed and non-normal. The non-normality is also confirmed by the Jarque-Bera statistic. The values of LB statistic and autocorrelation coefficients show that the weekly stock return series is not independent and is autocorrelated. The stationarity is guaranteed by the ADF statistic. The absolute value of the ADF statistic, which is equal to 9.954 is greater than the MacKinnon critical values, which are equal to -3.44, -2.86, and -2.56 at 1%, 5%, and 10% significance level respectively. The Hurst exponent, which is equal to 0.4806, describes weekly stock return as antipersistent or ergodic series with frequent reversals and high volatility.

4.2.2 Model Specification

We start with model selection for neural network followed by model selections for LAR and RW. The whole data set, which consists of 557 observations, has been divided into training set (in-sample data) and testing set (out-of-sample data). The training and testing set consist of 420 and 137 observations respectively. The whole data set has been normalized to the value between 0 and 1 before it is used in the training process of network. A single hidden layer feedforward is used in which hidden layer's transfer function is sigmoid. For the output unit in the output layer, the transfer function is taken as linear. The weights are initialized on the basis of Nguyen and Widrow (1990) technique and mean square error has been taken as the cost function to show the performance of neural network in the training process. Resilient backpropagation algorithm has been taken as the training algorithm keeping its advantage in view (see section 3.2.2 in chapter III).

The number of input nodes and hidden nodes has been chosen through systematic experimentation with different number of input and hidden nodes. Twenty levels of the number of input nodes ranging from 1 to 20 lagged values of the dependent variable i.e. weekly stock returns are experimented along with five levels of hidden nodes 4, 8, 12, 16, and 20.Thus the combination of 20 input nodes and 5 hidden nodes result in a total of 100 neural network architectures to be experimented with for training. We also train each of the 100 architectures 10 times by using 10 different sets of initial random weights to avoid the network getting stuck in local minima and to reach the true global minima. The effects of neural network factors such as input nodes and hidden nodes on the training set in terms of RMSE and MAE are shown in Table 4.11. The RMSEs and MAEs at each level of input in combination with each of the five levels of hidden nodes are the averages of ten runs. The results in Table 4.11 show that as the number of input nodes increases, average RMSE at each level of input across the five hidden node levels decreases. This falling pattern is observed consistently at all levels of input nodes. Thus, we get the smallest RMSE value, which is equal to 0.0752, at input node level 20. Hence the vector of inputs for the problem of forecasting weekly stock returns includes all the 20 lagged values of the dependent variable. As far as the effects of hidden nodes are concerned, the results show that in general, the RMSE falls as the number of hidden nodes increases. This pattern is observed at all levels of input nodes except for input node levels 11,12, 15 16, 17, and 19. However, we also observe that the neural network's model fitting performance is not that sensitive to the number of hidden nodes as to the number of input nodes. This is because network with the combination of more inputs and less hidden units gives much better performance than the network with less inputs and more hidden units. For example, network with 2 input nodes and just 8 hidden nodes gives much better performance in terms of RMSE, which is equal to 0.1032, than the network with 1 input node and as large as 20 hidden nodes where the RMSE is equal to 0.1040.

So far as the effects of input nodes and hidden nodes in terms of MAE are concerned, we observe a different pattern. The results show that when the number of input nodes is in the range of 1 to 12, the average MAE decreases from 0.0817 to 0.0604. When the number of input nodes is increased from 12 to 13, the average MAE increases, but falls as the number is increased thereafter. MAE decreases in general as the number of hidden nodes increases within each level of input node except for input node levels 11, 12, 15, 16, 17, and 20. However, we select the optimum architecture according to the patterns suggested by RMSE. As we find that there is no clear hidden node effect in terms of both RMSE and MAE and neural network's performance is not sensitive to the number of hidden nodes, we could not fix the optimum number of hidden nodes. After trying with several hidden nodes, we find that two hidden nodes in combination with 20 inputs achieve the best result. Hence the network configuration 20-2-1 is used for in-sample and out-of-sample forecasting of weekly stock returns.

The appropriate model for LAR is selected by regressing the dependent variable y_t (i.e. weekly stock return) on its own 20 lagged values i.e. from y_{t-1} to y_{t-20} and then selecting the significant lags as explanatory variables to predict y_t. . The regression results presented in Table 4.12 show that three lags i.e. y_{t-12},y_{t-13} and y_{t-18} are significant as suggested by their corresponding t-values and p-values. Hence y_{t-12}, y_{t-13}, and y_{t-18} are used as explanatory variables to predict weekly stock returns. For random walk model, current period's value has been taken as the best predicting value for tomorrow.

4.2.3 Empirical Findings

The empirical results for in-sample and out-sample forecasting of weekly stock returns by neural network, linear autoregressive and random walk models are presented here.

In-sample Forecasting

In-sample performances of neural network, linear autoregressive, and random walk models are presented in Table 4.13. The table shows that neural network has superior in-sample forecasting by all performance measures than both linear autoregressive and random walk models. ANN has smaller values of RMSE, MAE, and MAD which are equal to 1.5508, 1.2008, and 0.9590 respectively than the corresponding figures of LAR and RW. Neural network has also higher correlation coefficient than both linear autoregressive and random walk model. The values of CORR for ANN, LAR, and RW are 0.4274, 0.2413, and 0.0146 respectively. The direction accuracy for ANN, which is equal to 0.7775, is higher than that of LAR (0.7675). For random walk it is zero. In sign prediction also neural network has an edge over both linear autoregressive and random walk model. The value of SIGN for ANN is 0.5800 whereas this value is smaller i.e. 0.5550 for LAR and 0.5300 for RW. Between LAR and RW, LAR outperforms RW by all six performance measures. From the above findings, we sum up that both ANN and LAR surpass random walk in in-sample forecasting of weekly stock returns.

Figure 4.9, which plots the in-sample errors of ANN, LAR, and RW shows that neural network predicted errors, in general, are smaller in magnitude than linear autoregressive and random walk predicted errors. The figure also shows that the in-sample errors of ANN have smaller fluctuations than the in-sample errors of LAR and RW models. This is further confirmed by the values of error variance of each model. The error variance for ANN, which is equal to 2.4110, is less than the error variances of LAR and RW, which are equal to 2.7782 and 5.8140 respectively.

Out-of-sample Forecasting

The out-of-sample results presented in Table 4.14 show that neural network outshines random walk by five out of six performance measures. The neural network gives better out-of-sample forecasting than random walk in terms of RMSE, MAE, MAD, CORR, and DA. ANN has smaller values of RMSE, MAE, and MAD, which are equal to 1.7495, 1.2810, and 0.9187 respectively than the corresponding values of RW, which are equal to 2.2863, 1.6255, and 1.0990 respectively. ANN is found to have higher values of correlation coefficient and direction accuracy than RW. However, random walk performs better than neural network in out-of-sample forecasting when the percentage of correct signs predicted (SIGN) is considered as performance measure. RW has higher SIGN value i.e. 0.6068 than neural network in which the value is 0.4615.

If we compare the out-of-sample performances of ANN and LAR, we find that LAR is outperformed by ANN by all performance measures except for SIGN. The values of RMSE, MAE, and MAD for ANN are smaller than the corresponding values of LAR, which are equal to 1.8037, 1.2994, and 0.9693 respectively. Neural network fitted values have higher correlation with the actual values than linear autoregressive model in which the correlation coefficient is negative i.e.-0.1055. The direction accuracy for ANN, which is equal to 0.7350, is found to be higher than that of LAR (0.7299). However, LAR beats neural network in terms of SIGN in out-of-sample forecasting. So far as comparison between LAR and RW is concerned, LAR overtakes RW in out-of-sample forecasting by four out of six performance measures. It is found that LAR improves upon RW in terms of RMSE, MAE, MAD, and DA in out-of-sample forecasting of weekly stock returns. On the other hand, RW performs better than LAR when CORR and SIGN are considered as performance measures. The above empirical findings convey that both ANN and LAR outperform RW in out-of-sample forecasting of weekly stock returns. This further confirms the evidence against efficient market hypothesis (EMH).

By a mere look at the Figure 4.10, which shows the out-of-sample errors of ANN, LAR, and RW in the forecasting of weekly stock returns, it can be said that neural network predicted errors have smaller fluctuations than linear autoregressive and random walk predicted errors. In fact, the error variance for ANN is found to be 3.0296, which is smaller than the corresponding values of LAR and RW, which are found to be 3.2295 and 5.2726 respectively. From these findings, it can be concluded that neural network has better out-of-sample fit than linear autoregressive and random walk models in forecasting weekly stock returns.

Forecast Horizon Effect

The forecast horizon effect on the in-sample performances of ANN, LAR, and RW in predicting weekly stock returns is shown in Table 4.15. We have taken four different forecast horizons such as 1 month, 3 months, 6 months, and 12 months, under which the performances of ANN, LAR, and RW are evaluated. RMSE and SIGN are used as the performance criteria to facilitate the comparison. The results in Table 4.15 show that the neural network's in-sample forecast, in terms of RMSE, deteriorates as the forecast horizon increases. For example, in 1 month forecast horizon, the RMSE value is 0.3359. It increases to 1.0844 when the forecast horizon increases to 3 months. Again the RMSE increases to 1.5714 and 1.7652 respectively as the forecast horizon increases to 6 months and 12 months. From this we conclude that the longer the forecast horizon the worse the neural network's in-sample performance in terms of RMSE in predicting weekly stock returns. However, under all forecast horizons neural network outperforms both linear autoregressive and random walk model in in-sample forecasting when RMSE is considered as performance criteria.

The table also shows that neural network's in-sample performance gets worse as the forecast horizon increases when the sign prediction is considered as performance measure. The only exception occurs at 12 months forecast horizon where neural network gives slightly better sign prediction in comparison to 6 months forecast horizon. Neural network is found to outshine random walk models in all forecast horizons in in-sample forecasting of weekly stock returns. For example, in 1 month forecast horizon, ANN gives 100% correct sign prediction whereas RW gives only 25%. Similarly, in other forecast horizons also, ANN gives better percentage of correct sign prediction than RW. However, between ANN and LAR, ANN is found to outperform LAR when the forecast horizon is short, say for instance in 1 month and 3 months forecast horizon. When the forecast horizon further increases, performances of ANN and LAR are found to be equal. LAR also outperforms RW in all forecast horizons.

The out-of-sample RMSEs and sign predictions of ANN, LAR, and RW under different forecast horizons are presented in Table 4.16. Here, we do not find a very clear forecast horizon effect on neural network's out-of-sample performance in terms of RMSE. In 1 month forecast horizon, ANN takes on an RMSE value, which is equal to 2.1878. It increases to 2.6711 when the forecast horizon extends to 3 months and then starts falling until it reaches at 2.0448 in 12 months forecast horizon. However, we can see that ANN gives better out-of-sample forecasting in terms of RMSE in 6 months and 12 months forecast horizon than 1 month forecast horizon. From this finding, we conclude that neural network has better out-of-sample forecasting in longer forecast horizon rather than shorter forecast horizon. If we compare the performances of three studied models under four forecast horizons, we find that ANN outclasses RW in 6 months and 12 months forecast horizon and is being outperformed by RW in 1 month and 3 months forecast horizon. This finding suggests that neural network gives better out-of-sample forecasting of weekly stock returns, in terms of RMSE, than random walk in longer horizons than shorter horizons. Between ANN and LAR, LAR performs better than ANN in all forecast horizons. LAR is also found to have better out-of-sample forecasting than RW in all forecast horizons.

We do not find a clear forecast horizon effect on out-of-sample performances of ANN and LAR when sign prediction is considered as performance measure. For instance, ANN gives equal sign prediction, which is equal to 0.6000, when the forecast horizon increases from 1 month to 3 months. It falls to 0.5262 when the forecast horizon increases to 6 months. Further, ANN gives slightly better sign prediction, which is equal to 0.5283, when the forecast horizon extends from 6 months to 12 months. In spite of this finding, we can say that neural network performs better in terms of sign prediction in short run than in long run. Random walk is found to perform better in short forecast horizon than long forecast horizon. Random walk model outperforms neural network in all forecast horizons. ANN outperforms LAR in 1 month forecast horizon and is outperformed by LAR in 6 months forecast horizon in terms of sign prediction. The sign predictions arc equal for ANN and LAR in 3 months and 12 months forecast horizon. From this we can say that ANN gives better out-of-sample forecasting of weekly stock returns, in terms of sign prediction, than LAR in short forecast horizon than long forecast horizon.

Forecasting Weekly Stock Price

The predicted values of weekly stock price are obtained by taking exponential values of weekly predicted stock returns. Seven performance measures, RMSE, MAE, MAPE, MAD, CORR, R^2, and DA arc used to compare the in-sample and out-of-sample performances of ANN, LAR, and RW in predicting weekly stock prices. Table 4.17 presents in-sample performances of ANN, LAR, and RW. The table shows that ANN has better in-sample forecasting than RW by all seven performance measures. ANN is also found to outweigh LAR by all performance measures except for MAD in in-sample forecasting. Between LAR and RW, LAR outperforms RW by all performance measures. To sum up, neural network and linear autoregressive model are found to have better in-sample forecasting of weekly stock price than random walk. The plots of in-sample errors of ANN, LAR, and RW in Figure 4.11 look similar. Hence, we find it difficult to conclude anything about the predicted errors of each model by just looking at the figure.

Out-of-sample forecasting results are presented in Table 4.18. The results in the table are little bit surprising where random walk performs better than neural network by six out of seven performance measures. Random walk outshines neural network in out-of-sample forecasting in terms of RMSE, MAE, MAPE, MAD, CORR and R^2. ANN could only manage to give better out-of-sample forecasting in terms of DA. However, as far as the performances of ANN and LAR are concerned, the out-of-sample **results** are mixed. ANN outperforms LAR in terms of RMSE, CORR, and R^2 and is being outperformed by LAR when MAE, MAPE, MAD, and DA are considered as performance measures. Between LAR and RW, RW gives better out-of-sample forecasting by all performance measures except for DA in comparison to LAR. The plots of out-of-sample errors of each model in Figure 4.12 look alike. So, it is difficult to say anything from there.

From the above in-sample and out-of-sample analysis, we find that both ANN and LAR are found to achieve better results than RW in in-sample forecasting. However, their performances go down when it comes to out-of-sample forecasting. RW is found to thrash both neural network and linear autoregressive model by all performance measures except for DA.

4.3 Forecast Encompassing Test

The forecast encompassing test (sec the detailed discussion in section 3.4 of chapter III) results for both daily and weekly out-of-sample forecasts of stock returns are given in Table 4.19 and Table 4.20 respectively. The name of the dependent variable is listed down the left side of the table, while the independent variable is listed along the top. The entries in the Table 4.19 and Table 4.20 are estimated coefficients θ_l with the associated p-values in parenthesis. The daily out-of-sample forecast encompassing test shown in Table 4.19 shows that all the estimated coefficients are significant as indicated by their associated p-values. For example, the estimated coefficient from the regression of forecast error from ANN on the forecast error from LAR, which is equal to 0.8534, is significant as indicated by its associated p-values. Conversely, the estimated coefficient from the regression of forecast error from LAR on the forecast error from ANN is also significant. Hence, from this finding, we can say that neither ANN nor LAR encompasses each other i.e. neither of the model can explain part of **other's** forecast error. In Table 4.19, we find that all the coefficients are significant and hence we fail to reject the null hypothesis that none of ANN, LAR and RW model encompasses the other. Similarly, for weekly exchange rate returns, the out-of-sample encompassing test, which is shown in Table 4.20, reveals that all the estimated coefficients are significant. As a result, in this case also, we find that none of three studied models encompasses the other.

4.4 Conclusion

In this chapter, we have employed neural network to forecast daily and weekly stock returns and have compared its performance with linear autoregressive and random walk models. We find that neural network outperforms linear autoregressive and random walk models by all performance measures in both in-sample and out-of-sample forecasting of daily stock returns. Neural network is found to perform better in terms of RMSE in long forecast horizon than in short forecast horizon in forecasting daily stock returns. On the other hand, neural network is found to give better out-of-sample sign prediction in short forecast horizon than long forecast horizon. In case of stock price prediction, neural network performs better than linear autoregressive and random walk models.

As far as the forecasting of weekly stock return is concerned, neural network outshines both linear autoregressive and random walk models. We also find that the longer the forecast horizon the worse the neural network's in-sample performance in terms of RMSE in predicting weekly stock returns. However, neural network is found to have better out-of-sample forecasting in long horizon rather than short horizon. On the contrary, when sign prediction is considered as performance measure, neural network performs better in short forecast horizon than long forecast horizon. In weekly stock price series prediction, our results suggest that neural network is better than linear autoregressive and random walk models in in-sample forecasting. However, random walk is found to perform better than neural network and linear autoregressive model in out-of-sample forecasting of weekly stock price. The out-of-sample results are mixed in case of neural network and linear autoregressive model. Forecast encompassing test shows that no model encompasses other in both daily and weekly stock returns prediction.

Table 4.1 Summary statistics for the daily stock returns: log first difference January 2, 1991-December 31, 2001

Description	SENSEX
Sample size	2552
Mean	0.0201
Median	0.0204
SD	0.8295
Skewness	0.1045
Kurtosis	6.5106
Maximum	5.3598
Minimum	-4.4584
Jarque-Bera	1315.170
	(0.000)
ρ_1	0.097
ρ_5	-0.002
ρ_{10}	-0.006
ρ_{15}	0.034
ρ_{20}	-0.043
LB statistic (20)	61.98
	(0.000)
ADF	-21.839
Hurst exponent	0.5619

Note: The p-value for Jarque-Bera and LB statistic is given in parentheses. The MacKinnon critical values for ADF test are –3.43, -2.86, and –2.56 at 1%, 5%, and 10% significance level respectively.

Table 4.2 Effects of inputs and hidden units on the training performance of ANN in forecasting daily normalized stock returns

Input	Hidden	RMSE	MAE	Input	Hidden	RMSE	MAE
1	4	0.0821	0.0600	8	4	0.0763	0.0586
1	8	0.0818	0.0598	8	8	0.0763	0.0573
1	12	0.0814	0.0597	8	12	0.0774	0.0579
1	16	0.0811	0.0596	8	16	0.0757	0.0570
1	20	0.0807	0.0594	8	20	0.0755	0.0569
	Avg	0.0814	0.0597		Avg	0.0762	0.0575
2	4	0.0817	0.0600	9	4	0.0785	0.0586
2	8	0.0800	0.0592	9	8	0.0768	0.0578
2	12	0.0795	0.0589	9	12	0.0756	0.0567
2	16	0.0788	0.0583	9	16	0.0751	0.0566
2	20	0.0783	0.0581	9	20	0.0761	0.0572
	Avg	0.0796	0.0589		Avg	0.0764	0.0573
3	4	0.0812	0.0598	10	4	0.0789	0.0588
3	8	0.0796	0.0591	10	8	0.0757	0.0571
3	12	0.0786	0.0586	10	12	0.0748	0.0565
3	16	0.0778	0.0580	10	16	0.0738	0.0558
3	20	0.0769	0.0576	10	20	0.0739	0.0559
	Avg	0.0788	0.0586		Avg	0.0754	0.0568
4	4	0.0810	0.0595	11	4	0.0785	0.0585
4	8	0.0794	0.0588	11	8	0.0760	0.0574
4	12	0.0783	0.0583	11	12	0.0746	0.0565
4	16	0.0775	0.0579	11	16	0.0738	0.0559
4	20	0.0775	0.0580	11	20	0.0745	0.0562
	Avg	0.0787	0.0585			0.0754	0.0569
5	4	0.0807	0.0595	12	4	0.0783	0.0584
5	8	0.0784	0.0586	12	8	0.0755	0.0568
5	12	0.0781	0.0582	12	12	0.0741	0.0558
5	16	0.0771	0.0580	12	16	0.0741	0.0561
5	20	0.0767	0.0575	12	20	0.0730	0.0553
	Avg	0.0782	0.0583		Avg	0.0750	0.0564
6	4	0.0802	0.0593	13	4	0.0790	0.0588
6	8	0.0784	0.0586	13	8	0.0748	0.0562
6	12	0.0779	0.0581	13	12	0.0744	0.0561
6	16	0.0767	0.0574	13	16	0.0732	0.0553
6	20	0.0765	0.0573	13	20	0.0735	0.0555
	Avg	0.0779	0.0581		Avg	0.0749	0.0563
7	4	0.0796	0.0590	14	4	0.0778	0.0580
7	8	0.0776	0.0580	14	8	0.0746	0.0561
7	12	0.0763	0.0572	14	12	0.0734	0.0553
7	16	0.0762	0.0569	14	16	0.0731	0.0553
7	20	0.0758	0.0567	14	20	0.0736	0.0557
	Avg	0.0771	0.0575		Avg	0.0745	0.0560

continued on next page

continued from previous page

Input	**Hidden**	RMSE	MAE	Input	**Hidden**	RMSE	MAE
15	4	0.0772	0.0575	18	4	0.0776	0.0578
15	8	0.0740	0.0555	18	8	0.0737	0.0555
15	12	0.0730	0.0564	18	12	0.0741	0.0560
15	16	0.0725	0.0549	18	16	0.0722	0.0546
15	20	0.0719	0.0541	18	20	0.0728	0.0551
	Avg	0.0737	**0.0556**		Avg	0.0740	0.0558
16	4	0.0776	0.0572	19	4	0.0764	0.0572
16	8	0.0740	0.0555	19	8	0.0744	0.0563
16	12	0.0733	0.0552	19	12	0.0723	0.0546
16	16	0.0727	0.0551	19	16	0.0723	0.0548
16	20	0.0749	0.0554	19	20	0.0708	0.0536
	Avg	0.0745	0.0556		Avg	0.0732	0.0553
17	4	0.0774	0.0558	20	4	0.0768	0.0572
17	8	0.0738	0.0555	20	8	0.0738	0.0555
17	12	0.0728	0.0549	20	12	0.0728	0.0552
17	16	0.0725	0.0547	20	16	0.0717	0.0537
17	20	0.0720	0.0544	20	20	0.0708	0.0537
	Avg	0.0737	0.0550		Avg	0.0731	0.0550

Note: The RMSEs and MAEs at each level of input node in combination with each of the five hidden node levels arc the averages of ten runs.

Table 4.3 Results for the regression of daily stock return y_t on its own 20 lagged values

Variable	Coefficient	Std. Error	t-Statistic	Prob.
Constant	0.369418	0.038352	9.632182	0.0000
y_{t-1}	0.084417	0.021895	3.855554	0.0001
y_{t-2}	-0.002091	0.021975	-0.095174	0.9242
y_{t-3}	0.024959	0.021979	1.135591	0.2563
y_{t-4}	0.020510	0.021990	0.932704	0.3511
y_{t-5}	0.005267	0.022007	0.239356	0.8109
y_{t-6}	-0.025864	0.022041	-1.173467	0.2407
y_{t-7}	0.007352	0.022052	0.333414	0.7389
y_{t-8}	-0.034829	0.022070	-1.578131	0.1147
y_{t-9}	0.082704	0.022091	3.743855	0.0002
y_{t-10}	-0.027096	0.022197	-1.220707	0.2223
$y_{t 11}$	0.005003	0.022174	0.225635	0.8215
y_{t-12}	0.023577	0.022096	1.066995	0.2861
y_{t-13}	0.017641	0.022097	0.798332	0.4248
y_{t-14}	-0.021869	0.022090	-0.990004	0.3223
y_{t-15}	0.034730	0.022087	1.572382	0.1160
y_{t-16}	0.018484	0.022104	0.836208	0.4031
y_{t-17}	0.001235	0.022094	0.055896	0.9554
y_{t-18}	0.019862	0.022093	0.899003	0.3688
y_{t-19}	-0.021650	0.022098	-0.979733	0.3273
y_{t-20}	-0.019850	0.021988	-0.902733	0.3668

Table 4.4 In-sample performance of ANN, LAR, and RW models on dally stock return series for the period February 20, 1991-March 7, 2000

	ANN	LAR	RW
RMSE	0.8101	0.8164	1.1174
MAE	0.5932	0.5979	**0.7843**
MAPE	—	—	...
MAD	0.4469	0.4496	0.5723
CORR	0.1664	0.1127	0.0813
DA	0.7153	0.7086	0
SIGN	0.5658	0.5360	0.5572

Table 4.5 Out-of-sample performance of ANN, LAR, and RW models on daily stock return series for the period April 7, 2000-December 31, 2001

	ANN	LAR	RW
RMSE	0.7432	0.8529	1.0666
MAE	0.5366	**0.6317**	0.7713
MAPE	—	—	—
MAD	0.3885	0.4649	0.5667
CORR	0.6736	-0.1839	0.1615
DA	0.7182	0.6882	**0**
SIGN	0.7598	0.4133	0.5750

Table 4.6 In-sample performance of ANN, LAR, and RW models on daily stock return series in different forecast horizons

		1 month	3 months	6 months	12 months
RMSE	ANN	1.0699	0.7690	0.7671	0.7792
	LAR	1.1247	0.7934	0.7859	**0.7891**
	RW	2.0729	**1.3769**	1.2173	**1.1811**
SIGN	ANN	0.5000	0.5370	0.5575	0.5388
	LAR	0.3333	0.4818	**0.5132**	0.5205
	RW	0.3333	0.3888	0.4778	0.4885

Table 4.7 Out-of-sample performance of ANN, LAR, and RW models on daily stock return series in different forecast horizons

		1 month	3 months	6 months	12 months
RMSE	ANN	1.5021	1.0701	0.9244	0.8254
	LAR	1.6488	**1.2311**	**1.0567**	**0.9413**
	RW	1.6927	1.3953	**1.3087**	**1.1917**
SIGN	ANN	0.8947	0.8095	**0.7698**	0.7588
	LAR	0.3684	0.3492	**0.3650**	0.4166
	RW	0.7894	0.6349	**0.5634**	**0.5714**

Table 4.8 In-sample performance of ANN, LAR, and RW models on daily stock price series for the period February 20, 1991-March 7, 2000

	ANN	LAR	RW
RMSE	64.81	65.11	65.35
MAE	45.55	45.80	45.99
MAPE	0.0137	0.0137	0.0138
MAD	33.55	33.89	34.15
CORR	0.9973	0.9973	0.9972
R^2	0.9946	0.9946	0.9945
DA	0.5658	0.5360	0

Table 4.9 Out-of-sample performance of ANN, LAR, and RW models on daily stock price series for the period April 7, 2000-December 31, 2001

	ANN	LAR	RW
RMSE	73.22	77.63	76.66
MAE	51.78	55.61	54.71
MAPE	0.0132	0.0142	0.0140
MAD	36.65	40.24	39.33
CORR	0.9921	0.9912	0.9914
R^2	0.9843	0.9823	0.9827
DA	0.7575	0.4110	0

Table 4.10 Summary statistics for the weekly stock returns: log first difference January 3, 1992- November 8, 2002

Description	Sensex
Sample size	557
Mean	0.0306
Median	-0.0076
SD	1.8395
Skewness	0.2793
Kurtosis	5.2729
Maximum	9.9890
Minimum	-6.0352
Jarque-Bera	127.1430
	(0.000)
ρ_1	0.050
ρ_5	-0.021
ρ_{10}	0.006
ρ_{15}	-0.027
ρ_{20}	0.001
LB statistic (20)	22.03
	(0.338)
ADF	-9.954
Hurst exponent	0.4806

Note: The p-value for Jarque-Bera and LB statistic is given in parentheses. The MacKinnon critical values for ADF test are -3.44, -2.86, and -2.56 at 1%, 5%, and 10% significance level respectively.

Table 4.11 Effects of inputs and hidden units on the training performance of ANN in forecasting weekly normalized stock returns

Input	Hidden	RMSE	MAE	Input	Hidden	RMSE	MAE
1	4	0.1064	0.0829	8	4	0.0963	0.0751
1	8	0.1060	0.0825	8	8	0.0924	0.0726
1	12	0.1051	0.0814	8	12	0.0895	0.0698
1	16	0.1047	0.0812	8	16	0.0850	0.0661
1	20	0.1040	0.0807	8	20	0.0838	0.0650
	Avg	0.1052	0.0817		Avg	0.0894	0.0697
2	4	0.1050	0.0824	9	4	0.0980	0.0749
2	8	0.1032	0.0814	9	8	0.0916	0.0716
2	12	0.1027	0.0804	9	12	0.0867	0.0669
2	16	0.1025	0.0803	9	16	0.0834	0.0643
2	20	0.1015	0.0791	9	20	0.0820	0.0635
	Avg	0.1029	0.0807		Avg	0.0883	0.0682
3	4	0.1033	0.0807	10	4	0.0971	0.0752
3	8	0.1031	0.0805	10	8	0.0894	0.0690
3	12	0.1009	0.0789	10	12	0.0864	0.0664
3	16	0.1003	0.0779	10	16	0.0819	0.0633
3	20	0.0994	0.0767	10	20	0.0826	0.0618
	Avg	0.1014	0.0789		Avg	0.0874	0.0671
4	4	0.1022	0.0797	11	4	0.0937	0.0734
4	8	0.0996	0.0773	11	8	0.0843	0.0652
4	12	0.0982	0.0760	11	12	0.0825	0.0638
4	16	0.0973	0.0753	11	16	0.0797	0.0613
4	20	0.0956	0.0737	11	20	0.0803	0.0622
	Avg	0.0985	0.0764		Avg	**0.0841**	0.0651
5	4	0.0994	0.0777	12	4	0.0931	0.0727
5	8	0.0969	0.0761	12	8	0.0850	0.0659
5	12	0.0939	0.0735	12	12	0.0808	0.0618
5	16	0.0937	0.0732	12	16	0.0784	0.0595
5	20	0.0908	0.0705	12	20	0.0793	0.0604
	Avg	0.0949	0.0742		Avg	0.0833	0.0604
6	4	0.1001	0.0782	13	4	0.0922	0.0717
6	8	0.0955	0.0750	13	8	0.0838	0.0640
6	12	0.0927	0.0720	13	12	0.0814	0.0631
6	16	0.0884	0.0684	13	16	0.0798	0.0620
6	20	0.0870	0.0668	13	20	0.0776	0.0595
	Avg	0.0927	0.0720		**Avg**	0.0829	**0.0640**
7	4	0.0979	0.0768	14	4	0.0910	0.0708
7	8	0.0938	0.0727	14	8	0.0825	0.0639
7	12	0.0893	0.0694	14	12	0.0812	0.0630
7	16	0.0894	0.0693	14	16	0.0771	0.0594
7	20	0.0859	0.0669	14	20	0.0738	0.0562
	Avg	0.0912	0.0710		Avg	**0.0811**	**0.0626**

continued from previous page

Input	Hidden	RMSE	MAE	Input	Hidden	RMSE	MAE
15	4	0.0903	0.0701	18	4	0.0900	0.0694
15	8	0.0822	0.0626	18	8	0.0794	0.0603
15	12	0.0761	0.0582	18	12	0.0758	0.0580
15	16	0.0781	0.0602	18	16	0.0721	0.0551
15	20	0.0719	0.0545	18	20	0.0685	0.0531
	Avg	**0.0797**	**0.0611**		**Avg**	**0.0771**	**0.0591**
16	4	0.0916	0.0701	19	4	0.0893	0.0688
16	8	0.0817	0.0627	19	8	0.0761	0.0575
16	12	0.0773	0.0604	19	12	0.0716	0.0568
16	16	0.0715	0.0550	19	16	0.0718	0.0550
16	20	0.0748	0.0568	19	20	0.0709	0.0542
	Avg	0.0793	**0.0610**		**Avg**	0.0759	**0.0584**
17	4	0.0921	0.0714	20	4	0.0867	0.0673
17	8	0.0802	0.0607	20	8	0.0773	0.0599
17	12	0.0726	0.0568	20	12	0.0734	0.0566
17	16	0.0733	0.0560	20	16	0.0699	0.0540
17	20	0.0735	0.0560	20	20	0.0691	0.0540
	Avg	**0.0783**	**0.0601**		**Avg**	**0.0752**	**0.0583**

Note: The RMSEs and MAEs at each level of input node in combination with each of the five hidden node levels are the averages of ten runs.

Table 4.12 Results for the regression of weekly stock return y_t on Its own 20 lagged values

Variable	Coefficient	Std. Error	t-Statistic	Prob.
Constant	0.041387	0.085402	0.484618	0.6282
y_{t-1}	0.000354	0.051339	0.006887	0.9945
y_{t-2}	-0.002245	0.051193	-0.043860	0.9650
y_{t-3}	0.033739	0.050287	0.670920	0.5027
y_{t-4}	0.020287	0.049670	0.408429	0.6832
y_{t-5}	-0.024256	0.049519	-0.489838	0.6245
y_{t-6}	0.027533	0.049523	0.555968	0.5786
y_{t-7}	0.031256	0.050007	0.625041	0.5323
y_{t-8}	0.012757	0.049144	0.259581	0.7953
y_{t-9}	0.031524	0.047987	0.656925	0.5116
y_{t-10}	0.020387	0.048002	0.424712	0.6713
y_{t-11}	0.000534	0.047792	0.011168	0.9911
yt-n	-0.147088	0.046451	-3.166540	0.0017
y_{t-13}	-0.111751	0.046656	-2.395219	0.0171
y_{t-14}	-0.022430	0.046901	-0.478229	0.6328
y_{t-15}	0.032864	0.046968	0.699712	0.4845
y_{t-16}	0.020163	0.046488	0.433710	0.6647
y_{t-17}	-0.029180	0.046290	-0.630363	0.5288
y_{t-18}	0.156778	0.046228	3.391372	0.0008
y_{t-19}	-0.012007	0.046803	-0.256537	0.7977
y_{t-20}	0.023435	0.046788	0.500880	0.6167

Table 4.13 In-sample performance of ANN, LAR, and RW models on weekly stock return series for the period May 29, 1992-March 24, 2000

	ANN	LAR	RW
RMSE	1.5508	1.6647	2.4082
MAE	1.2008	1.2888	1.8731
MAPE	---	—	—
MAD	0.9590	1.0203	1.5126
CORR	0.4274	0.2413	0.0146
DA	0.7775	0.7675	0
SIGN	0.5800	0.5550	0.5300

Table 4.14 Out-of-sample performance of ANN, LAR, and RW models on weekly stock return series for the period August 18, 2000-November 8, 2002

	ANN	LAR	RW
RMSE	1.7495	1.8037	2.2863
MAE	1.2810	1.2994	1.6255
MAPE	—	—	—
MAD	0.9187	0.9693	1.0990
CORR	0.0388	-0.1055	- 0.0320
DA	0.7350	0.7299	0
SIGN	0.4615	0.5401	0.6068

Table 4.15 In-sample performance of ANN, LAR, and RW models on weekly stock return series in different forecast horizons

		1 month	3 months	6 months	12 months
RMSE	ANN	0.3359	1.0844	1.5714	**1.7652**
	LAR	2.2467	2.1200	2.0722	2.1029
	RW	4.7625	3.5142	3.2526	3.3542
SIGN	ANN	1.000	0.9285	0.6551	0.6600
	LAR	0.7500	0.7857	0.6551	0.6000
	RW	0.2500	0.4285	0.4827	0.4800

Table 4.16 Out-of-sample performance of ANN, LAR, and RW models on weekly stock return series in different forecast horizons

		1 month	3 months	6 months	12 months
RMSE	ANN	2.1878	2.6711	2.1790	2.0448
	LAR	1.4574	2.1660	1.9110	1.9851
	RW	1.5804	2.6424	2.3739	2.7591
SIGN	ANN	0.6000	0.6000	0.5262	0.5283
	LAR	0.4000	0.6000	0.5862	0.5283
	RW	0.8000	**0.6666**	0.6206	0.6037

Table 4.17 **In-sample** performance of **ANN,** LAR, and RW models on weekly stock price series **for** the period May **29, 1992-March** 24, 2000

	ANN	LAR	RW
RMSE	129.95	135.05	137.76
MAE	100.83	104.51	106.68
MAPE	0.0987	0.1026	0.1048
MAD	82.41	82.21	84.89
CORR	0.9826	0.9812	0.9805
R^2	0.9655	0.9627	0.9612
DA	0.5775	0.5525	0

Table **4.18 Out-of-sample** performance of ANN, LAR, and RW models on weekly stock price series for the period August 18, 2000-November 8, 2002

	ANN	LAR	RW
RMSE	132.28	132.39	129.68
MAE	96.01	94.99	92.80
MAPE	0.0275	0.0272	0.0266
MAD	74.27	71.13	**69.93**
CORR	0.9551	0.9546	0.9566
R^2	0.9097	0.9092	0.9128
DA	0.4529	0.5470	0

Table 4.19 Daily out-of-sample forecast encompassing coefficient θ_1 (p-values in parenthesis) from $E_{jt} = \theta_0 + \theta_1 F_{kt} + \eta_t$

Forecast error E_{jt} from ↓	Forecast F_{kt} from ↓		
	ANN	LAR	RW
ANN	-----	0.8534 (0.000)	0.4414 (0.000)
LAR	0.9973 (0.000)	------	0.5042 (0.000)
RW	0.9174 (0.000)	0.7923 (0.000)	-----

Table 4.20 Weekly out-of-sample forecast encompassing coefficient θ_1 (p-values in parenthesis) from $E_{jt} = \theta_0 + \theta_1 F_{kt} + \eta_t$

Forecast error E_{jt} from ↓	Forecast F_{kt} from ↓		
	ANN	LAR	RW
ANN	-----	0.9509 (0.000)	0.4401 (0.000)
LAR	0.8938 (0.000)	------	0.5121 (0.000)
RW	0.7660 (0.000)	0.9482 (0.000)	-----

Figure 4. 1 Plots of daily BSE 30 stock prices and returns for the period January 2, 1991-December 31, 2001

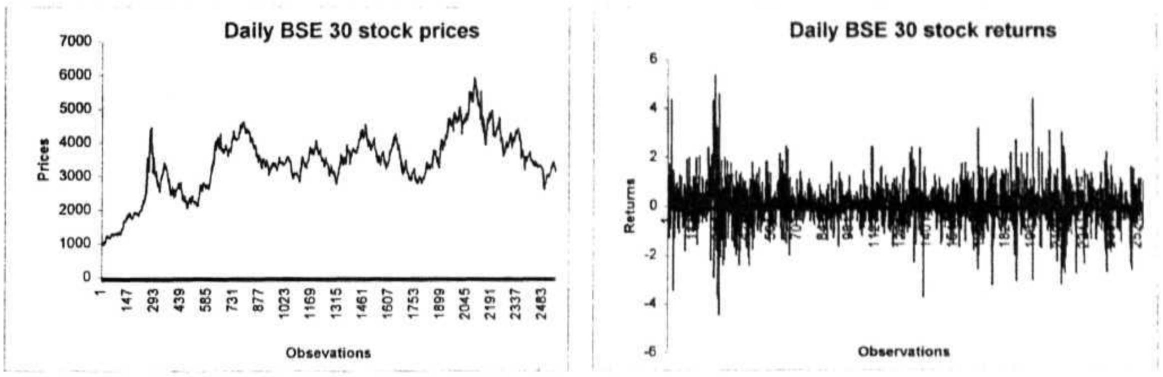


Figure 4.2 Quantile-Quantile (QQ) plots of daily stock prices and returns against normal distribution

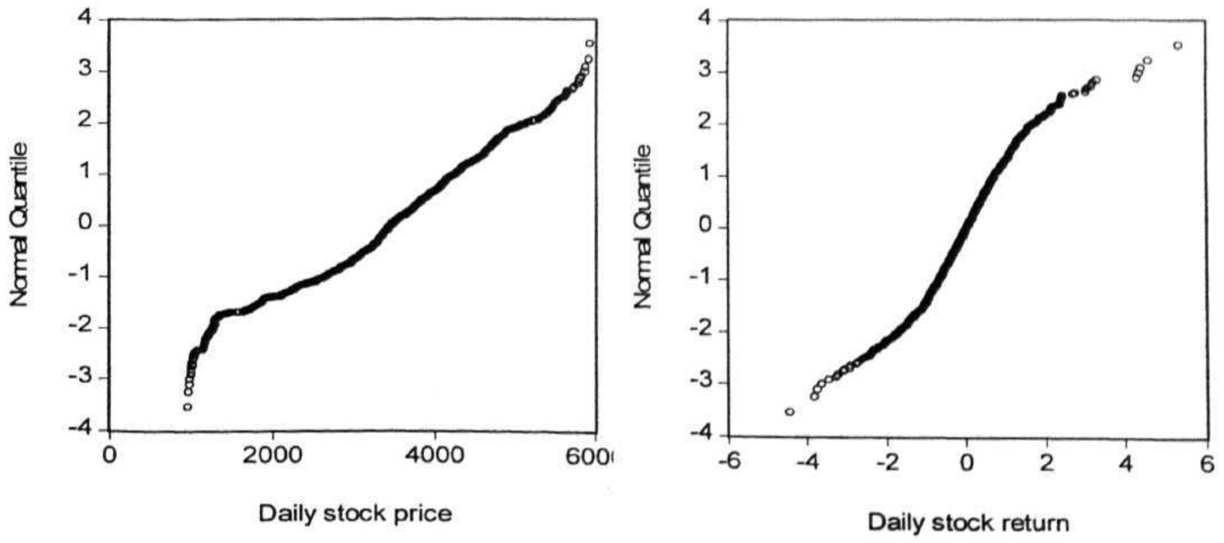


Figure 4.3 Plots of in-sample errors of **ANN, LAR, and RW** in **predicting daily stock returns for the period February 20, 1991-March** 7, 2000

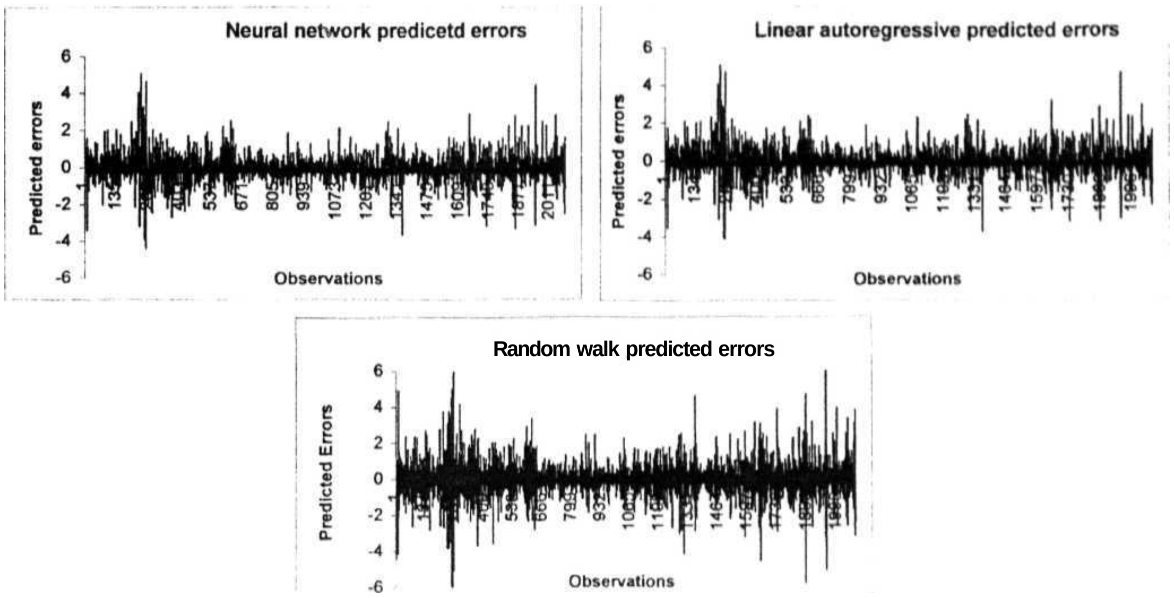


Figure 4.4 Plots of **out-of-sample** errors of ANN, LAR, RW in predicting daily stock returns for the period April 7, 2000-December 31, 2001

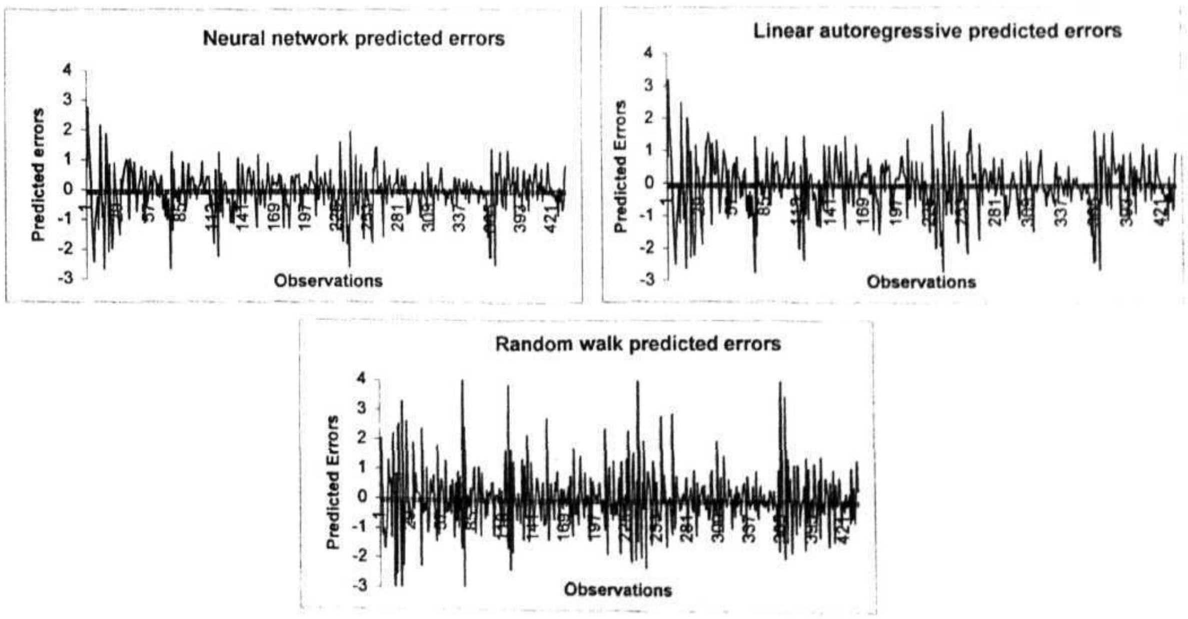


Figure 4.5 Plots of in-sample errors of ANN, LAR and RW in predicting daily stock prices for the period February **20, 1991-March 7,** 2000

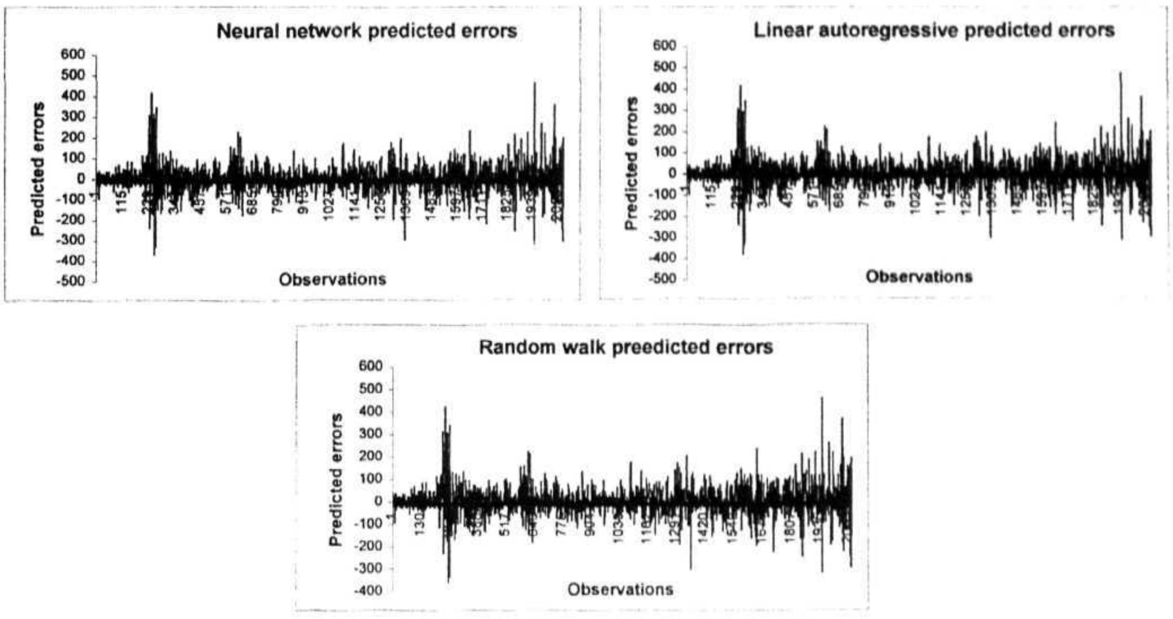


Figure 4.6 Plots of out-of-sample errors of ANN, LAR and RW in predicting daily stock prices for the period April 7, 2000-December 31, 2001

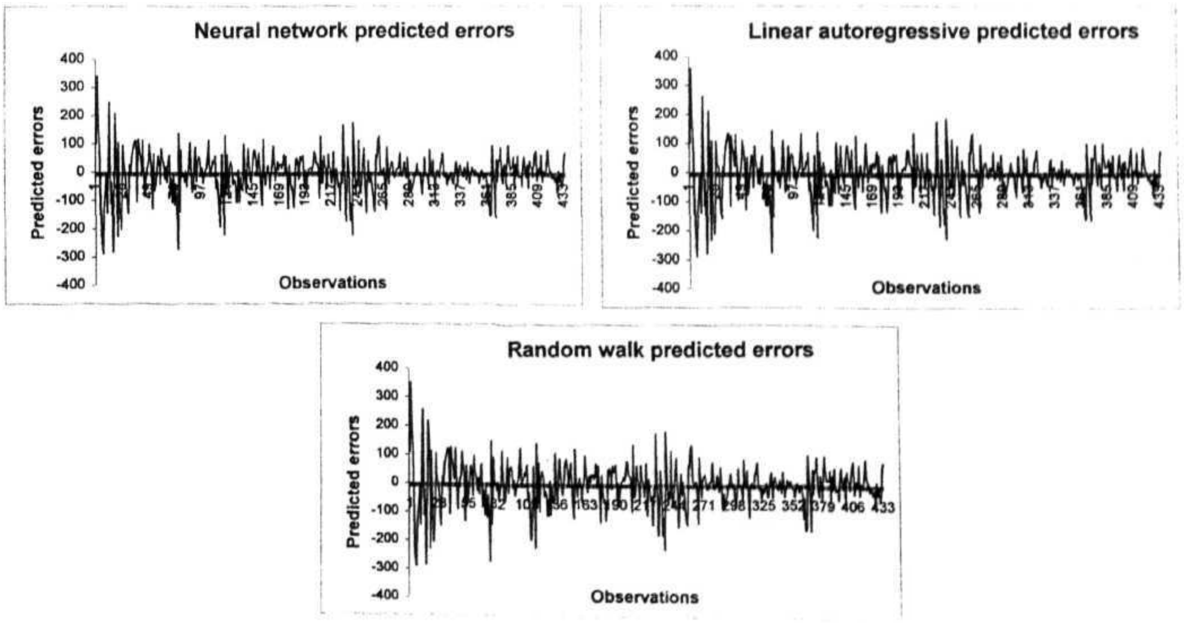


Figure 4.7 Plots of weekly BSE 30 stock prices and returns for the period January 3, 1992-November 8, 2002

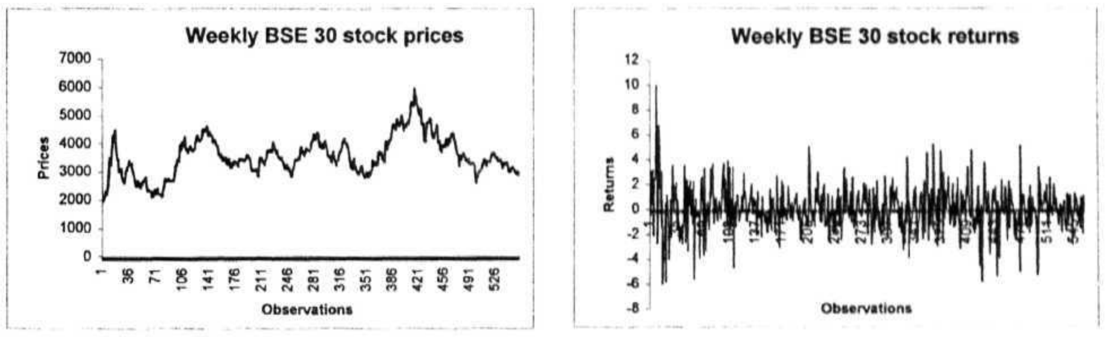


Figure 4.8 Quantile-Quantile (QQ) plots of weekly stock prices and returns against normal distribution

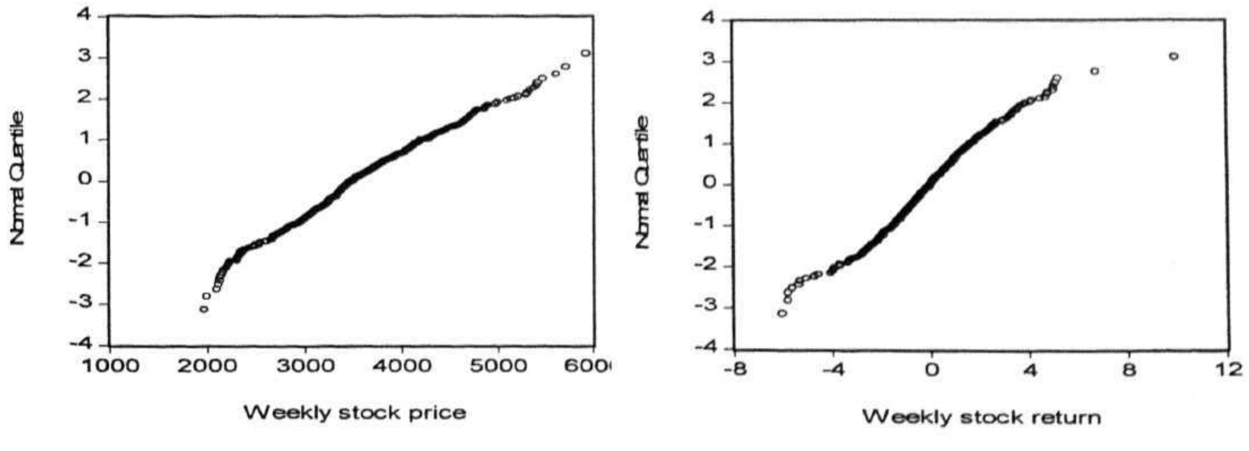


Figure 4.9 Plots of in-sample errors of ANN, LAR and RW in predicting weekly stock returns for the period May 29, 1992-March 24, 2000

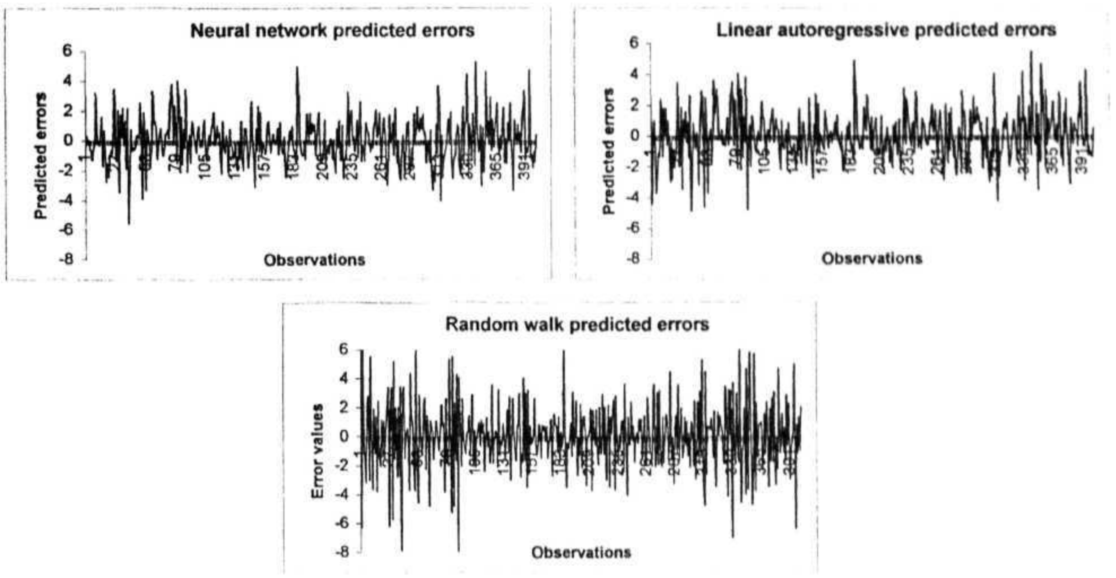


Figure 4.10 Plots of out-of-sample errors of ANN, LAR and RW in predicting weekly stock returns for the period August 18, 2000-November 8, 2002

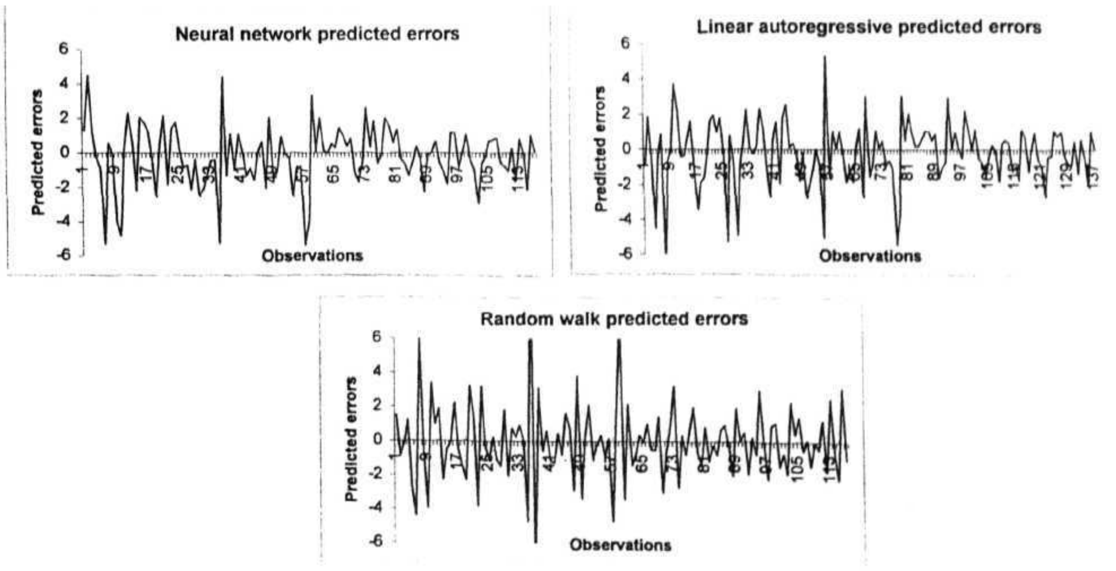


Figure 4.11 Plots of in-sample errors of ANN, LAR and RW in predicting weekly stock prices for the period May 29, 1992-March 24, 2000

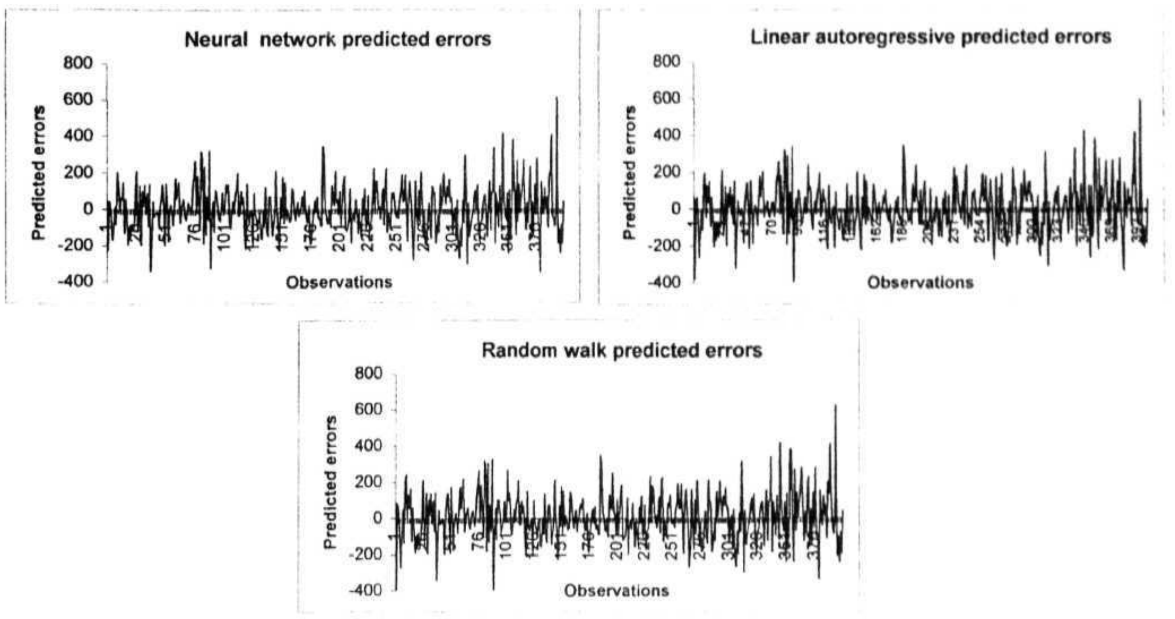
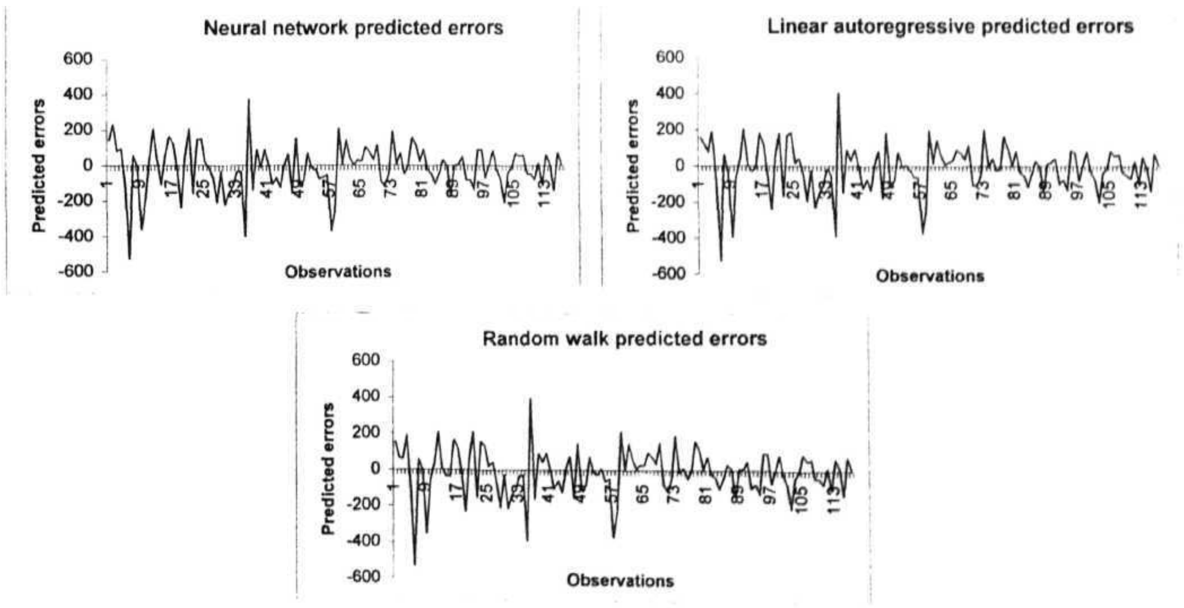


Figure 4.12 Plots of out-of-sample errors of ANN, LAR and RW in predicting weekly stock prices for the period August 18, 2000-November 8, 2002



Chapter V

Improving Generalization Performance of Neural Network

An economist is an expert who will know tomorrow why the things he predicted yesterday didn 7 happen day.

5.0 Introduction

In the last two chapters, we compared the performance of neural network with linear autoregressive and random walk models in predicting daily and weekly exchange rate and stock returns. The empirical findings suggested that, in general, neural network gives better in-sample forecasting of exchange rate and stock returns than linear autoregressive and random walk models. However, sometimes, neural network was found to have failed to improvc upon the linear autoregressive model and random walk model in out-of-sample forecasting of exchange rate and stock returns. To illustrate, we found no clear winner modcl between neural network and linear autoregressive model in out-of-sample forecasting of daily exchange rate returns (see section 3.2.3 in chapter III) and found random walk outperforming neural network in out-of-sample forecasting of weekly stock prices (see section 4.3.3 in chapter IV). The present chapter delves into the issue of whether the generalization or out-of-sample performance of neural network can be improved in predicting exchange rate and stock returns. We have applied two methods of improving generalization, namely, regularization and early stopping technique to improve the out-of-sample performance of neural network in predicting weekly exchange rate and stock returns. We have not taken daily exchange rate and stock return for the study in view of the fact that neural network estimation involves computational complexities and is a very time consuming process. Further, our aim here is basically to test whether regularization and early stopping techniques improve the out-of-sample performance of neural network on a given data set, and sampling frequency is not a major issue here.

The rest of the chapter proceeds as follows. In section 5.1, we discuss the two methods, regularization and early stopping, which are used to improve the generalization or out-of-sample performance of neural network. Section 5.2 presents the empirical results pertaining to improvement in out-of-sample performance of neural network by each of these techniques in forecasting weekly exchange rate returns followed by empirical results in case of weekly stock returns in section 5.3. Finally, section 5.4 concludes the chapter.

5.1 Methods of Improving Generalization

The goal of neural network training is to produce a network which minimizes the errors on the training set, but which will also respond properly to new inputs. When a network is able to perform as well on new inputs or the out-of-sample data as on training set inputs or in-sample data, we say that the network generalizes well. However, one of the problems that occurs during neural network training is Overfitting. Overfitting (for detailed discussion see section 2.1.2 in chapter II) is an outcome of a situation where the neural network is overtrained or has a high complexity. In this circumstance, the error on the training set is driven to a very small value, but when new data is presented to the network the error is large. This is because the network has memorized the training examples, including all of their peculiarities, but it has not learned to generalize to new situations and hence the bad generalization or out-of-sample performance.

One method for improving network generalization is to use a network, which is just large enough to provide an adequate fit. The larger a network we use, the more complex the functions that the network can create. If we use a small enough network, it will not have enough power to overfit the data. But the problem is that it is difficult to know beforehand how large a network should be for a specific application. Regularization and early stopping are two other methods that are used for improving out-of-sample performance of neural network. These are discussed below.

5.1.1 Regularization

The first method for improving generalization is called regularization. This method constrains the size of network weights. The idea is that the true underlying function is assumed to have a degree of smoothness. When the weights in a network are kept small, the network response will be smooth. With regularization, any reasonably oversized network should be able to adequately represent the true function. Thus, regularization tries to limit the complexity of the network such that it is unable to learn peculiarities.

The typical performance function that is used for training a feedforward neural network is the mean of squares of the network errors:

$$F = E_D = 1/N \sum_{t=1}^{N} (y_t - \hat{y}_t)^2 \quad \ldots \tag{5.1}$$

However, regularization modifies the performance function E_D by adding an additional term that consists of the mean of the sum of squares of the network weight and the objective function now becomes

$$F = \beta E_D + \alpha E_W \quad \ldots \tag{5.2}$$

where E_W is the mean of the sum of squares of the network weights and α and β are objective function or regularization parameters. The relative size of the objective function parameters dictates the emphasis for training. If $\alpha \ll \beta$, then the training algorithm will drive the errors smaller. On the other hand, if $\alpha \gg \beta$ then training will emphasize the weight size reduction at the expense of network errors. This will produce a smoother network response and thus the network is less likely to overfit. However, the problem with regularization is that it is difficult to determine the optimum values for objective function or regularization parameters. To overcome this problem, it is desirable to determine the optimal regularization parameters in an automated fashion. One approach to this process is the Bayesian framework of

MacKay (1992), who has done extensive work on the application of Bayes' rule to neural network training and to optimizing regularization.

The Bayesian school of statistics is based on a different view of what it means to learn from data, in which probability is used to represent uncertainty about the relationship being learned. Before we have seen any data, our opinions about what the true relationship might be can be expressed in a probability distribution over the network weights that define this relationship. After we look at the data, our revised opinions are captured by a posterior distribution over network weights. Nctwork weights that seemed plausible before, but which don't match the data very well, will now be seen as being much less likely, while the probability for values of the weights that do fit the data well will have increased.

Typically, the purpose of training is to make predictions for future cases in which only the inputs to the network are known. The result of conventional network training is a single set of weights that can be used to make such predictions. In contrast, the result of Bayesian training is a posterior distribution over network weights. In the Bayesian framework the weights of the network are considered random variables. After the data is taken, the density function for the weights can be updated according to Bayes' rule:

$$P(\mathbf{w} \mid D,\alpha,\beta,M) = \frac{P(D \mid \mathbf{w},\beta,M)P(\mathbf{w} \mid \alpha,M)}{P(D \mid \alpha,\beta,M)} \tag{5.3}$$

where *D* represents the data set, *M* is the particular neural network model used, and *w* is the vector of network weights. $P\ (\mathbf{w} \mid \alpha,\ M)$ is the prior density, which represents our knowledge of the weights before any data is collected. $P\ (D \backslash w,\ \beta,\ M)$ is the likelihood function, which is the probability of the data occurring, given the weights $\mathbf{w}$. $p\ (D \mid \alpha,\ p,\ M)$ is a normalization factor, which guarantees that the total probability is 1.

Assuming that the noise in the training set is Gaussian and that the prior distribution for the weights is Gaussian, the probability densities can be written

$$p(D \mid \mathbf{w},\beta,M) = \frac{1}{Z_D(\beta)}\exp(-\beta E_D) \quad \text{and}$$

$$P(\mathbf{w} \mid \alpha,M) = \frac{1}{Z_W(\alpha)}\exp(-\alpha E_W) \tag{5.4}$$

where $Z_D(\beta) = (\pi/\beta)^{n/2}$ and $Z_W(\alpha) = (\pi/\alpha)^{N/2}$. Substituting these probabilities into (5.3), we get

$$\begin{aligned} P(\mathbf{w} \mid D,\alpha,\beta,M) &= \frac{\frac{1}{Z_w(\alpha)}\frac{1}{Z_w(\beta)}\exp(-(\beta E_D + \alpha E_w))}{P(D \mid \alpha,\beta,M)} \\ &= \frac{1}{Z_F(\alpha,\beta)}\exp(-F(\mathbf{w})) \end{aligned} \tag{5.5}$$

In this Bayesian framework, the optimal weights should maximize the posterior probability $P(\mathbf{w} \mid D, \alpha \beta, M)$. Maximizing the posterior probability is equivalent to minimizing the regularized objective function $F = fiEo + \alpha E_W$. The values of the regularization parameters *a* and β, in the objective function, are optimized by applying David MacKay's Bayesian techniques (1992). This technique requires the computation of the Hessian matrix of the objective function at the minimum point to get the optimum values of α and β. A Gauss-Newton approximation is used to compute the Hessian matrix. This approximation and the optimum values for regularization parameters are readily available once we use the Bayesian regularization as the training algorithm for network training. One important feature of this algorithm is that it provides a measure of how many network parameters are being effectively used by the network to reduce the error on the training set. If the final effective number of parameters is very close to the actual number of parameters in neural network, then the neural network may not be large enough to properly represent the true function. In this case, more hidden layer units should be added to retrain the network. If the larger network has the same final effective number of parameters, then the smaller network was large enough. Otherwise, more hidden layer units may need to be added.

5.1.2 Early Stopping

The second method for improving generalization is called early stopping. Early stopping technique aims at stopping the training at the point of optimal generalization. Thus, it prevents overtraining in neural nctwork, which is one cause of Overfitting and consequently bad generalization. In this technique the available data is divided into three subsets. The first subset is the training set which is used for computing the gradient and updating the network weights and biases. The second subset is the validation set. The error on the validation set is monitored during the training process. The validation error will normally decrease during the initial phasc of training, as does the training set error. However, when the network begins to overfit the data, the error on the validation set will typically begin to rise. When the validation error increases for a specified number of iterations, the training is stopped, and the weights and biases at the minimum of the validation error are returned.

The third set is the test set. The error on the test set is not used during the training, but it is used to compare different models. It is also useful to plot the test error during the training process. If the error in the test set reaches a minimum at a significantly different iteration number than the validation set error, this may indicate a poor division of data set.

5.2 Improving Out-of-sample Performance of Neural Network: Weekly Exchange Rate Returns

In this section, we report the results of our attempt to increase the out-of-sample performance of neural network by using the techniques of Bayesian regularization and early stopping in predicting weekly Indian rupee/US dollar exchange rate returns. The network with Bayesian regularization technique is represented as BANN and the network with early stopping technique is represented as EANN. The results are presented from Table 5.1 to Table 5.5.

Table 5.1 shows the effects of input nodes on out-of-sample performance of network with Bayesian regularization technique (BANN) in predicting weekly normalized exchange rate returns. The RMSE and MAE at each level of input node are the averages across five hidden node levels viz. 4, 8, 12, 16, and 20. The results in the table convey that out-of-sample performance of BANN is not varying greatly in terms of RMSEs and MAEs even if we make the network more complex by adding more input units. Among 20 input units, we find that input units such as 1, 2, 5, 6, and 7 contribute to bring down out-of-sample RMSEs and MAEs. Except for these, the network's performance is found to be worse at al! other input levels in terms of high values of RMSE and MAE. Hence, we take these five input units, 1, 2, 5, 6, and 7 as inputs to BANN for out-of-sample forecasting of weekly exchange rate returns.

The effects of hidden nodes on out-of-sample performance of BANN in predicting weekly normalized exchange rate returns are shown in Table 5.2. The RMSEs and MAEs at each level of input in combination with each of the five hidden node levels 4, 8,12, 16, and 20 are the averages of ten runs. The results in Table 5.2 show that when the input units are in the range of 1 to 5, the effects of hidden nodes are not clear in terms of RMSEs and MAEs. However, in general, within this input range, the RMSEs and MAEs increase as the number of hidden nodes increases. For example, at input level 1, if we put 4 hidden nodes, the network gives the RMSE value of 0.0221. Adding 8 hidden nodes has no further effect in terms of RMSE and MAE. However, the addition of further hidden nodes decreases BANN's performance in terms of increasing values of RMSE and MAE.

However, when the input units are in the range of 6 to 20, we find that at each level of input unit, the network's performance is not at all sensitive to the number of hidden nodes. This conclusion is arrived at on the ground of invariability of RMSEs and MAEs at each level of input level after input level 5 in combination with each of the five hidden node levels. For example, input level 6 in combination with 4 hidden nodes produces the values of RMSE and MAE, which are equal to 0.0220 and 0.0144 respectively. Adding of more hidden units such as 8, 12, 16, and 20 produce no better result as the values of RMSEs and MAEs at these hidden node levels are all equal to

0.0220 and 0.0144. From these findings we conclude that BANN's out-of-sample performance is not sensitive to the number of hidden nodes. As a result, we try with less than 4 hidden nodes and find that only one hidden node achieves the best result. Hence, neural network with Bayesian regularization technique uses architecture of 5-1-1 to make out-of-sample forecasting of weekly exchange rate returns. One more point to be noted here is that the effective number of parameters, which the network uses to reduce the training error, arc found to be 6 out of 8 total network parameters. We also find that the larger networks with more hidden units also yield the same number of effective parameters. Thus, smaller network i.e. 5-1-1 is large enough to approximate the function.

Table 5.3 shows the effects of input nodes on out-of-sample performance of network with early stopping technique in predicting weekly normalized exchange rate returns. The whole data set is divided into three subsets. Out of a total number of 496 observations, training, validation and testing set consist of 270, 130, and 96 observations respectively. The RMSE and MAE at each level of input node are the averages across fivc hidden nodc levels viz. 4, 8, 12, 16, and 20. The results presented in the table show that the performance of the network with early stopping technique gets worse in terms of both RMSE and MAE as the number of input nodes increases. Among 20 input levels, the best performance is achieved at input level 1 in terms of lowest RMSE and MAE, which are equal to 0.0185 and 0.0125 respectively. Hence, only one input is supplied to EANN as input for out-of-sample forecasting of weekly exchange rate returns.

The effects of hidden nodes on out-of-sample performance of EANN in predicting weekly normalized exchange rate returns are shown in Table 5.4. The RMSEs and MAEs at each level of input node in combination with each of the five hidden node levels 4, 8,12, 16, and 20 are the averages of ten runs. The results demonstrate that, in general, at each input level, out-of-sample RMSEs and MAEs increases as the number of hidden node increases. However, input level 1 in combination with 8 hidden nodes achieves the best performance in terms of lowest RMSE and MAE, which are equal to 0.0178 and 0.0114 respectively. Hence, neural network with early stopping technique uses architecture of 1-8-1 to make out-of-sample forecasting of weekly exchange rate returns. As shown in Figure 5.1, initially the error on both training and validation sets falls with an increase in the number of iterations. After 50 iterations, the error on the validation set starts increasing. So training stops when the number of iterations reaches 50.

In Table 5.5, we have presented the out-of-sample performance in predicting weekly exchange rate returns of the simple neural network model (ANN) together with that of each of the alternative methods of improving generalization, viz. BANN and EANN. We have retained the same network configuration i.e. 16-3-3-1 for simple neural network model (ANN) as we have in section 3.3.2 in chapter III for the out-of-sample forecasting of weekly exchange rate returns. BANN uses network architecture of 5-1-1 and EANN uses network architecture of 1-8-1. In the same table, we have also presented the results for the linear autoregressive (LAR) and random walk (RW) models. We note here that, for LAR, we have retained the same specification as in section 3.3.2 of chapter III, for out-of-sample forecasting of weekly exchange rate returns.

The results in Table 5.5 provide evidence that network with Bayesian regularization technique (BANN) improves the out-of-sample performance of ANN in terms of MAD, CORR, and SIGN. BANN gives smaller value of median absolute deviation (MAD), which is equal to 0.0303, than the corresponding value of ANN (0.0306). BANN also gives higher values of correlation coefficient (CORR) and sign prediction (SIGN), 0.2461 and 0.5657 respectively, than the corresponding values of ANN, which are equal to 0.2455 and 0.5394 respectively. However, BANN fails to improve upon ANN in terms of RMSE, MAE, and DA in out-of-sample forecasting weekly exchange rate returns.

The results also show that network with early stopping technique (EANN) improves the out-of-sample performance of ANN in terms of all performance measures except for direction accuracy (DA) in forecasting weekly exchange rate returns. EANN returns smaller values of RMSE, MAE, and MAD, which are equal to

0.0832, 0.0540, and 0.0297 respectively than the corresponding values of ANN, which are equal to 0.0835, 0.0547, and 0.0306 respectively. EANN gives higher correlation coefficient (0.2579) than ANN, in which the value for correlation coefficient is 0.2455. Network with early stopping technique also has higher sign prediction than ANN. In addition to this, if we compare the out-of-sample performances of BANN and EANN, we find that EANN gives better out-of-sample performances than BANN. EANN outshines BANN by all performance measures except for SIGN, whose value is equal for both. From this we conclude that early stopping is a better method of improving generalization than Bayesian regularization.

The results in the Table 5.5 also show that ANN outperforms LAR in terms of all performance measures except for SIGN. Nctwork with Baycsian regularization and network with early stopping are also found to give better out-of-sample forecasting than linear autoregressive model by all performance measures except for SIGN. Though these two techniques improve upon ANN in sign prediction, they are still outperformed by LAR in this. It is also found that ANN, BANN, and EANN outperform RW by all performance measures in out-of-sample forecasting of weekly exchange rate returns. From these findings, we conclude that neural networks gives better out-of-sample forecasting than both linear autoregressive and random walk model. And if we use BANN and EANN, they yield much better out-of-sample performance than LAR and RW in forecasting weekly exchange rate returns.

5.3 Improving Out-of-sample Performance of Neural Network: Weekly Stock Returns

In this section, we turn our attention to the out-of-sample prediction of weekly stock returns, and the issue of whether this can be improved by using techniques of Bayesian regularization and early stopping. The results are presented from Table 5.6 to 5.10.

Table 5.6 presents the effects of input nodes on the out-of-sample performance of neural network with Bayesian regularization (BANN) in terms of RMSE and MAE in predicting weekly normalized stock returns. The RMSE and MAE at each level of input node are the average across five hidden node levels. We have experimented with twenty levels of the input nodes ranging from 1 to 20 lagged values of the dependent variable i.e. weekly normalized stock returns. Out of these twenty levels of input nodes, the input nodes, which contribute to bring down out-of-sample RMSE and MAE, are taken as inputs to BANN. The results presented in the table demonstrate that the out-of-sample performance of BANN in terms of RMSE and MAE is invariant to the number of input nodes upto input level 11. To illustrate, the values of RMSE and MAE, which are equal to 0.0999 and 0.0717, are same at each level of input upto input level 11. After input level 11, the out-of-sample performance decreases as the number of input nodes increases. As a result of these findings, only one input i.e. input level 1 is taken as input to BANN.

The effects of hidden nodes on the out-of-sample performance of BANN in terms of RMSE and MAE in predicting weekly normalized stock returns are presented in Table 5.7. The RMSEs and MAEs at each level of input in combination with each of the five hidden node levels, 4, 8,12, 16, and 20 are the averages of ten runs. The table shows that the out-of-sample performance of BANN in terms of RMSE and MAE is completely invariant with the number of hidden nodes at each input level. Because of this finding, less than 4 hidden nodes are tried and we find that only 1 hidden unit is enough to achieve the best result. Hence network architecture of 1-1-1 is used for BANN for out-of-sample forecasting of weekly stock returns. The effective numbers of parameters, which the network uses to reduce the training error, are found to be 1 out of 4 total network parameters. We also find that the larger networks with more hidden units also yield the same numbers of effective parameters. Thus, smaller network i.e. 1-1-1 is considered as large enough to approximate the function.

Table 5.8 displays the effects of input nodes on out-of-sample performance of neural network with **early** stopping technique (EANN) in predicting weekly normalized stock returns. The whole data set, which consists of 557 observations, is divided into three subsets, namely, training set consisting of 300 observations, validation set consisting of 160 observations, and the testing set consisting of 97 observations. The RMSE and MAE at each level of input node are the average across five hidden node levels, 4, 8, 12, 16, and 20. The table reveals that adding of more and more input units, instead of increasing the out-of-sample performance, decreases the performance in terms of RMSE and MAE. Thus, we find the best performance at input level 1 in terms of lowest RMSE and MAE, which arc equal to 0.0831 and 0.0621, among all input levels. Therefore, only one input is taken as the input to EANN to make out-of-sample forecasts of weekly stock returns.

The effects of hidden nodes on out-of-sample performance of EANN are shown in Table 5.9. The RMSEs and MAEs at each level of input in combination with each of the five hidden node levels, 4, 8,12, 16, and 20 are the averages of ten runs. The table puts on view that, in general, the out-of-sample performance of EANN in terms of RMSE and MAE goes down as the number of hidden nodes increases at each level of input. However, input level 1 in combination with 20 hidden units achieves the best result in terms of lowest RMSE and MAE, which are equal to 0.0806 and 0.0610 respectively among all the combinations. Hence network architecture of 1-20-1 is used for EANN to make out-of-sample forecasts of weekly stock returns. Figure 5.2 demonstrates how the training of network is stopped through early stopping technique once the error on validation set starts increasing. As shown in Figure 5.2, the training stops when the number of iterations reaches 79, because after that the error on the validation set starts increasing.

Table 5.10 shows the out-of-sample performances of ANN, BANN, and EANN in predicting weekly stock returns. We have retained the same network configuration i.e. 20-2-1 for ANN as we have in section 4.3.2 of chapter IV in predicting weekly stock returns. BANN uses network architecture of 1-1-1 and EANN uses network architecture of 1-20-1 to make out-of-sample forecasting of weekly stock returns. In this table, we have also presented the results for the linear autoregressive (LAR) and random walk (RW) models. We have retained the same specification for LAR (lags 12, 13, and 18 are significant explanatory variables) as used in section 4.3.2 of chapter IV, for out-of-sample forecasting of weekly stock returns.

The results presented in Table 5.10 reveal that neural network with Bayesian regularization technique (BANN) improves upon the out-of-sample performance of neural network (ANN) in terms of all performance measures. BANN returns smaller values of RMSE, MAE, and MAD in comparison to the corresponding values of ANN. BANN has also higher values of CORR, DA, and SIGN than the equivalent values of ANN. Similarly, EANN improves upon out-of-sample performance of ANN in terms of all performance measures. Between BANN and EANN, EANN outperforms BANN by all performance measures except for DA. From these findings we conclude that both BANN and EANN improve upon the ANN in out-of-sample forecasting of weekly stock returns. And we also find that neural network with early stopping technique is a better method of improving generalization performance than neural network with Bayesian regularization technique.

The results in the table also show that ANN outperforms LAR by all performance measures except for DA and SIGN. But, BANN is now found to outshine LAR by all performance measures. Similarly, except for direction accuracy (DA), EANN outclasses LAR by all performance measures. It is also found that both ANN and BANN give better out-of-sample forecasting of weekly stock returns than RW by all performance measures except for correlation coefficient (CORR) and sign prediction (SIGN). Even though BANN improves upon ANN in correlation coefficient and sign prediction, it does not improve upon it enough to outweigh RW in these two performance measures. However, EANN is found to outperform RW by all performance measures. From these findings, we conclude that neural network performs better than both linear autoregressive and random walk models in out-of-sample forecasting of weekly stock returns. However, two methods of improving generalization such as neural network with Bayesian regularization technique (BANN) and neural network with early stopping technique (EANN) produce even much better out-of-sample forecasts of weekly stock returns.

5.4 Conclusion

In this chapter, we have applied two methods of improving generalization performance of neural network, namely, Bayesian regularization and early stopping in the forecasting of weekly exchange rate returns and stock returns. We find that both the techniques do indeed greatly improve the out-of-sample performance of neural network. The results also suggest that network with early stopping technique improves the out-of-sample forecasting of neural network to a greater extent than network with Bayesian regularization technique does in both the exchange rate and stock returns prediction. We also find that these two techniques, Bayesian regularization and early stopping improve upon neural network's out-of-sample performance better in the case of weekly stock returns than in the case of weekly exchange rate returns. In addition to these findings, we find that both network with Bayesian regularization and early stopping technique outperform linear autoregressive and random walk models except in out-of-sample forecasting of weekly exchange rate and stock returns except for very few cases. In other words, barring few cases, after using these two techniques of improving generalization, neural network is found to outperform linear autoregressive and random walk models by the performance measures in which they had earlier proved superior to neural network.

Table 5.1 Out-of-sample performance of BANN on weekly normalized exchange rate returns: effects of input nodes

Input	RMSE	MAE	Input	RMSE	MAE
1	0.0227	0.0151	**11**	0.0220	0.0150
2	0.0225	0.0148	**12**	0.0221	0.0151
3	0.0248	0.0156	**13**	0.0223	0.0152
4	0.0244	0.0151	**14**	0.0223	0.0152
5	0.0224	0.0146	15	0.0222	0.0152
6	0.0220	0.0144	**16**	0.0223	0.0152
7	0.0218	0.0143	**17**	0.0223	0.0152
8	0.0220	0.0149	**18**	0.0224	**0.0153**
9	0.0220	0.0150	**19**	0.0229	0.0156
10	0.0220	0.0150	**20**	0.0229	0.0156

Table 5.2 Out-of-sample performance of BANN on weekly normalized exchange rate returns: effects of hidden nodes

Input	Hidden	RMSE	MAE	Input	Hidden	RMSE	MAE
1	4	0.0221	0.0146	8	4	0.0220	0.0149
1	8	0.0221	0.0146	8	8	0.0220	0.0149
1	12	0.0232	0.0155	8	12	0.0220	0.0149
1	16	0.0235	0.0155	8	16	0.0220	0.0149
1	20	0.0227	0.0153	8	20	0.0220	0.0149
2	4	0.0227	0.0152	9	4	0.0220	0.0150
2	8	0.0236	0.0159	9	8	0.0220	0.0150
2	12	0.0223	0.0145	9	12	0.0220	0.0150
2	16	0.0224	0.0145	9	16	0.0220	0.0150
2	20	0.0219	0.0143	9	20	0.0220	0.0150
3	4	0.0283	0.0153	10	4	0.0220	0.0150
3	8	0.0253	0.0156	10	8	0.0220	0.0150
3	12	0.0252	0.0156	10	12	0.0220	0.0150
3	16	0.0252	0.0159	10	16	0.0220	0.0150
3	20	0.0250	0.0159	10	20	0.0220	0.0150
4	4	0.0220	0.0143	11	4	0.0220	0.0150
4	8	0.0220	0.0143	11	8	0.0220	0.0150
4	12	0.0243	0.0157	11	12	0.0220	0.0150
4	16	0.0309	0.0169	11	16	0.0220	0.0150
4	20	0.0232	0.0143	11	20	0.0220	0.0150
5	4	0.0220	0.0144	12	4	0.0221	0.0151
5	8	0.0220	0.0144	12	8	0.0221	0.0151
5	12	0.0240	0.0155	12	12	0.0221	0.0151
5	16	0.0220	0.0144	12	16	0.0221	0.0151
5	20	0.0220	0.0144	12	20	0.0221	0.0151
6	4	0.0220	0.0144	13	4	0.0223	0.0152
6	8	0.0220	0.0144	13	8	0.0223	0.0152
6	12	0.0220	0.0144	13	12	0.0223	0.0152
6	16	0.0220	0.0144	13	16	0.0223	0.0152
6	20	0.0220	0.0144	13	20	0.0223	0.0152
7	4	0.0218	0.0143	14	4	0.0223	0.0152
7	8	0.0218	0.0143	14	8	0.0223	0.0152
7	12	0.0218	0.0143	14	12	0.0223	0.0152
7	16	0.0218	0.0143	14	16	0.0223	0.0152
7	20	0.0218	0.0143	14	20	0.0223	0.0152

continued on next page

continued from previous pa

Input	Hidden	RMSE	MAE	Input	Hidden	RMSE	MAE
15	4	0.0222	0.0152	18	4	0.0224	0.0153
15	8	0.0222	0.0152	18	8	0.0224	0.0153
15	12	0.0222	0.0152	18	12	0.0224	0.0153
15	16	0.0222	0.0152	18	16	0.0224	0.0153
15	20	0.0222	0.0152	18	20	0.0224	0.0153
16	4	0.0223	0.0152	19	4	0.0229	0.0156
16	8	0.0223	0.0152	19	8	0.0229	0.0156
16	12	0.0223	0.0152	19	12	0.0229	0.0156
16	16	0.0223	0.0152	19	16	0.0229	0.0156
16	20	0.0223	0.0152	19	20	0.0229	0.0156
17	4	0.0223	0.0152	20	4	0.0229	0.0156
17	8	0.0223	0.0152	20	8	0.0229	0.0156
17	12	0.0223	0.0152	20	12	0.0229	0.0156
17	16	0.0223	0.0152	20	16	0.0229	0.0156
17	20	0.0223	0.0152	20	20	0.0229	0.0156

Table 5.3 Out-of-sample performance of EANN on weekly normalized exchange rate returns: effects of input nodes

Input	RMSE	MAE	Input	RMSE	MAE
1	0.0185	0.0125	11	0.0223	0.0161
2	0.0218	0.0151	12	0.0217	0.0154
3	0.0237	0.0168	13	0.0217	0.0175
4	0.0246	0.0175	14	0.0237	0.0172
5	0.0221	0.0155	15	0.0218	0.0158
6	0.0206	0.0143	16	0.0232	0.0171
7	0.0246	0.0179	17	0.0219	0.0160
8	0.0220	0.0157	18	0.0234	0.0175
9	0.0222	0.0163	19	0.0243	0.0194
10	0.0207	0.0149	20	0.0238	0.0176

Table 5.4 Out-of-sample performance of EANN on weekly normalized exchange rate returns: effects of hidden nodes

Input	Hidden	RMSE	MAE	Input	Hidden	RMSE	MAE
1	4	0.0180	0.0120	8	4	0.0184	0.0125
1	8	0.0178	0.0114	8	8	0.0195	0.0139
1	12	0.0183	0.0125	8	12	0.0214	0.0155
1	16	0.0189	0.0131	8	16	0.0233	0.0167
1	20	0.0198	0.0137	8	20	0.0275	0.0202
2	4	0.0182	0.0125	9	4	0.0182	0.0138
2	8	0.0200	0.0133	9	8	0.0222	0.0164
2	12	0.0227	0.0158	9	12	0.0189	0.0133
2	16	0.0210	0.0151	9	16	0.0262	0.0191
2	20	0.0271	0.0192	9	20	0.0256	0.0189
3	4	0.0184	0.0124	10	4	0.0184	0.0128
3	8	0.0265	0.0200	10	8	0.0221	0.0162
3	12	0.0269	0.0190	10	12	0.0190	0.0135
3	16	0.0240	0.0171	10	16	0.0223	0.0163
3	20	0.0229	0.0158	10	20	0.0219	0.0159
4	4	0.0239	0.0165	11	4	0.0179	0.0119
4	8	0.0202	0.0137	11	8	0.0198	0.0145
4	12	0.0298	0.0220	11	12	0.0222	0.0167
4	16	0.0236	0.0171	11	16	0.0310	0.0221
4	20	0.0255	0.0185	11	20	0.0213	0.0151
5	4	0.0190	0.0131	12	4	0.0187	0.0128
5	8	0.0196	0.0135	12	8	0.0201	0.0142
5	12	0.0279	0.0201	12	12	0.0266	0.0192
5	16	0.0197	0.0142	12	16	0.0211	0.0151
5	20	0.0243	0.0170	12	20	0.0222	0.0161
6	4	0.0179	0.0118	13	4	0.0206	0.0149
6	8	0.0198	0.0138	13	8	0.0199	0.0138
6	12	0.0192	0.0130	13	12	0.0215	0.0156
6	16	0.0244	0.0173	13	16	0.0233	0.0174
6	20	0.0222	0.0157	13	20	0.0234	0.0175
7	4	0.0193	0.0133	14	4	0.0202	0.0139
7	8	0.0217	0.0164	14	8	0.0242	0.0179
7	12	0.0354	0.0260	14	12	0.0213	0.0151
7	16	0.0219	0.0158	14	16	0.0305	0.0229
7	20	0.0248	0.0180	14	20	0.0224	0.0165

continued on next page

continued fr<

Input	Hidden	RMSE	MAE	Input	Hidden	RMSE	MAE
15	4	0.0203	0.0144	**18**	**4**	0.0197	0.0144
15	8	0.0194	0.0147	**18**	**8**	0.0223	0.0168
15	12	0.0219	0.0157	**18**	**12**	0.0191	0.0139
15	16	0.0204	0.0144	**18**	**16**	0.0259	0.0195
15	20	0.0272	0.0199	**18**	**20**	0.0301	0.0230
16	4	0.0186	0.0132	**19**	**4**	0.0184	0.0124
16	8	0.0183	0.0128	19	**8**	0.0239	0.0176
16	12	0.0212	0.0153	**19**	**12**	0.0205	0.0153
16	16	0.0216	0.0162	**19**	**16**	0.0205	0.0147
16	20	0.0366	0.0283	19	**20**	0.0382	0.0370
17	4	0.0195	0.0140	20	**4**	0.0196	0.0136
17	8	0.0195	0.0138	20	**8**	0.0198	0.0141
17	12	0.0213	0.0155	20	**12**	0.0227	0.0169
17	16	0.0240	0.0174	20	**16**	0.0315	0.0237
17	20	0.0253	0.0194	20	**20**	0.0257	0.0199

Table 5.5 Out-of-sample performance of ANN, BANN, EANN, LAR, and RW on weekly exchange rate return series

	ANN	BANN	EANN	LAR	RW
RMSE	0.0835	0.0846	0.0832	0.0877	0.1074
MAE	0.0547	0.0585	0.0540	0.0590	0.0676
MAPE					
MAD	0.0306	0.0303	0.0297	0.0319	0.0430
CORR	0.2455	0.2461	0.2579	0.2002	0.2232
DA	0.7368	0.6973	0.7105	0.6052	0
SIGN	0.5394	0.5657	0.5657	0.5921	0.5263

Table 5.6 Out-of-sample performance of BANN on weekly normalized stock returns: effects of input nodes

Input	RMSE	MAE	Input	RMSE	MAE
1	0.0999	0.0717	11	0.0999	0.0717
2	0.0999	0.0717	12	0.1008	0.0725
3	0.0999	0.0717	13	0.1031	0.0747
4	0.0999	0.0717	14	0.1029	0.0745
5	0.0999	0.0717	15	0.1032	0.0748
6	0.0999	0.0717	16	0.1032	0.0748
7	0.0999	0.0717	17	0.1043	0.0758
8	0.0999	0.0717	18	0.1053	0.0766
9	0.0999	0.0717	19	0.1051	0.0764
10	0.0999	0.0717	20	0.1048	0.0762

Table 5.7 Out-of-sample performance of BANN on weekly normalized stock returns: effects of hidden nodes

Input	Hidden	RMSE	MAE	Input	Hidden	RMSE	MAE
1	4	0.0999	0.0717	8	4	0.0999	0.0717
1	8	0.0999	0.0717	8	8	0.0999	0.0717
1	12	0.0999	0.0717	8	12	0.0999	0.0717
1	16	0.0999	0.0717	8	16	0.0999	0.0717
1	20	0.0999	0.0717	8	20	0.0999	0.0717
2	4	0.0999	0.0717	9	4	0.0999	0.0717
2	8	0.0999	0.0717	9	8	0.0999	0.0717
2	12	0.0999	0.0717	9	12	0.0999	0.0717
2	16	0.0999	0.0717	9	16	0.0999	0.0717
2	20	0.0999	0.0717	9	20	0.0999	0.0717
3	4	0.0999	0.0717	10	4	0.0999	0.0717
3	8	0.0999	0.0717	10	8	0.0999	0.0717
3	12	0.0999	0.0717	10	12	0.0999	0.0717
3	16	0.0999	0.0717	10	16	0.0999	0.0717
3	20	0.0999	0.0717	10	20	0.0999	0.0717
4	4	0.0999	0.0717	11	4	0.0999	0.0717
4	8	0.0999	0.0717	11	8	0.0999	0.0717
4	12	0.0999	0.0717	11	12	0.0999	0.0717
4	16	0.0999	0.0717	11	16	0.0999	0.0717
4	20	0.0999	0.0717	11	20	0.0999	0.0717
5	4	0.0999	0.0717	12	4	0.1008	0.0725
5	8	0.0999	0.0717	12	8	0.1008	0.0725
5	12	0.0999	0.0717	12	12	0.1008	0.0725
5	16	0.0999	0.0717	12	16	0.1008	0.0725
5	20	0.0999	0.0717	12	20	0.1008	0.0725
6	4	0.0999	0.0717	13	4	0.1031	0.0747
6	8	0.0999	0.0717	13	8	0.1031	0.0747
6	12	0.0999	0.0717	13	12	0.1031	0.0747
6	16	0.0999	0.0717	13	16	0.1031	0.0747
6	20	0.0999	0.0717	13	20	0.1031	0.0747
7	4	0.0999	0.0717	14	4	0.1029	0.0745
7	8	0.0999	0.0717	14	8	0.1029	0.0745
7	12	0.0999	0.0717	14	12	0.1029	0.0745
7	16	0.0999	0.0717	14	16	0.1029	0.0745
7	20	0.0999	0.0717	14	20	0.1029	0.0745

continued on next page

continued from previo

Input	Hidden	RMSE	MAE	Input	Hidden	RMSE	MAE
15	4	0.1032	0.0748	18	4	0.1053	0.0766
15	8	0.1032	0.0748	18	8	0.1053	0.0766
15	12	0.1032	0.0748	18	12	0.1053	0.0766
15	16	0.1032	0.0748	18	16	0.1053	0.0766
15	20	0.1032	0.0748	18	20	0.1053	0.0766
16	4	0.1032	0.0748	19	4	0.1051	0.0764
16	8	0.1032	0.0748	19	8	0.1051	0.0764
16	12	0.1032	0.0748	19	12	0.1051	0.0764
16	16	0.1032	0.0748	19	16	0.1051	0.0764
16	20	0.1032	0.0748	19	20	0.1051	0.0764
17	4	0.1029	0.0746	20	4	0.1048	0.0762
17	8	0.1047	0.0761	20	8	0.1048	0.0762
17	12	0.1047	0.0761	20	12	0.1048	0.0762
17	16	0.1047	0.0761	20	16	0.1048	0.0762
17	20	0.1047	0.0761	20	20	0.1048	0.0762

Table 5.8 Out-of-sample performance of EANN on weekly normalized stock returns: effects of input nodes

Input	RMSE	MAE	Input	RMSE	MAE
1	0.0831	0.0621	11	0.0941	0.0706
2	0.0899	0.0662	12	0.0926	0.0696
3	0.0901	0.0670	13	0.0939	0.0702
4	0.0925	0.0678	14	0.0939	0.0699
5	0.0931	0.0697	15	0.0933	0.0696
6	0.0915	0.0685	16	0.0932	0.0700
7	0.0916	0.0683	17	0.0958	0.0725
8	0.0940	0.0685	18	0.0977	0.0742
9	0.0909	0.0678	19	0.0947	0.0720
10	0.0926	0.0694	20	0.0944	0.0715

Table 5.9 Out-of-sample performance of EANN on weekly normalized stock returns: effects of hidden nodes

Input	Hidden	RMSE	MAE	Input	Hidden	RMSE	MAE
1	4	0.0835	0.0622	8	4	0.0860	0.0633
1	8	0.0845	0.0626	8	8	0.0883	0.0651
1	12	0.0856	0.0632	8	12	0.1022	0.0757
1	16	0.0815	0.0612	8	16	0.0919	0.0671
1	20	0.0806	0.0608	8	20	0.1018	0.0716
2	4	0.0851	0.0628	9	4	0.0862	0.0639
2	8	0.0899	0.0663	9	8	0.0917	0.0681
2	12	0.0903	0.0667	9	12	0.0893	0.0666
2	16	0.0919	0.0672	9	16	0.0934	0.0699
2	20	0.0924	0.0681	9	20	0.0943	0.0709
3	4	0.0863	0.0642	10	4	0.0861	0.0640
3	8	0.0859	0.0640	10	8	0.0895	0.0667
3	12	0.0884	0.0666	10	12	0.0943	0.0708
3	16	0.0938	0.0696	10	16	0.0958	0.0714
3	20	0.0965	0.0707	10	20	0.0973	0.0743
4	4	0.0862	0.0639	11	4	0.0878	0.0655
4	8	0.0898	0.0659	11	8	0.0912	0.0684
4	12	0.0976	0.0706	11	12	0.0917	0.0693
4	16	0.0924	0.0674	11	16	0.0970	0.0744
4	20	0.0969	0.0713	11	20	0.1028	0.0757
5	4	0.0892	0.0661	12	4	0.0877	0.0656
5	8	0.0887	0.0657	12	8	0.0912	0.0679
5	12	0.0905	0.0684	12	12	0.0942	0.0714
5	16	0.1034	0.0759	12	16	0.0940	0.0714
5	20	0.0939	0.0724	12	20	0.0963	0.0721
6	4	0.0853	0.0629	13	4	0.0881	0.0644
6	8	0.0882	0.0667	13	8	0.0906	0.0677
6	12	0.0936	0.0698	13	12	0.0966	0.0721
6	16	0.0936	0.0704	13	16	0.0974	0.0731
6	20	0.0971	0.0731	13	20	0.0970	0.0737
7	4	0.0865	0.0642	14	4	0.0884	0.0655
7	8	0.0869	0.0646	14	8	0.0918	0.0685
7	12	0.0913	0.0686	14	12	0.0944	0.0701
7	16	0.0919	0.0683	14	16	0.0942	0.0696
7	20	0.1018	0.0760	14	20	0.1010	0.0759

continued on next page

continued from previous page

Input	Hidden	RMSE	MAE	Input	Hidden	RMSE	MAE
15	**4**	0.0869	0.0647	**18**	**4**	0.0880	0.0657
15	**8**	0.0910	0.0665	18	**8**	0.0957	0.0731
15	12	0.0938	0.0690	18	**12**	0.0983	0.0743
15	16	0.0981	0.0751	18	**16**	0.1011	0.0765
15	20	0.0970	0.0728	18	**20**	0.1056	0.0815
16	4	0.0858	0.0632	19	**4**	0.0877	0.0655
16	8	0.0903	0.0666	19	**8**	0.0919	0.0694
16	12	0.0946	0.0715	19	**12**	0.0936	0.0713
16	16	0.0957	0.0731	19	**16**	0.0982	0.0752
16	20	0.0998	0.0758	19	**20**	0.1021	0.0790
17	4	0.0888	0.0666	20	**4**	0.0872	0.0647
17	8	0.0907	0.0680	20	**8**	0.0923	0.0698
17	12	0.0966	0.0738	20	**12**	0.0922	0.0700
17	16	0.1046	0.0811	20	**16**	0.1018	0.0777
17	20	0.0987	0.0733	20	20	0.0985	0.0756

Table 5.10 Out-of-sample performance of ANN, BANN, EANN. LAR, and RW on weekly stock return series

	ANN	BANN	EANN	LAR	RW
RMSE	1.4179	1.3627	1.2261	1.4183	1.8220
MAE	1.0482	0.9950	0.7843	1.0706	1.3634
MAPE					
MAD	0.8485	0.8313	0.8265	0.9165	1.0223
CORR	0.0305	0.0986	0.4347	-0.0113	0.0986
DA	0.7272	0.7631	0.7316	0.7532	0
SIGN	0.4155	0.5324	0.6103	0.5194	0.5844

Figure 5.1 Early stopping: Weekly exchange rate returns

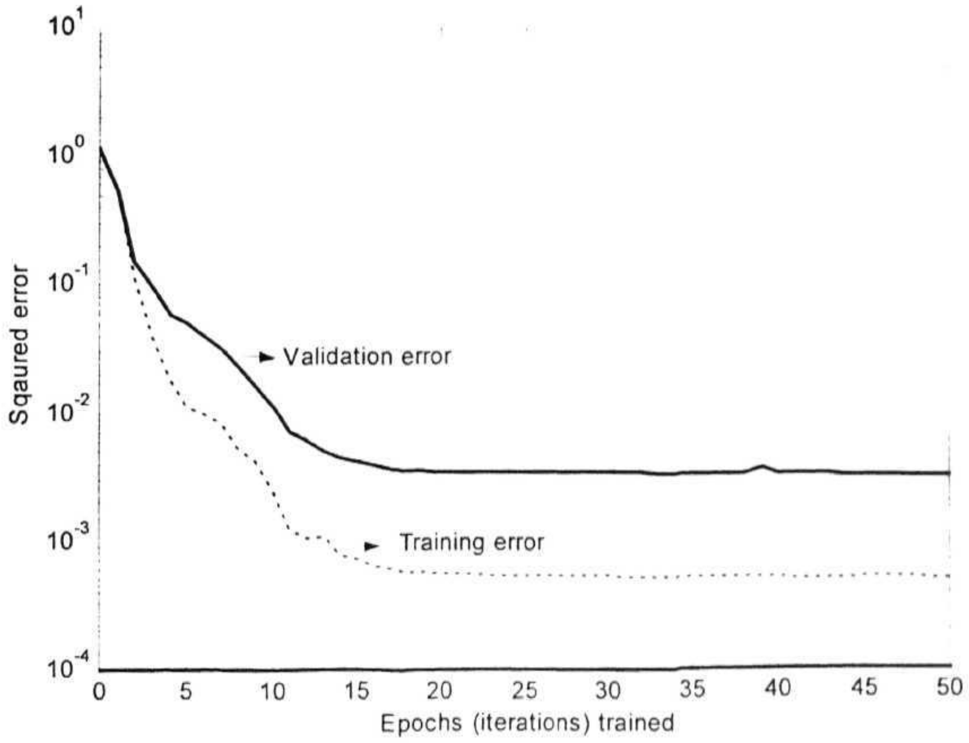
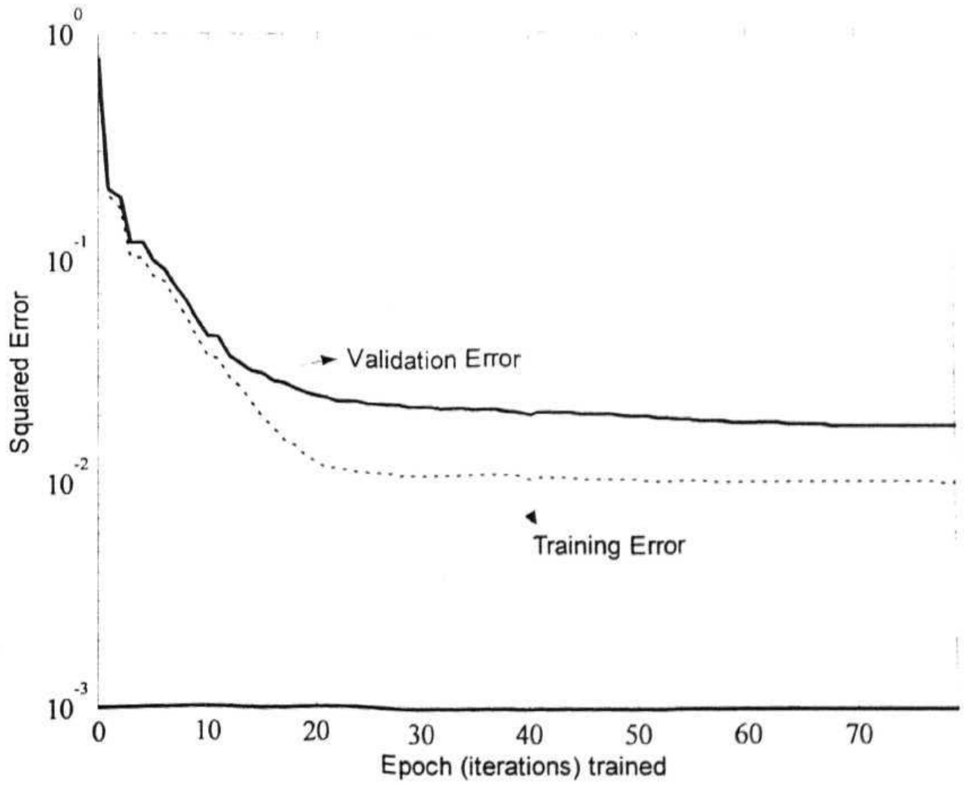


Figure 5.2 Early stopping: Weekly stock returns



Chapter VI

Summary, Implications and Scope for Further Research

A religious sect is predicting that the world will end at 10 P.M tonight. For more details, watch the news at eleven

Forecasting exchange rate and stock returns is rather a hazardous operation as these variables are notorious for their unpredictability. These variables are often characterized by high volatility, complexity, and noise. It is widely agreed that both exchange rate and stock returns are conditionally heteroskedastic, unconditionally leptokurtic, and nonlinear without having a specific nonlinear pattern. Finally, it is believed by some researchers that both foreign exchange rate and stock markets are weakly efficient i.e. they follow a random walk. As a result, linear and parametric nonlinear forecasting methods find it difficult to trace the movements of exchange rate and stock returns.

In this thesis, our concern has been to use a nonparametric and nonlinear model called artificial neural network to model and forecast the exchange rate and stock returns. We also have compared the performance of neural network with two benchmark forecasting models, linear autoregressive and random walk in the prediction of exchange rate and stock returns. After noting, in chapter I, the peculiarities of exchange rate and stock returns, we go on to discuss the artificial neural network in detail in chapter II. We also spell out the linear autoregressive and random walk model in brief in this chapter. This is followed up by a select review of the literature on the application of neural network to forecast exchange rate and stock returns. The general picture obtained from the review suggests that the neural network technique is superior to other conventional forecasting techniques in exchange rate and stock returns prediction. With the motivation provided by these findings, we take up the task of applying neural network to the one-step-ahead prediction of exchange rate and stock returns, starting from chapter III. We have used eight performance measures, root mean square error, mean absolute error, mean absolute percentage error, median absolute deviation, Pearson correlation coefficient, goodness of fit, direction accuracy and sign predictions for the comparison.

We first look at both daily and weekly exchange rate returns prediction in chapter III. In case of daily exchange rate returns, we find that neural network gives better in-sample prediction than both linear autoregressive and random walk model. However, in out-of-sample prediction, while neural network is better than random walk, the results are mixed with respect to linear autoregressive model. There is no clear winner between the two. These findings appear consistent with those of Kuan and Liu (1995). Linear autoregressive model is found to outperform random walk model in both in-sample and out-of-sample forecasting of daily exchange rate returns.

We also look at forecast horizon effects on the performances of neural network, linear autoregressive and random walk models under 1 month, 3 months, 6 months and 12 months forecast horizon using root mean square error and sign prediction as the performance measures. It is found that neural network performs better than random walk models under all four forecast horizons in terms of both measures in both in-sample and out-of-sample forecasting of daily exchange rate returns. As far as linear autoregressive model is concerned, neural network outshines it in in-sample prediction in all forecast horizons with respect to both measures. But linear autoregressive model outperforms neural network in out-of-sample prediction in all horizons with respect to root mean square error. However, neural network outperforms linear autoregressive model with respect to sign prediction. It is also found that as length of forecast horizon increases neural network's performance in out-of-sample forecasting in terms of RMSE increases and in terms sign prediction it decreases.

As far as forecasting of weekly exchange rate returns is concerned, our findings suggest that neural network has superior in-sample forecast than linear autoregressive and random walk models. As far as the out-of-sample forecasting is concerned, neural network outperforms random walk by all evaluation criteria, except for Pearson correlation coefficient. Neural network is also found to beat linear autoregressive model by four out of six evaluation criteria in out-of-sample forecasting. We find that neural network gives superior in-sample forecasting in terms of RMSE as compared to linear autoregressive and random walk models under short as well as long forecast horizons. As far as the in-sample sign prediction is concerned neural network outperforms random walk in all forecast horizons and outshines linear autoregressive model in long forecast horizon. Here, contrary to results with daily exchange rate return, the findings suggest that neural network's out-of-sample performance, in terms of RMSE, decreases as the length of horizon increases but increases as the length of forecast horizon increases in terms of sign prediction. It is also found that neural network and linear autoregressive models are outperformed by random walk model in short forecast horizon in out-of-sample forecasting of weekly exchange rate returns. But as the length of forecast horizon increases, they start outperforming random walk model. However, the sign predictions of neural network and linear autoregressive models are equal in short forecast horizon and mixed in long forecast horizon.

We take up the task of predicting daily and weekly stock returns using the three studied alternative models in chapter IV. The findings suggest that neural network gives better in-sample forecasting of daily stock returns than linear autoregressive and random walk models by all performance measures. Linear autoregressive model also outperforms random walk model by all performance measures in in-sample forecasting. Neural network is found to have better generalizing capability, doing better in out-of-sample forecasting of daily stock returns than both linear autoregressive and random walk models. Linear autoregressive model outperform random walk by most of the performance measures.

As far as the effects of horizon are concerned the results convey that irrespective of forecast horizons, neural network performs better than linear autoregressive and random walk models in terms of RMSE and sign predictions in in-sample forecasting of daily stock returns. The neural network's performance improves in terms of RMSE and sign predictions as forecast horizon extends. We find that the

out-of-sample performance of **neural network in terms** of **RMSE gets better** as **the length** of forecast horizon increases. However, network's out-of-sample performance becomes worse in terms of sign prediction as the forecast horizon increases. In addition, it is also found that neural network gives better in-sample and out-of-sample predictions in terms of RMSE and sign prediction than linear autoregressive and random walk models under all forecast horizons.

In weekly forecasting of stock returns, neural network has better in-sample and out-of-sample predictions than linear autoregressive and random walk models. The results show that the neural network's in-sample performance gets worse as the forecast horizons increase in terms of both RMSE and sign predictions. Similarly, neural network gives better out-of-sample sign prediction in the short run than in the long run. On the other hand, it is found that its out-of-sample performance improves in terms of RMSE as the length of forecast horizon increases. The findings also suggest that neural network gives better out-of-sample forecasting of weekly stock returns, in terms of RMSE, than random walk in longer horizons than shorter horizons. However, random walk is found to outperform neural network in all forecast horizons when sign prediction is considered as performance measure. **Linear** autoregressive performs better than neural network in out-of-sample forecasting in all forecasting horizons in terms of RMSE. On the other hand, neural network outperforms linear autoregressive model in short forecast horizon than long forecast horizon in terms of sign prediction.

There are two other points worth noting. The forecasting encompassing test, which tests for statistical significance of in-sample and out-of-sample results of **neural** network, linear autoregressive and random walk models was carried out for daily and weekly exchange rate and stock returns. The test shows that no model encompasses the other in both daily and weekly out-of-sample forecasting of exchange rate and stock returns. In addition to these findings, it is also found that neural network's performance is more sensitive to the number of input nodes than the number of hidden nodes both in case of daily and weekly exchange rate and stock returns prediction, This is consistent with the findings of Tang and Fishwick (1993), Zhang **and** Hu (1998) and Hu *et al* (1999).

One problem in neural network training is overfitting which in fact leads to poor out-of-sample forecasting. We address this problem in chapter V. In this chapter, we have tried to improve the out-of-sample performance of neural network in the forecasting of weekly exchange rate and stock returns by using Bayesian regularization and early stopping technique. Results show that both techniques improve out-of-sample performance of neural network, but the improvement is more in the case of weekly stock returns than in weekly exchange rate returns. Also, early stopping technique results in greater improvement in out-of-sample forecasting than Bayesian regularization. In addition to these findings, we also find that network with Bayesian regularization and early stopping technique outperform linear autoregressive and random walk models in the out-of-sample forecasting of weekly exchange rate and stock returns.

Implications of the Study

In a nutshell, this study has the following key implications. First, neural network is found superior to linear autoregressive and random walk models in forecasting exchange rate and stock returns. Further, linear autoregressive model is found to do better than random walk in most cases. From this, we can draw the inference that financial markets i.e. foreign exchange and stock markets do not follow the random walk. Thus, the weak form of efficient market hypothesis stands refuted. This opens up the possibility of extracting information from prices to explain the future behaviour of an asset. Second, superiority of neural network model in forecasting over linear autoregressive and random walk models is because of its flexibility to account for potentially complex nonlinear relationships not easily captured by traditional forecasting methods. Third, the findings suggest that in most cases neural network gives better out-of-sample short-term forecasts than both linear autoregressive and random walk models. This should clearly make neural network the preferred forecasting tool for investors and traders in the foreign exchange and stock markets because their buy-and-sell decisions are primarily based on short-term forecasts. Finally, we find neural network gives a better performance in the stock market than in the foreign exchange market. Thus, neural network can be more profitably used in the stock market than the foreign exchange market.

Limitations of the Study and Scope for Further Research

This work must be treated only as a starting point. The results are encouraging, but they provide only limited evidence supporting the usefulness of neural network models. First, we have limited our study to only the Indian rupee/US dollar exchange rate returns on the one hand, and BSE 30 stock returns on the other. Needless to say, we must exercise caution in generalizing to other exchange rate and stock returns. Second, the same can be said regarding the performance measures used. We have restricted our attention to only eight performance measures. The findings may change if other performance measures are used. Third, we have also considered only one-step-ahead forecasting. Multi-step-ahead forecasting could yield different results. Fourth, we have not used any other exogenous variable and have used only past returns as inputs or explanatory variables for neural network and linear autoregressive model. Finally, we have been limited by the software we have used. We have had to use fairly primitive software i.e. Matlab, since we did not have access to more specialized software on neural network. We hope to bring all this into our further research.

The other fruitful areas for further research are the following. First, is to combine methodology of linear models and neural networks. It is suspected that most time series contain a linear trend and a nonlinear component. Hence, a combination approach in which linear model captures linear patterns and neural network captures nonlinear patterns is expected to produce even better results than either linear or neural network model used singly. Second, in this thesis we have optimized the network architecture through systematic and rigorous experimentation. However, recently a new optimizing technique called genetic algorithm has come into increasing use to find the optimum network architecture. This we hope to do in our further research. Third, the robustness of neural networks to the changing structures, or turning points typically associated with exchange rates and stock prices can be investigated in further research by using multiple training and test samples systematically chosen from the original series. Fourth and finally, we have only used past returns as inputs to the network. However, technical trading rules can be profitably used in the set of inputs to the network to make more accurate forecast of' exchange rate and stock returns.

Bibliography

Abhyankar, A., L. S. Copeland and W. Wong (1997), "Uncovering nonlinear structure in real-time stock-market indices: The S&P 500, the DAX, the Nikkei 225, and the FTSE-100", *Journal of Business and Economic Statistics*, 15, 1-14.

Abu-Mostafa, Y. S. (1995), "Financial market applications of learning from hints", In: A.P. Refenes, (eds.), *Neural Networks in the Capital Markets*, John Wiley & Sons, Chichester, 221-232.

Anastasakis, L. and N. Mort (2000), "Neural network based prediction of the USD/GBP exchange rate: The utilization of data compression techniques for input dimension reduction", *From the Internet.*

Baestaens, D. E. and W. M. Van Den (1995), "Tracking the Amsterdam Stock Index using neural networks", In: A.P. Refenes, (eds.), *Neural Networks in the Capital Markets,* John Wiley & Sons, Chichester, 149-161.

Baillie, R. T. and P. C. McMahon (1989), *The foreign exchange market: Theory and econometric evidence*, Cambridge University Press, New York.

Barron, A. R. (1991), "Universal approximation bonds for superpositions of a sigmoidal function", *Technical Report No. 58,* Department of Statistics, University of Illinois, Urbana Champaign.

Bautista, B. (2001), "Predicting Philippine stock market using neural networks", *From the Internet.*

Boothe, P. and D. Glasssman (1987), "The statistical distribution of exchange rates", *Journal of International Economics,* 22, 297-319.

Bordoloi, S (2001), "Genetically engineered neural network: An application for predicting daily exchange rate", *37t Indian Econometric Conference.*

Borisov, A. N. and V. A. Pavlov (1995), "Prediction of a continuous function with the aid of neural networks", *Automatic Control and Computer Sciences,* 29(5), 39-50.

Box, G. E. P. and G. M. Jenkins (1976), *Time Series Analysis: Forecasting and Control,* revised edition, San Francisco: Holden-Day.

Brock, W. A., W. D. Dechert, J. Scheinkman and B. LeBaron (1995), "A test for independence based on the correlation dimension", *Econometric Reviews,* 15(3), 197-235.

Brooks, C. (1996), "Testing for non-linearity in daily sterling exchange rates", *Appl. Finance. Econ.*, 6, 307-317.

Campbell, J. Y. (1987), "Stock returns and the term structure", *Journal of Financial Economics,* 18, 373-399.

Chen, N. F., R. Roll and S. A. Ross (1986), "Economic forces and stock market", *Journal of Business*, 59, 383-403.

Chiang, W. C. and G. W. Baldridge (1995), "A neural network approach to mutual fund net asset value forecasting", *Omega,* 205-214.

Chinn, M. D. (1991), "Some linear and nonlinear thoughts on exchange rates", *Journal of International Money and Finance,* 10, 214-230.

Chong, Y. Y. and D. F. Hendry (1986), "Econometric evaluation of linear macroeconomic models", *Review of Economic Studies,* 53, 671-690.

Chuah, K. L. (1993), "A nonlinear approach to return predictability in the securities markets using feed forward neural network", *Dissertation,* Washington State University.

Clark, P. K. (1973), "A subordinated stochastic process model with finite variance for speculative prices", *Econometrica*, 41, 135-155.

Cogger, K. O., P. D. Koch and D. M. Lander (1997), "A neural network approach to forecasting volatility in international equity markets", *Advances in Financial Economies*, volume 3, Hirschey, Mark Marr, M. Wayne, eds., Greenwich, Conn and London: J AI Press, 117-157.

Collins, E., S. Ghosh and C. Scofield (1988), "An application of a multiple neural-network system to emulation of mortgage underwriting judgments", *Proc. IEEE International Conference of Neural Networks,* 2, 459-466.

Cumby, R., and M. Obstfeld (1984), "International interest-rate and price-level linkages under flexible exchange rate: A review of recent evidence" In: Bilson, J.F.O., R. Marston, (eds.), *Exchange Rate Theory and Practice,* University of Chicago Press, Chicago, 121-151.

Cybenko, G. (1989), "Approximations by superpositions of a sigmoidal function" *Mathematics of Control, Signals, and Systems,* 2, 303-314.

De Grauwe, P., H. Dewachter and M. Embrechts (1993), *Exchange Rate Theory: Chaotic Models of Foreign Exchange Markets,* Blackwell.

Denoeux, T. and R. Lengell (1993), "Initializing backpropagation networks with prototypes", *Neural Networks,* 6, 351-363.

Diebold, F.X. (1988), *Empirical modeling of exchange rate dynamics,* Springer-Verlag, New York.

Diebold, F.X. and Mariano, R.S. (1995), "Comparing predictive accuracy", *Journal of Business and Economic Statistics,* 13, 253-263.

Diebold, F.X. and M. Nerlove (1989), "The dynamics of exchange rate volatility: A multivariate latent-factor ARCH model", *Journal of Applied Econometrics,* 4, 1-22.

Dicky, D. A. and W. A. Fuller (1979), "Distribution of the estimators for autoregressive times series with a unit root", *Journal of the American Statistical Association,* 74, 427-431.

Diebold, F. X. and J. A. Nason (1990), "Nonparametric exchange rate prediction?", *Journal of International Economics,* 28, 315-332.

Domowitz, I. and C. S. Hakkio (1985) "Conditional variance and the risk premium in the foreign exchange market", *Journal of International Economics,* 19, 47-66.

Donaldson, R. Glen and M. Kamstra (1996), "Forecast combining with neural networks", *Journal of Forecasting,* 15,49-61

Donaldson, R. Glen and M. Kamstra (1997), "An artificial neural network-GARCH model for international stock return volatility", *Journal of Empirical Finance,* 4, 17-46.

Drunat, J., G. Dufrenot, C. Dunis and L. Mathiew (1996), "Stochastic or chaotic dynamics in high frequency exchange rates?", In: C. Dunis, (eds.), *Forecasting Financial Markets,* J. Wiley & Sons.

Dutta, S. and S. Shekhar (1988), "Bond rating: A non-conservative application of neural networks", *Proc. IEEE International Conference of Neural Networks,* 2, 443-450.

Elman, J. L. (1990), "Finding structure in time", *Cognitive Science,* 14,179-211.

Embrechts, M. J. (1995), "Forecasting foreign exchange rates with artificial neural networks" *From the Internet.*

Engel, R. F., T. Ito and W. Lin (1990), "Meteor showers or heat waves? Heteroskedastic daily volatility in the foreign exchange market", *Econometrica,* 58, 525-542.

Engel, C. (1991), "Can the Markov switching model forecast exchange rates", *Working Paper,* University of Washington.

Engel, C. and J. Hamilton (1990), "Long swings in the exchange rates: Are they in the data and do markets know it?" *American Economic Review,* 80, 689-713.

Episcopos, A. and J. Davis (1995), "Predicting returns on Canadian exchange rates with artificial neural networks and EGARCH-M models", In: A.P.N. Refenes, Y. Abu-Mostafa, J. Moody and A. Weigend, (eds.), *Neural Networks Financial Engineering, Proceedings of the third International Conference on Neural Networks in the Capital Markets,* Singapore, London, World Scientific, 135-145.

Fahlman, S. E. and C. Lebiere (1990), "The cascade-correlation learning algorithm", In: D. S. Touretzky, (eds.), *Advances in Neural Information Processing Systems,* 2, Morgan Kaufmann, San Mateo, CA, 525-532.

Fama, E. F. and F. R. French, (1988), "Dividend yields and expected stock returns", *Journal of Financial Economics,* 22, 3-25.

Fama, E. F. and F. R. French, (1989), "Business conditions and expected returns on stocks and bonds", *Journal of Financial Economics,* 25, 23-49.

Feige, E. L. and D. K. Pearce (1976), "Economically rational expectations: Are innovations in the rate of inflation independent of innovations in measures of monetary and fiscal policy?" *Journal of Political Economy,* 84, 499-522.

Gallant, S. I. (1986), "Three constructive algorithms for neural learning", *Proc. 8th Annual Conference of Cognitive Science Society.*

Gencay, R. (1999), "Linear, non-linear, and essential foreign exchange rate prediction with simple technical trading rules", *Journal of International Economics,* 47, 91-107.

Geman, S., E. Bienenstock and R. Doursat (1992), "Neural networks and the bias/variance dilemma ", *Neural Computation,* 4, 1, 1-58.

Gracia, R. and R. Gencay (2000), "Pricing and hedging derivative securities with neural networks and a homogeneity hint", *Journal of Econometrics,* 94 (1 -2), 93-115.

Gradojevic, N and Jing Yang (2000), "The application of artificial neural networks to exchange rate forecasting: The role of microstructure variables", *Working Paper,* Bank of Canada.

Gray, M., C. Coen and K. Frost (2000), "Mutual fund net asset value forecasting using neural networks", *From Internet.*

Grudnitski, G. and L. Osburn (1993), "Forecasting S&P and gold futures prices: An application of neural networks", *Journal of Futures Markets,* 13, 631 -643.

Haefke, C. and C. Helmenstein (1994), "Stock price forecasting of Austrian initial public offerings using artificial neural networks", *Proc. Neural Networks Capital Markets.*

Haefke, C. and C. Helmenstein (1995), "Predicting stock market averages to enhance profitable trading strategies", Proc. *Neural Networks Capital Markets.*

Hann, T. H. and E. Steurer (1996), "Much ado about nothing? Exchange rate forecasting: Neural networks vs. linear models using monthly and weekly data", *Neurocomputing,* 10,323-339.

Hecht-Nielsen, R. (1987), "Kolmogorov's mapping neural network existence theorem", *Proc. IEEE Ist Ineternational Conference on Neural Networks,* June, San Diego, 3, 11-14.

Hinich, M. J. and D.M. Patterson (1985), "Evidence of nonlinearity in daily stock returns" *Journal of Business and Economic Statistics,* 3, 69-77.

Hinton, G. E. (1987), "Connectionist learning procedures", *Technical Report,* Computer Science Department, Carnegie-Mellon University, June.

Hoptroff, R. G. (1993), "The principles and practice of time series forecasting and business modeling using neural nets", *Neural Computing and Applications,* 1, 59-66.

Hornik, K., M. Stinnchcombe and H. White (1989), "Multi-layer feed forward networks are universal approximators", *Neural networks, 2,* 359-366.

Hsieh, D. A. (1988), "The statistical properties of daily foreign exchange rates: 1974-1983", *Journal of International Economics,* 24, 129-145.

Hsieh, D. A. (1989), "Testing for nonlinear dependence in daily foreign exchange rates", *Journal of Business,* 62, 329-368.

Hu, M. Y., G, Zhang, C. Y. Jiang, and B. E. Patuwo (1999), "A cross validation analysis of neural networks out-of-sample performance in exchange rate forecasting", *Decision Sciences,* 30(1), 197-216.

Hurst, H. E. (1951), Long term storage of reservoirs, *Transactions of the American Society of Civil Engineers,* pp. 116.

Hutchinson, J., A. Lo and T. Poggio (1994), "A nonparametric approach to pricing and hedging derivative securities via learning networks", *Journal of Finance,* 99, 851-889.

Ingber, L. (1996), "Statistical mechanics of non-linear nonequilibrium financial markets: Applications to optimized trading", *Mathematical Computer Modeling.*

Jordan, M. (1986), "Serial order: A parallel distributed processing approach", *ICS Report 8604,* University of California San Diego, Institute for cognitive science.

Jordan, M. (1992), "Constrained supervised learning", *J. Math. Psychol,* 36, 396-425.

Kaastra, I. and M. S. Boyd (1995), "Forecasting futures trading volume using neural networks", *Journal of Futures Markets,* 15. 953-970.

Kamath, M. V. (2001), "Empirical investigation of multifactor asset pricing model using artificial neural networks", *NSE Research Initiative*, Paper No. 10.

Kuan, C. and H. White (1994), "Artificial neural networks: An econometric perspective" *Econometric Reviews,* 13, 1 -91.

Kuan, C. and T. Liu (1995), "Forecasting exchange rates using feed forward and recurrent networks", *Journal of Applied Econometrics,* 10, 347-364.

Kuo, C. and A. Reitsch (1995), "Neural networks vs. conventional methods of forecasting", *The Journal of Business Forecasting,* Winter, 17-22.

Kurkova, V. (1991), "Kolmogorov's theorem is relevant", *Neural Computation,* 3(4), 617-622

Landt, F. W. (1997), Stock price prediction using neural networks, *Master Thesis,* Leiden University.

Lippmann, R. P. (1987), "An introduction to computing with neural nets", *IEEE ASSP Magazine,* April, 4-22.

Ljung, G. and G. Box (1979), "On a measure of lack of fit in time series models", *Biometrika*, 66, 265-270.

Lo, W. A. and A. Craig MacKinlay, (1999), *A Non-Random Walk Down Wall Street,* Princeton University Press.

MacKay, D. J. C. (1992), "Bayesian interpolation", *Neural Computation,* 4, 415-447.

Malkiel, B. G. (1996), *A Random Walk Down Wall Street,* Norton, New York, NY.

Mandelbrot, B., (1963), "The variation of certain speculative prices", *Journal of Business,* 36,394-419.

Medeiros, Alvaro Veiga and Carlos E. Pedreira (2001), "Modeling exchange rates: Smooth transitions, neural networks, and linear models ", *IEEE Transactions on Neural Networks,* Special Issue on Financial Engineering, 1-32.

Meese, R.A. and K. Rogoff (1983a), Empirical exchange rate models of the seventies: Do they fit out of sample?, *Journal of international Economics,* 14, pp. 3-24.

Meese, R.A., and K. Rogoff (1983b), "The out-of-sample failure of empirical exchange rate models: Sampling error or misspecification?, In: J. Frenkel, (eds.), *Exchange rates and International economics* (University of Chicago Press, Chicago, 1L).

Meese, R.A., and K. Rogoff (1988), Was it real? The exchange rate-interest differential relation over the modern-floating rate period, *Journal of finance,* 43, pp. 933-947.

Meese, R. A. and A. K. Rose (1990), "Nonlinear, nonparametric, nonessential exchange rate estimation", *The American Economic Review,* 80, 678- 91.

Meese, R. A. and A. K. Rose (1991), "An empirical assessment of non-linearities in models of exchange rate determination", *Review of Economic Studies*, 80, 603-619.

Mezard, M. and J. Nadal (1989), "Learning in feedforward layered network: The tiling algorithm", *Journal of Physics,* 22, 2191-2203.

Morris, R. J. T., L. D. Rubin and H. Tirri (1990), "Neural network techniques for object orientation; detection: Solution by optimal feedforward network and learning vector quantization approaches", *IEEE Transaction on PAMI*, 12, 1107-1115.

Nag, A., and A. Mitra (2002), "Time series modeling with genetic neural networks: Case studies of some important Indian economic and financial series", *Statistics and Applications,* 4, 37-57.

Nguyen, D. and B. Widrow (1990), "Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights", *Proc. of the International Joint Conference Neural Networks,* 3, 21 -26.

Pesaran, M. H. and A. Timmermann (1992), "A simple nonparametric test of predictive performance", *Journal of Business and Economic Statistics,* 10, 461-465.

Pesaran, M. H. and A. Timmermann (1994), "Forecasting stock returns. An approximation of stock market trading in the presence of transaction costs", *Journal of Forecasting,* 13,335-367.

Plasmans, J., W. Verkooijen and H. Daniels (1998), "Estimating structural exchange rate models by artificial neural networks", *Applied Financial Economics,* 8, 541-51.

Poli, L. and R. D. Jones (1994), "A neural net model for prediction", *Journal of American Statistical Association,* 89, 117-121.

Qi, M. and G. S. Maddala (1999), "Economic factors and the stock market: A new perspective", *Journal of Forecasting,* 18, 151-166.

Refenes, A.-P. and S. Vithlani (1991), "Constructive learning by specialization", *Proc. International Conference of Artificial Neural Networks,* Helsinki, Finland.

Refenes, A.-P. (1993), "Constructive learning and its application to currency exchange rate forecasting", In: R. Trippi and E. Turban, (eds.), *Neural Networks in Finance and Investing,* Irwin Professional Publishing, New York.

Refenes, A.-P. (1995), eds., *Neural Networks in the Capital Markets*, John Wiley & Sons, Chichester.

Refenes, A.-P., A. D. Zapranis and G. Francis (1994), "Stock performance modeling using neural networks: A comparative study with regression models", *Neural Networks,* 7, 375-388.

Refenes, A.-P., A. D. Zapranis and G. Francis (1995), "Modeling stock returns in the framework of APT: A comparative study with regression models", In: A.-P. Refenes, (eds.), *Neural Networks in the Capital Markets,* John Wiley & Sons, Chichester, 101-125.

Rumelhart, D.E., G.E. Hinton and R.J. Williams (1986), "Learning representations by backpropagation errors", *Nature,* 323, pp. 533-536.

Schalkoff, R.J. (1997), *Artificial Neural Networks*, McGraw-Hill International Editions.

Schoneburg, E. (1990), "Stock price prediction using neural networks: A project report", *Neurocomputing,* 2, 17.

Shachmurove, Y. and D. Witkowska (2000), "Utilizing artificial neural network model to predict stock markets", *CARESS Working Paper, #* 00-11.

Sietsma, J. and R. F. J. Dow (1991), "Creating artificial neural networks that generalize", *Neural networks, A,* 67-79.

Steiner, M. and H. S. Wittkemper (1995), "Neural networks as an alternative stock market model", In: A.-P. Refenes, (eds.), *Neural Networks in the Capital Markets,* John Wiley & Sons, Chichester, 137-147.

Tang, Z. and P. A. Fishwick (1993), "Feedforward neural nets as models for time series forecasting", *ORSA Journal on Computing,* 5, pp. 374-385.

Taylor, S. J. (1980), "Conjectured models for trends in financial prices, tests and forecasts", *Journal of Royal Statistical Society, A,* 143, 338-362.

Taylor, S. J. (1982), "Tests of the random walk hypothesis against a price trend hypothesis", *Journal of Financial and Quantitative Analysis,* 17, 37-61.

Taylor, S. J. (1994), *Modeling Financial time series,* J. Wiley & Sons, Chichester.

Trippi, R. and E. Turban (1993), eds., *Neural Networks in Finance and Investing*, **Probus Publishing Company.**

Trippi, **R. R. and D.** Desieno (1992), "Trading equity index futures with a neural network", *Journal of Portfolio Management,* 19, 27-33.

Tsibouris, G. C. (1993), Essays on nonlinear models of foreign exchange, *Dissertation,* University of Wisconsin-Madison.

Tsibouris, G. and M. Zeidenberg (1995), "Testing the efficient markets hypothesis with gradient descent algorithms", In: A.-P. Refenes, (eds.), *Neural Networks in the Capital Markets,* John Wiley & Sons, Chichester, 127-136.

Verkooijen, W. (1996), "A neural network approach to long-run exchange rate prediction", *Computational Economics, 9,* 51-65.

Wasserman, P. D. (1989), *Neural Computing: Theory and Practice*, Van Nostrand Reinhold: New York.

Westerfield, J.M. (1977), "An examination of foreign exchange risk under fixed and floating rate regimes", *Journal of International Economics*, 7, 181 -200.

White, H. (1989a), "Learning in artificial neural networks: A statistical perspective", *Neural Computation,* 1, 425-464.

White, H. (1989b), "Some asymptotic results for learning in single hidden-layer feed forward network models", *Journal of American Statistical Association,* 84, 1003-1013.

White, H. (1989c), "Economic prediction using neural networks: The case of IBM daily stock returns", *Proc. IEEE Internal. Conf. Neural Networks.*

White, H. (1990), "Connectionist nonparametric regression: Multilayer feedforward networks can learn arbitrary mappings", *Neural Networks,* 3, 535-549.

White, H. (1992), Artificial Neural Networks: Approximation and Learning Theory, Oxford: Blackwell.

White, H. (1996), "Option pricing in modern finance theory and the relevance of artificial neural networks", *Discussion Paper,* Econometrics Workshop.

Whitelaw., R. F. (1994), "Time variations and covariations in the expectation and volatility of stock market returns", *Journal of Finance,* 49, 515-541.

Wu, B. (1995), "Model-free forecasting for non-linear time series (with application to exchange rates)", *Computational Statistics & Data Analysis,* 19, 433-459.

Wurtz, D. and C. De Groot (1992), "Nonlinear time series analysis with connectionist nets: towards a robust methodology", *Proceedings of the SPIE Conference on Applications of Artificial Neural Networks.,* Orlando.

Yao, J., H.L. Poh. and T. Jasic (2000), "Foreign exchange rates forecasting with neural networks", *From the Internet.*

Yoon, Y. and G. Swales (1990) "Predicting stock price performance", *Proceeding of the 24th Hawaii International Conference on System Sciences,* 4,156-162.

Zhang, G. and M. Y. Hu (1998), Neural network forecasting of the British pound / US dollar exchange rate", *International Journal of Management Science,* 26(4), 495-506.