Forecasting of Foreign Exchange Rate Time Series Using Neural Networks

A thesis submitted towards partial fulfillment of

BS-MS duel degree programme

by

SHASHANK AGRAWAL

Under the guidance of

DR. RAGHU NANDAN SENGUPTA

Associate Professor,

Department of Industrial and Management Engineering,

Indian Institute of Technology, Kanpur

Department of Physics

Indian Institute of Science Education and Research Pune

Certificate

This is to certify that this dissertation entitled "Forecasting of Foreign Exchange Rate Time Series Using Neural Networks" towards the partial fulfillment of the BS-MS duel degree programme at the Indian Institute of Science Education and Research Pune, represents original research carried out by Shashank Agrawal at Indian Institute of Technology, Kanpur under the supervision of Dr. Raghu Nandan Sengupta, Associate Professor, Department of Industrial & Management Engineering, IIT Kanpur during the academic year 2010-2011.

[illegible]	[illegible]
SHASHANK AGRAWAL	DR. RAGHU NANDAN SENGUPTA
Name and signature of the student	Thesis Supervisor
	Date: 07-04-2011
	Place: IIT Kanpur

Head (Biological/Chemical/Physical/Mathematical) Sciences

Date:

Place:

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I heartily feel a deep sense of gratitude towards my Thesis Guide Dr. Raghu Nandan Sengupta for his invaluable guidance and support at every stage of my thesis work. I feel highly indebted to him for his highly efficient supervision, valuable suggestions and encouragement, which inspired me for taking up the challenging task. I was very fortunate to have a supervisor like him.

I would like to take this opportunity to thank Prof. Ashu Jain without whom this project would not have been possible and whose immense support in innumerable ways always helped me find out my ways out of all my problems. I pay my sincere thanks to Dr. Vinay Gupta for his enlightening words in times of crisis. I would like to thank Mr. G. Ravikumar, who provided me constant support as an elder brother. There have been many friends who made my stay at IIT Kanpur a relishable and cherishable one. Very special thanks goes to Shanu Singla whose positivity in every situation and sparkling vibes always held me high spirited throughout. Great support from friends like Kamalkant, Nitin Tripathi, Rambhaj, Pradeep Pandey and others was of immense help. I am also thankful to Mrs. Savita Jain for her special concern throughout my stay. I would like to express my sincere thanks to many others not mentioned here and everyone who made my stay at IIT Kanpur a memorable one.

My acknowledgements would be incomplete without expressing my deepest gratitude to the Almighty and my parents for their constant blessings, love and affection.

Shashank Agrawal.

ABSTRACT

The prediction of stock market returns and prices and other variables have always been a topic of great interest to researchers and financial analysts since long as a large amount of capital is being traded all over the world and therefore predicting important time series like financial foreign exchange rate proves to be of immense help in systematizing and planning financial gain. Also, some theorists and researchers believe that the markets are efficient in themselves, they absorb every another new information coming up and thus there is no scope of prediction. This poses as a challenge and thus provides another motivation to try and devise mechanisms for market prediction to prove that the markets are not completely random and carry an element of predictability.

We also believe that the markets don’t follow complete random walks and can be modeled carefully and efficiently to get quite encouraging results. We have used Neural Networks for modeling, which are one of the very interesting machine learning techniques that can approximate nonlinear continuous functions without any priori information about the nature of the generating process i.e. the underlying data generating process. They are said to be the models following data driven approach because they perform nonlinear modeling without any knowledge of the relationship between the input and the output variables. A neural network is a system consisting of many simple units called neurons which are highly interconnected and are organized as layers. Each neuron performs the simple task of information processing by converting received inputs into processed outputs. These Neural Networks can perform wide variety of tasks and achieve remarkable results.

We have modeled the spot exchange rate time series of the daily data of about 10 years from 2000 to 2010 using neural networks which are trained using various algorithms. Training of the neural networks and finding the optimal set of weights and therefore, the optimal network to generate the best of results is not an easy task. We have used the Conjugate Gradient Method (CGM) to train the network and the algorithms like Genetic Algorithm and Simulated Annealing Algorithm to optimize the neural network

configuration in terms of weights. We have also used parametric complex statistical models like Auto-Regressive Moving Average (ARMA) coupled with Generalized Auto-Regressive Conditional Heteroskedasticity (GARCH) for forecasting the same time series data. The results obtained from various models are thereby compared. It has been observed that the neural networks trained with CGM and optimized with GA perform the best among all models.

CHAPTER 1

INTRODUCTION

In financial research, predicting stock market index and stock prices is the topic of great interest to people. First, the discoveries about the nature of the stock prices and returns are made and then, these developments lead to the design of prediction models for the stock prices. After the coming of the floating exchange regime, liberalization of trade and rapid expansion of global trade, these issues have come to a forefront. To construct models that are able enough to predict and explain reasonably enough the future value of the exchange rates is in the interest of strategic investors as well as the common man. Recently, the markets have become a more accessible investment tool for everyone. Thus, it is not only related to the macroeconomic parameters, but it also affects the everyday life in a more direct manner. There is no straightforward equation which decides the behavior of the stock market or the stock price. Thus, the characteristic of unpredictability is always associated with the stock markets and the exchange rates. People tend to extend this concept of unpredictability to declare that the markets are completely random and prediction about them is impossible. But it has been shown and proved that the markets do carry an element of predictability. The main motive of making such prediction models are thus obvious, first, the financial gain and then, to prove that the markets are not totally random but predictable to an extent.

1.1 Objective

The objectives of the thesis are as follows:

1. Forecasting using Neural Networks using Conjugate Gradient after pre-processing of the data to find the number of the inputs to be given on the basis of the autocorrelations.

2. Using some metaheuristic techniques for the optimization of the Neural Network Configuration and thus compare the different models formed.

1.2 Financial Time Series

An ordered sequence of the values of a variable at equally spaced time intervals is known to be a Time Series. Now, a time series may have can have identifiable stochastic or deterministic components. A stochastic component is one in which each data value of the series can be considered as a sample mean of a probability distribution of an underlying population at each point in time. Whereas a deterministic time series is one which is not driven by stochastic process but by some predefined laws of which the corresponding time series is the data. Some examples of stochastic time series are foreign exchange rate, rainfall data, stock prices, earthquake data and Gross Domestic Product (GDP). Some examples of deterministic time series are AC single phase voltage across household mains, seasonal flooding of Nile river data.

Our study concerns with the stochastic type of time series of Foreign Spot Exchange Rates. Foreign Exchange Rate is the ratio of the currencies of two countries. It is a very important factor in understanding the dynamics of the trading of goods in the import-export markets as well as in the exchange markets, as it expresses the currency of one country in terms of another. These foreign exchange rates are determined and influenced by a large variety of factors. For any given currency, time is one of the most important influencing factors in determining the foreign exchange rate. The other factors i.e. various economic factors have differing influence on the foreign exchange rate differing from country to country. Some of the factors determining the foreign exchange rate movement are relative growth of the economy of one country with respect to that of the other country, inflation differential, equity flow, and market volatility. The spot exchange rate of United States Dollar vs. Australian Dollar (US$/AU$) which we have also used as one of our time series data in our analysis is shown below in Figure 1.1 as it varies with time over a period of 10 years from January 2000 to December 2010.

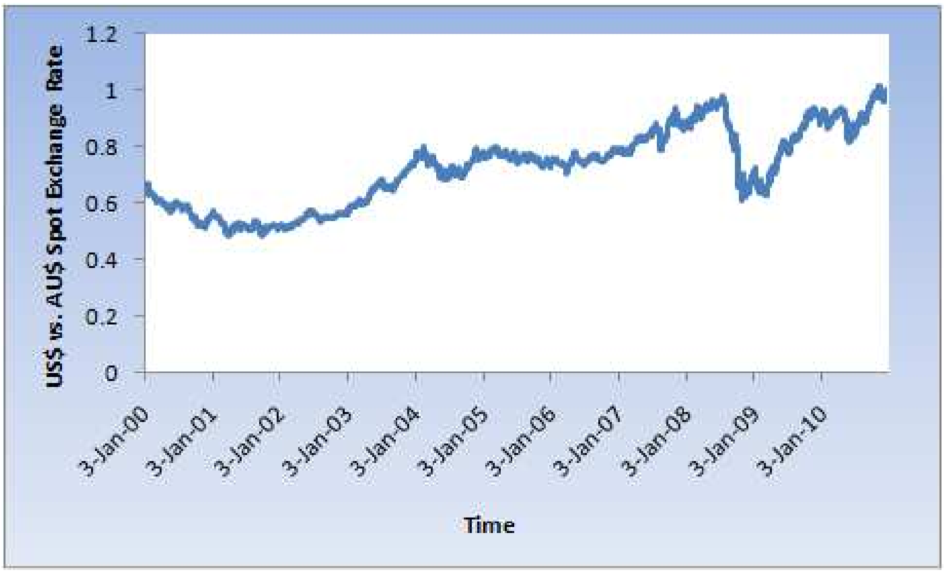

Figure 1.1: US$/AU$ Spot Exchange Rate from 03/01/2000 to 24/12/2010

1.3 Stationarity of time series

Time Series may be stationary or non-stationary. A time series r_t is said to be stationary if the joint distribution of (r_t,……….. r_k) is identical to that of (r_{t-1},……….. r_{k+1}) for all t, where k is an arbitrary positive integer. This means that for a series to be strictly stationary, the joint distribution must not vary with time or it is invariant under time shift. This means that the first and the second moments, i.e. the means, variances and covariances are constant with respect to time. Thus, the stationary time series are characterized by a kind of statistical equilibrium around a constant mean level as well as a constant dispersion around the mean level.

On the other hand, a non-stationary time series is one which does not exhibit this type of equilibrium and show random behavior. Also, the first and second moments are expected to change. The non-stationary time series are thus characterized by properties like random walk, drift, trend, changing variance, etc.

Fortunately, for most of the financial time series data such as the stock prices and Gross Domestic Product, logarithmic first-differencing usually transforms the non-stationary time series into stationary time series.

1.4 Research Background

The stock market prediction task divides the researchers into two belief groups. On one side, there are people who believe that the market is efficient and whenever some new information comes up, it absorbs it by correcting itself. That the market follows random walk, there is no room for prediction and the best prediction you can have about tomorrow's value is today's value. The other belief is that the markets are not completely random and we can devise mechanisms to predict it. However, due to the non-linear characteristics of the exchange rates, it has been generally difficult to model the financial forecasting. In the past, various researchers have used different methods for evaluating the dynamic behavior of the financial series and have claimed some success in being able to forecast the financial time series data.

1. **Technical Analysis** – Chartists or the technical analysts attempt to predict the market by tracing patterns that come from the study of the charts which describe the historic data of the market. Technical market analysis is based on market generated statistics and is used for the market data i.e., price movements (high, low, open, close, volume) for an individual exchange rate. This analysis cannot be used because it often trivializes market behavior by reducing it to two dimensional charts that are susceptible to subjective interpretation and incomplete analysis.

2. **Fundamental Analysis** – Fundamental Analysts study the intrinsic value of a stock and they invest on it if they estimate that its current value is lower than its intrinsic value. They study the effects of supply and demand of each currency. However, fundamental analysis is not capable of capturing the behavior of the nonlinear markets and the inherent complexity of inter-

market relationships. The models are limited to the financial analyst's ability to identify dominant factors which affect the supply and demand.

3. **Linear Time Series Forecasting** – In traditional time series forecasting, we try to create linear prediction models to trace patterns in historic data. Depending on whether they use one or more variables to approximate the stock market time series, they can be divided as the univariate and the multivariate regression models. The linear methods have long been the dominant technique for the analysis of economic and financial time series as they provide the ease of interpretation and simple computation. Auto-regressive (AR) models are useful in the prediction of foreign exchange rate i.e., financial time-series. However, traditional methods cannot be used to track the complexity of market behavior and the intricacies of economic theory. Traditional methods fail in regard to predicting financial time series because the dynamic exchange rates are found to be strongly nonlinear.

4. **Non-Linear Modeling** – Recently, the discovery of the nonlinearity in the financial markets have been largely emphasized by various researchers and financial analysts. There are many new potentially promising nonlinear methods and techniques introduced for prediction. Some of the nonlinear time series models developed are the bilinear model, the Threshold Auto-Regressive (TAR) model and the Auto-Regressive Conditional Heteroscedastic (ARCH) model. ARCH model was later extended to Generalized Auto-Regressive Conditional Heteroscedastic (GARCH). Even though a number of non-linear statistical techniques have been used to produce better predictions of the stock returns or prices, most techniques require that the nonlinear model be specified before the estimation of parameters can be determined. These techniques are better known as the model driven approaches. That is, an explicit relationship between the inputs and output variables for the data series at hand has to be hypothesized with little knowledge of the underlying law. In fact, the formulation of a

nonlinear model to a particular dataset is a very difficult task since many possible nonlinear patterns may not be captured by a pre-specified nonlinear model.

Neural Networks, as opposed to the above model based nonlinear methods; do not require a pre-specification during the modeling process because they can independently learn the relationships inherent in the variables. Neural Networks are thus said to be nonlinear data driven approaches. They are capable of performing nonlinear modeling without a priori knowledge about the relationships between input and output variables. Neural networks, therefore, are a more general and flexible modeling tool for nonlinear problem forecasting. As it continues to operate on the data, a properly constructed network can subsequent learn by itself. This is the most essential advantage of neural networks over other forecasting models.

Neural Networks have been used for forecasting since long. They have been equally and more extensively used for the forecasting of other types of stochastic data apart from the financial time series data like rainfall data, earthquake data. The first application dates back to 1964. Hu (1964), in his thesis, uses the Widrow's adaptive linear network for weather forecasting. Due to the insufficiency of a training algorithm for general multi-layer networks at the time, the research was quite limited. It was not until 1986 when the backpropagation algorithm was introduced (Rumelhart *et. al.* (1986)) that there had been much development in the use of neural network for forecasting. Werbos (1974, 1988) first formulated the backpropagation network and found that neural networks trained with backpropagation outperform the traditional statistical methods such as regression and Box-Jenkins approaches. Tang *et. al.* (1991), Sharda and Patil (1992), and Tang and Fishwick (1993) report results of several forecasting comparisons between Box-Jenkins and neural networks. Weigend *et. al.* (1990, 1992) and Cottrell *et. al.* (1995) addresses the issue of network structure for forecasting real-world time series. Lapedes and Farber (1987) concluded that neural networks can be used for modeling and forecasting of a nonlinear time series.

One of the many other abilities of a neural network is that it can generalize and "see through" noise and distortion and abstract essential characteristics in the presence of irrelevant data. The neural network model also provides a high degree of robustness and fault tolerance. In addition, the well-built model can find the right transformations for variables and can also represent complex and highly nonlinear relationships through independent variable data patterns. They have the potential to capture nonlinear properties of time series and they are non-parametric in nature. Hence, neural networks seem appropriate for forecasting because of their self-adaptive, automatic modeling properties.

1.5 Models employed for Exchange Rates

As discussed about neural networks in the previous section, their various characteristics of self-learning, non-linearity, adaptability, arbitrary function mapping ability, make them quite suitable and useful forecasting tools. However, the performance of neural networks is affected by many factors. Some of these factors are, the design of the neural network, the configuration of the network, the algorithm used to train the network and alike. There are different algorithms which can be used to optimize the neural network. Initially, in our work, we use the Conjugate Gradient Method (CGM) for the purpose of training the neural network. This training algorithm back propagates the changes needed to be incorporated to minimize the difference between the actual output and the desired output. Further, in our study, we utilize some metaheuristic techniques for the optimization of the neural network configuration in terms of weights and number of nodes. We have used two metaheuristic techniques, Genetic Algorithm (GA) and Simulated Annealing (SA). The results obtained after the employment of the metaheuristic techniques, when compared with those obtained from CGM show that GA outperforms SA and CGM. This comparison is done on the basis of different performance measures.

The different performance measures or metrics used to compare the performance of different models are (1) Mean Square Error (MSE), (2) Mean Absolute Error (MAE), (3) Direction Accuracy (DA), (4) Pearson Correlation Coefficient and (5) Theil's Inequality Coefficient. The results we have obtained are quite motivating and encouraging. For all of our different simulations, we have used MATLAB R009B as the programming tool for our whole analysis.

1.6 Thesis Organization

The thesis is organized as follows:

In Chapter 1, we have discussed and explained the problems in financial forecasting and the different methods utilized by people in the past in their quest to develop efficient predicting models for forecasting. Chapter 2 discusses about the structure of Neural Network, its parts and processing techniques, the learning rules and the training methods for a neural network configuration. Chapter 3 discusses in detail the two metaheuristic techniques used by us in our study namely Genetic Algorithm (GA) and Simulated Annealing (SA) while Chapter 4 describes the parametric model ARMA-GARCH used by us for better comparison of results. In Chapter 5, we have briefly described how the various models based on different techniques discussed work to produce prediction results. Chapter 6 shows the various steps followed by us for data analysis and then the results obtained by us from different models. Finally Chapter 7 concludes with a brief about our conclusions and the further future possibilities of our work.

CHAPTER 2
ARTIFICIAL NEURAL NETWORKS

2.1 An Overview

Artificial Neural Networks or simply neural networks are computing models for information processing and pattern identification. These are models based on data driven approaches and considered as a data processing technique that relates a set of inputs to a set of outputs. Artificial Neural Network technique is motivated by the way biological neural system works and it grew out of the research interest in modeling neural systems, especially human brains. It can be considered as a massively parallel distributed processor made up of simple processing units, which can be called as artificial neurons, and which carry the natural tendency to store experimental knowledge and making it available for later use. An artificial neuron is again a computational model inspired by the natural neuron. Neuron is known to be the simplest processing unit, receives and processes the signal from other neurons through its input paths known as the dendrites. If the combined signal is enough, i.e. it exceeds the threshold, it generates the output signal to its path called axon which splits up and connects to other neurons' input paths through a junction known a synapse. The amount of the signals transferred depends on the synaptic strength of the junction which is chemical in nature. This synaptic strength is modified during the learning process of the brain. Now, the mathematical representation of these biological processes is known to be the Artificial Neural Network (ANN). ANN resembles the brain in two respects: 1) The network acquires knowledge from its environment through a learning process and 2) To store the acquired knowledge, interneuron connection strengths, known as synaptic weights are used.

2.2 Processing Elements: The neural network consists of many simple computing units known as neurons or cells, which are highly interconnected and organized in layers. Each neuron performs the simple task of information processing by converting the received inputs into processed outputs. The output from the neuron is mapped to its inputs through a transfer function. Each path making the connection between the neurons has its weight which represents the strength of that path. These weights are modified in the process of network learning to find out the optimal set of weights giving the best output. A typical processing element or a neuron receiving two inputs is shown in Figure 2.1. It has two paths with weights W_1 and W_2. If the net input is higher than the threshold, then an output is obtained from the output path.

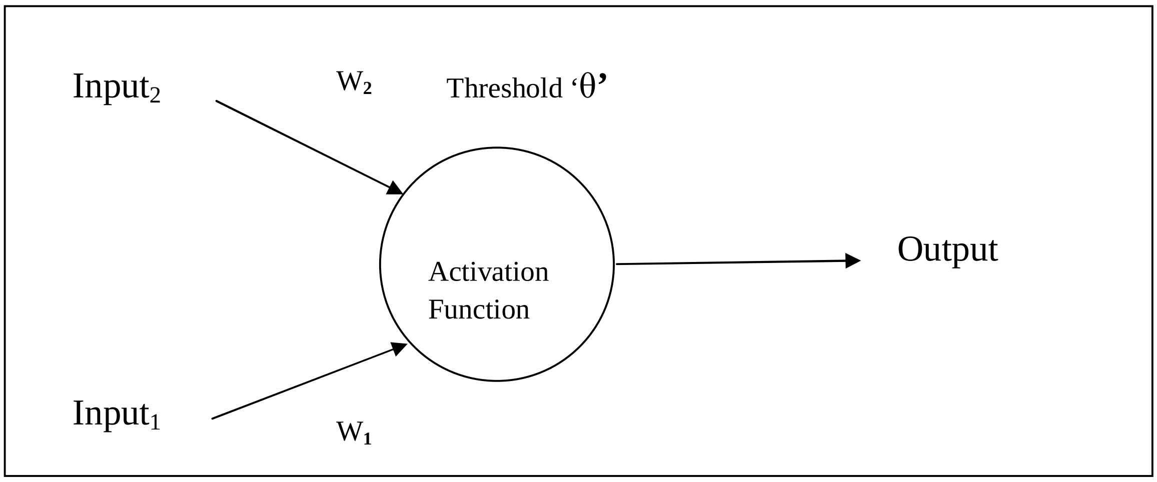

Figure 2.1: A Processing Element

2.3 Transfer Function: The output of any neuron is related with the inputs by a transfer function which gives the values of the outputs for the given inputs. To calculate the output from a neuron, the first step is to calculate the net input for that neuron, which is obtained by summing up the multiplication of each input to the weight of the corresponding path of the neuron that connects it to the input. The weights are the strength of connections between the neurons. Thus higher the weights of paths are, stronger will be the net effect of the inputs. Weights can also be negative, which signifies that the signal can be inhibited by the negative weight. The net input is converted into the output with the activation function which is the transfer function for each neuron. There are many types of transfer function defined in the literature i.e. Logsigmoid Transfer Function, Tansigmoid Transfer Function, Cumulative Gaussian Function, etc.

Logsigmoid Transfer Function: For our study, we have used the logistic or the logsigmoid transfer function as the transfer function. This is the most commonly used transfer function. The transformation defining the logsigmoid transfer function can be written in the form of following equation:-

$$y = \frac{1}{1 - e^{-x}}$$

To illustrate the operation of a typical logsigmoid activation function on a series ranging -5 to +5, we represent in the form of the following figure:

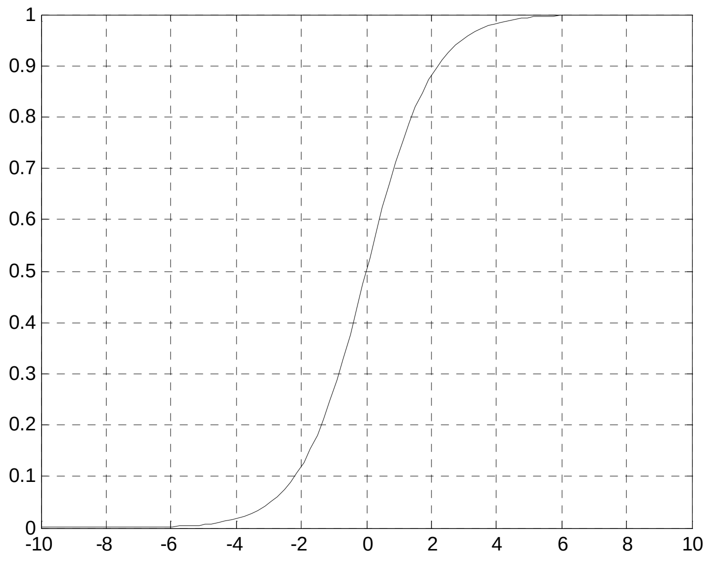

Figure 2.2 Logsigmoid Transfer Function

The above function becomes steep increasingly until some inflection point. Thereafter, the function becomes increasingly flat and its slope moves exponentially to zero. In the neural network paradigm, this threshold feature is described as the fundamental characteristic of the nonlinear response. It is described that the certain types of neurons remain inactive up to certain levels of input activity, become active after it passes this threshold level, and while beyond this, increase in input activity have again little effect.

2.4 Layers: Neural Networks consists of neurons distributed across layers. The three types of layers in a neural network configuration are (1) The input layer, (2) the hidden layers, and (3) the output layer. There is exactly one input layer and one output layer in each network. The way these neurons are linked to each other and the way they are distributed in the network configuration define the structure of the neural network starting from the input layer to the output layer with a number of hidden layers i: i=1,…..,m in between these two layers. The neurons in an ANN are denoted as nodes, (i,j) where each node performs the simple task of information processing by converting the received input signals/information into some processed output signals/information. These nodes are connected to the next layer neurons through directed arcs or links each characterized by a weight ($w_{i,j}$).This typical structure of ANN can be represented in the form of Figure 2.3 as follows:

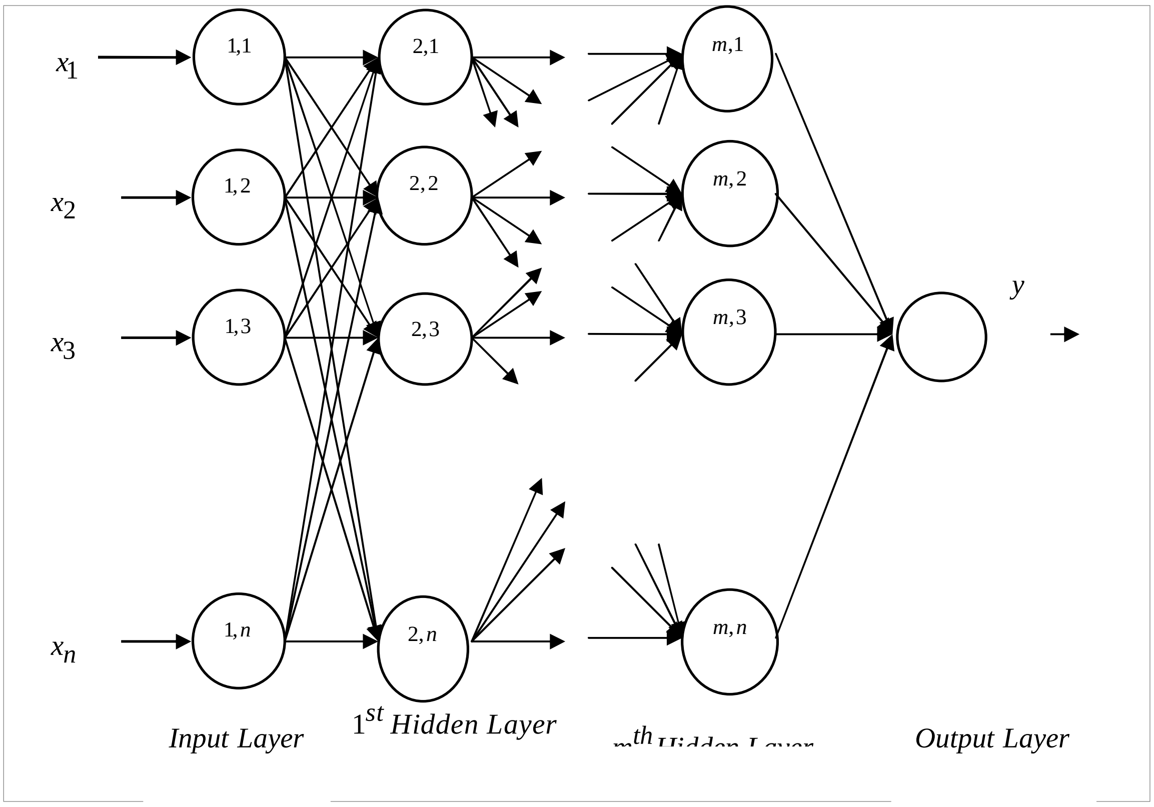

Figure 2.3: A typical structure of an Artificial Neural Network (ANN)

2.5 Supervised Learning: Supervised Learning involves a mechanism of providing the network with the desired output by providing the desired outputs with the inputs or by manually “grading” the network’s performance. Supervised training is thus also known as learning with a teacher. The neural network is provided with an input vector and the corresponding output vector. These vector pairs are used to determine the mappings that exist on the data set. The weights of the paths decide the computations of the neurons and varying these weights of the paths will therefore give different outputs from the network. The actual output when compared with the desired output, computes the error which is thereby used to further modify the weights of the paths. The resulting errors are then used to update the weights of the network which also represents the memory of the network and control the network. As the weights are changed, this process is repeated over and over again. The set of data which enables the training is called the training set. During the training of a network, the same set of data is processed many times to improve the connection weights each time. For a particular data set, there may be a specific network configuration that maps the inputs and the corresponding outputs most efficiently. There are many ways of varying the output from the network. It can be varied by varying the configuration of the network, i.e. varying the number of hidden layers, varying the number of nodes in each hidden layer or by changing the number of inputs to the network. For our study, we have used the conjugate gradient method which is also a type of supervised learning and is described in the subsequent section. Supervised learning can also be illustrated with the help of the following Figure 2.4.

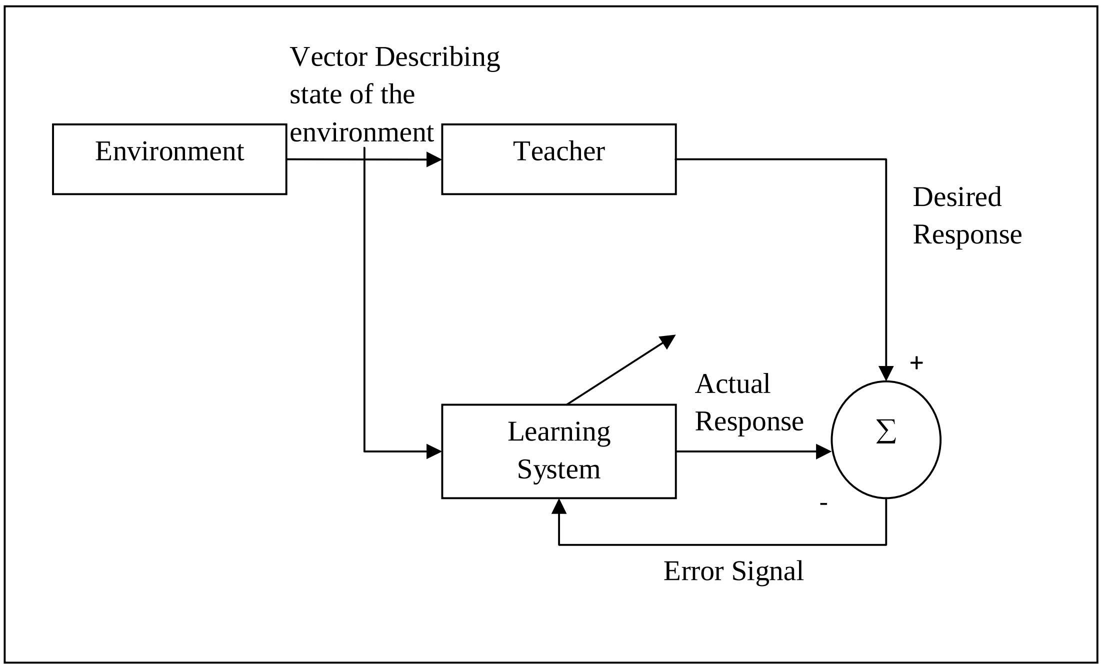

Figure 2.4: Supervised Learning – A Block Diagram

2.6 Rules of Learning: Apart from the configuration of the network and the learning method used, another important part is the rules of training. Many laws or algorithms are used to implement the adaptive feedback required to adjust the weights during training. Some examples of such rules are Hebb’s Rule, Hopfield Law, The Delta Rule, The Gradient Descent Rule, Kohonen’s learning Law, etc. The most commonly used technique is the Gradient Descent Rule using backward error propagation, more commonly known as the back-propagation. This method utilizes the derivative of the transfer function and a learning rate acting as the proportional constant in modifying the connection weights.

There is a possibility of the limitation that the error function falls into local optima in the methods discussed above thus giving inferior results. Thus, the algorithms that give the global optimal point can be used to compare the results obtained from the previous algorithms. Some such algorithms are Genetic Algorithm, Simulated Annealing, Tabu Search, etc.

An artificial neuron can be depicted in the form of a mathematical model. To describe a general neuron mathematically, we can use the following equations:

$$u_{i,j} = \sum_{j=1}^{n} w_{i,j} * x_{i,j} \qquad \forall \text{ i} = 1\ldots, \text{m}$$

$$y_{i,j} = f(v_{i,j}) = f(u_{i,j} + b_{i,j}) \qquad \forall \text{ i} = 1\ldots, \text{m}$$

Where, $u_{i,j}$ is the output from the (*i, j*) neuron which has $x_{i,j}$′s as the inputs and $w_{i,j}$′s as the corresponding weights. Furthermore $b_{i,j}$′s denote the bias, f(.) is the transfer function and $y_{i,j}$ is the output. This mathematical form of a neuron can be represented in the form of a figure as follows:

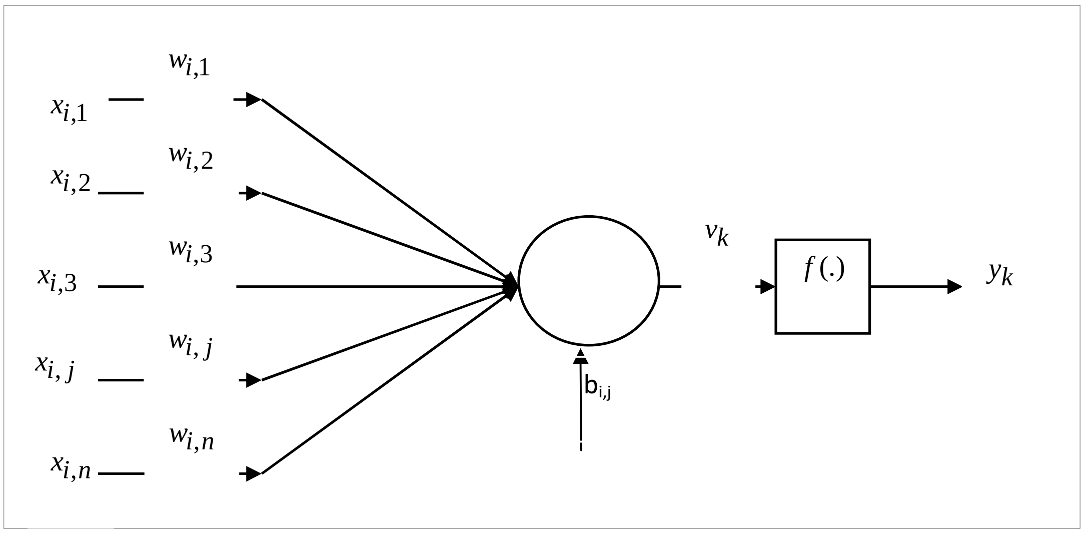

Figure 2.5: Mathematical model of a neuron

2.7 Backpropagation Method: For the feed-forward neural networks, backpropagation is used as a supervised learning procedure for the purpose of training. The training set, i.e. the series of the test cases are presented before the network, one at a time. Thus obtained errors between the actual output and the desired output of the network are propagated backwards to the internal layers, i.e. first to the hidden layers and then to the input layer. The weights are thereby adjusted in accordance and in proportion to their contribution to the error. This error used in our case is the Mean Square Error (MSE). A typical feed-forward back-propagation network in which the errors are back propagated and the weight metrics are modified accordingly is shown in Figure 2.6.

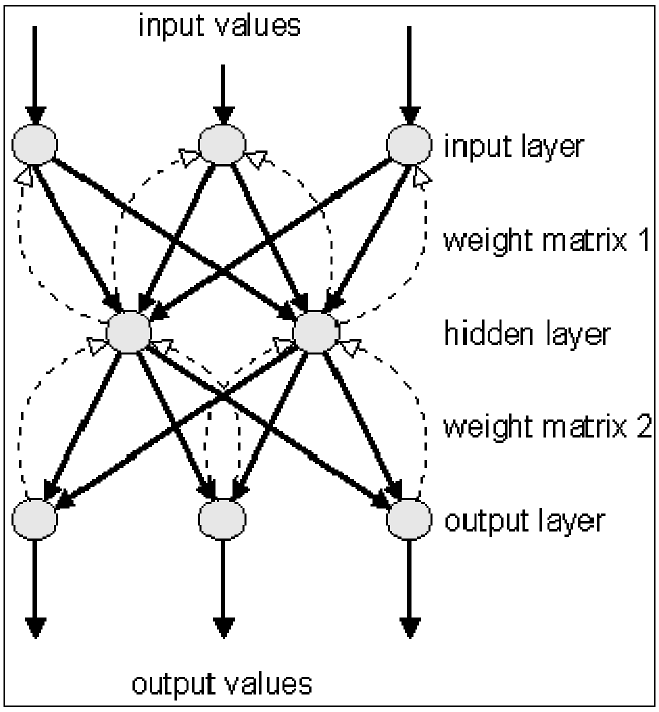

Figure 2.6: A feed-forward back-propagation network

2.8 Conjugate Gradient Method

This method, Conjugate Gradient Method belongs to the class of Second order optimization methods which are collectively known as conjugate-direction methods and they are generally simple and easy to implement. The basic idea of this method is that for the convergence to the solution to be accelerated, it is more beneficial to minimize our objective function (Q) over the hyper plane that contains all the previous search directions, than to minimize Q over just the line that points down gradient. The Conjugate Gradient Method is considered superior to the Steepest Descent Method but the Newton's method when compared to the CGM is better. However, in case of large number of variables, again the Conjugate Gradient method is better because it only uses vectors and takes O (n) operations per step, where n is the number of parameters.

First, we consider the minimization of the quadratic equation $f(x) = \frac{1}{2} x^T Ax \ - b^T x + c$ where x is a (WX1) parameter vector, A is a (W X W) symmetric, positive definite matrix, b is a (W X 1) vector, and c is a scalar. Now, we say that the quadratic equation is minimized by assigning the unique value to x and i.e. $x^* = A^{-1}b$. Thus minimizing *f(x)* and solving the linear system of equations $Ax^* = b$ are equivalent problems.

Given the matrix A, we say that a set of nonzero vectors s(0), s(1)…, s(W-1) is A-Conjugate if the following condition satisfies: $\lfloor s^T(n) * A * s(j) \rfloor = 0$ ∀ n and j such that n≠j.

For a given set of A-conjugate vectors s(0), s(1)…, s(W-1), the corresponding conjugate direction method for unconstrained minimization of the quadratic error function is defined by $x(n+1) = x(n) + \eta(n)s(n)$ ∀ n = 0,…, W-1 where x(0) is an arbitrary starting vector and η(n) is a scalar defined by $f(x(n)) + \eta(n)s(n) = \min_{\eta} f(x(n) + \eta s(n))$.

If the residual be

$$r(n) = b - Ax(n) \tag{2.1}$$

We use a linear combination of r(n) and s(n-1), as shown by

$$s(n) = r(n) + \beta(n)s(n-1), \ \forall \text{ n=1, 2… W-1} \tag{2.2}$$

Where, β(n) is a scaling factor to be determined. Multiplying Eq. (2.2) by A, taking the inner product of the resulting expression with s (n-1), invoking the A-conjugate property of the direction vectors, and then solving the resulting expression for β(n), we get:

$$\beta(n) = \frac{s^T(n-1)Ar(n)}{s^T(n-1)As(n-1)} \qquad (2.3)$$

Using equations 2.1 and 2.3, we find that the vectors s(0), s(1),…, s(W-1) and these are indeed A-Conjugate.

To use the conjugate gradient method, we do two things (i) approximate the cost function ($\xi_{av}(w)$) by a quadratic function and (ii) formulate the computation of coefficients β(n) and η(n). To compute the coefficient of β(n), we can use the Polak Rebiere formula (Haykin (2004)) which is given by $\beta(n) = \frac{r^T(n)(r(n)-r(n-1))}{r^T(n-1)r(n-1)}$ or Fetcher Reeves formula (Haykin (2004)) given by $\beta(n) = \frac{r^T(n)r(n)}{r^T(n-1)r(n-1)}$. For the computation of η(n) which determines the learning rate of the conjugate-gradient algorithm, the preferred method would be a line search routine, the purpose of which is to minimize the function $\xi_{av}(w+\eta s)$ with respect to η.

There is a possibility of the limitation that the error function falls into local optima in the methods discussed above thus giving inferior results. Thus, the algorithms that give the global optimal point can be used to compare the results obtained from the previous algorithms. Some such algorithms are Genetic Algorithm (GA), Simulated Annealing (SA), Tabu Search (TS), Artificial Immune System (AIS), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), to name a few. Out of these, for our study, we use GA and SA.

CHAPTER 3

METAHEURISTIC TECHNIQUES

An important factor in concern regarding optimization problems of practical and theoretical importance is that the best configuration of the set of variables is chosen to achieve certain goals. They are thus divided into two broad categories, (1) Those whose solutions are encoded with real-valued variables, (2) those where solutions are encoded with discrete variables. Among the later class, we find a sub-category or sub-class of problems which are known as Combinatorial Optimization (CO) problems.

Various famous CO problems encountered are Travelling Salesman Problem (TSP), quadratic assignment problem and neural network weight selection. Now, since these problems have a lot of practical importance, various algorithms have been developed to handle them. These can be classified as either complete or approximate algorithms. A kind of approximate algorithms which basically try to combine basic heuristic methods in higher level frameworks aimed at efficiently and effectively exploring a search space are Metaheuristic Techniques.

A metaheuristic is formally defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions. Some of these types of algorithms are Genetic Algorithm (GA), Simulated Annealing Algorithm (SA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO) and Tabu Search (TS). We have used GA and SA for our study and they are described as follows:

3.1 Genetic Algorithm

The Genetic Algorithm is the most commonly and successfully used method for solving the optimization problems which is based on the well known concept of natural selection, the process that drives biological evolution. Accordingly, there is constant and repeated modification of the population of individual solutions in the Genetic Algorithm. At each step, the genetic algorithm selects individuals at random from the current population to be parents and then uses them to produce the children for the next generation. Thereby, over successive generations, the population evolves toward an optimal solution. Genetic Algorithm is also used to solve a wider variety of optimization problems that are not well suited for the standard optimization algorithms, including the problems in which the objective function is discontinuous, non-differentiable, stochastic, or highly nonlinear.

3.1.1 Population Criterion: This method starts with a population *N* (an even number) of random vectors and not with one random coefficient vector w. If we take *n* to be the size of each column vector, representing the total number of coefficients to be estimated in the neural network, we create a population *N* of (*n* X 1) random vectors.

$$\begin{array}{l}
(w_1, w_2, w_3, \ldots\ldots\ldots\ldots\ldots\ldots\ldots w)_1 \\
(w_1, w_2, w_3, \ldots\ldots\ldots\ldots\ldots\ldots\ldots w)_2 \\
(w_1, w_2, w_3, \ldots\ldots\ldots\ldots\ldots\ldots\ldots w)_3 \\
. \\
. \\
. \\
(w_1, w_2, w_3, \ldots\ldots\ldots\ldots\ldots\ldots\ldots w)_N
\end{array}$$

Each individual member of the population is known as chromosome. Chromosomes could be bit strings (1010……….011011), real numbers (89.2 66.4 88.3………45.6), permutations of elements (M23 M2 M7 M16 M12), lists of rules (R6 R8 R14 R17 R25), program elements (genetic programming), etc.

3.1.2 Selection: During each successive epoch, some part or proportion of the existing population is selected to breed a new generation. This selection is done through a fitness based process to select individual solutions, when, as measured by a fitness function, the fittest solutions are more likely to get selected. Each solution is rated for its fitness by predefined certain selection methods and preferentially, the best solutions are selected. Other methods rate only a random sample of the population, as this process may be very time consuming. The most commonly used, well-studied as well as popular selection methods are the roulette wheel method and tournament selection method.

In the roulette wheel selection method, a roulette wheel is designed, where each member is represented in the wheel in proportion to their fitness value. This obviously means that the better chromosomes proportionally get more space on the wheel and are thus more likely to survive than the poorer chromosomes. Thus for obtaining the next generation of chromosomes, these winning vectors are retained for the breeding purposes. But this is only a proportion of the current population as the poorer chromosomes are dropped during the selection process. Thus to get the original size of the population, the population is needed to be refilled after every generation. The refilling is achieved by "mating" the selected members using crossover and mutation.

3.1.3 Crossover: The first step of refilling or the next step in the process is crossover. In this step, the two parent chromosomes "breed" to give two children chromosomes. On each given pair of coefficient vector i and j, the algorithm allows crossover to be performed with a fixed probability $p > 0$. There are three different types of crossover operations defined, and the algorithm chooses one of the three methods for the crossover operation to be performed based on an equal probability of (1/3) of each method to be chosen. The three techniques of crossover operation are as follows:

1. **Shuffle Crossover:** For each given pair of vectors, k random draws are made from a binomial distribution. Based on if the k^{th} draw is equal to 1, the coefficients $w_{i,p}$ and $w_{j,p}$ are swapped; otherwise, no change is made.

2. **Arithmetic Crossover:** For each given pair of vectors, a random number θ is chosen, such that $\theta \in (0,1)$. This number is used to create two new parameter vectors which are linear combinations of the two parent factors, $\theta w_{i,p} + (1-\theta) w_{j,p}$.
3. **Single-point Crossover:** For each given pair of vectors, an integer ω is randomly chosen from the set [*l*, *k* −*l*], where $\omega \in [l, k-l]$. The two vectors are then cut at integer *I* and the coefficients to the right of this cut point, $w_{i,l+\omega}$, $w_{j,l+\omega}$ are swapped.

3.1.4 Mutation: After obtaining the children chromosomes from the crossover, the next (fourth) step in the process is to mutate these children. Each element or coefficient of the two children vectors is subjected to mutation with a small probability *p*, which decreases over time. The probability of mutation of each element depends on the generation *g*. Now, to apply the mutation formula to obtain the mutated coefficients, we need some random numbers. If two real numbers r_1 and r_2 are randomly drawn from the interval [0, 1] and one random number *s* from a standard normal distribution, then the mutation formula to generate the mutated coefficients can be given by:

$$\left.\begin{aligned} w_{i,p} &= w_{i,p} + s[1 - r^{(1-g/G)^b}] \qquad if\ r_1 < 0.5 \\ w_{i,p} &= w_{i,p} - s[1 - r^{(1-g/G)^b}] \qquad if\ r_1 \geq 0.5 \end{aligned}\right\} \tag{3.1}$$

Where, *g* is the generation number and *G* is the maximum number of generations. The parameter *b* is one which governs the degree to which the mutation operation is non-uniform. Generally, we set the value of the parameter to be 2. Now, as *g* approaches *G*, i.e. the maximum number of generations, there is a decrease in the probability of generating a new coefficient which is quite far away from the current coefficient value, by mutation. Thus there is an evolution in the probability of mutation itself over time. The mutation operation is non-uniform, because, with time, the probability of getting a far away value from the current coefficient value decreases and the algorithm thus returns values in the neighborhood of the existing coefficient values intensively. This more localized search thus thereby leads to some fine tuning of the coefficient vector in the later stages of the research, when the vectors approach the global optima.

3.1.5 Iterations: Now, in the next step, the selected members from the previous generation and the members generated after applying crossover and mutation are compared for their fitness value. Then, on the basis of the fitness values, the best N chromosomes are selected to populate the second generation. However, the search for the best member is continued for a fixed number of generations. That is, some iterative search is carried on, or until we meet some convergence criterion.

3.1.6 Convergence: This process of the algorithm is continued up to the maximum number of generations, i.e. G. But the value of G to be chosen is not very well defined, even in the existing literature. Thus, for better results, since we have narrowed down on the convergence criterion based on the fitness value, we say that the value of G should be large enough so that for several generations, there are no changes in the fitness value of the best member.

The general pseudo code for GA can be given as follows:

<u>Pseudo Code for Genetic Algorithm:</u>

```
1    Generate initial population of solutions N
2    while stopping criteria not met do
3        select mating pool $N' \subset N$, initialize $N'' = \varnothing$ (set of children)
4        for i=1 to n do
5        select individuals $x_a$ and $x_b$ at random from $N'$
6        apply crossover to produce $x_{child}$
7        randomly mutate produced child $x_{child}$
8        $N'' = N'' \cup x_{child}$
```

```
9     end for
10    N=survive(N′, N″)
11    end while
```

3.2 Simulated Annealing

Simulated Annealing algorithm finds its origins in statistical mechanics (Metropolis Algorithm) and it was initially presented as a search algorithm for Combinatorial Optimization (CO). It is one of the oldest methods introduced in 1983 by Kirkpatrick and used among the metaheuristics and is one of the algorithms that had an explicit strategy to avoid local minima. It is a generic probabilistic meta-algorithm for the global optimization problem, used to locate a good approximation of any multi-optimal function to the global optimal point. The Simulated Annealing algorithm gets its name as its is based on the simulation of the annealing of solids. Annealing is a process of heat treatment technique which involves heating and controlled cooling of a material to reduce the internal defects. In the process, the solid is first heated by increasing the temperature to a maximum value such that it is transformed into its liquid phase, and then after that, it is cooled slowly. When the atoms of the solid reach the high temperature, they leave their initial position and begin to wander through states of high energy randomly. Subsequently, when they are cooled down slowly, they thereby acquire a position of lower internal energy than their initial energy level. It is said that if the cooling is performed from a very high temperature and is carried out sufficiently slow, then all the atoms of the liquid arrange themselves in the low energy ground state of a corresponding lattice. The solid keeps on achieving thermal equilibrium at every temperature. The thermal equilibrium achieved by the solid at each temperature T can be given by a probability of being in a state with energy E by the Boltzmann distribution as given below:

$$\Pr(E = e) = \frac{1}{Z(T)}.\exp\left[-\frac{E}{k_B.T}\right] \tag{3.2}$$

Where, *Z (T)* is a normalization factor which depends on the temperature T and k_B is the Boltzmann constant. As it can be easily deduced from the above equation, the Boltzmann distribution concentrates on the states with lowest energy. Finally, when the temperature reaches zero, with a non-zero probability, the minimum energy states are achieved.

Long back in 1953, in order to simulate the evolution of thermal equilibrium of a solid, Metropolis proposed a Monte Carlo method to generate the sequences of states of the solid. The principle is that, under the given state of the solid, a perturbation is applied by small displacement of a randomly chosen particle and if its perturbation leads to decrease in the energy of the solid, i.e. ΔE, between the current state and the new state is negative, and then the process is continued from the new state. On the contrary, if the difference is positive, i.e. ΔE is > 0, and then the probability of accepting the new state will be given by $\exp\left[-\frac{\Delta E}{k_B T}\right]$. This is known as the acceptance rule for the new state better known as the Metropolis criterion. This criterion finally evolves into the thermal equilibrium of the solid.

For the better understanding of the detailed working of the Simulated Annealing algorithm, it can be illustrated in the form of the following pseudo code:

<u>Pseudo Code for Simulated Annealing:</u>

```
1. Generate initial solution $x_0$, initialize starting temperatures T, maximum number of iterations $R_{max}$
2. for r=1 to $R_{max}$ do
3.         while stopping criteria not met do
4.         compute $x_n$ (Neighbor to current solution)
5.         compute $\Delta=f(x_n)-f(x_0)$ and generate u (uniform random variable)
6.         if $(\Delta<0)$ or $(e^{-\Delta/T}>u)$ then $x_0=x_n$
7.         end while
```

```
8. reduce T
9. end for
```

Mathematical Model of Simulated Annealing: For realizing a mathematical model of the simulated annealing algorithm, a control parameter and is realized as the temperature and a cost function as the energy respectively for determining the different configurations of the variables to be optimized. As the control parameter is decreased, the new sequences are obtained. Let the cost function be denoted by C and the control parameter be denoted by c. When during the iterations, if the configuration changes from *i* to *j*, which is in the neighborhood of the previous state (i.e. *i*), then the change in the cost function is given by $\Delta C_{ij} = C(j)-C(i)$. Now, the probability of accepting this new state j will be decided according to the Metropolis criterion. Thus, the configuration will be 1 if $\Delta C_{ij} < 0$ and $\exp(-\Delta C_{ij} / c)$ if $\Delta C_{ij} > 0$. Thus, the probability of accepting the higher state is non-zero. This process is thereby continued until the equilibrium is established. We lower down the control parameter in steps and the previous sequence of operations is again followed to reach equilibrium. The control parameter is lowered up to a certain value below which there is no significant improvement observed in the outputs. Thus, we can see that when the value of the control parameter is high, the probability of accepting a new solution is higher, and as the value of control parameter decreases, there is a gradual decrease in this probability due to less change in the equilibrium state value solution.

The Simulated Annealing technique is famous for finding the global optima by accepting both the increasing as well as decreasing values of the transition function. The former is accomplished with certain probability which depends on stochastic acceptance criterion. For the minimization problems, the probability of accepting the higher state descends slowly towards zero by some decreasing schedule. Because of this decreasing acceptance of a new state, the algorithm is able to escape from the local minima.

CHAPTER 4

PARAMETRIC FORECASTING METHODS (GARCH)

Nonlinearity has always posed problems towards those trying to model financial time series for forecasting. The nonlinear models which have been developed try to realize the underlying nonlinear processes through parametric assumptions with specific nonlinear functional forms. Some examples of the many nonlinear functional forms used are Auto-Regressive Conditional Heteroskadisticity (ARCH) (Engle, 1982), Generalized Auto-Regressive Conditional Heteroskadisticity (GARCH), Self-Exciting Threshold Auto-Regressive (SETAR) (Chappel, 1996), chaotic dynamics (Hsieh, 1991), which have been proposed and applied to forecasting financial time series.

GARCH models have been extensively used in finance and macroeconomics because of their attractive approximation-theoretic properties. This model provides useful approximations to uncomplicated volatility dynamics. The basic concept of the model is that it considers that the variance of the current error term is the function of the variances of the previous time period’s error terms. Thus, it is able to capture the important property of volatility clustering, i.e. large/small changes of either sign are tend to be followed by large/small changes of the small sign. The Auto-Regressive Moving Average (ARMA) model, when is assumed for the error variance, then it is converted to GARCH model. Thus, the time varying variance is therefore called the conditional variance or volatility for this reason. However, to extract the inherent auto regressions in the time series, it is modeled with the help of Auto-Regressive Moving Averages (AR-MA). Thus, the model becomes a comprehensive ARMA-GARCH model and is described in the following equations as follows:

$$y_t = \sum_{i=1}^{r} \psi_i y_{t-i} + \varepsilon_t + \sum_{j=1}^{m} \varphi_j z_{t-j} + k \qquad (5.1)$$

$$\sigma_t^2 = \sum_{i=1}^{p} G_i \sigma_{t-i}^2 + \sum_{j=1}^{q} A_i \varepsilon_{t-j}^2 + c \tag{5.2}$$

$$\varepsilon_t \sim N(0, \sigma^2) \tag{5.3}$$

The first equation, i.e. (5.1) is the ARMA part of our model where r is the number of past lagging terms; ψ_i's is the coefficients of the lag values and φ_i's are the coefficients of the moving averages. The main part of the model, the variance is modeled in the equation (5.2), which as we know is known as the conditional variance as it depends on the past variance. G_i's are the coefficients of lag values of variance terms and A_is, are coefficients of squared errors. Equation (5.3) shows that the errors ε_t are normally distributed.

Since the distribution of the shock is normal, we can use the maximum likelihood function for the estimation of the parameters G_i, A_i and *c*. The likelihood function L is the joint probability function for $\hat{y}_t = y_t$. The likelihood function for the GARCH model can be represented in the following form:

$$L_t = \prod_{t=1}^{T} \sqrt{\frac{1}{2\pi\hat{\sigma}_t^2}} \exp\left[\frac{(y_t - y_t)^2}{2\pi\hat{\sigma}_t^2}\right] \tag{5.4}$$

$$\hat{\sigma}_t^2 = \sum_{i=1}^{p} \hat{G}_i \sigma_{t-i}^2 + \sum_{j=1}^{q} \hat{A}_i \hat{\varepsilon}_{t-j}^2 + \hat{c} \tag{5.5}$$

$$\hat{y}_t = \sum_{i=1}^{r} \hat{\psi}_i \hat{y}_{t-i} + \sum_{j=1}^{m} \hat{\varphi}_j \hat{\varepsilon}_{t-j} + \hat{k} \tag{5.6}$$

Where, $\hat{G}_i, \hat{A}_i, \hat{c}, \hat{k}$ are the estimates of the underlying parameters and $\prod$ is the multiplication operator. The usual method for obtaining the parameter estimates is that we maximize the sum of the logarithm of the likelihood function, or log-likelihood function, over the entire sample *T*, (*t* = 1,……, *T*). The important point to be remembered here is that the conditional variance is a nonlinear transformation of the past values, in the same way that the variance measure is a nonlinear transformation of the past prediction errors. Thus, the GARCH model is quite beneficial in this way that it pins down the source of nonlinearity in the process.

The GARCH model has its specific limitations due to its structure and parameters. As we have a defined set of parameters which we want to estimate in the GARCH model, and which carry a well-defined meaning, interpretation, and rationale, the parametric approach of GARCH to the specification of the nonlinear process, thus become restrictive. The GARCH model, being capturing the property of volatility clustering is able to show an important observed phenomenon in the financial time series, that is, the periods of high and low volatilities do not dampen out fast. This restrictiveness of the GARCH approach proves to be its drawback as we are limited to a well-defined set of parameters, a well-defined distribution, a specific nonlinear functional form, and a specific parameter estimation method that does not always converge to the parameter estimates. Thus, certain alternative nonlinear processes are unable of being realized with specific nonlinear models as they lack in flexibility due to restrictiveness in specification.

CHAPTER 5

MODELS DESCRIPTION

5.1 Introduction

In the previous chapters, we had discussed various forecasting techniques, under the heading of non-linear forecasting techniques, namely (i) Non-Parametric non-linear forecasting methods i.e. Neural Networks and (ii) Parametric non-linear forecasting methods (GARCH). We had also discussed the metaheuristic techniques like Genetic Algorithm and Simulated Annealing Algorithm which are used to optimize the Neural Network configuration models in terms of its weights and nodes. In the following Section 5.2, we explain the various models based on the non-parametric approach, in section 5.3, the parametric models and then in the next section, we explain the performance metrics used for the comparison of the models.

5.2 Non-Parametric Approach based Models

We use Neural Network which is a kind of universal function approximator that can map any non-linear function without any assumption about the data and its parameters.

5.2.1 Neural Network trained with Conjugate Gradient Method (NN_CGM): The basic algorithm that we use to train the neural network is Conjugate Gradient Method (CGM). Firstly, we apply the autocorrelation function to determine the significant lags and then the lag matrix made up of these lags is input to the first layer as the input. The first layer, thus, has one node and the number of nodes in the hidden layer can vary from 2 to 15. The output from the hidden layer is fed into the third layer, the layer known as the output layer. We have used the log-sigmoid transfer function as the transfer function in our analysis which has been described in Chapter 2. The error is calculated according to the performance index, which we have used as the Mean Square Error (MSE) as described in Chapter 6. The error thus calculated is then backpropagated and the weights are thus adjusted according to the CGM as explained in Chapter 2. The iterative CGM algorithm attempts to reduce the training error on each epoch, which is used to decide the stopping criterion. The stopping criteria as to decide when to stop the network training depends on three basic following criterion (1) fix the number of epochs (2) check when the training error falls below an acceptable predefined level and finally (3) when the error fails to improve by a given amount over a given number of epochs. These three criterions may be used as the three stopping rules so as to decide when to stop the training of the neural network. The number of epochs is checked at each stage and the network is trained until the stopping criterion is met and we keep the maximum number of epochs fixed at the number 500. We assume that the number of 500 for the number of epochs is sufficient enough for the network to learn. To represent the model in the form of a flowchart, it can be done in the form of the following figure:-

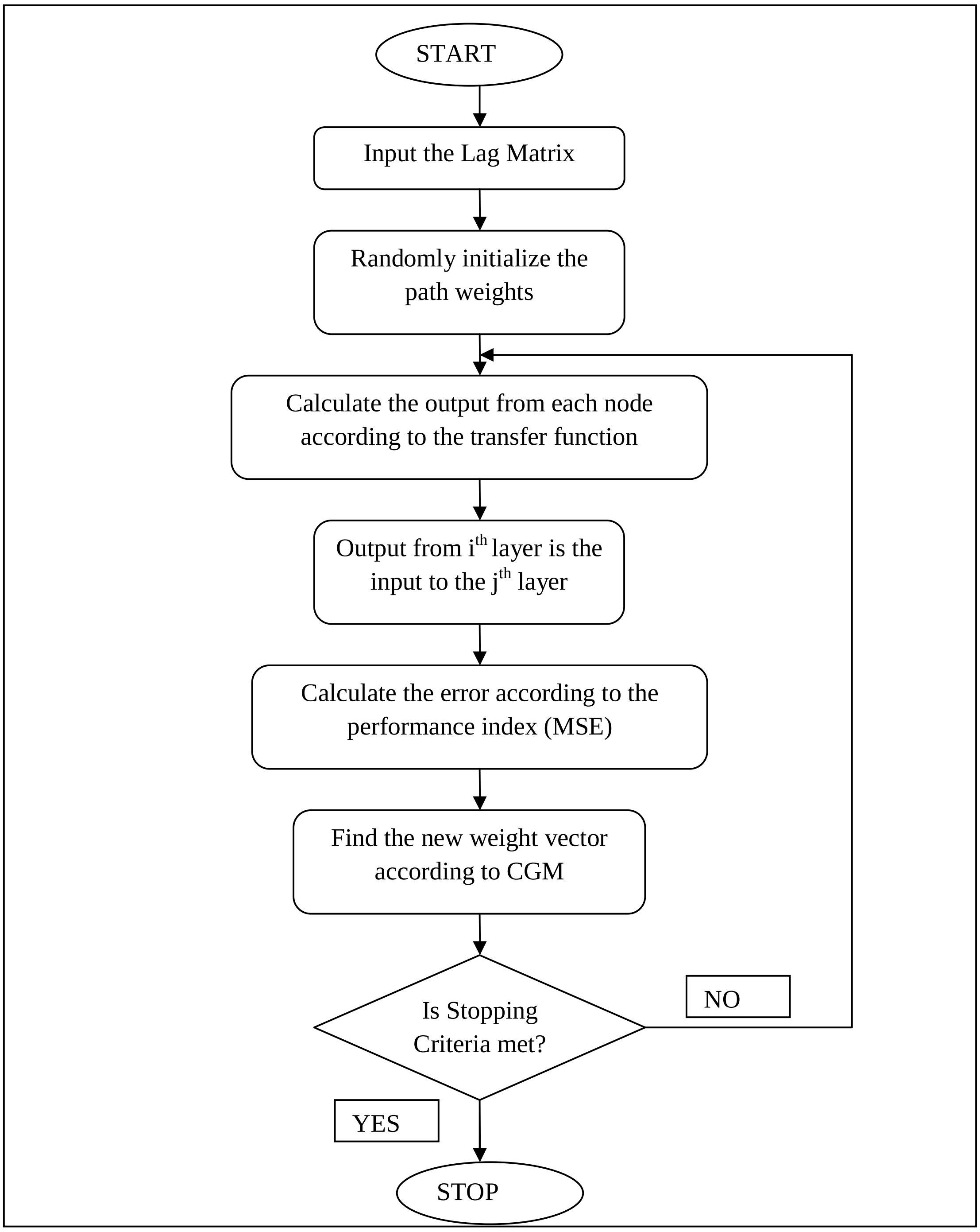
START
Input the Lag Matrix
Randomly initialize the path weights
Calculate the output from each node according to the transfer function
Output from ith layer is the input to the jth layer
Calculate the error according to the performance index (MSE)
Find the new weight vector according to CGM
Is Stopping Criteria met?
NO
YES
STOP

Figure 5.1 Flowchart Representing Neural Network trained with Conjugate Gradient Method.

5.2.2 Neural Network path weights Optimization: About the backpropogation algorithm, as we have discussed earlier in the Chapter 3, has the disadvantage that the solution may fall into the local optima and thereby result in the poorer forecast of the time series. There is where the metaheuristic techniques come into play. Although we have used Conjugate Gradient Method to obtain better results from the previous algorithm, but still it is not certain that whether the output we obtain is the global optima. Hence, we use the metaheuristic techniques o check whether these techniques outperform the backpropagation learning technique used for the neural network training.

5.2.2.1 Optimization with Genetic Algorithm (NN_GA): The Genetic Algorithm is the highly used and successful algorithm useful in optimizing the neural network configuration in terms of weights. This is because the Genetic Algorithm is able to find a near optimal solution. The objective is to find the optimum set of weights so that it minimizes the difference between the desired outputs and the actual outputs. Hence, the chromosomes are constructed as the real numbers which thereby represent the weights of the neural network. Thus, each chromosome is a weight vector which can be utilized in the GA optimization technique. After once the chromosome is designed, we create or initialize an initial population at the length 60, while each of the chromosomes is composed of 15 real numbers, each of which represents the path weights. Crossover fraction is varied from 0.6 to 0.9, while the mutation probability is set to 0.001. Roulette wheel selection method is then used for selecting those two chromosomes which are used for reproducing. After the crossover process is over, the children chromosomes are produced and evaluated using the fitness function. The flow process of the detailed working of the Genetic Algorithm can be represented in the form of the schematic flowchart to understand the working of the GA, as follows:-

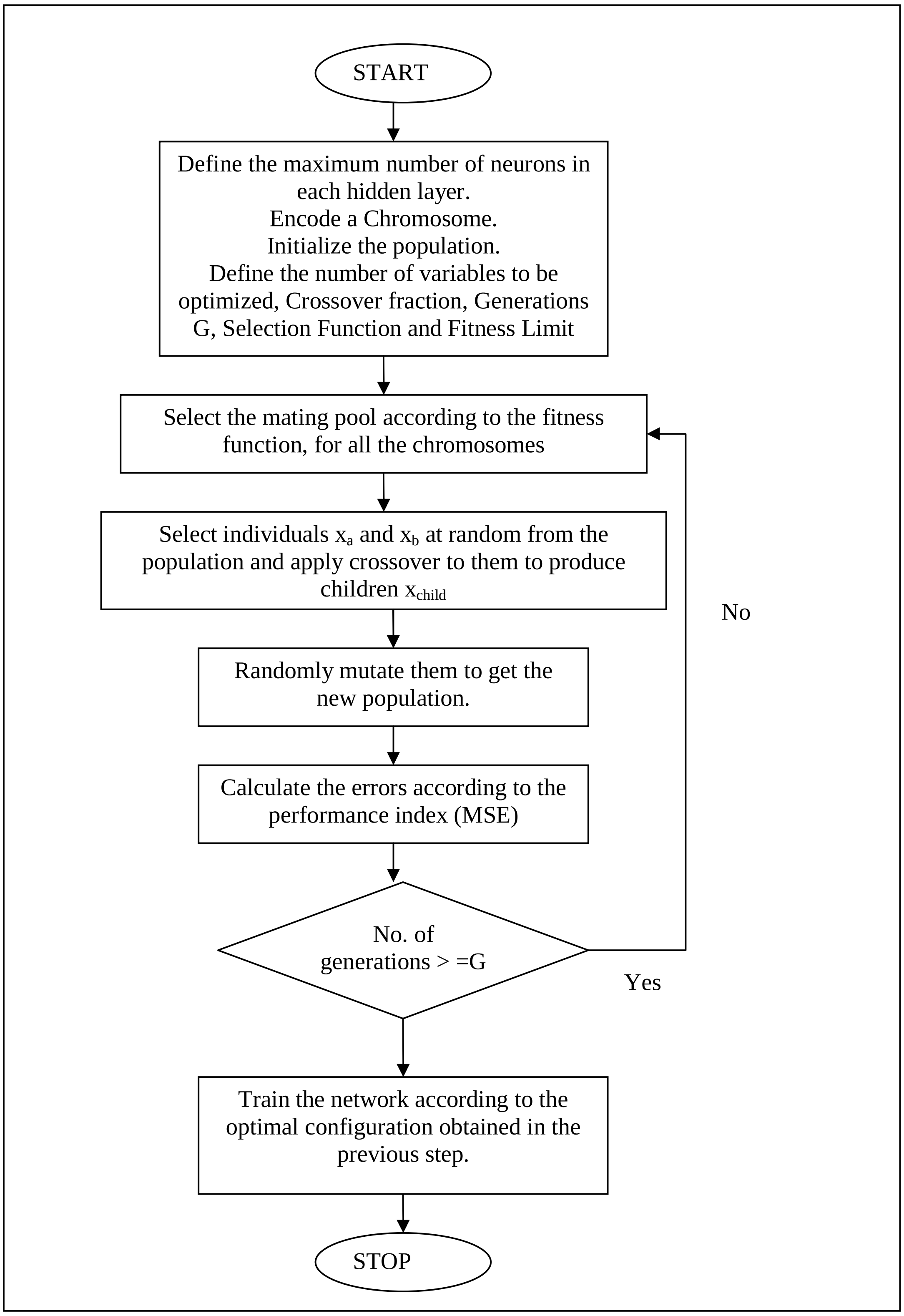
START
Define the maximum number of neurons in each hidden layer.
Encode a Chromosome.
Initialize the population.
Define the number of variables to be optimized, Crossover fraction, Generations G, Selection Function and Fitness Limit
Select the mating pool according to the fitness function, for all the chromosomes
Select individuals x_a and x_b at random from the population and apply crossover to them to produce children x_{child}
Randomly mutate them to get the new population.
Calculate the errors according to the performance index (MSE)
No. of generations > =G
No
Yes
Train the network according to the optimal configuration obtained in the previous step.
STOP

Figure5.2: Flowchart Representing Neural Network path weight Optimization using GA

5.2.2.2 Optimization using Simulated Annealing (NN_SA): - The Simulated Annealing (SA) algorithm is also one of the most frequently used techniques for obtaining the global optima in terms of weights for the neural network configuration. The results obtained from SA optimization can be compared to those from GA optimization. In our SA optimization study, initially, the set of weights x_0 are taken randomly in the range of [-6, 6] which is known as the starting point. The starting step vector v_o is the vector which decides about the changes in each of the weight element in the weight vector meaning the movement in each direction. The controlling parameter acting here is the starting temperature T_o whose value is varied from 16 to 30 with increments of 4. The stopping criterion here depends on the variable e, which denotes the error difference between the observed and the target values of the predicted time series. The value of e needs to be specified such that the SA optimization stops or terminates as soon as the actual value of e falls below a predefined value. We have taken this value to be 0.0030. The acceptance and rejection of a new state formed at each step depends on and is decided according to the Metropolis Criterion, as explained in the Chapter 3. Thus, for each particular temperature, an optimal network configuration is found out and this process of obtaining the corresponding optimal neural network is continued for the temperature cooling schedule till it reaches the equilibrium state. The detailed working of the SA optimization algorithm can be represented in the form of a flowchart as follows:-

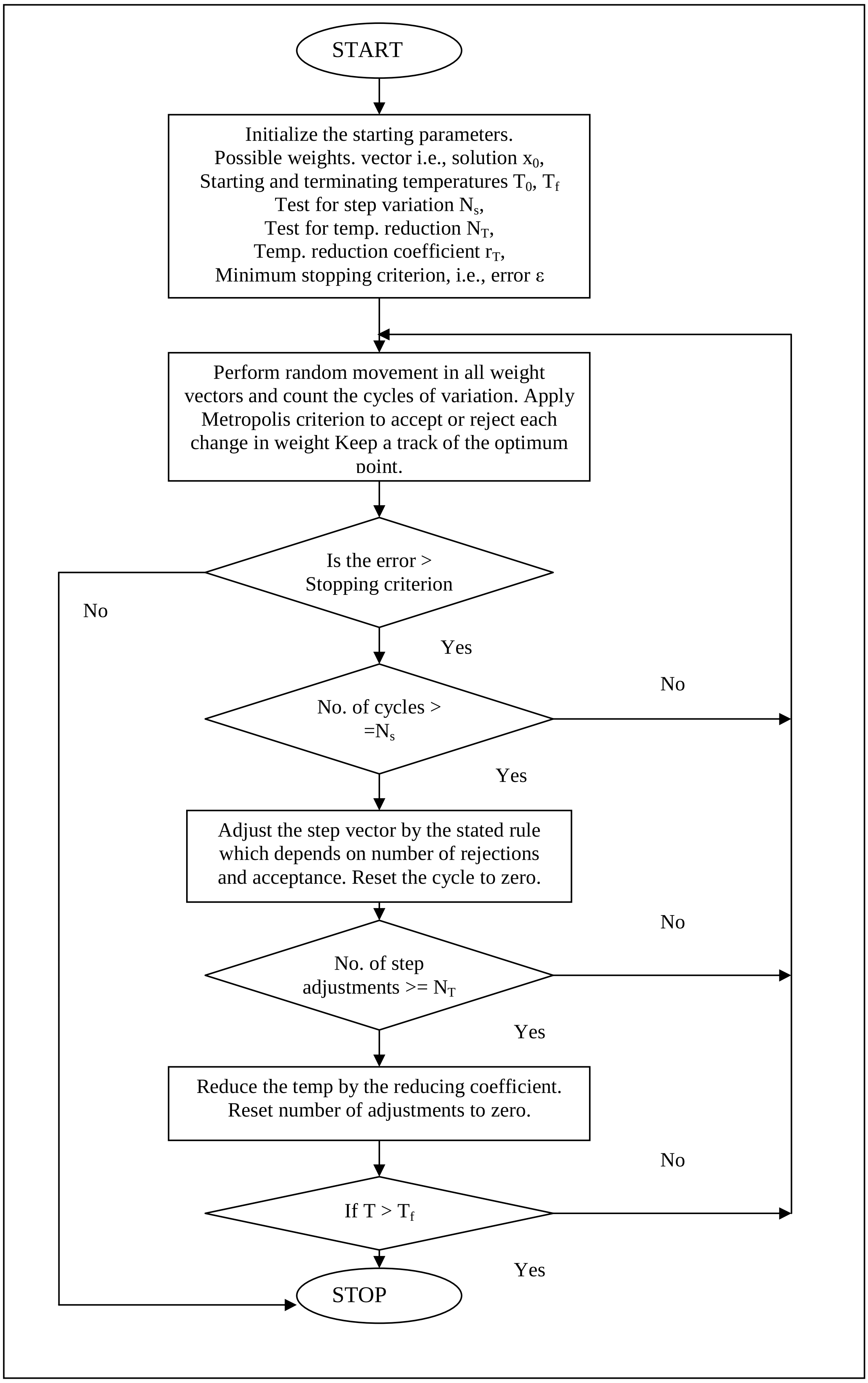
START
Initialize the starting parameters.
Possible weights. vector i.e., solution x_0,
Starting and terminating temperatures T_0, T_f
Test for step variation N_s,
Test for temp. reduction N_T,
Temp. reduction coefficient r_T,
Minimum stopping criterion, i.e., error ε
Perform random movement in all weight vectors and count the cycles of variation. Apply Metropolis criterion to accept or reject each change in weight Keep a track of the optimum point.
Is the error > Stopping criterion
No
Yes
No. of cycles > =N_s
No
Yes
Adjust the step vector by the stated rule which depends on number of rejections and acceptance. Reset the cycle to zero.
No. of step adjustments >= N_T
No
Yes
Reduce the temp by the reducing coefficient. Reset number of adjustments to zero.
If $T > T_f$
No
Yes
STOP

Figure5.3: Flowchart Representing Neural Network path weight Optimization using SA

5.3 Parametric Approach based Models

5.3.1 ARMA-GARCH: Apart from the non-parametric models described above for nonlinear modeling, we have also used the parametric model ARMA-GARCH for the better comparison of results from the non-parametric models and to realize the nonlinear properties of the exchange rate time series. In this model, the fours parameters, i.e. r, m, p and q are varied from 1 to 2. Thus, this gives rise to a total of 16 possible combinations of configurations. For each of these configurations, the values of $\hat{G}_i, \hat{A}_i, \hat{c}, \hat{k}$ are calculated, which are the estimates of the underlying parameters of the time series. These estimates are then further used to forecast the future values. The performance of each of the configuration is again measured by using the performance metrics defined below, where MSE serving as the main performance measurement index. The configuration giving the least or the optimum value is chosen and thereby compared with the non-parametric forecasting models.

5.4 Performance Metrics

The different models described above can be compared using the different performance metrics or the evaluation criteria and for our study, we have considered the following five performance metrics to compare these models which can be explained as follows:

If we say that $Y_1, Y_2, Y_3, Y_4 \ldots Y_N$ are the actual values and $\hat{Y}_1, \hat{Y}_2, \hat{Y}_3\ \hat{Y}_4, \ldots\ldots\ldots \hat{Y}_N$ be the forecasted values. Also $\overline{Y}$ and $\overline{\hat{Y}}$ be the mean of the actual and the forecasted values respectively and N be the sample size.

Mean Squared Error: The most common criterion used to evaluate the performance is the Mean Squared Error (MSE). It is the expected value of the square of the errors and is given by $MSE = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2$.

Mean Absolute Error: This gives the Absolute error between the actual output and the forecasted output and is given by $MAE = \frac{1}{N}\sum_{i=1}^{N}\left|(Y_i - \hat{Y}_i)\right|$.

Direction accuracy: This index basically measures how good the predicted direction is i.e., it is the measurement of correctness of predicted directions. It can be given by $DA = \frac{1}{N}\sum_{i=1}^{N} a_i$, where a_i=1 if $(Y_{i+1} - Y_i)(\hat{Y}_{i+1} - \hat{Y}_i) > 0$ and a_i = 0 otherwise.

Pearson correlation coefficient: Pearson's correlation reflects the degree of linear relationship between two variables. It ranges from +1 to -1.The correlation between the two variables reflects the degree to which the variables are related. +1 indicates perfect positive relationship while -1 indicates the negative relationship. The Person correlation is given by $\rho = \frac{\sum_{i=1}^{n}(Y_i - \overline{Y})(\hat{Y}_i - \overline{\hat{Y}})}{\sqrt{\sum_{i=1}^{N}(Y_i - \overline{Y})^2}\sqrt{\sum_{i=1}^{N}(\hat{Y}_i - \overline{\hat{Y}})^2}}$

Theil's coefficient of inequality (U): This performance index gives the prediction performance relative to the random walk prediction and the equation is $U = \frac{RMSE}{\sqrt{\frac{1}{N-1}\sum_{i=1}^{N-1}(Y_i - Y_{i-1})^2}}$, where RMSE is the Root Mean Squared Error and is given by $\text{RMSE}=\sqrt{\text{MSE}}$.

All the above models will be compared on the basis of these different performance indices. In the next chapter 6, the comparison has been shown among these models. Besides these performance indices, there are also other indices such as Akaike information criterion (AIC), Bayesian information criterion (SIC) which can be used to check the model complexity, while Correct Up trend, Correct Down trend can be used to measure the correctness of predicted up and down trend.

CHAPTER 6

DATA ANALYSIS AND SIMULATION

6.1 Data Analysis

In our analysis, we have taken the 12 different spot exchange rates to be our data. We have used the daily data series for all the exchange rates for our research from 03/01/2000 (January 3, 2000) to 24/12/2010 (December 24, 2010). We have taken the data from the Federal Reserve Bank of New York (http://www.federalreserve.gov/releases/h10/Hist) and it is the 12 noon buying rates at New York. The different exchange rates used are United States Dollar vs. Australian Dollar (US$/AU$), US Dollar vs. Canadian Dollar (US$/CAN$), US Dollar vs. Euro (US$/EURO), US Dollar vs. Hong Kong Dollar (US$/HK$), US Dollar vs. Japanese Yen (US$/JPY), US Dollar vs. Mexican Peso (US$/MXP), US Dollar vs. New Zealand Dollar (US$/NZ$), US Dollar vs. Singapore Dollar (US$/SING$), US Dollar vs. South Korean Won (US$/SKW), US Dollar vs. Sweden Kronor (US$/SWKR), US Dollar vs. Taiwan Dollar (US$/TW$) and US Dollar vs. United Kingdom Pound (US$/UKP). The reason for taking so many exchange rates as the data is to check the robustness of our models. Below shown in Figure 6.1 is the graph of Spot Exchange Rate US$/AU$ as it varies with time of 10 years from January 2000 to December 2010.

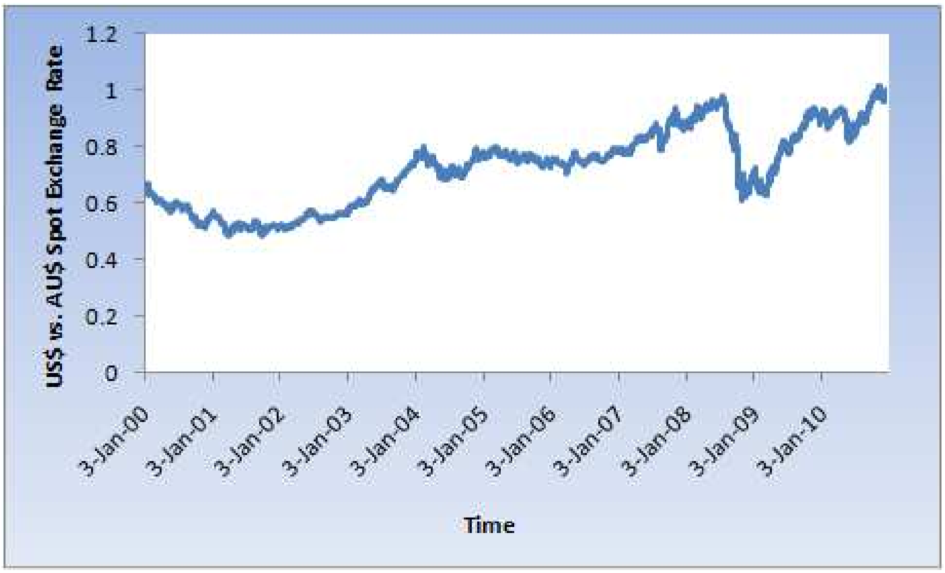

Figure 6.1: US$/AU$ Spot Exchange Rate from 03/01/2000 to 24/12/2010

6.1.1 Data Preprocessing

As the neural networks basically map the inputs and outputs, the data that should be fed into the neural network should be appropriate in the manner that it is already preprocessed. This means that data requires to be modified before feeding it to the neural network because only the necessary and the relevant patterns of the data must be learnt by the network. Also the data should be transformed such that the noise is minimized, thereby highlighting the important characteristics of the data, like we do the first differencing here in our study. The following steps have been followed in our study for the data to be accordingly transformed:

Missing Data Points: There are missing data points in the time series due to the presence of the non-occasional trading days. These missing data points can be handled in various ways, like omitting these missing data points, or by interpolation, or by taking the averages of the corresponding nearby values. In our study, we calculated the missing observations by taking the corresponding average of the immediate preceding and the succeeding values.

Removal of Outliers: In the datasets we encounter and work with and alike, we are ought to see some data points which does not match with the general behavior or the data model. These data points, which are not consistent with the remaining set of data, and are different are thus called outliers. Now, there is a specific method to check for the outliers in the data. We first calculate the first (Q1) quartile and the third (Q3) quartile of the data. For a distribution, the first quartile is defined as the 25th percentile and the third quartile is described as the 75th percentile respectively. The 25th percentile means that the value which the dataset into two subsets such that the first part contains the 25% of the data and the second part contains the remaining 75% of the data respectively. This logic would obviously define the 50th percentile as the value to be the median of the data. After calculating the first quartile (Q1) and the third quartile (Q3) for each dataset, we then find the corresponding value of Q3-Q1, which is known as the interquartile range (IQ). We then define an Upper Limit [Q3+3*IQ] and a Lower Limit [Q1-3*IQ]. Now, we call an extreme outlier to be any value in our dataset which is greater than the Upper Limit or lower than the Lower Limit of the corresponding of the dataset. We can thus eliminate the outliers from the data using this concept of Upper Limit and Lower Limit.

Differencing: Generally, the time series have some linear trends and these trends need to be removed. These trends can be generally removed by taking the first difference of the successive data points of the time series data. The method of first differencing is the most common step of the data transformation done before the processing. Sometimes, logarithmic transformation is also performed for the data sets which have both very small as well as very large extreme data points. After observing the time series datasets we have used for our study, we found that these time series datasets have trend which can be removed by taking the first difference. The first differencing results the time series in a stationary data series which makes it easier and comfortable in handling the data set for further processing. The first difference series of the data series US$/AU$ exchange rate is shown below in Figure 6.2.

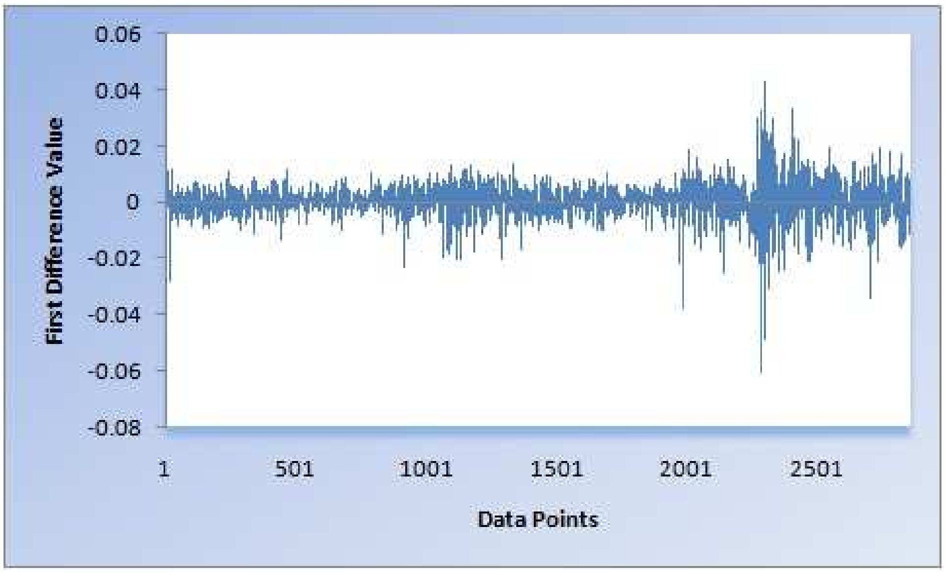

Figure 6.2: US$ vs. AU$ First Difference Series

Data Normalization: The data is to be further get normalized to match the actual output range. This depends on the activation function or the transfer function we have used. The neural networks which use the non-linear activation function in the output layer, there is a need of normalization to be done on the target data in order to match the range of the actual outputs. We have used the logistic activation function which has the typical range of [0, 1]. Therefore, the normalization of the data becomes necessary. There are three ways described in the literature to normalize the data into a specific range, which are:

1. Linear Transformation to [0, 1], where the maximum and the minimum value of the data is used for the data transformation process and it is given by $x_n = \left[\frac{(x_0 - x_{\min})}{(x_{\max} - x_{\min})} \right]$.

2. Statistical Normalization, where we use the mean and the standard deviation of the dataset for the data transformation process given by the following equation, $x_n = \left(\frac{x_0 - \bar{x}}{\sigma} \right)$.

3. Simple Normalization, in which we simply divide each data point with the maximum value of the corresponding data series to transform the data point between [-1, 1] using the equation $x_n = \left(\frac{x_0}{|x_{\max}|} \right)$.

In the above data transformation equations, $x_{\min}$, $x_{\max}$, $\bar{x}$ and σ denote the minimum, the maximum, the mean and the standard deviation of the corresponding data set respectively. While x_n represents the normalized data point and x_0 represents the original data point.

6.1.2 Data Post-Processing

The data post-processing is the comparison of the different model performances by the data outputs obtained. For performances comparison, we have different performances parameters as discussed in Chapter 5. These are Mean Square Error (MSE), Mean Absolute Error (MAE), Direction Accuracy (DA), Pearson Coefficient of Correlation (ρ) and Theil's Inequality Coefficient (U), etc..

6.1.3 Data Division

The neural network is a technique which is preferably used to generalize the relation between the inputs and output. Neural is a self-learning tool and is hardly used as a tool to memorize the data pattern; rather the main motive of using these networks remains the generalization of the pattern. One of the problems in neural networks is of overfitting which occurs when there are few number of observations with respect to the number of path weights and therefore the network memorizes the individual points rather than learning the general pattern. Thus, to keep the number of data points sufficient for training of the network, we utilize 2873 (2872) data points for all the data series and after removing the outliers, we take the first the first 70% of the data points to train the network. The next 10% of the data points is used for the validation and the remaining 20% of the data is utilized for the testing of the trained/optimized network.

6.2 Simulation

The different models discussed are compared on the basis of the evaluation criteria which are as mentioned in Chapter 5. The results obtained from the analysis are found in accordance to the expectations. We expect that the forecast errors should reduce as we incorporate more complex algorithms to train our neural network. The errors obtained from the benchmark model i.e. Neural Network optimized with Conjugate Gradient Method (NN_CGM), should reduce when we implement the Neural Network path weight optimization using Genetic Algorithm (NN_GA) and then when we do the Neural Network path weight optimization using Simulated Annealing (NN_SA).

In Section 6.2.1, we present the results of the autocorrelation of US$/AU$ time series as an example and the significant lags obtained through it. In the subsequent section 6.2.2, we compare the results obtained from the various models which employ different training algorithms.

6.2.1 Autocorrelation and Significant Lags

After the treatment of the missing values and correspondingly replacing them in the preprocessing process and taking the first difference, we now have to find the autocorrelation of the time series data to get the significant lags. We also tried to find the number of significant lags from the second difference data series, i.e. we obtained the second difference time series data from the first difference time series by again doing first differencing of the successive data points, and then calculated the autocorrelation from this series. But for our study, it was found that the significant lags obtained from the first difference time series were more appropriate than those obtained from the second difference time series. The corresponding autocorrelation values for the US$/AU$ spot exchange rate is shown in the following Figure 6.3.

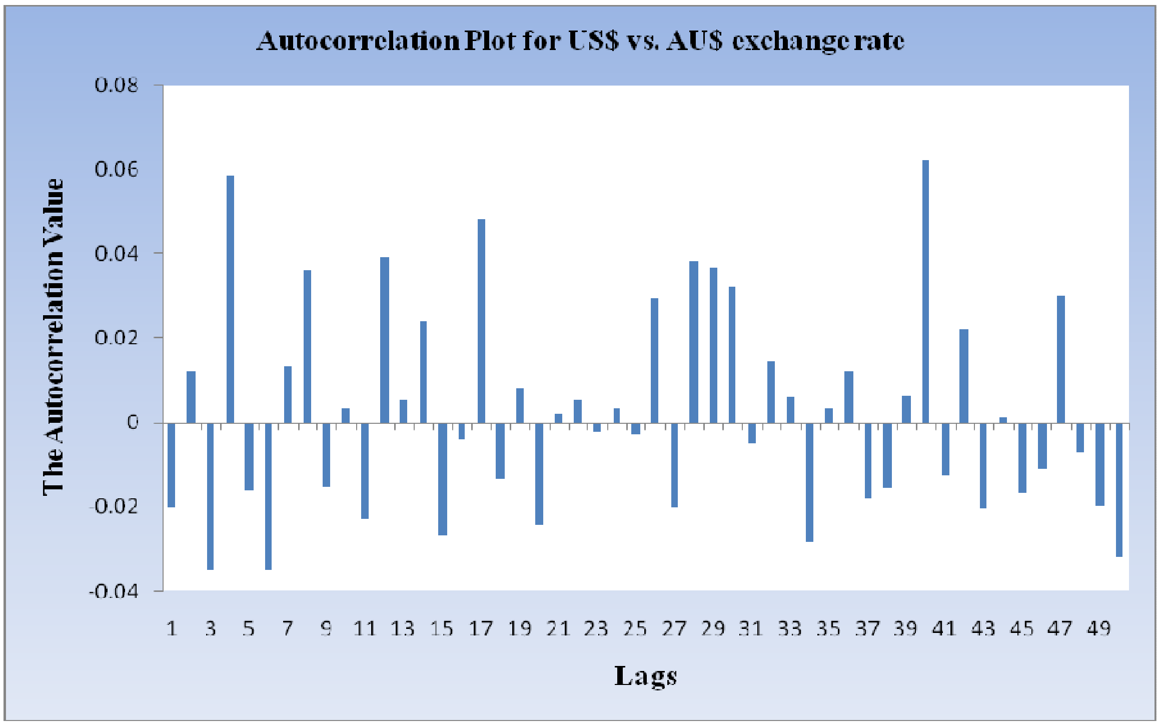

Figure 6.3: The Autocorrelation values for US$/AU$ Spot Exchange rate

From the graph above, we can clearly see that the significant lags for the US$/AU$ Spot exchange rate time series data are 4, 12, 17, 28 and 40. Similarly, we calculate the corresponding autocorrelation values from the first difference time series of the other spot exchange rate data sets and then find out the corresponding significant lags.

After obtaining the number of significant lags, which for example, for the data series US$/AU$ is 5, we form a lag matrix of the size (5 X 2872), where 2872 is the number of the data points in the data set. This matrix is then fed into the various models and the results thus obtained are then compared amongst themselves.

6.2.2 Results from Non-Parametric Models

For our models, the performance is mainly measured in terms of the Mean Square Error (MSE). We have presented the performance of our various models by MSE and other different comparison performance metrices which are Mean Absolute Error (MAE), Direction Accuracy (DA), Pearson Correlation Coefficient (Rho) and Theil's Inequality Coefficient (U). We hereby give the Error metrics tables and the comparison by MSE and other error indices for all of our 12 time series data. The final results of the five performance metrics for all the models, considering US$/AU$, are summarized in Table 6.1.

Metrics of comparison	**Models**		
	NN_CGM	**NN_GA**	**NN_SA**
Mean Square Error (MSE)	0.0374	0.0119	0.0539
Mean Absolute Error (MAE)	0.1560	0.0840	0.01914
Direction Accuracy (DA)	0.5028	0.4932	0.4972
Pearson Correlation (ρ)	0.9256	0.9932	1.0334
Theil's Inequality coefficient (U)	0.3609	0.2130	0.4335

Table 6.1: Comparison of Performance using various performance metrics for US$/AU$

From the table above, we can easily observe that the Mean Square Error (MSE) value is the least for the NN_GA model, while the highest value is obtained when we use NN_SA model. The model NN_CGM gives the value in between the two. In the case of Mean Absolute Error (MAE), we see that the minimum value is obtained when we use NN_SA, and the model NN_CGM performs the worst of the three. While comparing Direction Accuracy (DA), the results again suggest that the NN_GA model again outperforms the other two models giving the least value. For comparing Pearson Coefficient of Correlation (Rho), surprisingly, NN_CGM gives the least value. Whereas if we see the Theil's Inequality Coefficient (U), the values again prove that NN_GA is the best in terms of results.

The results obtained and explained above can be better understood and visualized when reperesnted in the form of the following bar graphs where we first compare the Mean Square Error (MSE) individually and then, we give the comparison using all the five error metrics.

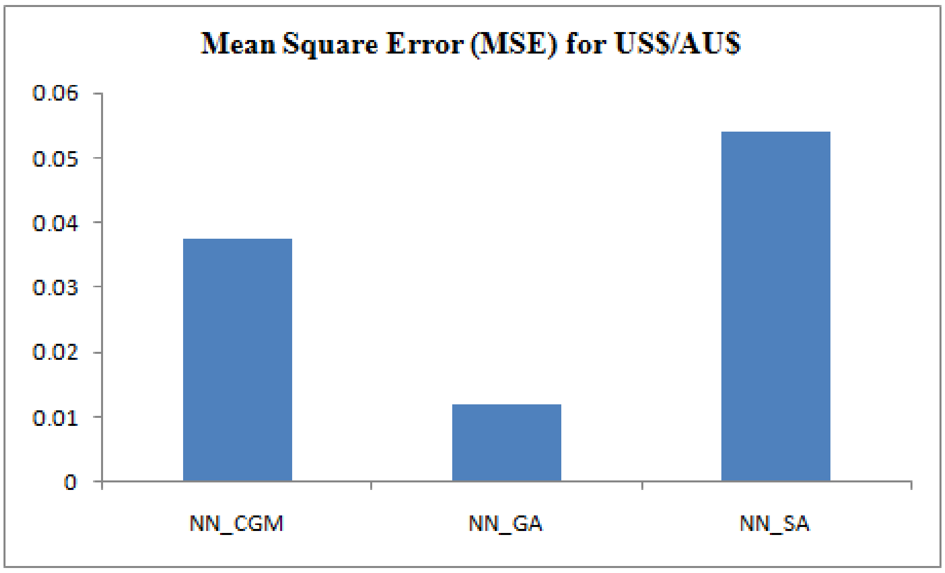

Figure 6.4: Comparison of Performance by MSE for data series US$/AU$

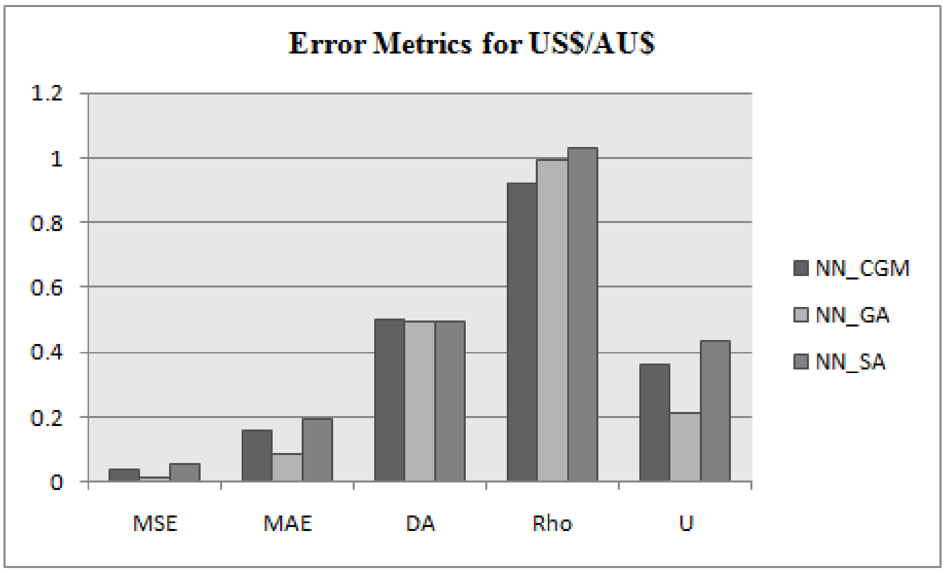

Figure 6.5: Comparison of Performance using various Error Metrics for data series US$/AU$.

Similarly, for the other time series data, we calculate the significant lags from the autocorrelation values and form the corresponding lag matrix which s fed into different models to give corresponding results. For the next time series data, i.e. US$/CAN$, the significant lags obtained are 3, which are 6, 9 and 29. The results thus obtained in terms of the performance metrics obtained are given in Table 6.2.

Metrics of comparison	**Models**		
	NN_CGM	**NN_GA**	**NN_SA**
Mean Square Error (MSE)	0.0251	0.0139	0.0584
Mean Absolute Error (MAE)	0.1237	0.0931	0.2000
Direction Accuracy (DA)	0.4430	0.4982	0.5478
Pearson Correlation (ρ)	1.0655	0.9766	0.8769
Theil's Inequality coefficient (U)	0.3076	0.2344	0.4692

Table 6.2: Comparison of Performance using various performance metrics for US$/CAN$

Here also, we can see the same trend being repeated of NN_GA model proving to be the best of the three in case of the main comparison parameter i.e. Mean Square Error (MSE). However, here it gives the minimum value also in the case of Mean Absolute Error (MAE) and Theil's Inequality Coefficient (U). Unlike US$/AU$, the NN_CGM proves to be the best comparing Direction Accuracy (DA) while NN_SA outperforms the other in case of Pearson Correlation (Rho). Again, following are the representation of the results in terms of bar graph plots for better visualization.

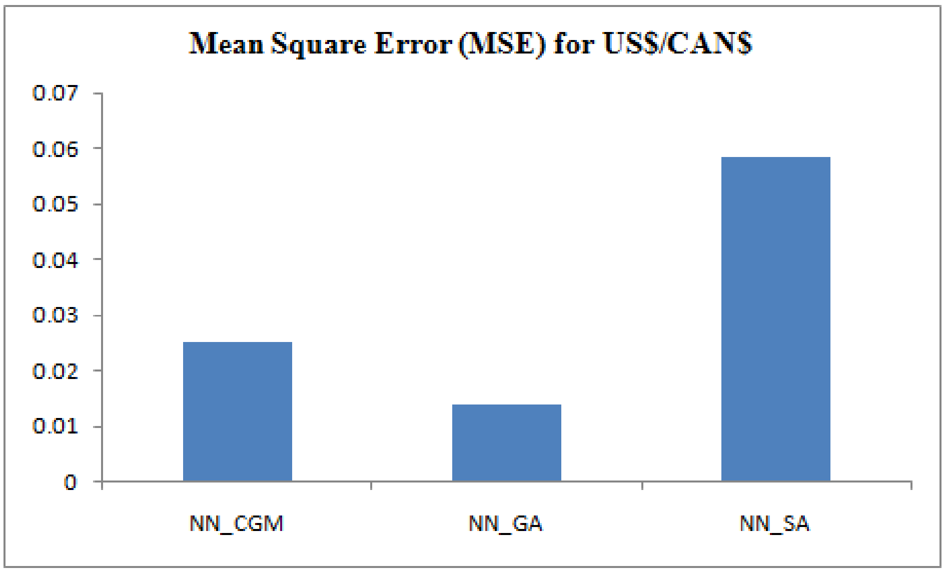

Figure 6.6: Comparison of Performance by MSE for data series US$/CAN$

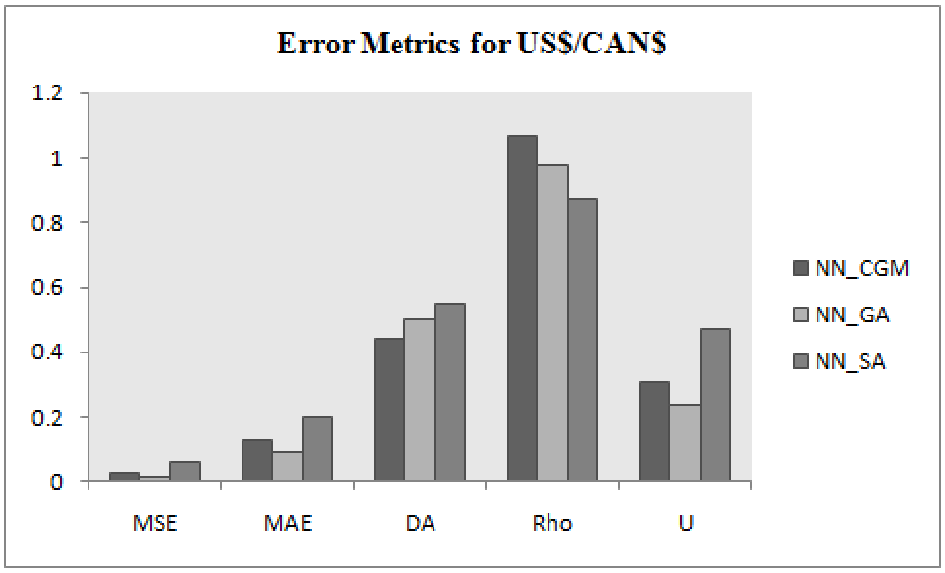

Figure 6.7: Comparison of Performance using various Error Metrics for data series US$/CAN$.

For the third time series data US$/EURO, the significant lags obtained are again 3, which are 4, 7 and 32. The results obtained as performance metrics are as follows in Table 6.3.

Metrics of comparison	Models		
	NN_CGM	NN_GA	NN_SA
Mean Square Error (MSE)	0.0301	0.0147	0.0419
Mean Absolute Error (MAE)	0.1387	0.0957	0.1671
Direction Accuracy (DA)	0.4881	0.5110	0.5170
Pearson Correlation (ρ)	0.9438	0.9599	1.0134
Theil’s Inequality coefficient (U)	0.3180	0.2289	0.3754

Table 6.3: Comparison of Performance using various performance metrics for US$/EURO

The bar graph plots comparing the Mean Square Error (MSE) individually and all the error metrics combined are as follows:

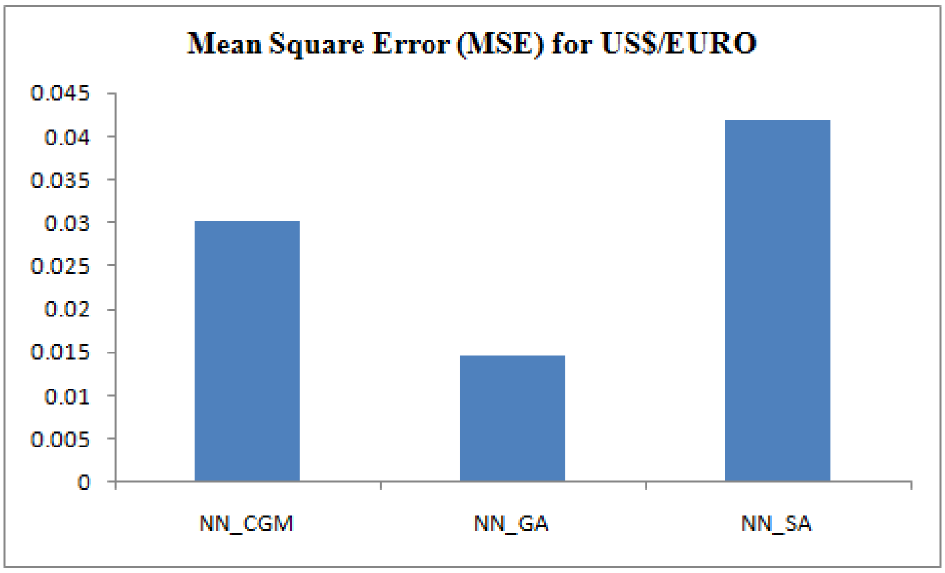

Figure 6.8: Comparison of Performance by MSE for data series US$/EURO

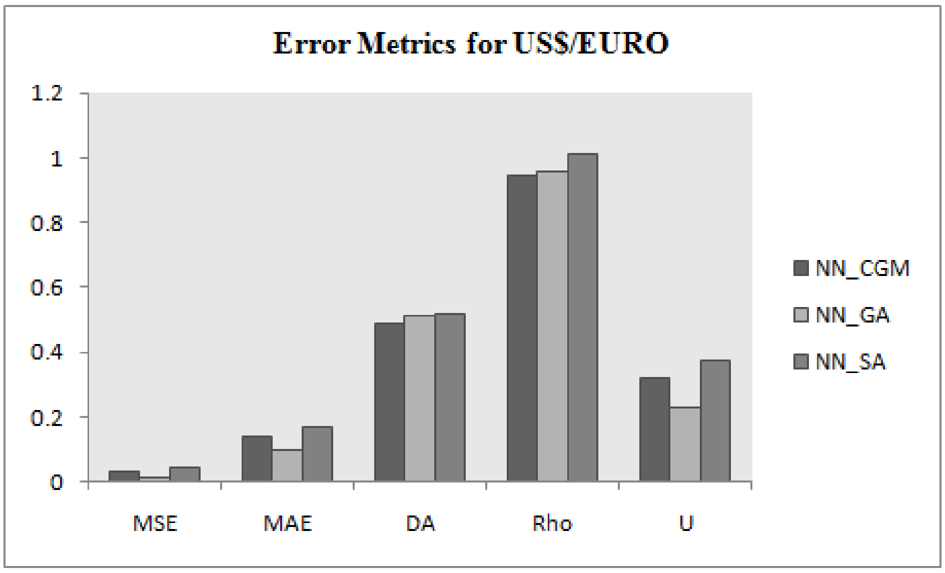

Figure 6.9: Comparison of Performance using various Error Metrics for data series US$/EURO.

Here also the NN_GA model performs the best in case of comapring Mean Square Error (MSE), MAE and U. However, surprisingly, the NN_CGM models performs better than the other two in case of the other two performance indices DA and Rho.

For the fourth data series, i.e. US$/HK$, the significant lags obtained are 5 again, which are 2, 8, 15, 23, 38. The results obatined are as follows:

Metrics of comparison	Models		
	NN_CGM	NN_GA	NN_SA
Mean Square Error (MSE)	0.0300	0.0085	0.0426
Mean Absolute Error (MAE)	0.1385	0.0621	0.1716
Direction Accuracy (DA)	0.5735	0.4298	0.5499
Pearson Correlation (ρ)	0.9612	1.0439	0.9693
Theil's Inequality coefficient (U)	0.3182	0.1761	0.3851

Table 6.4: Comparison of Performance using various performance metrics for US$/HK$

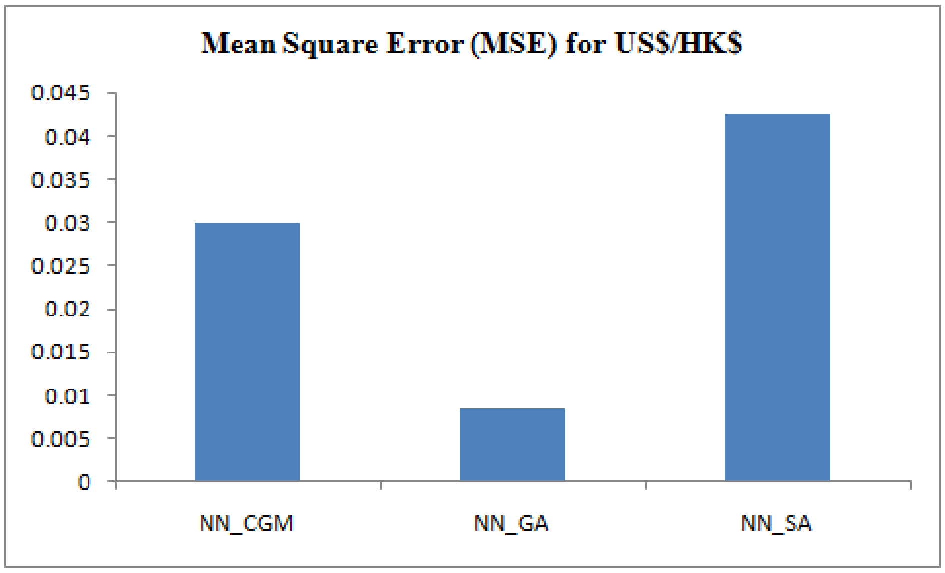

Figure 6.10: Comparison of Performance by MSE for data series US$/HK$

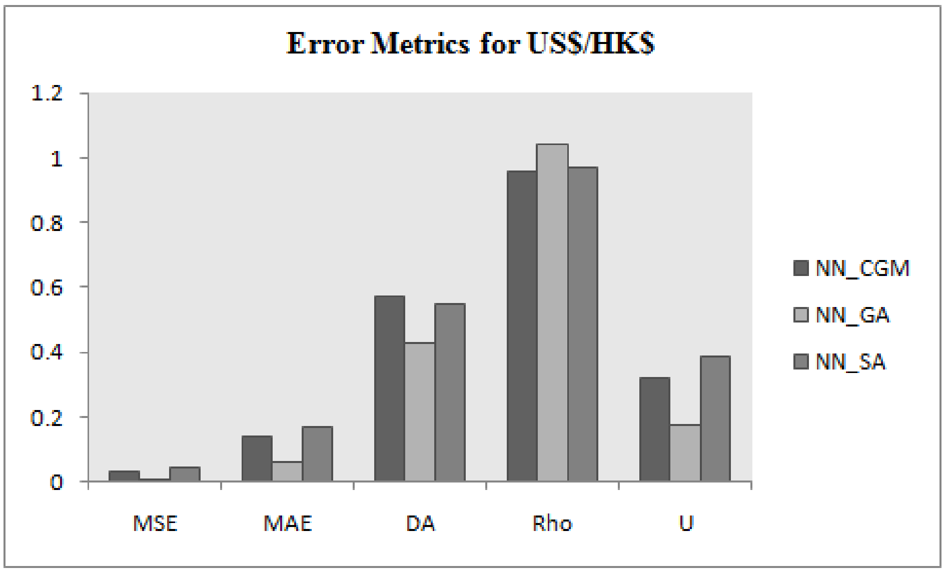

Figure 6.11: Comparison of Performance using various Error Metrics for data series US$/HK$.

For the fifth data time series US$/JPY, the significant lags obtained are 15, 21, 31, 40, 48.

Metrics of comparison	**Models**		
	NN_CGM	**NN_GA**	**NN_SA**
Mean Square Error (MSE)	0.0213	0.0240	0.0925
Mean Absolute Error (MAE)	0.1132	0.1233	0.02713
Direction Accuracy (DA)	0.4396	0.5311	0.4707
Pearson Correlation (ρ)	1.0239	0.9164	1.0247
Theil's Inequality coefficient (U)	0.2606	0.2751	0.5431

Table 6.5: Comparison of Performance using various performance metrics for US$/JPY

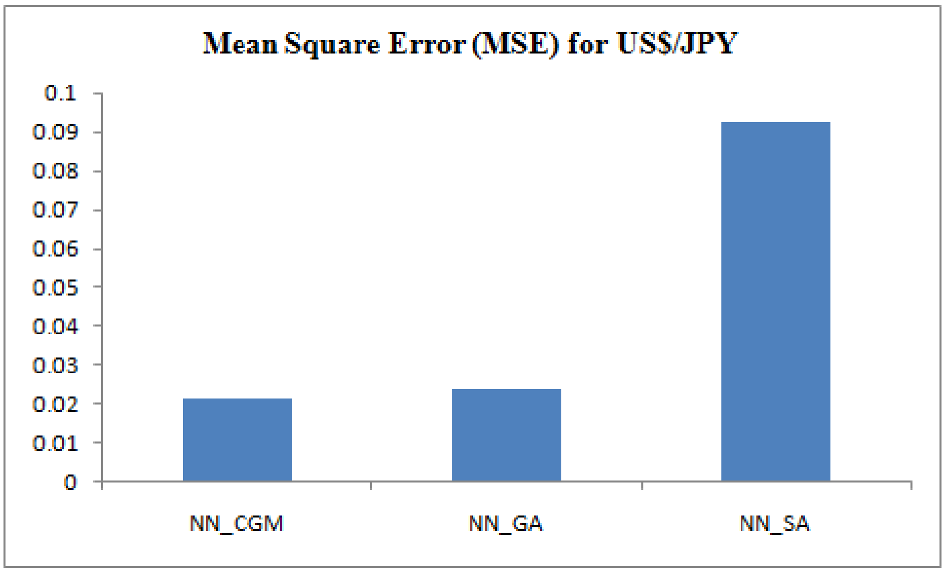

Figure 6.12: Comparison of Performance by MSE for data series US$/JPY

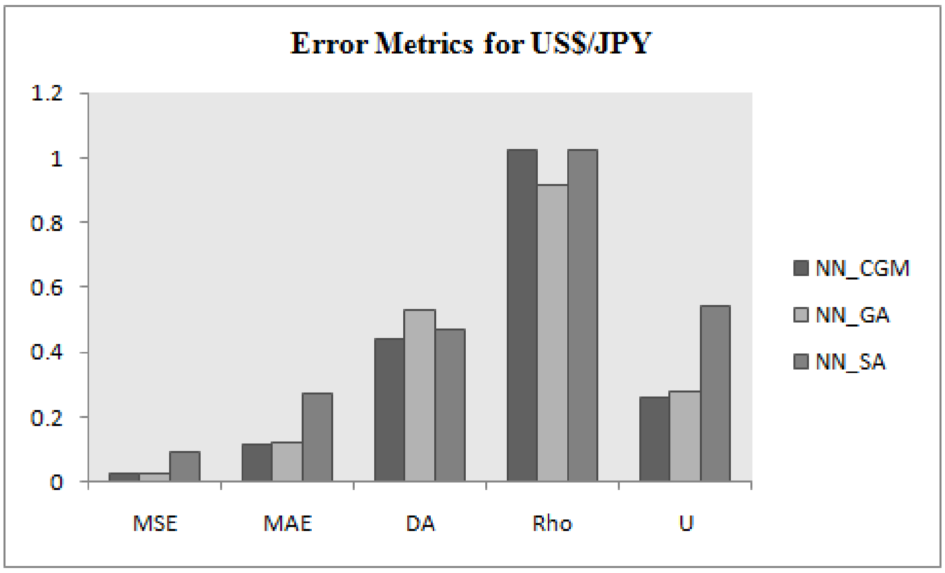

Figure 6.13: Comparison of Performance using various Error Metrics for data series US$/JPY

However, here we see an exception from the previous trend. The best performing model for this time series data comes out to be NN_CGM when comparing MSE. It is also the same for the other indices MAE, DA and U except for Rho, where NN_GA turns out to be the best.

For the sixth time series data i.e. US$/MXP, the significant lags obtained are again 5,which are 5, 13, 17, 26, 32. The results obtained in the form of performance indices and bar graph plots are shown as follows:

Metrics of comparison	**Models**		
	NN_CGM	**NN_GA**	**NN_SA**
Mean Square Error (MSE)	0.0286	0.0170	0.0463
Mean Absolute Error (MAE)	0.1326	0.0988	0.1775
Direction Accuracy (DA)	0.5224	0.5124	0.4832
Pearson Correlation (ρ)	1.0320	1.0241	1.0317
Theil's Inequality coefficient (U)	0.3102	0.2440	0.3951

Table 6.6: Comparison of Performance using various performance metrics for US$/MXP

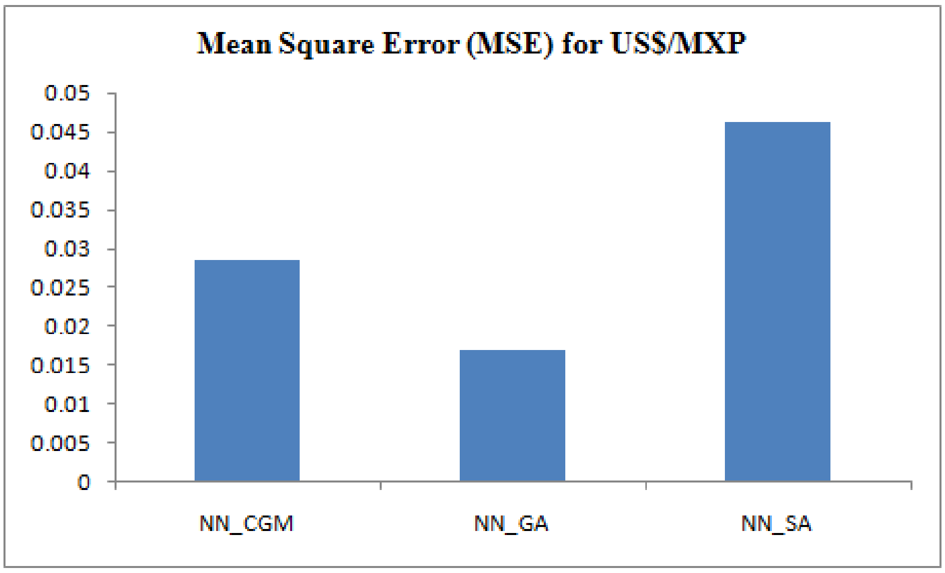

Figure 6.14: Comparison of Performance by MSE for data series US$/MXP

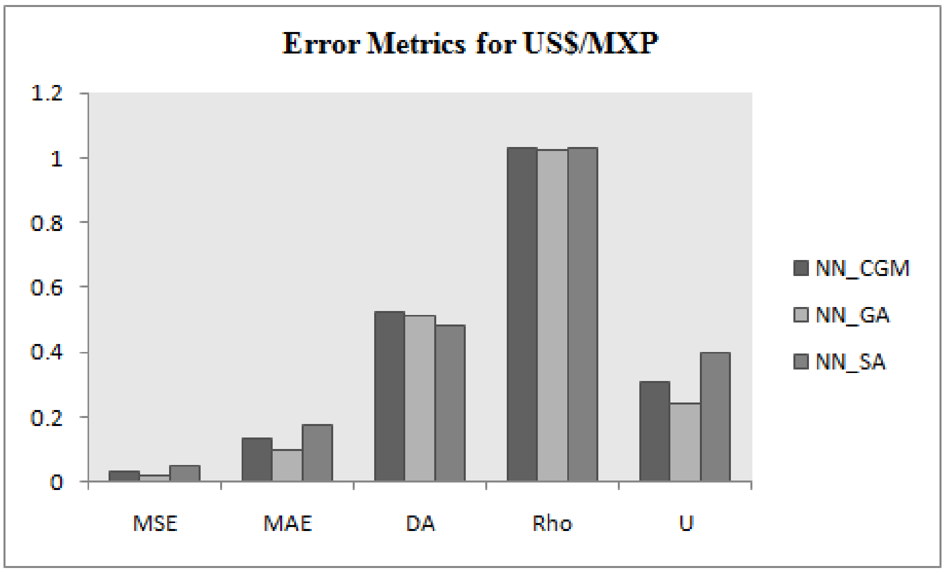

Figure 6.15: Comparison of Performance using various Error Metrics for data series US$/MXP

Here, NN_GA again turns out to be the best performing model in all the cases of performance indices except for the one which is DA. In case of DA, NN_SA proves to be the best.

For the seventh time series i.e. US$/NZ$, the significant lags are 3, which are 3, 29 and 40. The results thus obtained in terms of the performance indices and the corresponding bar graph plots are shown as:

Metrics of comparison	**Models**		
	NN_CGM	**NN_GA**	**NN_SA**
Mean Square Error (MSE)	0.0342	0.0112	0.1401
Mean Absolute Error (MAE)	0.1488	0.0810	0.3292
Direction Accuracy (DA)	0.4833	0.4630	0.4907
Pearson Correlation (ρ)	1.0141	1.0322	1.0016
Theil's Inequality coefficient (U)	0.3691	0.2217	0.7474

Table 6.7: Comparison of Performance using various performance metrics for US$/NZ$

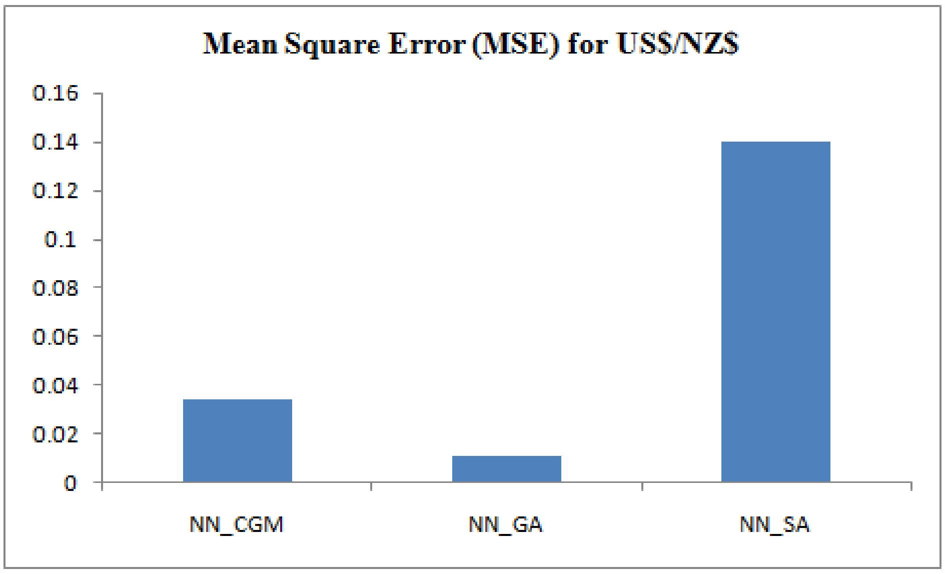

Figure 6.16: Comparison of Performance by MSE for data series US$/NZ$

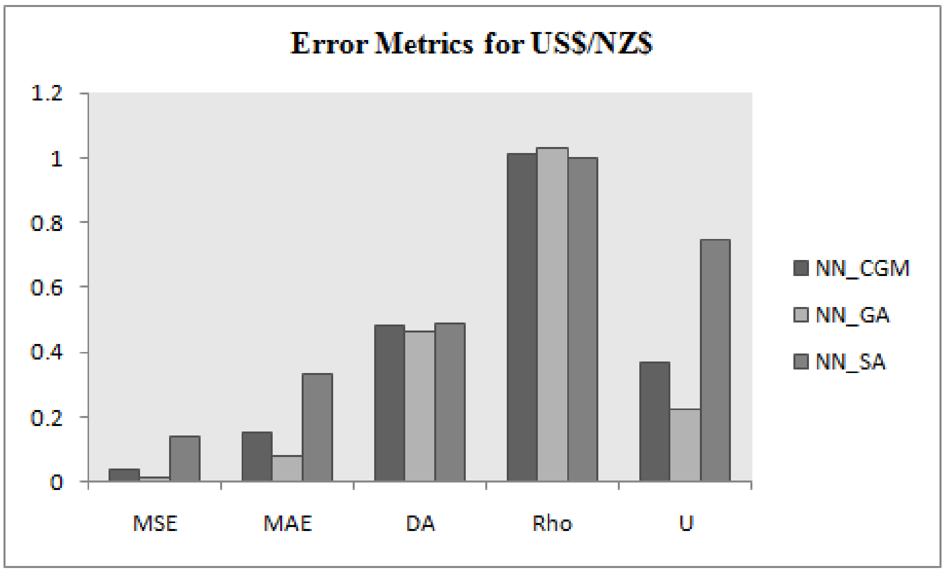

Figure 6.17: Comparison of Performance using various Error Metrics for data series US$/NZ$

For the eighth time series data, US$/SING$, the significant lags obtained are 3 again, 4, 7 and 25.

Metrics of comparison	**Models**		
	NN_CGM	**NN_GA**	**NN_SA**
Mean Square Error (MSE)	0.0235	0.0209	0.0650
Mean Absolute Error (MAE)	0.1194	0.1122	0.2177
Direction Accuracy (DA)	0.4945	0.4909	0.5255
Pearson Correlation (ρ)	0.8583	1.0575	0.9563
Theil's Inequality coefficient (U)	0.2797	0.2654	0.4650

Table 6.8: Comparison of Performance using various performance metrics for US$/SING$

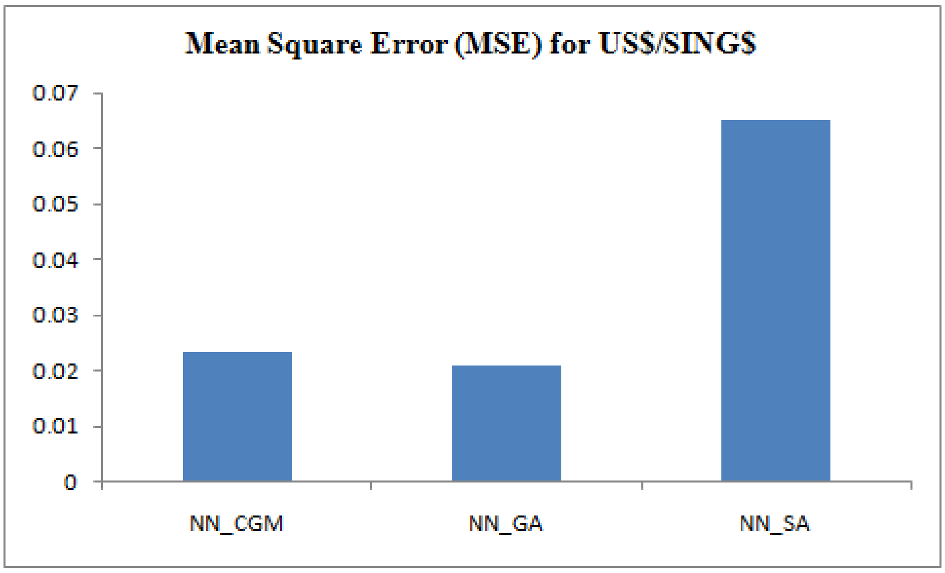

Figure 6.18: Comparison of Performance by MSE for data series US$/SING$

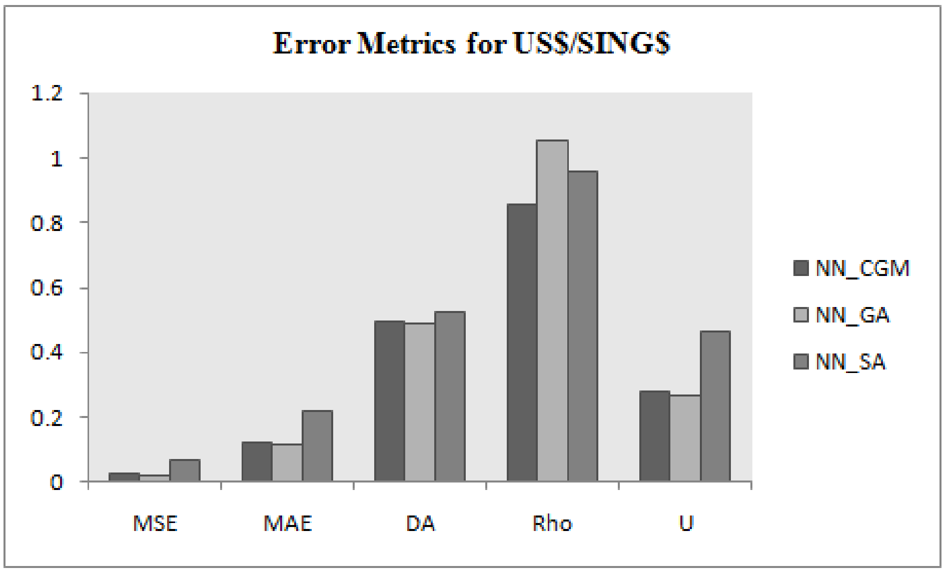

Figure 6.19: Comparison of Performance using various Error Metrics for data series US$/SING$

Here also, NN_GA proves to be the best model in terms of results except in case of index Rho.

For the ninth time series data US$/SKW, the significant lags obtained are 3, 7, 10, 14 and 31. The results thus obtained in the form of performance metrics and the bar graph plots are as below:

Metrics of comparison	**Models**		
	NN_CGM	**NN_GA**	**NN_SA**
Mean Square Error (MSE)	0.0153	0.0227	0.0231
Mean Absolute Error (MAE)	0.0941	0.1104	0.1225
Direction Accuracy (DA)	0.4556	0.5265	0.4813
Pearson Correlation (ρ)	0.8313	0.9504	1.0215
Theil's Inequality coefficient (U)	0.2459	0.2956	0.3021

Table 6.9: Comparison of Performance using various performance metrics for US$/SKW

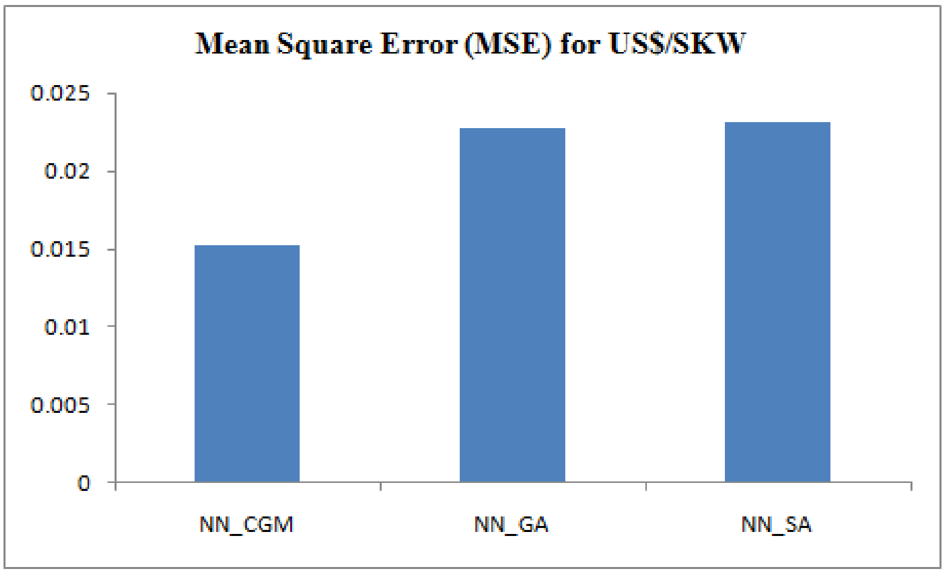

Figure 6.20: Comparison of Performance by MSE for data series US$/SKW

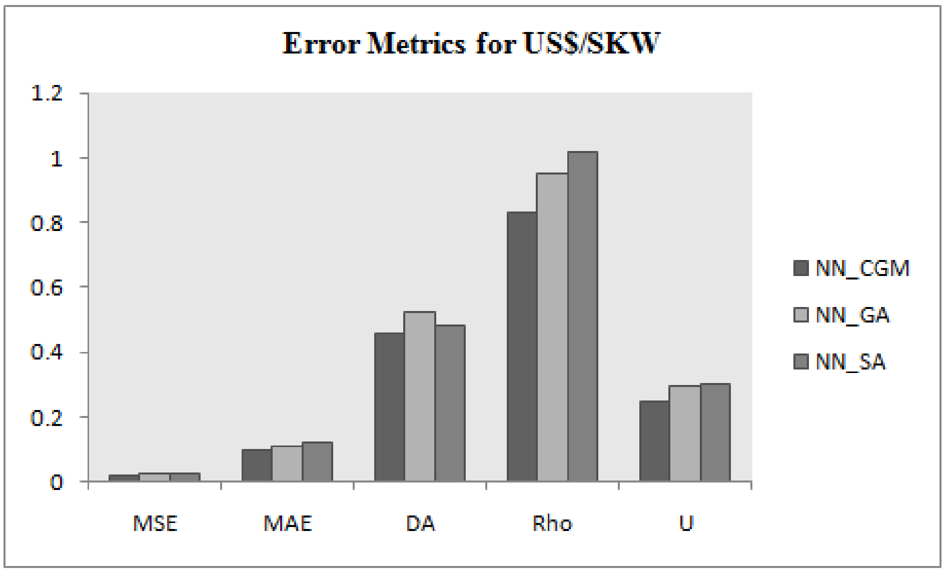

Figure 6.21: Comparison of Performance using various Error Metrics for data series US$/SKW

This time series data also proves to be an exception to the observed trend. Instead of NN_GA, the conventional NN_CGM model turns out to be the best performing with respect to all the five performance indices.

For the tenth time series data, i.e. US$/SWKR, the significant lags observed are 2, 5, 7, 18 and 32. The results thus obtained in terms of the performance indices and bar graph plots are following:

Metrics of comparison	**Models**		
	NN_CGM	**NN_GA**	**NN_SA**
Mean Square Error (MSE)	0.0335	0.0261	0.0339
Mean Absolute Error (MAE)	0.1468	0.1282	0.1474
Direction Accuracy (DA)	0.5101	0.4816	0.5562
Pearson Correlation (ρ)	0.9553	1.0070	0.8793
Theil's Inequality coefficient (U)	0.3587	0.3214	0.3609

Table 6.10: Comparison of Performance using various performance metrics for US$/SWKR

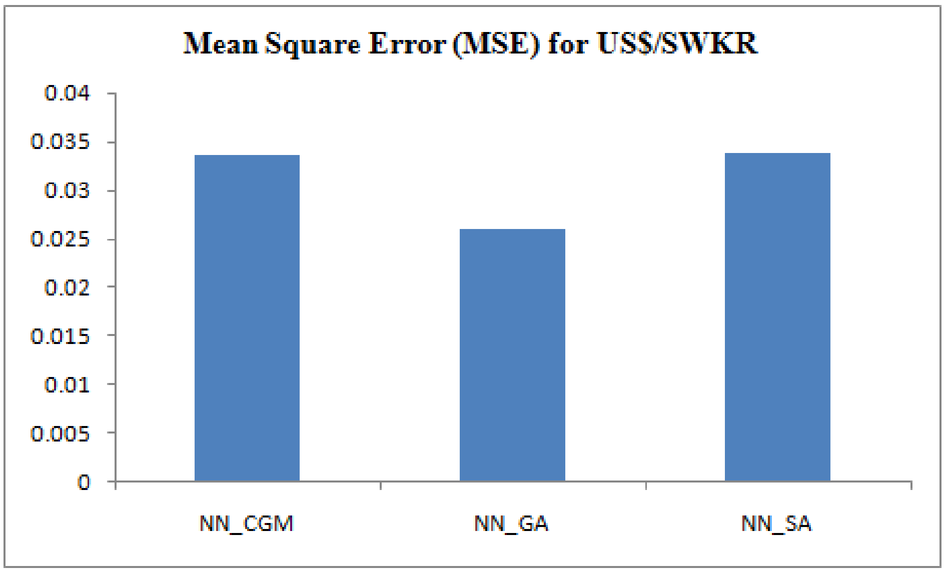

Figure 6.22: Comparison of Performance by MSE for data series US$/SWKR

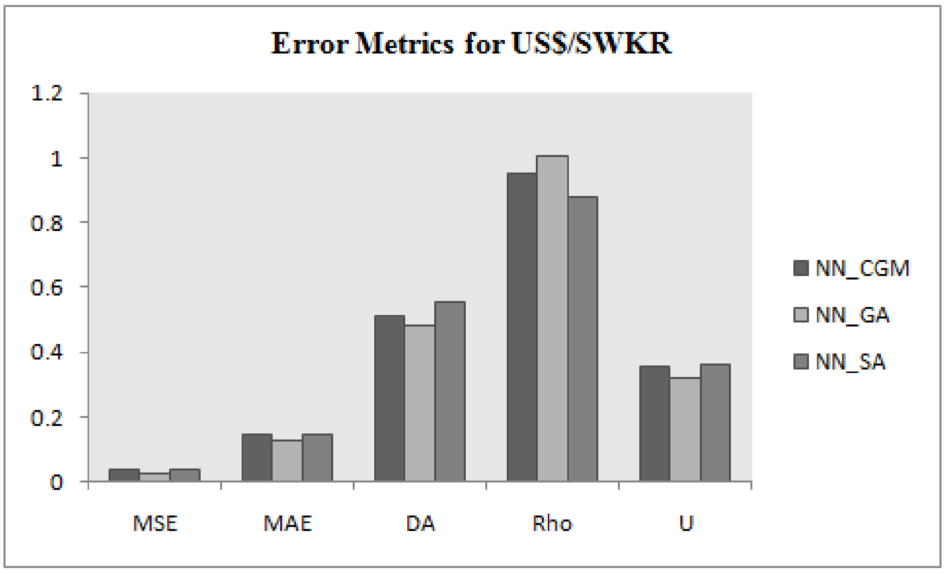

Figure 6.23: Comparison of Performance using various Error Metrics for data series US$/SWKR

Here, the NN_GA model gives the least value in terms of all the error indices except in the case of the Pearson Correlation Coefficient (Rho).

For the eleventh time series data US$/TW$, the significant lags are 2, 5, 11, 18 and 23. The results are:

Metrics of comparison	**Models**		
	NN_CGM	**NN_GA**	**NN_SA**
Mean Square Error (MSE)	0.0304	0.0147	0.0974
Mean Absolute Error (MAE)	0.1365	0.0884	0.2716
Direction Accuracy (DA)	0.5106	0.4648	0.4933
Pearson Correlation (ρ)	0.9404	1.0644	0.9186
Theil's Inequality coefficient (U)	0.3233	0.2316	0.5793

Table 6.11: Comparison of Performance using various performance metrics for US$/TW$

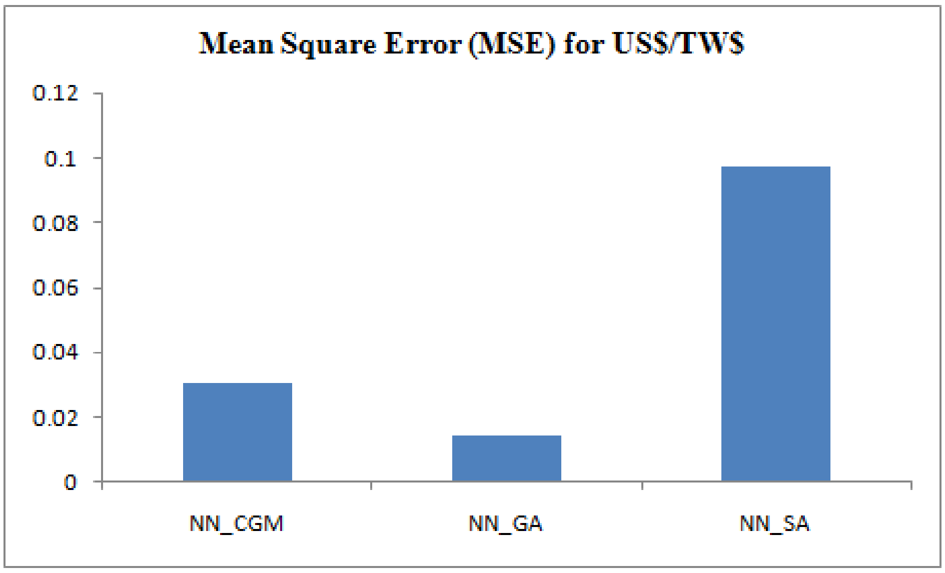

Figure 6.24: Comparison of Performance by MSE for data series US$/TW$

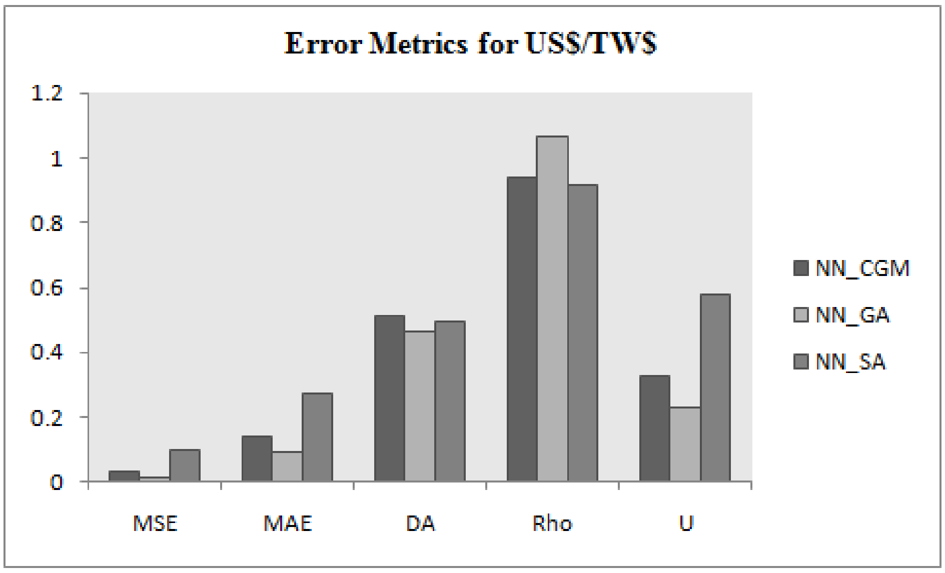

Figure 6.25: Comparison of Performance using various Error Metrics for data series US$/TW$.

Here also, the results obtained are similar to the previous time series data i.e. US$/SWKR, i.e. NN_GA turns out to be the best performing model.

For the last, i.e. twelth time series data, the significant lags obtained are 7, 11, 14, 22 and 27. The results obtained in terms of performance metrics and the bar graph plots are displayed below:

Metrics of comparison	**Models**		
	NN_CGM	**NN_GA**	**NN_SA**
Mean Square Error (MSE)	0.0310	0.0144	0.0346
Mean Absolute Error (MAE)	0.1392	0.0921	0.1499
Direction Accuracy (DA)	0.5145	0.4982	0.4800
Pearson Correlation (ρ)	0.9262	1.0092	0.9981
Theil's Inequality coefficient (U)	0.3437	0.2424	0.3631

Table 6.12: Comparison of Performance using various performance metrics for US$/UKP

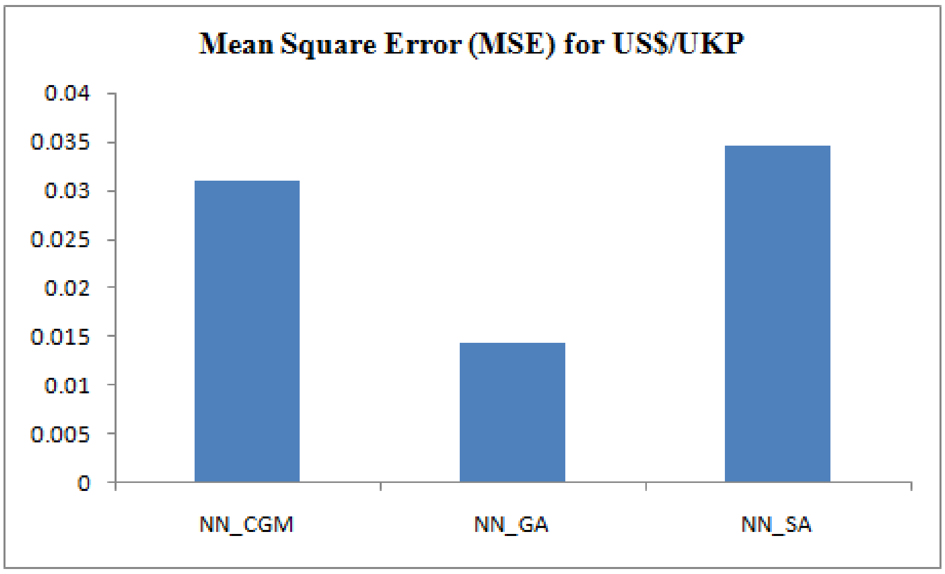

Figure 6.26: Comparison of Performance by MSE for data series US$/UKP

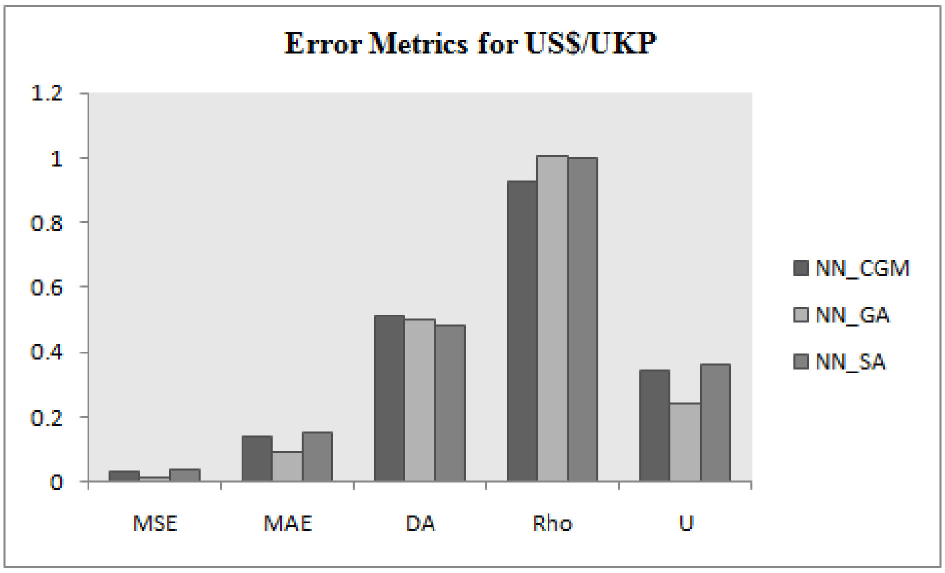

Figure 6.27: Comparison of Performance using various Error Metrics for data series US$/UKP.

Thus, here also, NN_GA model is performing as the best giving the least value of Mean Square Error (MSE), MAE and U. Thus, it can be concluded that by far from the above time series data and among the models implemented, NN_GA proves to be the best model in terms of most of the performance indices used. Thus, the neural network trained with weights optimized with the elp of Genetic Algorithm serves as the most successful model for the prediction of spot exchange rate in case of our study. However, it was also expected that NN_SA, that is, the one trained with Simulated Annealing algorithm will also perform greatly which is not observed as so. It performs badly than even our conventional NN_CGM model, with respect to Mean Square Error (MSE) and also other indices.

6.2.3 Results from Parametric Models

For the parametric model GARCH of our analysis, the four parameters r, m, p and q are varied each able to take values from 1 to 2. This gives rise to 16 possible combinations. The time series data is forecasted using all the 16 configurations and the best one is chosen among them. In our study, for the time series data US$/AU$, the best or the optimum results were obtained for the configuration r=2, m=1, p=2 and q =1. This configuration can also be written as AR(2) MA (1) GARCH (2, 1). The results in the form of error metrics can be displayed as below:

Metrics of comparison	**Model: ARMA-GARCH (2,1,2,1)**
Mean Square Error (MSE)	1.0001
Mean Absolute Error (MAE)	1.0000
Direction Accuracy (DA)	0.6714
Pearson Correlation (ρ)	0.6161
Theil's Inequality coefficient (U)	0.9999

Table 6.13 Performance Measurement using various Error Metrics for ARMA-GARCH model.

It can be clearly seen from the table above and comparing the results from the non-parametric models above that there is a great difference in the performance of the parametric and non-parametric models. For the main performance index, i.e. Mean Square Error (MSE), where the value for all the non-parametric models varies in the low ranges of 0.01-0.05, the corresponding MSE value for the statistical GARCH model is quite high, i.e. close to 1. The same is the observation with the next error metric i.e. Mean Absolute Error (MAE). However the values are quite comparable in the case of the other three performance error metrices. Thus, it can be concluded that the non-parametric models outperform the parametric models.

CHAPTER 7

CONCLUSION AND FUTURE POSSIBILITIES

It has been of great concern for the financial analysts to develop efficient models for the prediction of foreign exchange rate. In the financial market, the trading of foreign currency itself is one of the biggest market out of which most of the trading occurs in the spot market. Thus, the spot exchange rate prediction has been one of the most important issues in financial analysis. We, through our study, have tried to deal with the same issue and we have used artificial neural networks to predict the foreign exchange rate time series.

The Artificial Neural networks have wide variety of applications in the prediction of many time series similar to those of foreign exchange rate. We deal with the following issues regarding the forecasting methodology based on the neural networks. We employed different training algorithms for the training of the neural networks to generate different models for the financial time series forecasting. We have compared the conjugate gradient method, which is the most conventional algorithm used to train the neural network, with the Genetic Algorithm and the Simulated Annealing Algorithm. The second issue addressed in our work or study was to optimize the weights connecting the different layers, with the genetic algorithm and the simulated annealing algorithm, of the neural network which is trained using Conjugate Gradient Method. Thus, the three models proposed are thereby compared and the comparison is done on the basis of different performance indices suggested. We have also used the statistical nonlinear parametric model ARMA-GARCH which captures the nonlinear properties of the financial time series to compare the results with the above non-parametric models. Our results show that the model with weights of the neural network optimized using Genetic Algorithm and trained with Conjugate Gradient Method gives the best of the results among the non-parametric models for most of the data sets and that all the non-parametric models outperform the parametric models.

These different issues addressed in our study can improve the performance of forecasting of the neural networks. But, there is a further scope of improvement of the results by considering many other issues related to the neural networks. For the training and optimizing of the architecture, the parameters for genetic algorithms such as the number of generations, population size, crossover probability, etc. are obtained by hit and trial and offer poor justification. We can make the genetic algorithm itself to evolve these parameters simultaneously with training or optimization of networks. Such genetic algorithms are also known as the Evolutionary Genetic Algorithms. Also, for the initial connection weights, creation of population and mutation, genetic algorithm uses random number generators. This stochastic nature of the genetic algorithm and the neural networks make the results precisely, not reproducible. This can be improved by making improving the random nature of the generators or the number of runs is increased and the predicted value is the mean of all the runs taken. We can also use some other different optimization algorithms like Particle Swarm Optimization (PSO), Tabu Search (TS) Algorithm, Artificial Immune System (AIS), and Ant Colony Optimization (ACO) for the training of the neural networks. The data set forecasting (i.e. inherent input and output values) is based on the concept of technical analysis while it has been seen that a combination of technical factors, fundamental factors and the inter-market data can give a better prediction of the time series. Also, we believe that better results can be obtained by applying other training algorithms, like Quasi Newton Algorithm and Levenberg-Marquadt in Artificial Neural Networks.

REFERENCES

1. Bollerslev, T., (1986), Generalized Autoregressive Conditional Heteroscedasticity. *Journal of Econometrics*, **31**,307-327.
2. Deb, K. (2002), Multi-Objective Optimization using Evolutionary Algorithms, *John Wiley & Sons, Ltd.*
3. Cottrell, M., Girard, Y., Mangeas M. and Muller C., (1995), Neural modelling for time series: a statistical stepwise method for weight elimination. *IEEE Transactions on Neural Networks*, **6**, 1355-1364.
4. Borisov, A.N. and Pavlov, V.A., (1995), Prediction of a continuous function with the aid of neural networks. *Automatic Control and Computer Sciences* **29**, 39–50.
5. Goldberg, D. E. (1989), Genetic Algorithms in Search , optimization and machine learning, *Kluwer Academic publishers.*
6. Kirkpatrick, S., Gelatt, C. D. and Vecchi, M. P., (1983), Optimization by Simulated Annealing. *Science,* **220**, 671-680.
7. Kuan, C.-M. and Liu, T., (1995), Forecasting exchange rates using feedforward and recurrent neural networks. *Journal of Applied Econometrics* **10**, 347–364.
8. Lapedes A., and Farber R., (1987), Nonlinear signal processing using neural networks: prediction and system modeling. Technical Report LS-UR-87-2662, *Los Alamos National Laboratory*, NM, USA.
9. Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A. and Teller, H., (1953), “Equation of State Calculations by Fast Computing Machines”. *Journal of Chemical Physics*, **21**, 1087-1092.
10. Werbos P., (1974), Beyond regression; New tools for prediction and analysis in the behavioural sciences, *Dissertation Harvard University*, MA, USA.