ARTIFICIAL NEURAL NETWORKS APPLIED TO OPTION PRICING

Zaheer Ahmed Dindar

A Dissertation submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, in fulfilment of the requirements of the degree of Master of Science.

Johannesburg 2004

Preface

In the name of God, the Beneficent, the Merciful.

The work described in this dissertation was carried out in the University of Witwatersrand, School of Electrical and Information Engineering in 2004.

I would like to acknowledge a few people who made this thesis a possibility. Firstly, I'd like to thank my parents, where do I start? For unending support through this, and every other year of my studies, for the belief in me and kick in the behind whenever I needed it. Secondly, I'd like to thank my supervisor Prof Tshilidzi Marwala *OMB* for all the support and guidance during this year, financial and otherwise. Without Prof this really couldn't have been done. Thanks again.

Except where reference is made to the work of others, this thesis is the result of my own original work. No part of this work has already been, or is currently being, submitted for any other degree, diploma or other qualification. This thesis is 104 pages in length with approximately 21653 words.

Zaheer A. Dindar
2004

Abstract

Artificial Neural Networks has seen tremendous growth in recent years. It has been applied to various sciences, including applied mathematics, chemistry, physics, and engineering and has also been implemented in various areas of finance. Many researchers have applied them to forecasting of stock prices and other fields of finance. In this study we focus on option pricing. An option is a contract giving the buyer of the contract the right but not the obligation to purchase stock on or before a certain expiration date. Options have become a multi-billion dollar industry in modern times, and there has been a lot of focus on pricing these option contracts. Option pricing data is highly non-linear and its pricing has its basis in stochastic calculus. Since neural networks have excellent non-linear modeling capabilities, it seems obvious to apply neural networks to option pricing. In this thesis, many different methodologies are developed to model the data. The multilayer perceptron and radial basis functions are used in the stand-alone neural networks. Then, the architectures of the stand-alone networks are optimized using particle swarm optimization, which leads to excellent results. Thereafter, a committee of neural networks is investigated. A committee network is an average of a combination of stand-alone neural networks. In contrast to stand-alone networks, a committee network has great generalization capabilities. Many different methods are developed for attaining optimal results from these committee networks. The methods included different forms of weighting the stand-alone networks, a

non-linear combination of the committee members using another stand-alone neural network, a two layer committee network where the second layer was used for smoothing the output and a circular committee network. Lastly, genetic algorithm, with the Metropolis-Hastings algorithm, was used to optimize the committee of neural networks. Finally all these methods were analyzed.

Table of Contents

List of Tables and Figures

1. Thesis Layout

In this thesis different forms of neural networks are applied to option pricing data.

Chapter 2 introduces options. Options are an area of finance that has become a multi-billion dollar industry in recent years. Options can be divided into two types, namely, a call and a put option. There are two categories of options which are called the American option and the European option. Pricing options is deep rooted in stochastic calculus and there is no standard pricing technique. Current pricing techniques include variations of the binomial and Black/Scholes pricing models.

In Chapter 3 neural networks are investigated. The multi-layer perceptron (MLP) and radial basis functions (RBF) are two types of neural networks. A committee of networks is a combination of stand-alone networks. This type of network generalizes far better than individual stand-alone networks. The stand-alone networks within the committee of networks can be combined using various averaging methods. Many different techniques are developed to attain the best combination of the stand-alone networks. Three weighting techniques are investigated to attain good performing committee networks, namely, equal weighting, standard weighting and priority based weighting. A committee with neural network integrator, a double layered committee network and a circular committee network are the methods developed to improve the performance of a committee network.

Neural network architecture plays an important role in model accuracy. Chapter 4 discusses different optimization techniques that are used to find ‘optimal architecture’ networks.

Particle swarm optimization is a global optimization tool which simulates the social behaviour of a school of fish or a flock of birds during the hunt for food. This technique is used to optimize the stand-alone MLP and RBF networks. Thereafter, a modified genetic algorithm is introduced, where the Metropolis-Hastings Algorithm is used in the selection step. This is used to optimize the committee network.

All the above methodologies are applied to option pricing data from the South African Foreign Exchange and implemented in Chapter 5. The results are then summarized and discussed in Chapter 6.

From this thesis the following paper was published (Appendix B): ‘Option Pricing Using a Committee of Neural Networks and Optimized Networks’, IEEE International Conference on Systems, Man and Cybernetics, The Hague, Holland, 2004, pages 434 - 438.

2. Introduction to Options

"Even a cursory glance at the Wall Street Journal reveals a bewildering collection of securities, markets, and financial institutions. Although it may appear so, the financial environment is not chaotic: there is a rhyme or reason behind the vast array of financial instruments and the markets in which they trade." [1]

2.1 Introduction

A current pledge of money or other resources with the expectation of reaping future benefits is an investment [1]. The time an investor spends researching and purchasing the stock as well as the actual value of the stock is the investment one makes to reap future benefits.

Material wealth of a society is one of the factors determined by the productivity of its economy. This productivity is a function of the economy's real assets. Real assets include; the land, buildings, machines, etc. Financial assets, such as stocks and bonds, are the assets by which individuals hold their claims on real assets. Real assets generate income to the economy, while financial assets only define the allocation of the income among the investors.

Financial assets can be subdivided into three broad categories; fixed income, equity and derivatives. Fixed income assets guarantee either a fixed stream of income or a stream of income based on a certain formula. Equity represents ownership in a share of a firm. Derivatives provide payoffs based or derived from other assets such as stocks and bonds. Derivatives have become key investment tools. One of the most important uses of derivatives is to hedge risks or transfer these risks to other parties. [1] Hedging is a technique used to control risk. Intuitively high risk, or high volatility, stocks could develop into huge profits or possibly huge losses.

With the coming of globalization investors can benefit from a range of new choices. Investors can participate, in foreign markets easily with the increasingly efficient and cost effective communication currently established. The growth of derivative markets in recent years has been a significant development in financial markets.

2.2 Options

“Horror stories about large losses incurred by high-flying traders in the derivative markets such as those for futures and options periodically become a staple of the evening news. Indeed, there were some amazing losses to report in the last decade; several totaling hundreds of millions of dollars, and a few amounting more than a billion dollars. In the wake of these debacles, some venerable institutions have gone under…

…These stories, while important, fascinating, and even occasionally scandalous, often miss the point. Derivatives when misused can indeed provide a quick path to insolvency. When used properly, however, they are potent tools for risk management and control.” [1]

An option can be defined as a contract giving the buyer, of this contract, the right but not the obligation, to buy or sell a particular underlying asset at a specific price on or before a certain date.

Options are securities that were first traded in the 1970's, since then there has been dramatic growth in these markets and they are currently being exchanged all over the world. The Chicago Board Options Exchange (CBEO) was the first national exchange to start trading standardized options. The success of these contracts crowded out the existing over-the-counter traded options almost immediately. Options are also traded in huge volumes by banks and other institutions [2].

Option markets attract many investors and if used properly could be very lucrative. Options are written on common stock, stock indexes, foreign exchange, agricultural commodities, precious metals and interest rate futures as well as custom made options on the over-the-counter market. Forwards are contracts that protect the parties involved from price fluctuations. A forward contract is an arrangement of delivery of an asset at a future date at an agreed upon price. Futures markets are markets that use formalized and standardized forward contracts [1].

Derivatives provide a means to control risk that is qualitatively different from the techniques traditionally considered in the theory of managing portfolios. Derivatives are securities whose values are derived from other assets [2]. Derivatives are a powerful tool for hedging and speculation because its value is based on the value of other securities [1].

Businesses often use options to hedge, for example companies often use forwards and exchange listed futures to protect against fluctuations in currency or commodity prices, thereby helping to manage import and raw materials cost. Options can serve a similar purpose for businesses as they do for homeowners; interest rate options such as caps and floors help companies control financing costs in much the same way that caps on adjustable-rate mortgages do for homeowners. Options contracts are divided into two types, viz. the 'Call' and 'Put' option.

2.2.1 Call Option

A call option gives its holder the right to buy an asset for a specific price called the exercise price, on or before some specified expiration date. For example, a May call option on a particular stock with exercise price R80 entitles its owner to purchase the stock for a price of R80 at any time up to and including the expiration date in May. [1]

The holder of this call is not required to exercise this option. The holder will choose to exercise only if the market value or current value of the asset to be purchased exceeds the exercise price. Otherwise, the call option could be left unexercised. Therefore, if the stock price is greater than the exercise price on the expiration date, the value of the call option will equal the difference between the stock price and exercise price; but if the stock price is less than the exercise price at expiration, the call option will be worthless. The net profit on the call is the value of the option minus the price originally paid to purchase it. This is illustrated by Figure 2.1 and in the equation below:

$$\text{Payoff to call option holder} = \begin{cases} \text{Current} - \text{Exercise} & \text{if} \quad \text{Current} > \text{Exercise} \\ 0 & \text{if} \quad \text{Current} < \text{Exercise} \end{cases} \qquad [2.1]$$

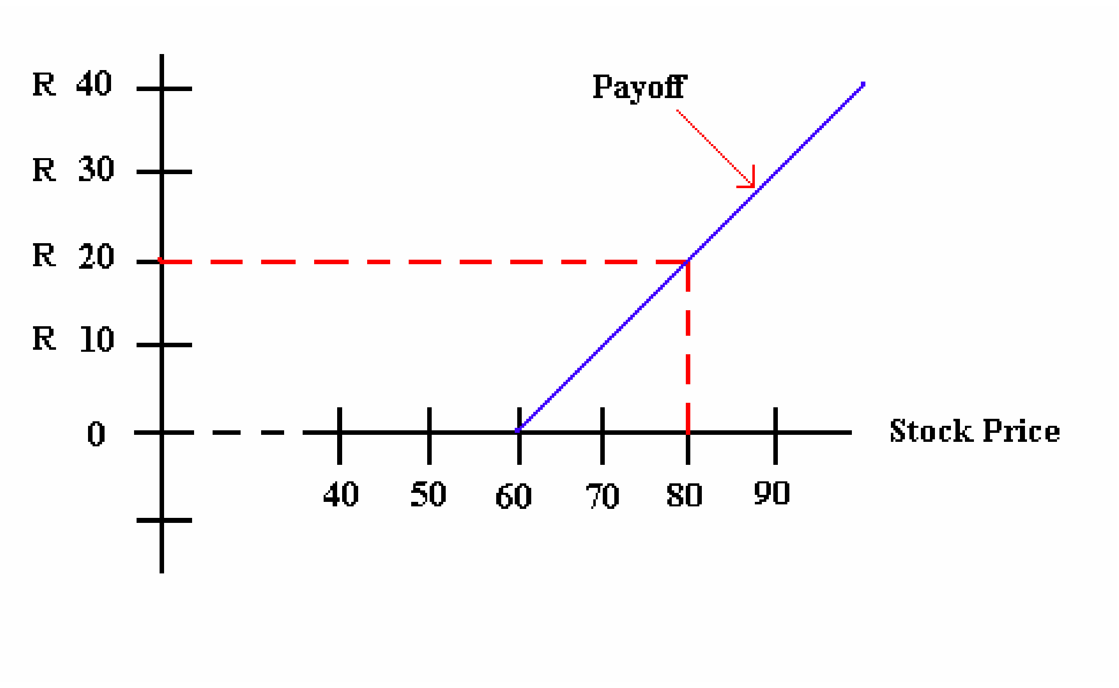

Figure 2-1 Payoff to call option where the exercise price is 60

2.2.2 Put Option

A put option gives its holder the right to sell an asset for a specified exercise price on or before a specified expiration date. A March put on a particular stock with exercise price of R80 entitles its owner to sell the stock to the put writer at a price of R80 at anytime before expiration in March [1].

While profits on calls increase when the asset value increases, profits on puts increase when the value of the asset decreases. A put option will only be exercised if the exercise price is greater than the market value of the asset. The value of the put option is the difference between the exercise price and the market value of the underlying asset. This is illustrated by Figure 2.2 and in the equation below:

Payoff to put option holder =	0	if	Current $\geq$ Exercise	[2.2]
	Strike – Current	if	Current < Exercise	

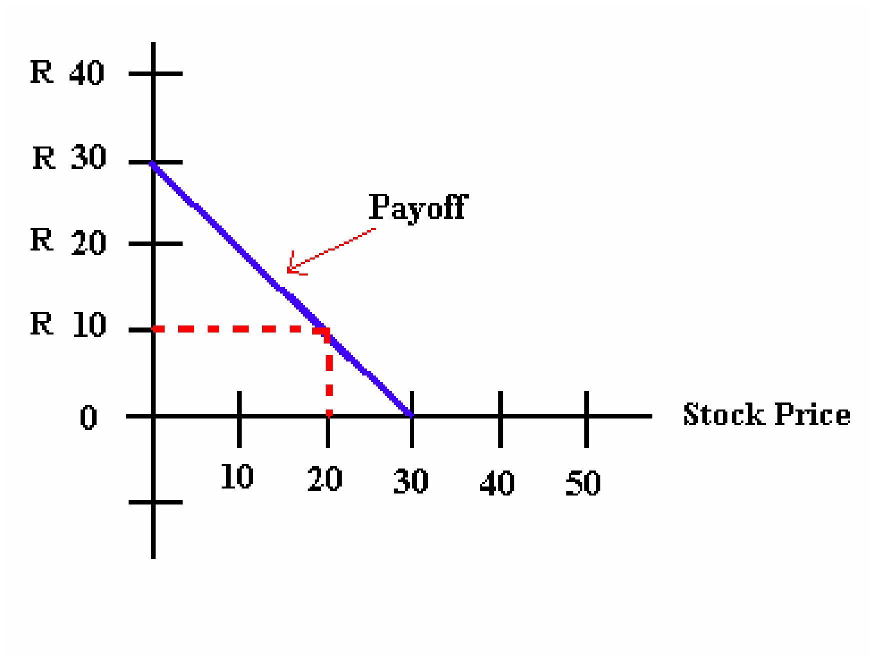

Figure 2-2 Payoff to put option with an exercise price of 30

An option is ‘in the money’ if exercising the option would result in profit; conversely, an option is ‘out of the money’ when exercising the option would not be profitable.

2.2.3 American and European Options

An American option allows its holder to exercise the right to buy (call option) or sell (put option) the underlying asset on or before the expiration date. A European option allows its owner to exercise the contract only on the contracts expiration date. Because of the flexibility of American options they are more valuable than European options. Most options traded are American.

2.2.4 Options in Practice

A hedge is an investment made in order to reduce the risk of adverse price movements in a security, by making an opposite transaction in a related security, such as an option [2].

Corporations in which individual investors place their money have exposure to fluctuations in all kinds of financial prices, as a natural by-product of their operations. The effect of changes in these prices on reported earnings can be overwhelming. Often, companies say in their financial statements that their income was reduced by falling commodity prices or that they enjoyed a windfall gain in profit [1].

Another reason for hedging the exposure of the firm to its financial price risk is to improve or maintain the competitiveness of the firm. Companies compete with other companies in their sector and with companies located in other countries that produce similar goods for sale in the global marketplace.

The hedging objective of the firm is the reduction in the variability of corporate income as an appropriate target. This is consistent with the notion that an investor purchases the stock of the company in order to take advantage of their core business expertise.

Foreign currency options can guarantee a certain rate or whatever is needed to secure prices and remain competitive.

Protective put is a strategy used to hedge. Investing in a stock alone might incur too much risk, since in principle; one could lose all the invested money. By purchasing a put option on the same stock you guaranteed a payoff equal to the put option's exercise price, regardless of how low the value of the stock falls [1].

2.2.5 Option Valuation

The fact that an option is 'out of the money' does not mean the option is valueless, even though immediate exercise would be profitless. The options' value still remains positive since there is always a chance that the underlying assets' value will increase sufficiently to allow for the option to be profitable at the expiration date. The value of a call option before expiration is illustrated in Figure 2.3. The time value of the option is the difference between the options' price and the value the option would have if it were expiring immediately. It is this reason that the options' value is positive when there is still positive time to expiration. Most of this time value is a type of volatility, as long as the option holder can choose not to exercise, the payoff cannot be worse than zero. The volatility value lies in the right not to exercise the option if exercising would deem unprofitable. Hence, this volatility decreases as it becomes more likely that the option holder would exercise.

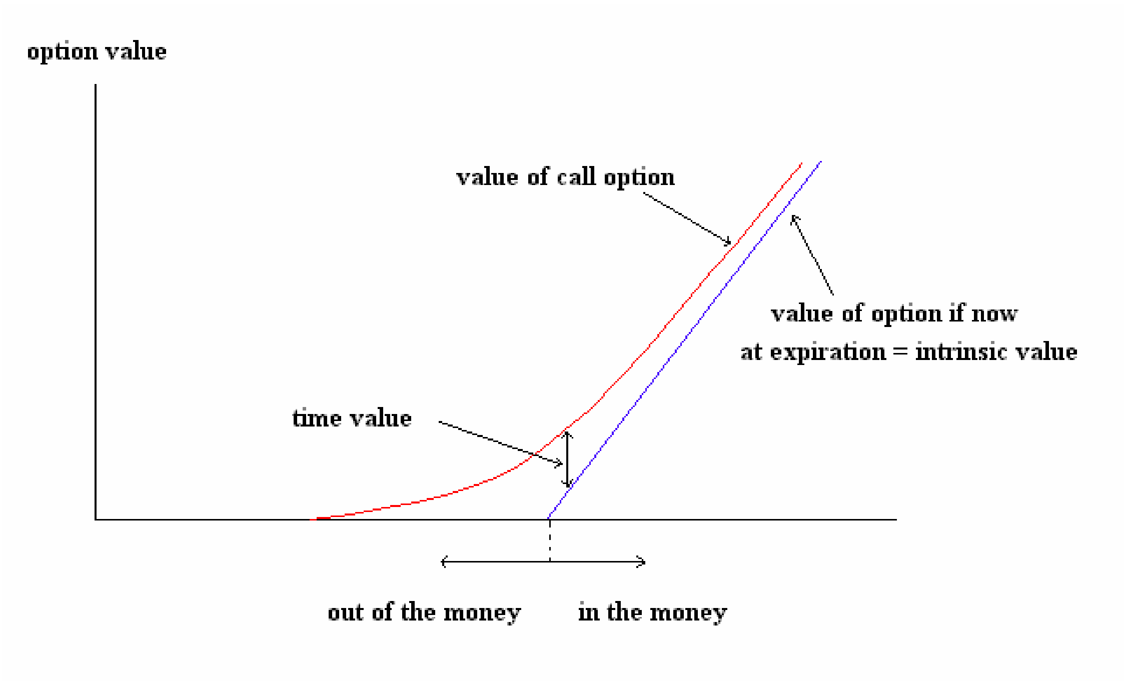

Figure 2-3 Call option value before expiration [1]

From Figure 2.3 we see that when the underlying asset price is low the option is worth very little, since there is little chance it would be exercised. When the underlying asset price is very high the option price increases one to one with it.

Factors affecting the option value:

The factors that affect the value of options (in the scope of this thesis) are; the underlying asset price, the exercise price, the volatility of the stock price, time to expiration and interest rate. This is tabulated in Table 2.1.

If this variable increases	The Value of the Call Option
Stock Price	Increases
Exercise Price	Decreases
Volatility	Increases
Time to Expiration	Increases
Interest Rate	Increases

Table 2-1 Determinants of call option values [1]

The options' value increases as the volatility increases due to the fact that the option holder cannot lose more than the actual price of the option, this means that volatility can only benefit the option holder positively. Longer time to expiration has a similar effect to increased volatility, the longer time the holder has to expiration the more the chances are that the underlying assets value can increase.

2.2.6 Option Pricing Techniques

Modern option pricing techniques, with roots in stochastic calculus, are often considered among the most mathematically complex of all applied areas of finance [3].

Binomial option pricing is a very popular option pricing technique; it involves constructing a binomial tree, illustrated in Figure 2.4. The tree represents possible paths that might be followed and or outcomes of a particular underlying asset's price over the time period of the option contract [2].

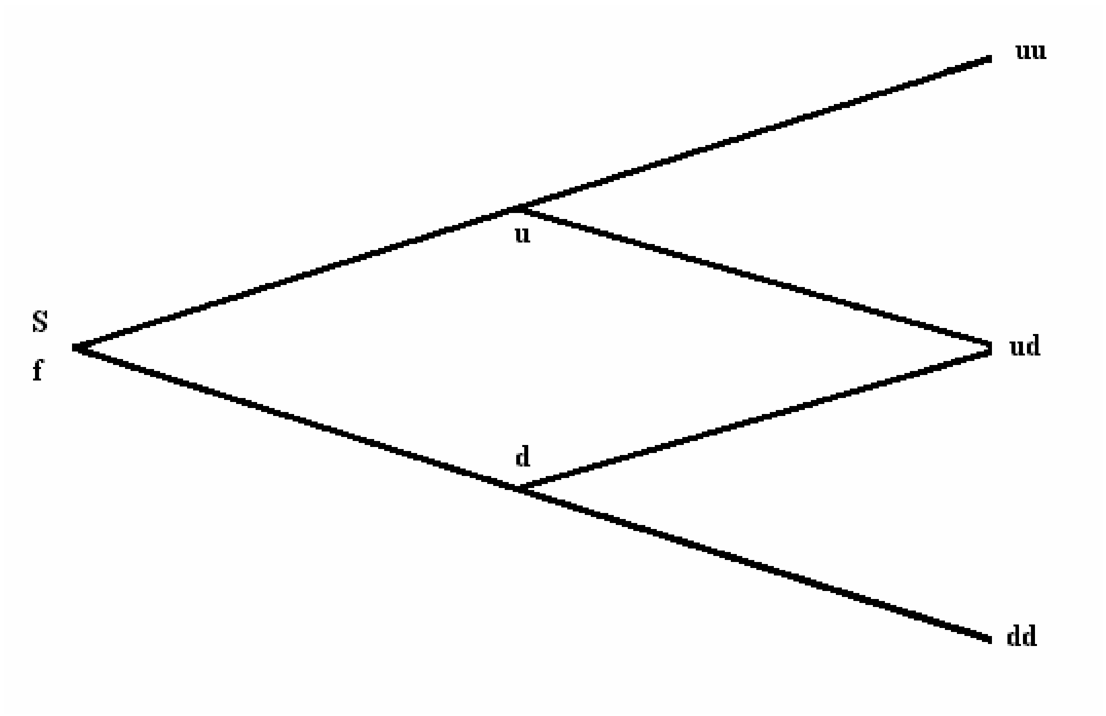

Figure 2-4 Stock and option prices in general two step tree .Here S represents the start and u, d, uu, ud and dd represent possible outcomes of the assets price over a certain time period.

The outcomes of this technique are all the possible values that the underlying asset could have at different time intervals. Generalizing this and using all these possible prices the option is priced using appropriate techniques. A number of financial firms use variations of this model to value options.

The year 1973 saw great inroads being made in this market, the work of Black and Scholes, with an option pricing model being developed [4]. In a Nobel Prize winning paper Black and Scholes derived a formula for calculating the exact value of European options, under certain assumptions.

$$C_0 = S_0 e^{-\delta T} N(d_1) - X e^{-rT} N(d_2) \qquad [2.3]$$

Where:

$$d_1 = \frac{\ln\left(S_0 / X\right) + \left(r - \delta + \sigma^2 / 2\right)T}{\sigma\sqrt{T}}$$

$$d_2 = d_1 - \sigma\sqrt{T}$$

And where

C_0 = Current call option value.

S_0 = Current stock price.

$N(d)$ = Probability that a random draw from a standard normal distribution will be less than d.

X = Exercise price

δ = Annual dividend yield of underlying stock.

r = Risk free interest rate

T = Time remaining until maturity (in years)

σ = standard deviation of the annualized continuously compounded rate of return of the stock.

The assumptions are:

- Geometric Brownian motion, of stock price movements,
- Options have to be exercised at the expiration date (European),
- Constant interest rate,
- Continuous trading without dividends and tax applied to the stocks,
- The market is frictionless [5].

Geometric Brownian motion is a continuous and stochastic process.

Although research shows that the Black/Scholes model outperforms other option pricing models, real data often violates the above assumptions. Because of this and since most options available in the market are American type options; other pricing methods need to be found [5].

Since this model is so easy to use, some of these assumptions are relaxed to adapt the model for real world applications. Many other techniques as well as variations of the Black/Scholes model have been developed in recent times.

2.3 Conclusion

Option contracts are presented in this chapter; first we discuss the different types of contracts, namely, a call option and a put option. We find that a call option has value even if the value of the underlying asset has dropped below the exercise price, and visa versa for a put option. American and European options are then introduced; American options are more valuable than European options, since they can be exercised at any date before the expiration date. The various factors that affect the value of an option are then explored. Option contracts are vital tools for risk management. Current methods for option pricing adopt variations of the binomial and Black/Scholes model. This chapter forms the basis of this thesis, as the methods developed are applied to data from these markets.

3. Neural Networks

"Artificial Intelligence as most people see it suggests machines that are something like brains and is potentially laden with science fiction connotations of the Frankenstein mythos." [24]

3.1 Introduction

A neural network is an assembly of interconnected processing elements, units or nodes, whose functionality is based on the animal neuron. The processing ability of the network is stored in the weights which are inter-unit connections. This ability is obtained by a process of adaptation to, or learning from, a set of training patterns [6]. The architecture of a network refers to how the processing elements are interconnected.

Neural networks are frequently applied to statistical analysis and data modeling, where it is used as a substitute to standard non-linear regression or cluster analysis techniques. Therefore, they are used for classification, or forecasting. Some examples include image and speech recognition, character recognition, and domains and human expertise such as medical diagnosis, and financial market prediction. Neural networks fall within the sphere of artificial intelligence, so that neural networks are perceived as an alternative to the algorithmic techniques that have dominated in machine intelligence [6].

Artificial neural networks could be seen as simplified models of the networks of neurons that arise in the animal brain [7].

Neuroscientists and psychologists use neural networks as computational models of the animal brain developed by abstracting, what is believed to be, properties of real nervous tissue that are essential for information processing. The artificial neurons are simplified versions of their biological equivalent and there are neuroscientists who are doubtful of the power of these models. Mathematicians and physicists are drawn to neural networks from an interest of non-linear dynamical systems [6].

All these groups use neural networks for different reasons from intelligent systems for computer scientists and engineers to understanding the complexities and properties of the network for mathematicians. Possibly the largest users of neural networks are people using it to analyze poorly understood data that arise in the workplace.

Neural networks are very sophisticated modeling techniques capable of modeling extremely complex functions. In particular, neural networks are non-linear. For many years linear modeling has been the commonly used technique in most modeling domains since linear models are simple to solve and also because non-linear models can be approximated by linear models. Where the linear approximation was not valid or the linear approximation was too localized to give globally valid results the models suffered accordingly [8]. The architecture of the networks plays an important role in the accuracy of the networks.

There are many different types of neural networks, we look at two; namely, the Multi-layer perceptron and Radial Basis Functions:

3.2 Multi-Layer Perceptron (MLP)

MLP's are feedforward neural networks. They learn how to transform input data into a desired response. Feed forward neural networks provide a general framework for representing non-linear functional mappings between a set of input variables and a set of output variables. This is achieved by representing the non-linear function of however many variables in terms of compositions of non-linear functions of a single variable, called activation functions [6].

Networks with just two layers of weights are capable of approximating any continuous functional mapping [9]. They are supervised networks, so they require a desired response to be trained. An illustration of an MLP is shown in Figure 3.1.

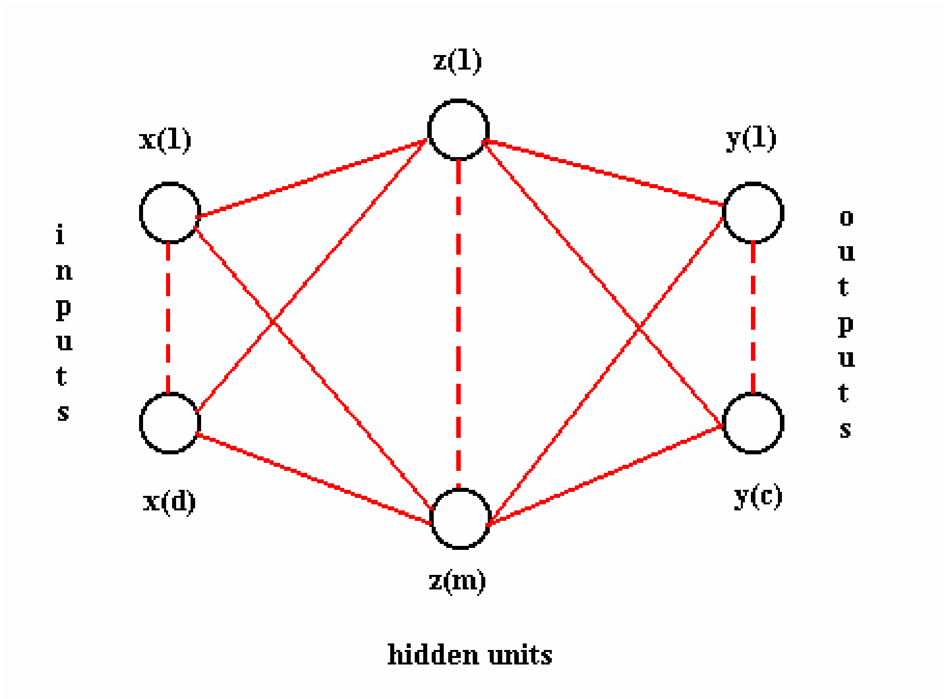

Figure 3-1 Multilayer perceptron with inputs x, hidden units z and outputs y

The above is an example of a MLP network. This network has d inputs, m hidden units and c outputs. The output of this figure analytically is as follows: to attain the i^{th} hidden unit, a weighted linear combination of the d input values and adding a bias, is formulated to give:

$$a_j = \sum_{i=1}^{d} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \qquad [3.1]$$

Here w_{ij} represents a weight in the first layer, going from input i to hidden unit j and w_{j0} denotes the bias for hidden unit j the hidden unit can be incorporated into the weight matrix by including an extra input variable x_o. Hence the equation now becomes:

$$a_j = \sum_{i=0}^{d} w_{ij}^{(1)} x_i \qquad [3.2]$$

The activation of hidden unit j is then obtained by transforming the linear sum in the above equation using an activation function $g(g)$ to give:

$$z_j = g(a_j) \qquad [3.3]$$

The outputs of the network are obtained by transforming the activations of the hidden units using the second layer of processing elements. Thus for each output unit k, we construct a linear combination of the outputs of the hidden units of the form:

$$a_k = \sum_{j=0}^{M} w_{kj}^{(2)} z_j \qquad [3.4]$$

The activation of the k^{th} output unit with activation $z_0 = 1$ is obtained by transforming this linear combination using a non-linear activation function, to give:

$$y_k = \bar{g}(a_k) \tag{3.5}$$

Combining all the above equations gives: [1]

$$y_k = \bar{g}\left(\sum_{j=0}^{M} w_{kj}^{(2)} g\left(\sum_{i=0}^{d} w_{ji}^{(1)} x_i \right)\right) \tag{3.6}$$

Another explanation has its basis in thinking of the neural network as implementing a mathematical function of its inputs, and is especially pertinent if we are dealing with continuous input and output signals. For example if we wanted to make a forecast P_n of a particular stock value based on previous values of the stock price $P_{n-1}, P_{n-2}, ...P_{n-k}$ we want to train a network to discover the functional relation between quantities: this is to discover the underlying function $P_n = P_n(P_{n-1}, P_{n-2}, ...P_{n-k})$ [8].

A MLP can perform categorization of an arbitrary number of classes and with an arbitrary decision surface. All that is required is that we have a set of inputs and targets, and that we fix the number of hidden units that are going to be used.

3.3 Radial Basis Functions (RBF)

Radial basis functions are another major class of neural networks; here the distance between the input vector and the prototype vector determines the activation unit. Radial Basis Functions have their roots in techniques for performing exact interpolation of a set of data points in a multi dimensional space. The exact interpolation problem required every input vector to be mapped exactly onto the corresponding target vector. [6]

The radial basis function approach introduces a set of N basis functions, one for each data point, which take the form $\phi\left(\left\|x-x^n\right\|\right)$ where $\phi(g)$ is some non-linear function. Thus the n^{th} function depends on the distance $\left\|x-x^n\right\|$, between x and x^n. The output of the mapping is then taken to be a linear combination of the basis functions: [9]

$$h(x)=\sum w_n\phi\left(\left\|x-x^n\right\|\right) \tag{3.7}$$

The most common form of basis function is the Gaussian:

$$\phi(x)=\exp\left(-\frac{x^2}{2\sigma^2}\right) \tag{3.8}$$

Where σ is a parameter that controls the smoothness properties of the interpolating function.

Generalizing this for several outputs: each input vector must be mapped exactly onto an output vector x^n having components t^n. The equation becomes: [9]

$$h(x)=\sum_n w_{kn}\phi\left(\left\|x-x^n\right\|\right) \tag{3.9}$$

Radial basis functions are illustrated in Figure 3.2; here the network has d inputs, m basis functions and c outputs.

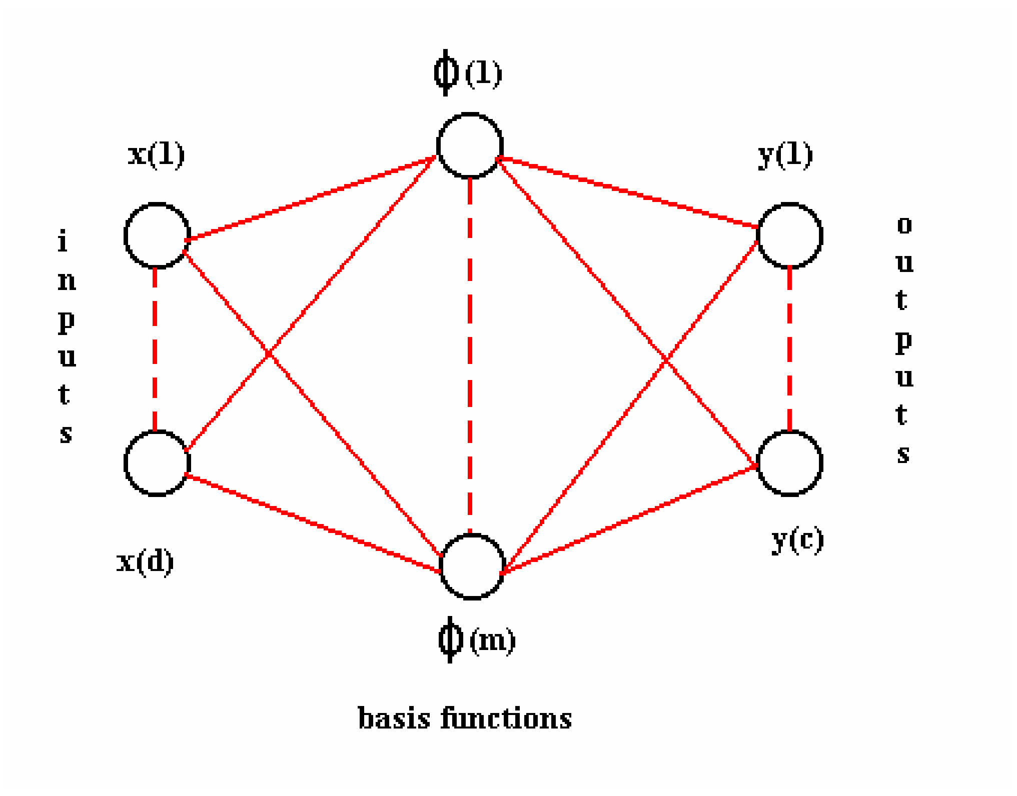

Figure 3-2 Radial basis functions with inputs x, basis functions ϕ and outputs y.

Figure 3.3 illustrates the fact that mappings past through each point. The exact interpolating function for noisy data is typically a highly oscillatory function. Functions such as these are generally undesirable. The interpolation function that gives the best generalization is typically smoother and averages over the noise in the data. Another disadvantage of exact interpolation, is the fact that the number of basis functions are equal to the number of patterns in the data set, and when computing can become costly to evaluate [6].

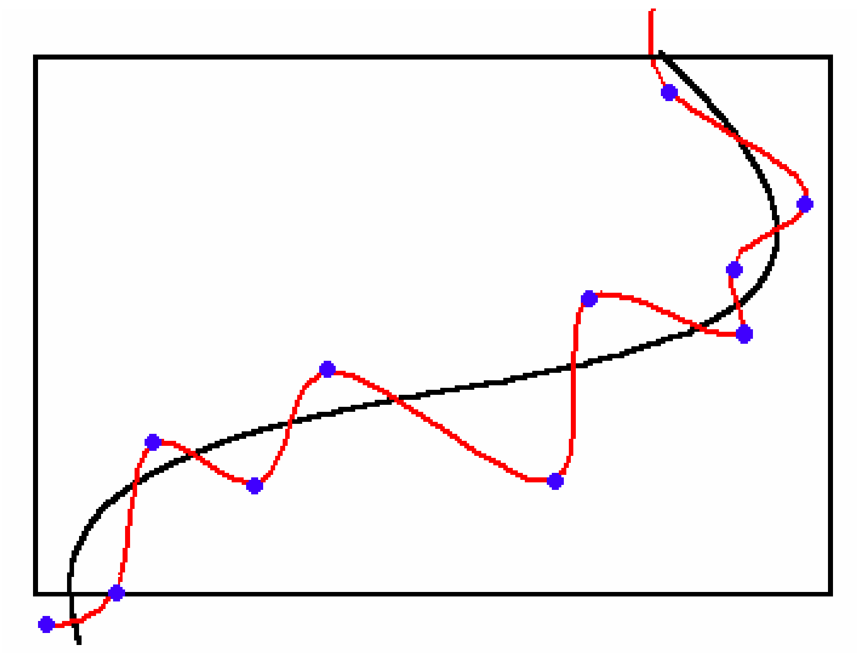

Figure 3-3 Exact interpolation using radial basis functions

There are a number of modifications to the exact interpolating procedure that provides a smooth interpolating function as well as the reduction of the number of basis functions, which are determined by the complexity of the mapping rather than by the size of the data set.

This technique makes use of a vector of parameters with the same dimension as the input. The difference between the weight *w* and input *x* vectors shows how much the input matches the feature template defined by the weights.

RBF's are usually two layer networks in which the first (hidden) layer is a series of RBF's and the second (output) layer is a set of linear units that can be thought of as computing a weighted sum of the evidence from each of the feature template RBF units.

3.4 MLP vs. RBF

The MLP and RBF techniques provide approximations for non-linear functional mappings. In both cases the mappings are in the form of functions of a single variable. The structures of the network, however, are different. Some of these differences are discussed below [9]:

1] A MLP network has very complex connectivity patterns that frequently consist of many hidden layers. A RBF network, however, consist of only two layers of weights. The first layer contains the parameters of the basis functions, and the second layer forms linear combinations of the activations of the basis functions.

2] The hidden units of MLP networks are dependent on weighted linear summations of the inputs and are then transformed by non-linear activation functions. By contrast, the hidden units of the RBF network are formed by the distance between the prototype vector and the input vector, transformed by a non-linear basis function.

3] The output from a MLP network is usually a non-linear combination of cross-coupled hidden units. This non-linearity often leads to problems with local minima when training these networks, which leads to slow convergence during training. By contrast, for a given input vector, the output of a RBF network is influenced by only a few hidden units which have significant activations this is because it has localized functions which form a representation in the hidden unit space.

4] The parameters in the MLP are all trained in one step using supervised training. The RBF, however, uses a two step training strategy where the first step determines the basis functions by unsupervised techniques and the second step uses supervised training to determine the second layer weights.

3.5 Training

Training a neural network is usually done by presenting a training set to the network, and at each step of an iterative process, adjusting the weights of the network to bring its output closer to the desired output. This process of changing or adapting the weights is referred to as the learning rule of the network [6].

3.6 Cross-validation

The objective of training a neural network is to have a network that performs best on unseen data. A simple method to compare the performance of neural networks is to test the errors of the networks using a separate validation / test data set. This is done by training many networks on a training set and comparing the errors of the networks on the validation set. The networks that performed best on the validation data set are then selected. This technique is called cross validation [6].

3.7 Architecture: Generalization and Overtraining

Another very important aspect of attaining good results from neural networks is choosing the correct architecture. Architecture, as it is dealt with here, refers to the amount of hidden units.

The training patterns, in Figure 3.4, are shown by circular symbols and the two classes shown by open and filled symbols. The two lines represent the output of neural networks; where the solid line has been trained more than the dotted line. The solid line classifies all the training patterns correctly. The dotted line, however, misclassifies four of the eighteen training patterns, and it may appear at first sight that this network has performed poorly since there will be some residual error. Suppose that some previously unseen test patterns are presented, as shown by the square symbols, again filled and open squares correspond to the different classes.

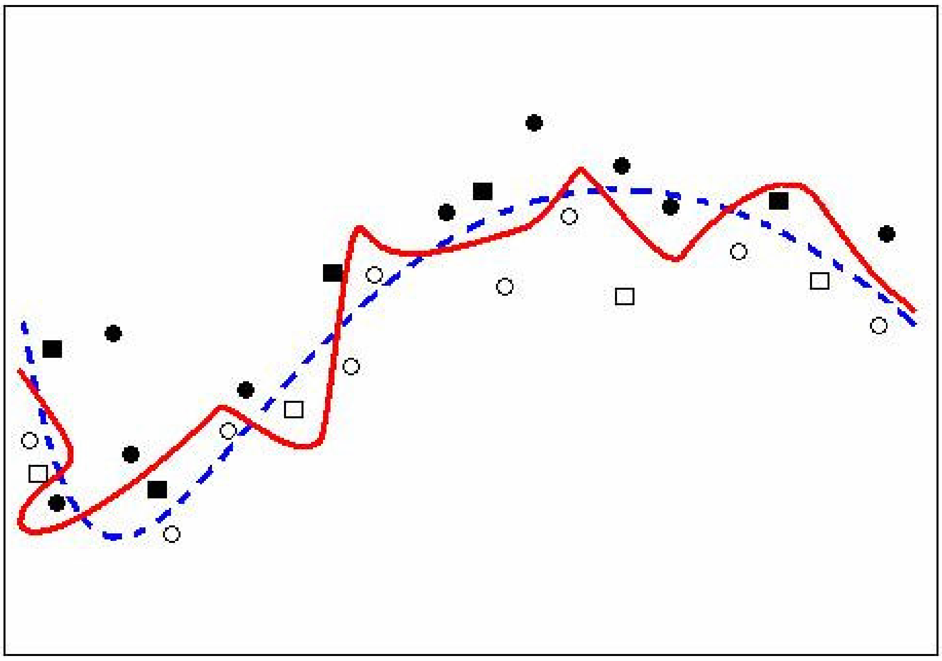

Figure 3-4 Generalization and overtraining

These have been classified correctly by the dotted line and the network is said to have generalized from the training data. This would seem to support the choice of using fewer training epochs since it may be that the two misclassified training patterns result from noisy data. In this case the network has implemented a good model of the data, which captures the essential characteristics of the data in pattern space.

Consider now the solid line. The training set is identical to that used in the previous example and each one has been successfully classified, resulting in a significantly smaller error. However, three of the nine test patterns have been incorrectly classified so that, even though the training data are all dealt with correctly, there may be many examples, especially those close to the decision boundary from each class, that are misclassified. The problem is that the network has too much freedom to choose its decision surface and has overfitted it to accommodate all the noise and intricacies in the data without regard to the underlying trends.

Generalization and overfitting is directly related to the architecture used in the neural network to model the data, since training iterations and the number of hidden units are key elements during the training of the network, and adjusting these elements could lead to great improvements in the networks modeling capability [6].

Figure 3.5 shows how increasing the training epochs eventually leads to greater error values on the validation set, and in turn leads to poor generalization, this illustrates the importance of finding the correct architecture for modeling accuracy. Cross-validation is used to determine when to stop training [6].

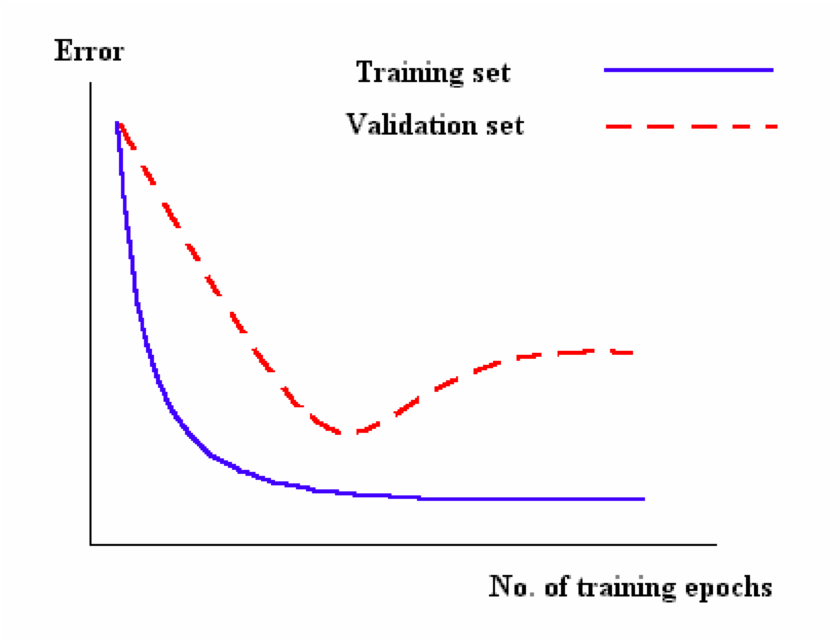

Figure 3-5 Cross-validation behaviour [6]

We may think of neural networks as discovering features in the training set that represent information essential for describing or classifying these patterns.

One of the criticisms sometimes leveled at neural networks is that, although they may generate good models of data, it is difficult to then analyze the structure of the resulting model or to discover the relative importance of the inputs.

3.8 Option Pricing and Neural Networks

Many of the recent applications of Neural Networks to finance, especially option pricing, have just touched on the comparison of a single method to the Black/Scholes option pricing model.

Yao et al [5] modeled the Japan Nikkei 225 future using a MLP, with one hidden layer, with the backpropagation technique, to optimize their option pricing model. They found that neural networks, because of the non-linear modeling ability, was able to get better results on more volatile markets when compared to the model by Black & Scholes, and suggest that when the Black/Scholes assumptions do not hold, neural networks should be used.

Duarte and Ait-Sahalia [10] proposed monotonic and convex shape restrictions on a nonparametric locally linear estimator, these methods could be very useful when there is a limited amount of data.

A nonparametric estimation procedure, proposed by Broadie et al [11], dealt with the computational complexity encountered in estimating the American Option price. They also compared the nonparametric methods to the current parametric model; they found that there were sometimes large discrepancies between the results of the two models which raised several questions with regard to the current models.

Path integrals where found to be fast and accurate for a large part of financial derivatives with early exercise features by Montagna et al [12], they also found that neural networks were a very flexible and powerful tool to predict the price of options to a high accuracy.

The daily S&P 500 index call options where modeled by Gencay and Qi [13]. They compared year by year pricing and hedging performance of feedforward networks without any regularization, where they found that these methods outperformed the baseline neural network model. Also, overfitting the network can be reduced by using regularization methods. Lastly, all of their neural network models outperform the Black/Scholes model significantly.

Statistical inference techniques are used to build neural networks to model the German stock index DAX [14]. By testing for the explanatory power of several variables serving as networks inputs, some insight into the pricing process of the option market is obtained. The results indicate that statistical specification strategies lead to economical networks which have a superior performance when compared to the Black/Scholes model.

Ghaziri et al have used a multi-layer feedforward neural network and neuro-fuzzy networks to price S&P 500 index call options and they have compared it to the Black/Scholes model. The results show that the neural network approach outperforms the Black/Scholes model provided that a sufficient number of patterns are presented [15].

A hybrid neural network was used by Lajbcygier et al to predict the difference between the conventionally accepted modified Black option pricing model and observed intraday option prices for stock index option futures. Their results reflect that a modified bootstrap predictor outperforms the hybrid and bagging predictors [16].

3.9 Committee of Neural Networks

Modeling using neural networks often involves trying multiple networks with different architectures and training parameters in order to achieve acceptable model accuracy. Selection of the best network is based on the performance of the network on an independent validation or test set for instance, and to keep only the best performing network and to discard the rest. There are two disadvantages with such an approach; first, all the effort involved in training the remaining neural networks is wasted, second, the networks generalization performance is greatly reduced.

These drawbacks can be overcome by combining the networks together to form a committee (Perrone and Cooper) [17]. The importance of such an approach is that it could lead to significant improvements in the predictions on new data, while involving the training of a few additional networks. In fact performance of a committee can be better than the performance of the best stand-alone network used in isolation [9]. The committee of networks contains neural networks with different architectures and/or different types of neural networks trained on the same training data set. It might even include different kinds of network models with a mixture of conventional models.

The error can never increase by using a committee of networks. Typically the error is reduced considerably by taking the average error of the combined networks [18].

$$E_{COM} \leq E_{AV} \qquad [3.10]$$

Another advantage of a committee of neural networks is the fact that it is more reliable than stand-alone networks.

The committee network can be represented by Figure 3.6:

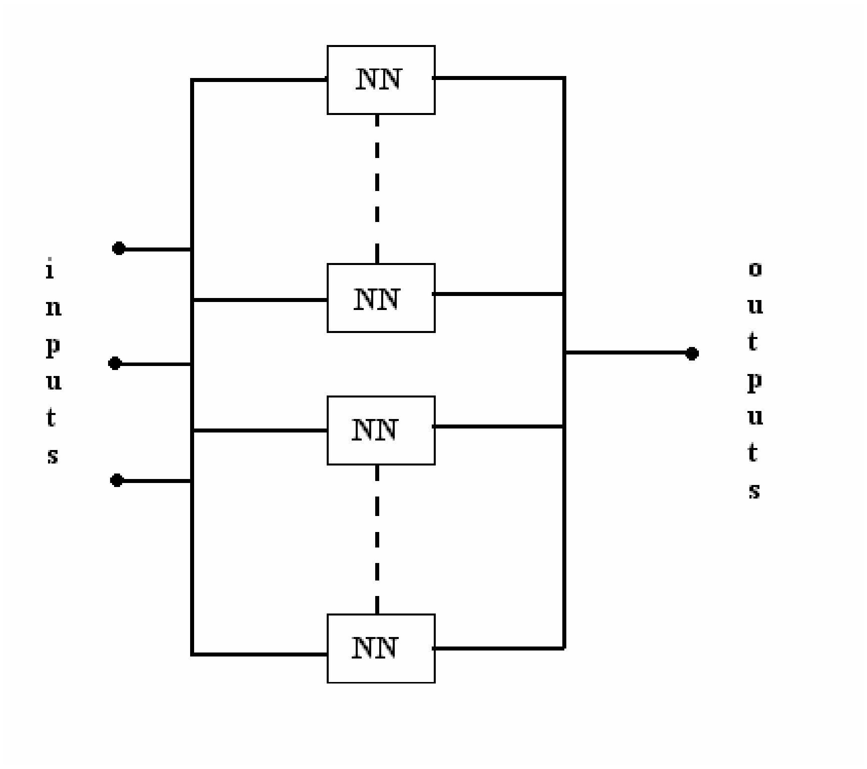

Figure 3-6 Standard committee network

Architectures of neural networks play a big role in capturing different aspects of data. Since the members of the committee network have different architectures, some will make better predictions than others; we expect to be able to reduce the error further if we give greater weight to the better performing committee members than to others. Thus, we consider different forms of committee networks given by weighted combinations of the member networks.

3.9.1 Weighted Averages

There are many possible ways, mathematically, to average the neural networks in a committee network:

1] One could divide the outputs of all the networks by the number of networks to attain an average, giving each network an equal weight, though not the most effective way to reduce the error, this method is easy to implement computationally.

2] Intuitively, one of the best ways to achieve minimum error from a committee of networks is to give greater weight to the stand-alone networks that give better results. This technique uses the following weighting function: [18]

$$Y_{COM} = \alpha_1 y_1 + \alpha_2 y_2 + \ldots + \alpha_n y_n \qquad [3.11]$$

Where the α's are the weighting variables and,

$$\alpha_1 + \alpha_2 + \ldots + \alpha_n = 1 \qquad [3.12]$$

n is the number of networks used in the committee. According to a theorem [18], there exists an optimal committee network that gives the least mean square error if the weight variables α are chosen to be:

$$\alpha_i = \frac{1}{\sum_{i=1}^{n} \frac{\varepsilon(e_i^2)}{\varepsilon(e_j^2)}} \qquad [3.13]$$

Where $\varepsilon(e)$ is the expected error. This method will be referred to as the standard weighting method (SW), in this text.

3] Another way of assigning weights to the outputs is to use priory based weighting, Figure 3.7. In this method weights are assigned according to the individual priority of the data points as well as satisfying the overall reduction in error.

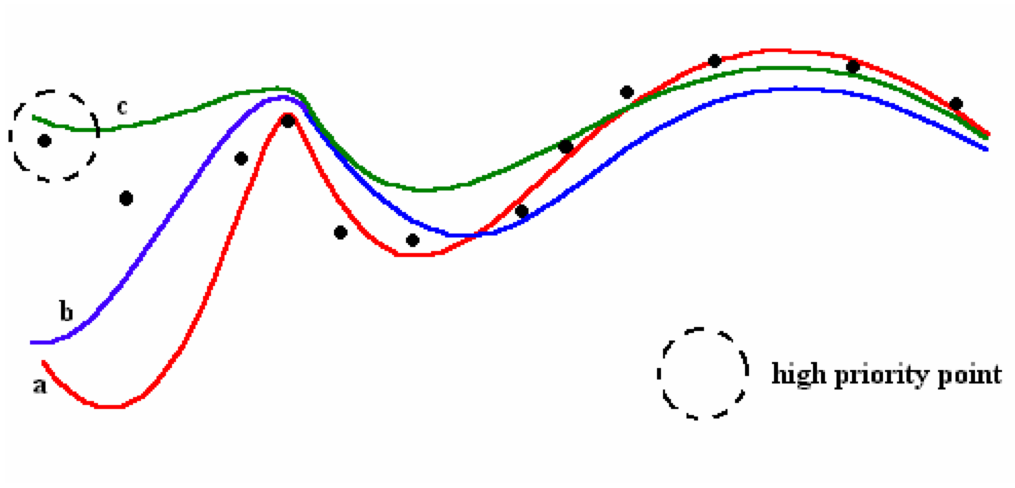

Figure 3-7 Priority based weighting

a, b and c are different approximations made by three different neural networks in a committee. As we can see from Figure 3.7 a, is the most accurate while c is the least accurate. Network c, however, does approximate the data better at a point that has high priority. High priority points are points that are more important to the modeler in a particular simulation. This weighting method satisfies both criteria, namely, better overall results as well as better approximations at points of high priority. It does this by using a strategy similar to the above weighting method, and by adding a priority variable:

$$Y_{COM} = (\alpha_1 + p_1)y_1 + (\alpha_2 + p_2)y_2 + ... + (\alpha_n + p_n)y_n \qquad [3.14]$$

Here

$$(\alpha_1 + p_1) + (\alpha_2 + p_2) + ... + (\alpha_n + p_n) = 1 \qquad [3.15]$$

Where

$$\sum_{i=1}^{n} p_i = k \qquad [3.16]$$

And

$$\sum_{i=1}^{n} \alpha_i = k - 1 \qquad [3.17]$$

k is a value the modeler chooses according to the model. The value *k* is divided according to which neural network approximates the high priority points better, the better the approximation of the i^{th} network, the bigger the chunk of *k*, p_i gets according to some division rule.

There are a number of other ways to average the outputs in a committee network. Some employ non-linear techniques. Below are a few different ways to average the committee network using a neural network as a non-linear combiner.

3.9.2 Committee with Neural Network Integrator (CNNI)

This committee has the same architecture as the one mentioned previously, the technique uses either a MLP or a RBF to combine the networks in the committee to create a non-linear weighted average of the neural networks, Figure 3.8. The inputs for the integrator MLP or RBF are the outputs of the first layer of networks. This is the same output that would be combined using the methods discussed in the previous section. This integrator network serves to smooth the outputs of the first layer networks.

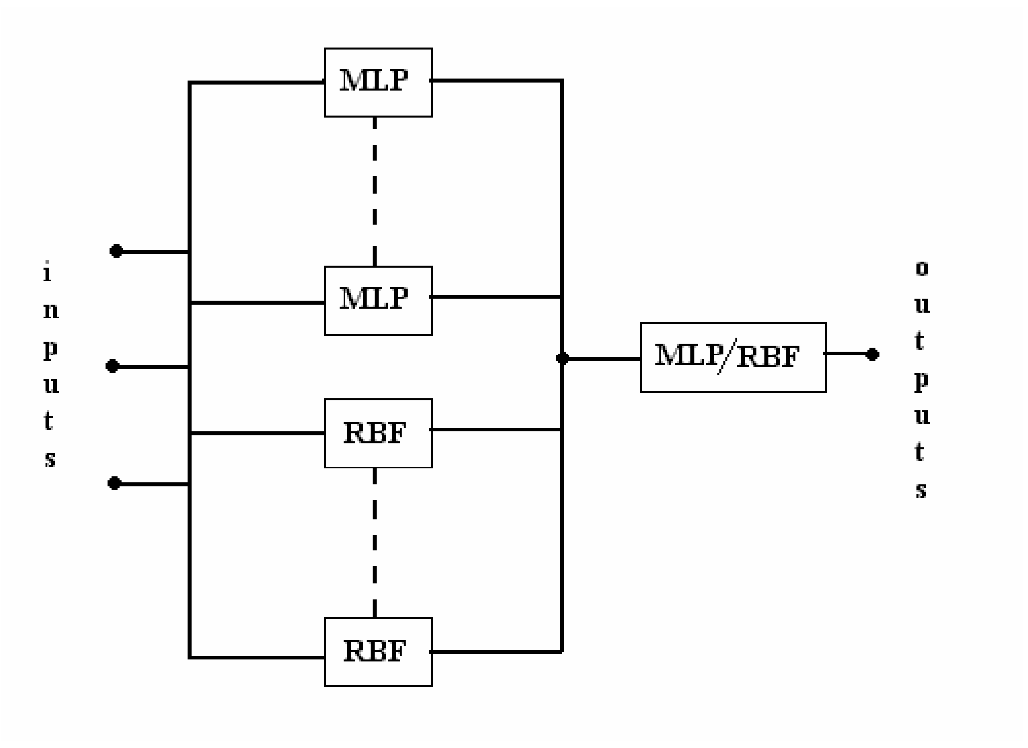

Figure 3-8 Committee with Neural Network Integrator (CNNI)

With regard to computational time, the ‘committee neural network integrator’ method only requires one more network to be trained; this does not affect computational time much.

3.9.3 Double Layered Committee Network (DLCN)

This technique, illustrated in Figure 3.9, uses a smoothing neural network after each of the first layer networks. The output of the first layer is the same as described in Section 3.9.1 for the standard committee network. This output provides the input for the second layer. The second layer smoothes these inputs and the second layer outputs are now combined using the weighting methods mentioned in Section 3.9.1.

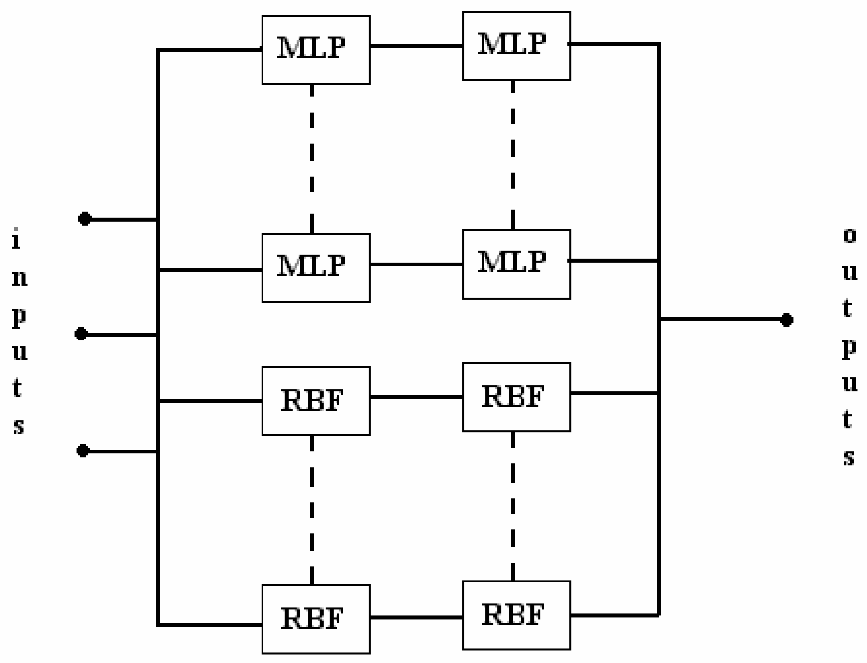

Figure 3-9 Double layered Committee Network (DLCN)

With the 'double layered committee network' the computational time is doubled, since twice the amounts of networks have to be trained.

3.9.4 Circular Committee Network (CCN)

The 'circular committee network' (CCN) uses a jump technique, Figure 3.10. Firstly, the inputs to all the stand-alone networks are the same. During the next step the outputs are jumped to the neighboring neural networks to serve as inputs to the second step of training. The second training step serves to smooth the outputs of the networks. Thereafter the weighting techniques in Section 3.9.1 are used to combine the outputs of this committee network.

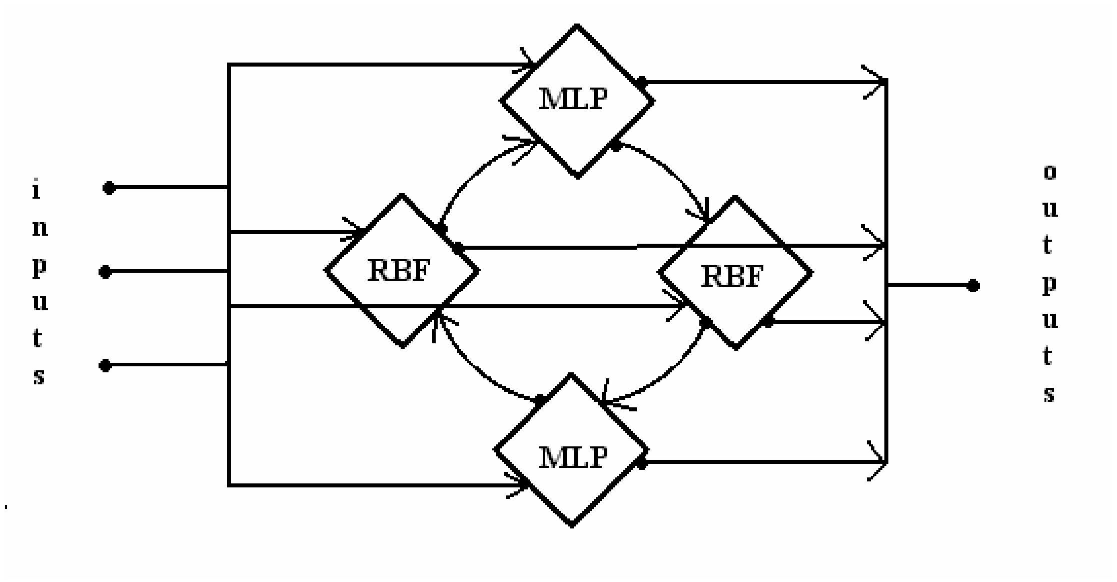

Figure 3-10 Circular Committee Network (CCN)

The computational time required by the (CCN) technique is much less than that required for all previous techniques as it requires only four neural networks being trained twice.

3.10 Conclusion

In this chapter neural networks were introduced. Neural networks originated from attempts to mimic the animal brain. Two types of stand-alone neural networks are investigated, namely, the MLP and RBF networks. Architectures of neural networks are then discussed in detail. In this text the number of hidden units, are referred to as the architecture. A committee of neural networks is an average of a few stand-alone networks. We investigated different approaches for averaging the networks using linear as well as non-linear methods. These included the following methods; the committee with neural network integrator (CNNI), the double layered committee network (DLCN) and the circular committee network (CCN). The methods introduced in this chapter are implemented in Chapter 5.

4. ‘Optimal Architecture’: Neural Networks and Optimization Techniques

4.1 Introduction

As mentioned before neural network architecture plays a big role in model accuracy. The architecture in this text refers to the number of hidden layers used to train the neural network. Finding the correct amount of hidden layers is crucial when dealing with such highly non-linear data, such as option pricing data. Because of the non-linearity, the amounts of hidden layers are almost counterintuitive, which leads, again, to high non-linearity with regard to the best architecture. In an attempt to find the best architectures that model this particular data, the techniques described below are proposed to find ‘Optimal Architecture’ networks. Particle Swarm Optimization is used to find stand-alone ‘Optimal Architecture’ networks, while Genetic Algorithm finds the optimal architecture for a committee of neural networks.

4.2 Optimization Techniques

Global optimization algorithms imitating principles of nature have been very useful in recent times and have been applied in various domains. Such phenomena can be found in annealing processes, central nervous systems and biological evolution, which in turn have lead to the field of Evolutionary Computation (EC) [19].

Evolutionary computation includes; Genetic algorithms, evolutionary programming, evolution strategies, classifier systems, genetic programming and numerous other problem solving approaches that are based on biological observations. These observations date back to Charles Darwin's 'theory of evolution', hence the term Evolutionary Algorithms [20].

This term 'evolutionary algorithm' refers to evolutionary processes used by computer-based problem solving systems. Evolutionary algorithms maintain a population of variables or structures, which evolve according to rules of selection, reproduction, recombination and mutation. Each individual in the population receives a measure of its fitness in its environment which is equivalent to its function value. Individuals with high fitness are chosen more readily for reproduction. Exploration across the domain is done by recombination and mutation by perturbing particular variables or individuals. Although simplistic from a biologist's viewpoint, these algorithms are sufficiently complex to provide robust and powerful adaptive search mechanisms [19].

4.2.1 Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) is an algorithm proposed by James Kennedy and R. C. Eberhart in 1995 [21], motivated by social behavior of organisms such as bird flocking and fish schooling. PSO as an optimization tool provides a population-based search procedure in which individuals called particles change their position with time. This method is related to evolutionary programming and was discovered through simulation of the simplified social model, namely, the synchronized flocking of birds or the movement of a school of fish. In a PSO system, particles fly around in a multidimensional search space. During flight, each particle adjusts its position according to its own experience, and according to the experience of a neighboring particle, making use of the best position encountered by itself and its neighbor. Thus, a PSO system combines local search methods with global search methods, attempting to balance exploration and exploitation [22].

Thus, PSO is a technique used to globally optimize non-linear functions. Since, a school of fish profits from previous experiences and discoveries of each member during the search for food [23]. As with most global optimization techniques particle swarm initializes a population of points, called particles, and uses them to calculate their function values either by selecting them randomly or at equal spacing across the space. At each iteration, this method influences the update of each particle according to the following:

- Each particle's best function value,
- The function value of its fittest neighbor,
- An element of randomness

This strategy is illustrated in Figure 4.1.

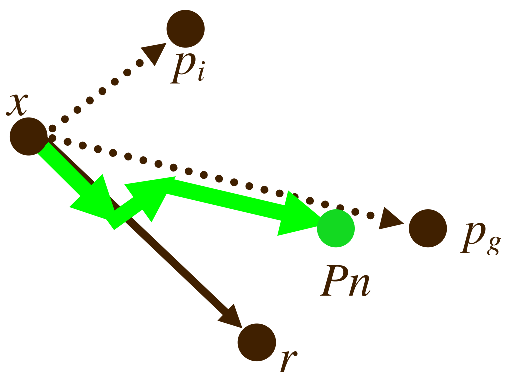

Figure 4-1 Pi = particle's best position, Pg = neighbor's best position, r = random direction, Pn = next move

Figure 4.1 shows how this method updates the particles, influenced by the above directions.

Algorithm:

1. Create random particle population and assign to them random positions,
2. Evaluate their function values at their current positions,
3. Choose a set of neighbors for each particle,
4. Update the particle population using the new direction, called the velocity,
5. Repeat until a stopping criterion is met.

The MLP and RBF network architectures were optimized using PSO, to find 'Optimal Architecture' MLP and RBF networks. The pseudo code for this technique is in Appendix A.

4.2.2 Metropolis-Hastings Algorithm (MHA)

In mathematics and physics, the Metropolis-Hastings algorithm is an algorithm to generate a sequence of samples from the joint distribution of two or more variables. The purpose of such a sequence is to approximate the joint distribution. This algorithm is an example of a Markov chain Monte Carlo algorithm. The Metropolis-Hastings algorithm can draw samples from any probability distribution *P(x)*, requiring only that the density can be calculated at *x* [24].

The key to the Metropolis-Hastings algorithm is to create a sampling strategy by which the probability of being in state *a* and moving to state *b* is the same as from *b* to *a*, subject to a few regularity conditions. This series of draws is accomplished by proposal and acceptance/rejection of candidate values x^* [24].

Algorithm:

```
for $j = 1 : N$
draw $x^*$ from $q(x^* | x_j)$, draw $u$ : $U(0,1)$
if $u < \frac{\left[q(x_j | x^*) f(x^*)\right]}{\left[q(x^* | x_j) f(x_j)\right]}$
then $x_{j+1} = x^*$
else $x_{j+1} = x_j$
```

The proposal of a candidate value x^* is conducted through a proposal function $q(x^* | x_j)$ the form of which is quite arbitrary. In order to get a lot of x values, the values of x^* should not be rejected too often. The arbitrariness of the proposal function $q(x^* | x)$ and the lack of theory guiding our choice leave lots of room for experimentation [25]. The two functions that suggest themselves are the uniform and normal distributions [24]. The arbitrariness of the proposal function is supported in the acceptance/rejection step, which corrects for unlikely steps from x_j to x^* by accepting them only with the ratio of moving from one to the other.

4.2.3 Genetic Algorithm (GA)

The genetic algorithm is a model of machine learning, which derives its behavior from a metaphor of the processes of evolution in nature. This is done by the creation within a machine of a population of individuals represented by chromosomes. The individuals in the population then go through a process of evolution. Some variables or individuals are better or fitter than others. Those that are better are more likely to survive and regenerate their genetic material. Reproduction allows the creation of genetically radically different offspring that are still of the same general species. At the molecular level what occurs is that a pair of chromosomes bump into one another, exchange chunks of genetic information and drift apart. This is the recombination operation, which genetic algorithms generally refer to as crossover, because of the way that genetic material crosses over from one chromosome to another. Crossover takes place when the selection of who gets to mate is made a function of the fitness of the individual. The genetic algorithm uses stochastic processes, but the result is distinctly non-random [19].

Firstly, an initial population of parent individuals (feasible solutions) is randomly created. Each individual is represented by a chromosome, a string of characteristic genes. Secondly, all the individuals are ranked with a fitness function appropriate to the problem at hand. The fittest of these, pass directly to the following generation; a process of 'elitism'. Thirdly, a breeding population is formed by selecting top-ranking individuals from those that remain. This is the natural selection step. Lastly, these selected individuals undergo certain transformation via genetic operators to reduce children for the next generation. Operators include recombination by crossover and mutation (a randomly generated gene that is somehow altered). Mutation ensures a genetic diversity in the population [19]. This process is repeated for a certain number of generations, or until some stopping criterion is met.

Algorithm:

1. generate initial population
2. generate offspring:
    - selection: probability of being accepted according to fitness
    - crossover: parents are paired to generate offspring
    - mutation: each string has a very small chance of being mutated
    - selection/rejection: selection or rejection of the new generation according to some criterion
3. Repeat step 2 unit stopping criterion is met.

4.2.4 Genetic Algorithm with Metropolis-Hastings Algorithm

In the above algorithm the selection/rejection step of the new generated population is not specified, and many methods can be used to do this step. Metropolis-Hastings Algorithm has been adopted in this text to make the selection or rejection of the new generation in the genetic algorithm.

The Genetic Algorithm Metropolis-Hastings Algorithm (GAMHA) method was then used to manipulate the architectures in the individual MLP and RBF networks so as to optimize the committee network. The pseudo code for this technique is in Appendix A.

4.3 Conclusion

The importance of neural network architecture is emphasized again in this chapter. Two optimization strategies are introduced. These strategies will be applied to neural networks in Chapter 5 to find 'optimal architecture' networks. Particle swarm optimization is a recently developed algorithm for global optimization; its algorithm is derived from the synchronized flocking of birds during the hunt for food. Genetic algorithm is then introduced. This algorithm is based on the theory of evolution or specifically 'survival of the fittest'. The Metropolis-Hastings Algorithm is used as the selection step in this modified genetic algorithm.

5. Implementation and Results

5.1 Introduction

The data set used was collected from the South African Foreign Exchange between January 2001 and December 2003 [26]. The data contained the underlying stock price, strike price, time to maturity, stock volatility, market to market price as well as the high and low prices.

The key parameters needed for option pricing are [2]:

- underlying asset price,
- volatility of the underlying asset,
- interest rate,
- time to maturity

Therefore these were the parameters used as inputs. The network output was the strike price. Approximately two thirds of the data set was used for training and the rest for the testing.

To implement these neural networks, the Netlab software toolbox [27] was used and in particular, the MLP and RBF software programs were used to achieve the results below. With the Netlab toolbox you can implement and train various types of neural networks, as well as tweak the architectures to achieve good modeling results.

5.2 Stand-alone MLP & RBF Networks

As expected, the architectures seemed to play a very important role in model accuracy. Architectures of the stand-alone MLP and RBF networks were chosen on a trial and error basis, whichever architecture modeled the test/validation data better would be chosen. This lead to many different architecture trials and sometimes particular neural networks took much longer than average. Below are the results of two different architectures for both the MLP and RBF. The first MLP is illustrated in Figure 5.1.

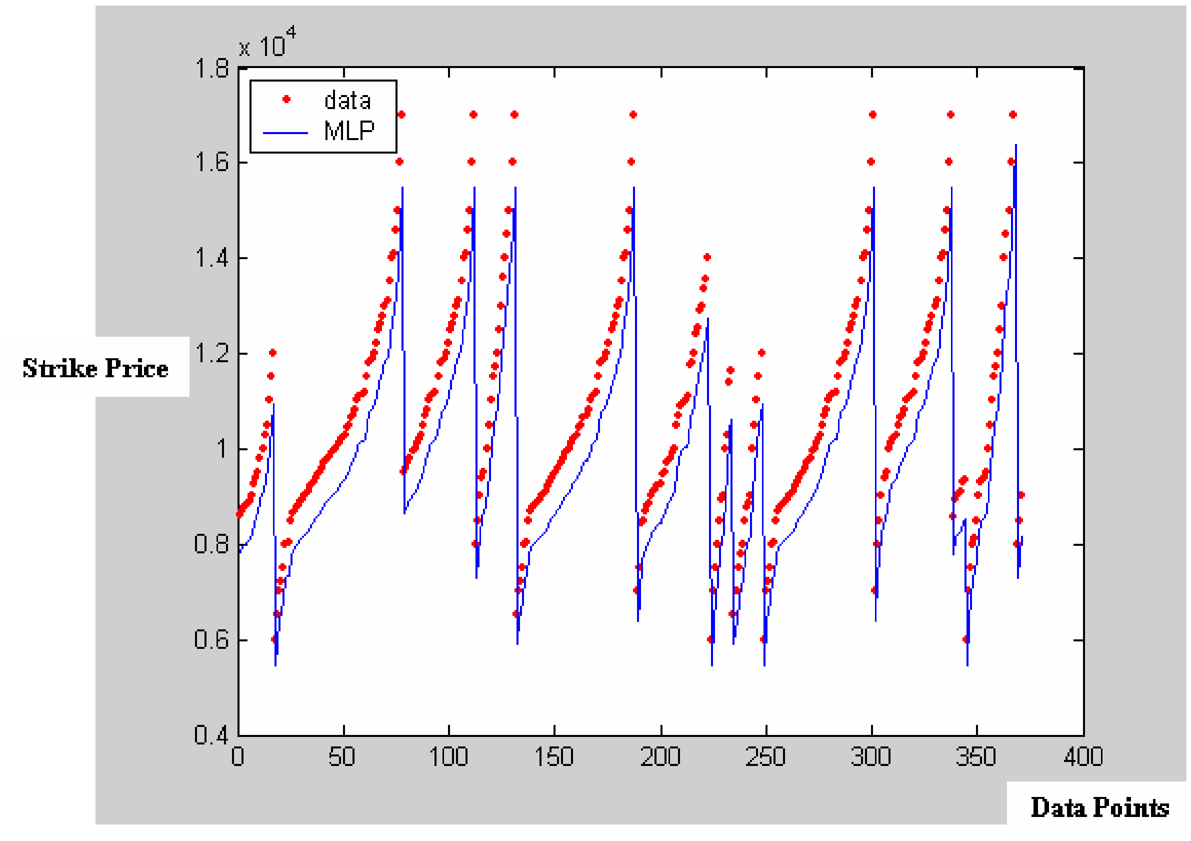

Figure 5-1 Standard MLP 1

This particular MLP took just over a second to train and has an error of almost 9 %. When increasing the number of hidden units 10 times, i.e. changing the architecture, it took almost ten times longer, it took almost 9 seconds and the error was reduced to about 3.5 %,

Figure 5.2. This shows just how much architecture affects the computational time of training MLP networks.

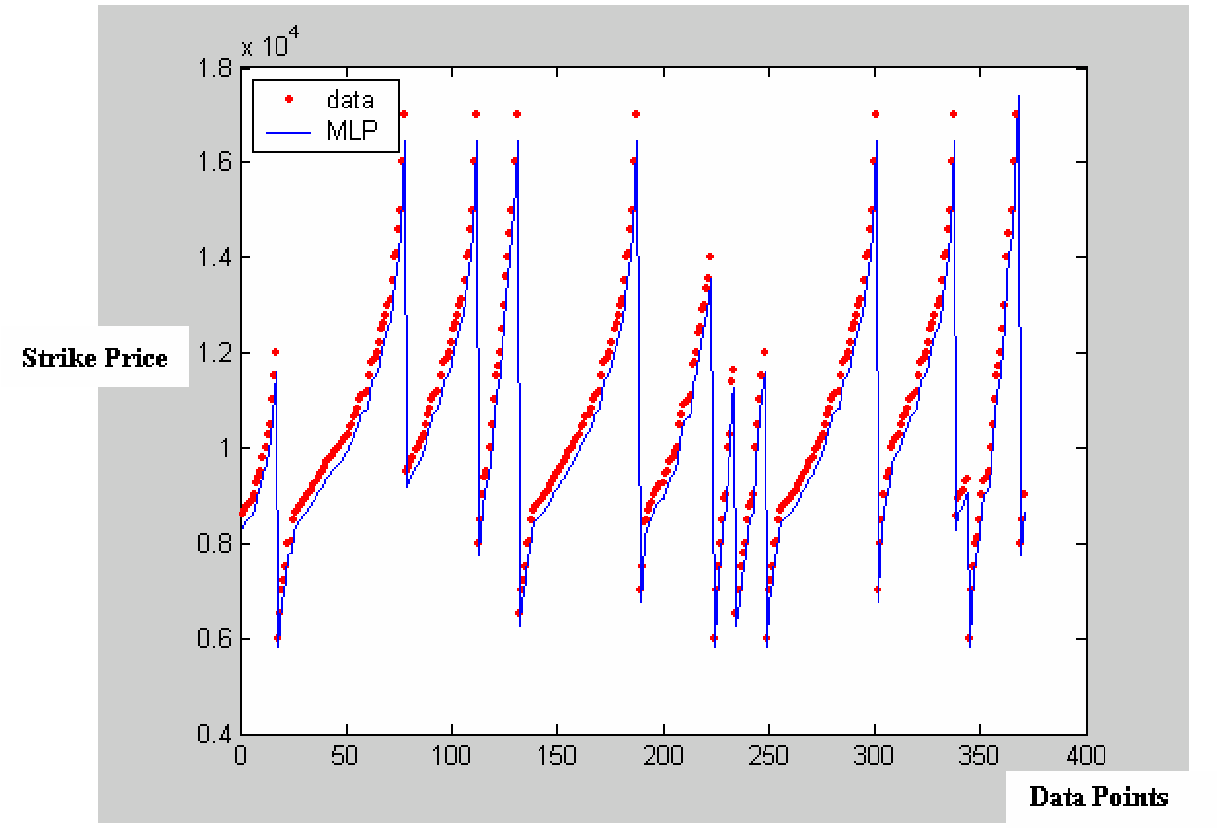

Figure 5-2 Standard MLP 2

RBF networks are characterized by fast training. Below we see how RBF attains similar results at a fraction of the computational cost. The error in Figure 5.3, just as the MLP in Figure 5.1 is about 9 %, whereas computational time used to train the RBF network is 5 times less, at about 0.2 seconds. Increasing the number of hidden units had a different effect on the results; the error increased using more hidden layers. This illustrates, again, the importance of attaining the correct architecture. The error in Figure 5.4 increased dramatically to about 60 %, with time required for computation still relatively fast, at just under a second.

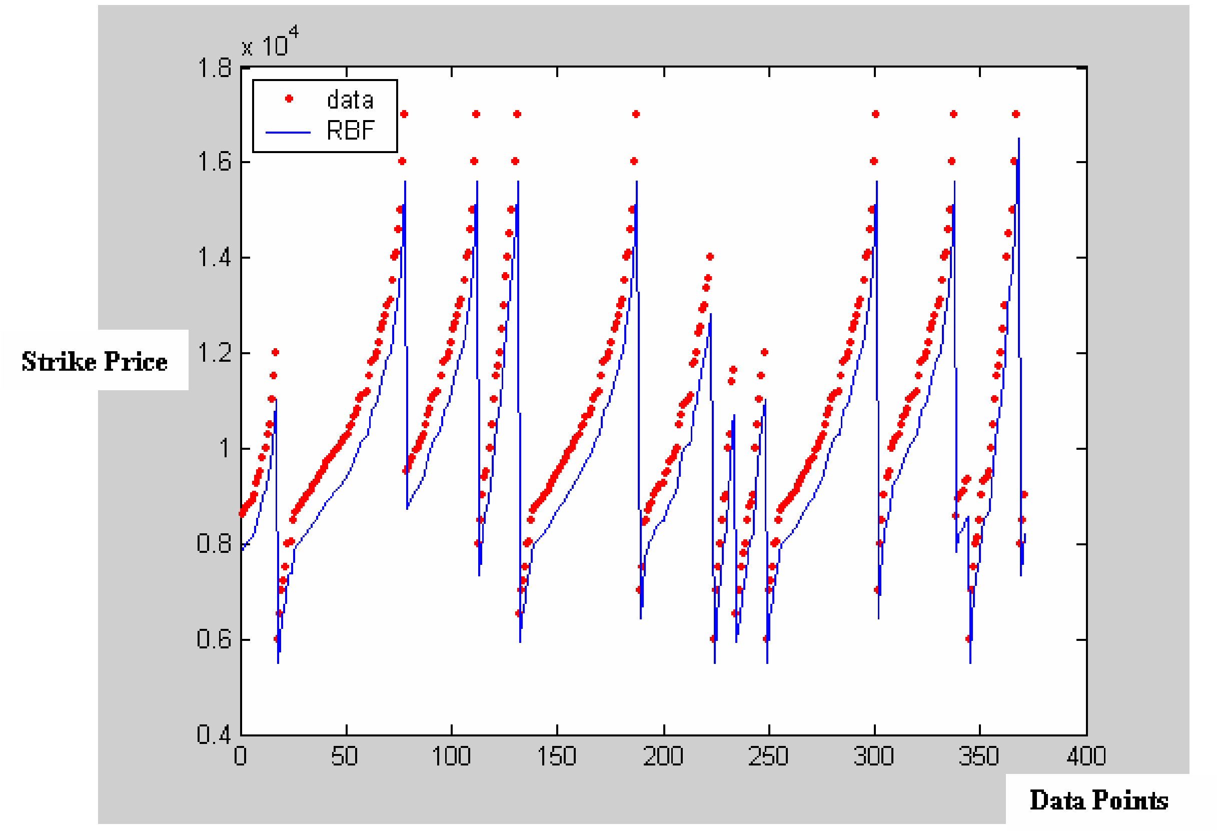

Figure 5-3 Standard RBF 1

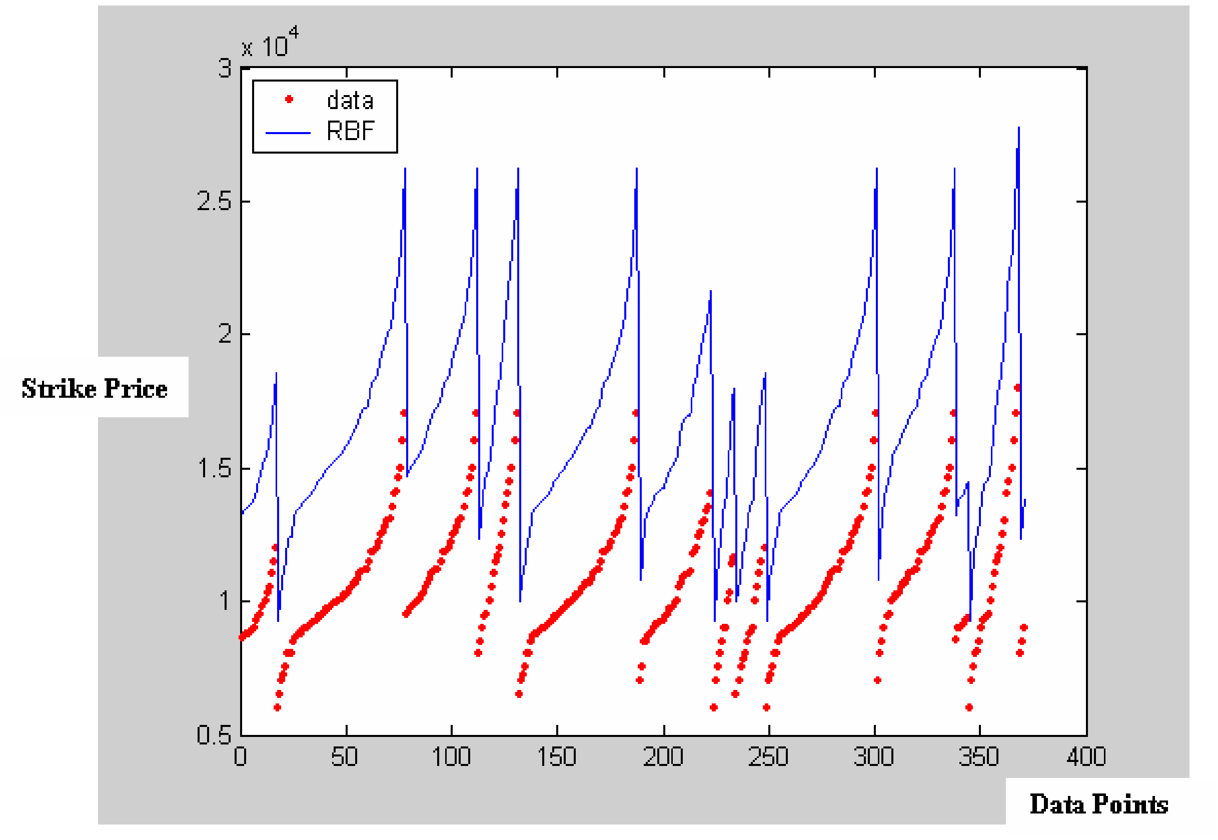

Figure 5-4 Standard RBF 2

5.3 'Optimal Architecture' Multi-layer Perceptron (MLP) and Radial Basis Functions (RBF) using Particle Swarm Optimization (PSO)

To solve the neural network optimal architecture problem, optimization techniques were employed. Particle Swarm Optimization was used to find the optimal architecture of neural networks by optimizing the error function; this was achieved by adjusting the number of hidden units. The results from these 'Optimal Architecture' MLP and RBF networks are as follows:

The error was greatly reduced achieving near perfect results. Results for both, MLP and RBF, 'optimal architecture' networks were less than 0.005%, Figure 5.5 and Figure 5.6, respectively. The time it took to achieve these results were unfeasible in a business or work environment; it sometimes took in excess of one hour to run an 'optimal architecture' simulation. This was due to the amount of networks that had to be trained and the highly non-linearity of the data, which meant that finding a global minimum could take a really long time. When running the simulation for a shorter period, say a few minutes, the results would still be a great improvement on the stand-alone MLP and RBF networks. This shows that the 'optimal architecture' program would choose local minima as its output.

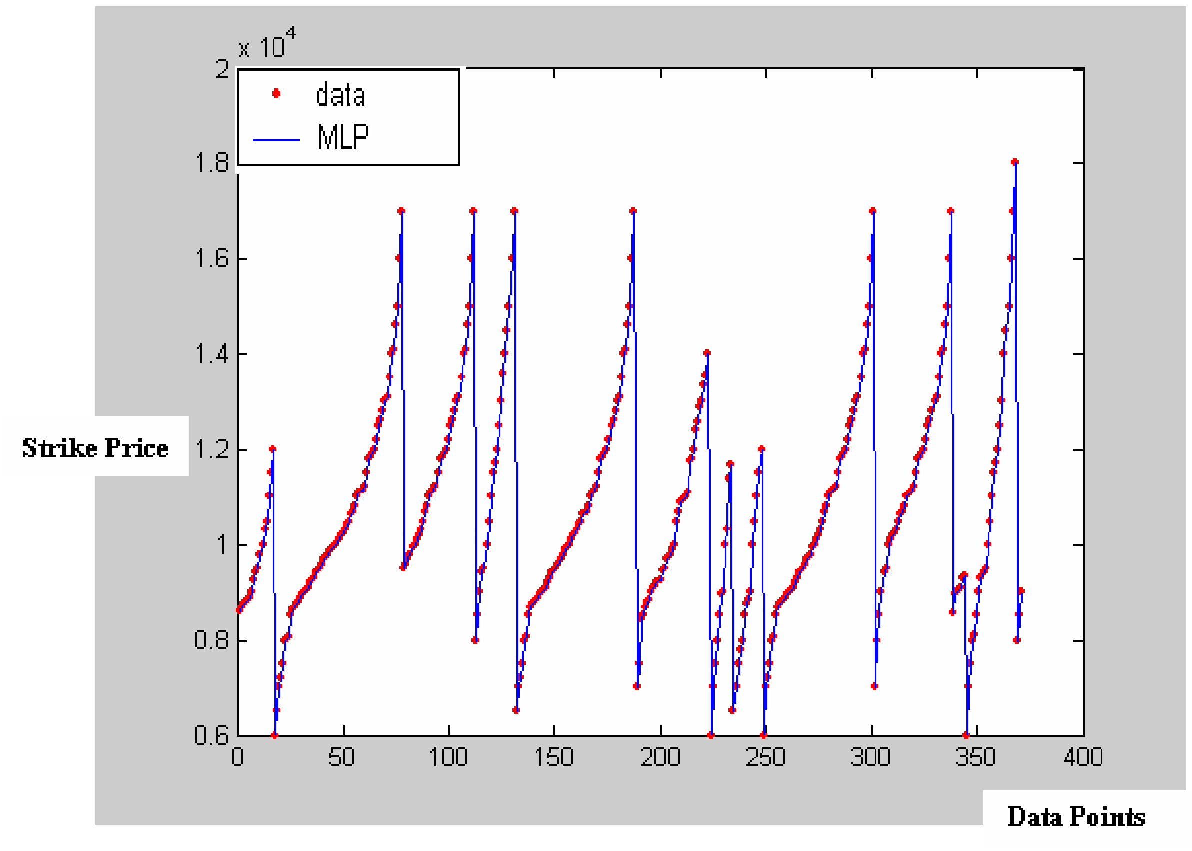

Figure 5-5 'Optimal Architecture' MLP

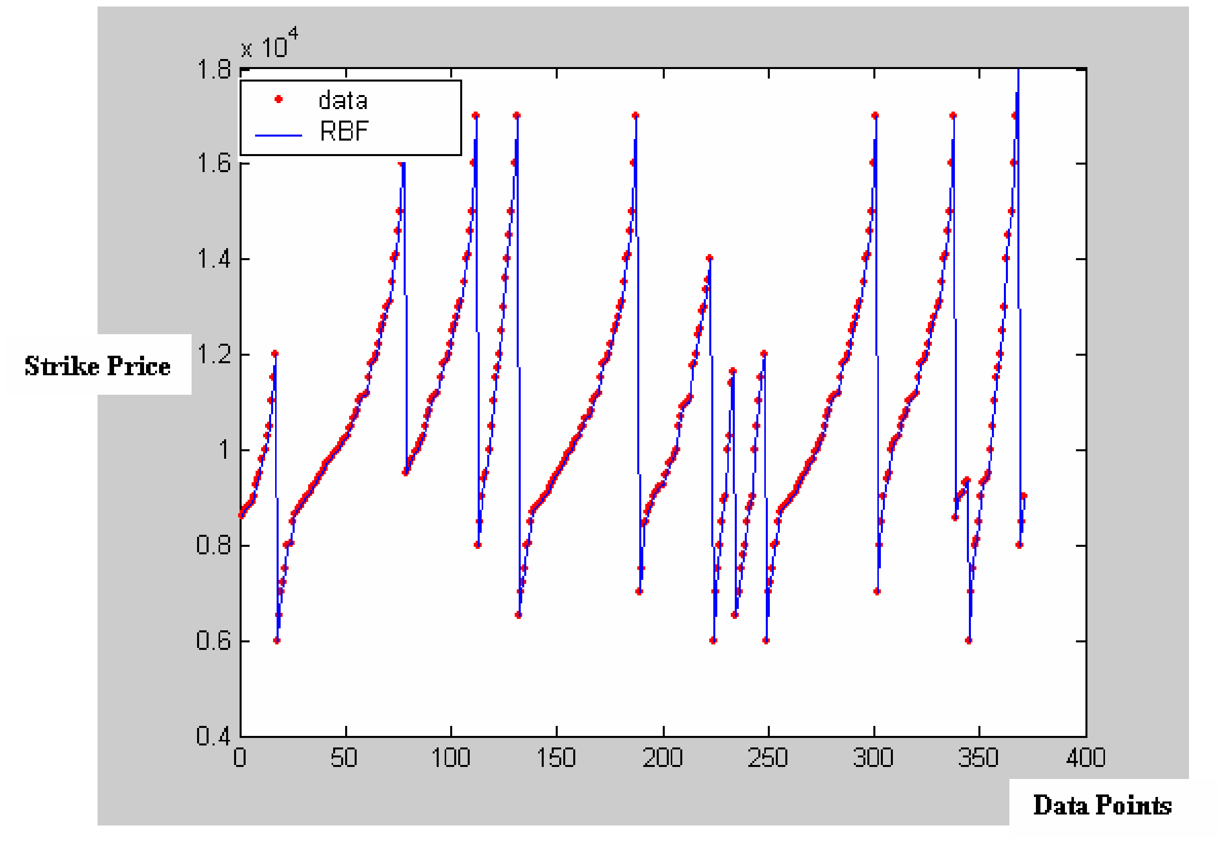

Figure 5-6 'Optimal Architecture' RBF

5.4 Committee Networks

When comparing committee networks to stand-alone networks, generalization capability, computational time as well as portability, always need to be considered. Committee networks in particular have excellent generalization capabilities (as mentioned in Chapter 3.9), but intuitively take a much longer time to compute. This is not always true, since the time taken during trial and error to find good performing stand-alone networks could take far longer than the computational time taken to train a committee network. Another very important aspect is portability. To be able to run the simulation on different data set without previous knowledge or understanding of the data is essential in a working environment. With regard to this attribute, the committee network outperforms any stand-alone network.

The standard committee network consists of 10 MLP and 10 RBF networks, each of which gets the same inputs. Each neural network in the standard committee had different architectures. The error from the individual stand-alone networks varied between 0.3 % and 13 %. The architectures were chosen in a random range of values. Figure 5.7 illustrates the output of the standard committee network.

Initially the stand-alone networks were each given equal weights followed by the Standard Weighting (SW) method mentioned in Chapter 3.9. The error was effectively reduced by using this technique to average the MLP and RBF networks, Figure 5.8. The error of the standard committee network with equal weights was just under 3 %, while the error when using the SW weighting method produced an error of only about 0.7 % with the computational time only increasing very slightly. The computational time both these committee networks were just under 33 seconds. This is excellent performance, when comparing these results to the 'optimal architecture' networks.

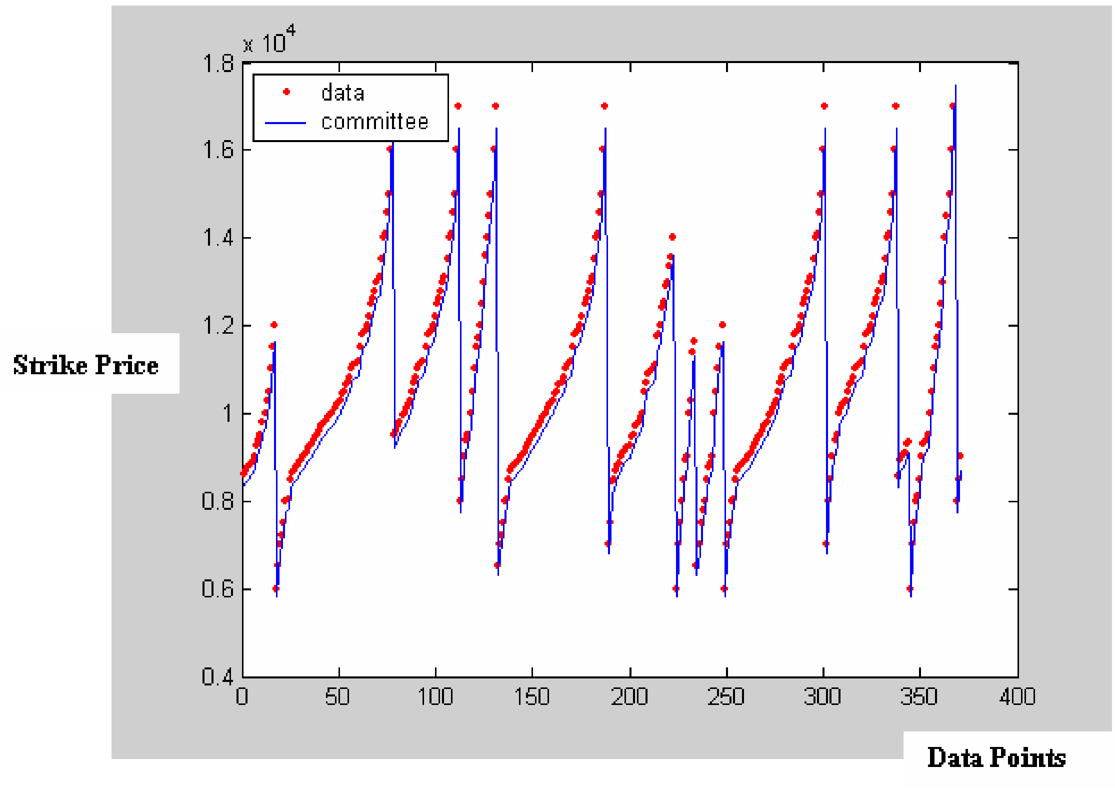

Figure 5-7 Standard Committee

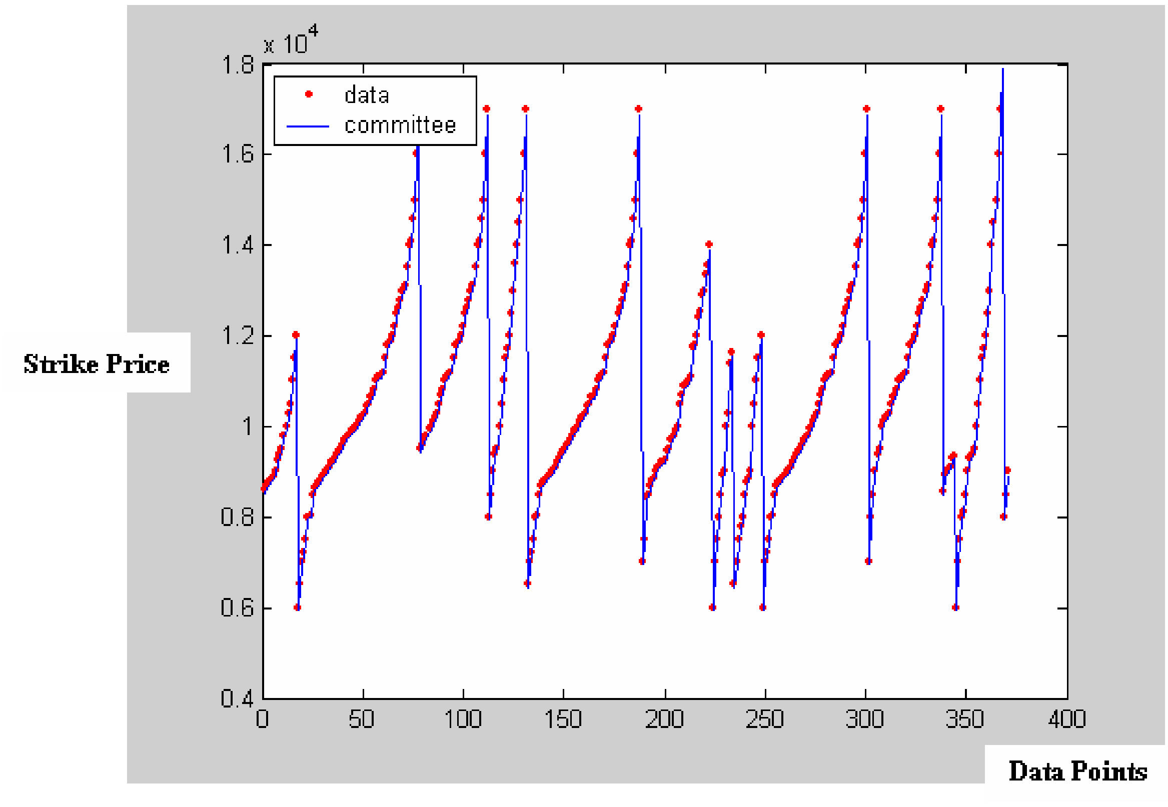

Figure 5-8 Standard Committee with SW weighting

5.4.1 Committee with Neural Network Integrator (CNNI)

As mentioned in Chapter 3.8, there are many different committee network architectures that can be implemented. The CNNI method uses a stand-alone MLP or RBF network to combine the outputs of the entire committee network. The final network in the CNNI method uses all the outputs of the first layer of networks as its input. When using a MLP network as the final network the results were again a great improvement on the standard committee with equal weights and it also performed better than the standard committee with the Standard Weighting (SW) method. The error for the CNNI method was just over 0.1 seconds with a computational time of about 34 seconds.

When using a RBF network as the final network an even better improvement was achieved. The error was reduced to less than 0.001 while the computational time was about the same as using the MLP network.

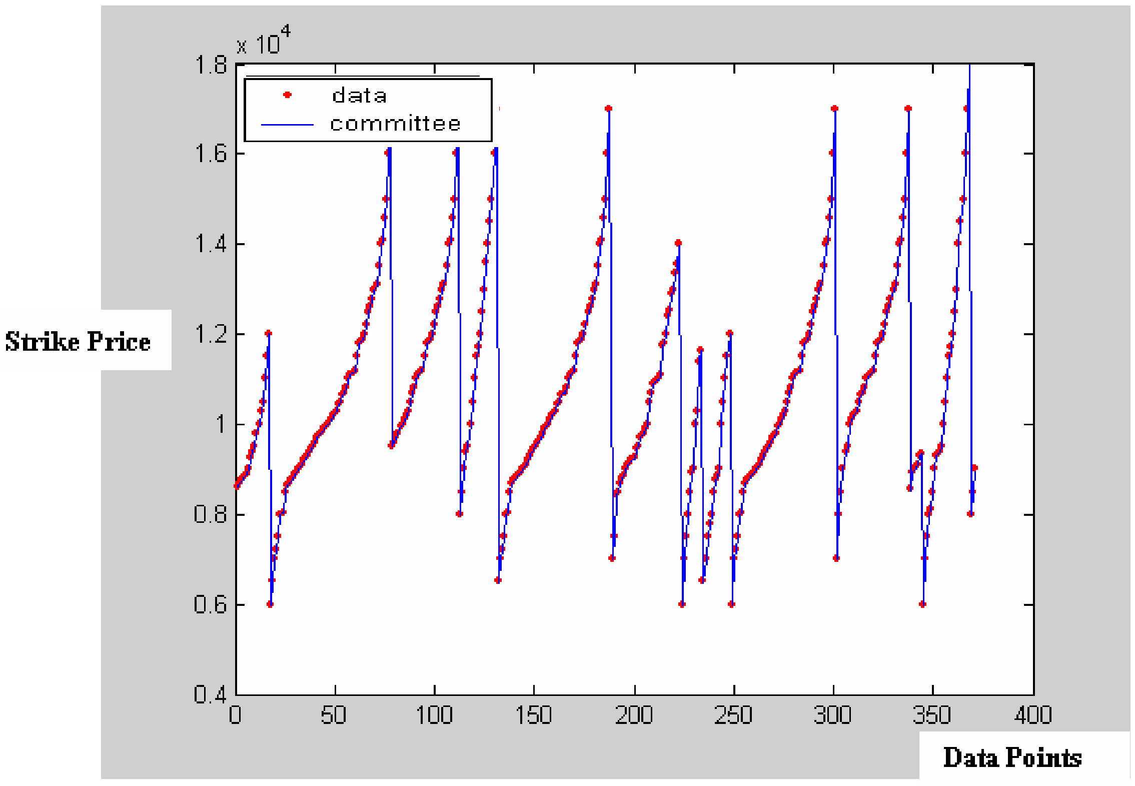

Figure 5-9 CNNI MLP

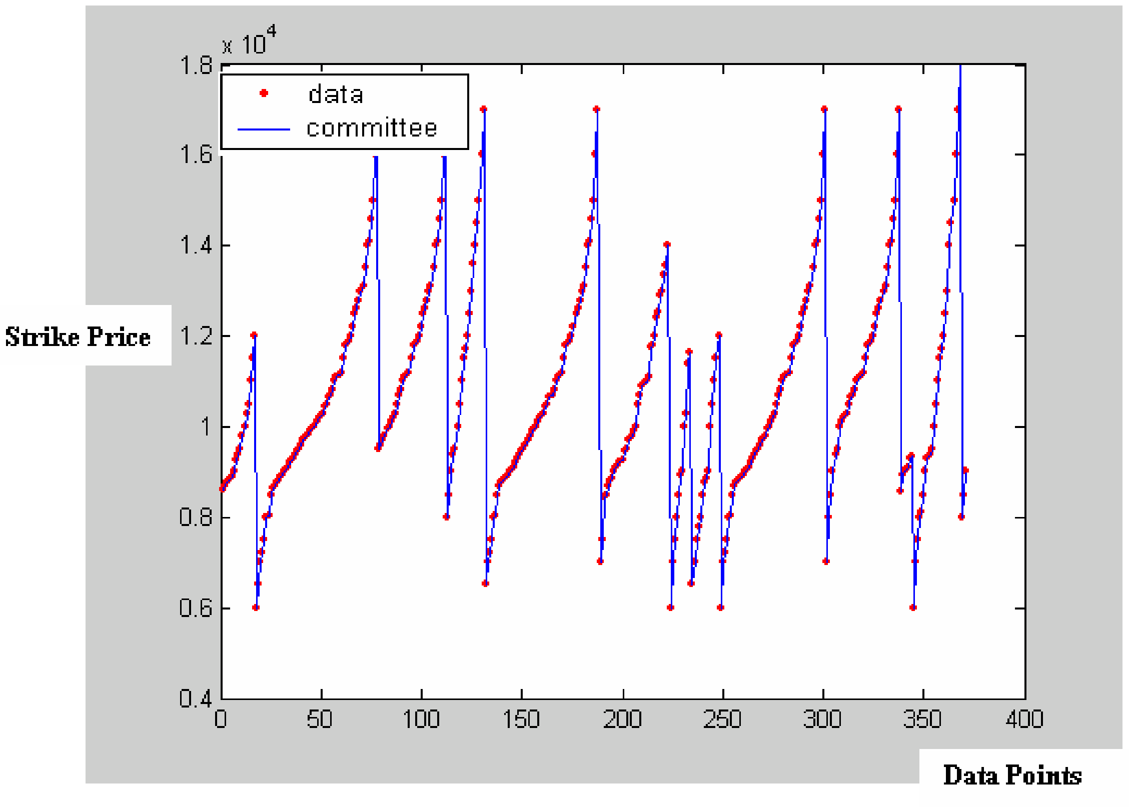

Figure 5-10 CNNI RBF

It’s worthwhile to note that the architecture of the final layer network also plays an important role in model accuracy. As in the first section of this chapter, trial and error was used to find good performing final networks.

5.4.2 Double Layered Committee Network (DLCN)

Intuitively, this method should double the computational time since there is a smoothing neural network after each of the first layer networks, thus doubling the amount of networks to be trained. The architectures of the second layer networks were substantially different thus leading to extra computational and programming time outside of the simulation time. The results were an improvement on the standard committee with equal weights but failed to improve on the results with the Standard Weighting (SW) method. The resulting error was just over 1 % for the final layer being combined with equal weights and just under 1 % for final layer being combined by the SW weighting method. Thus, the big increase in computational time doesn’t lead to better results. The computational time for the actual simulation without the extra setting up time was just under a minute.

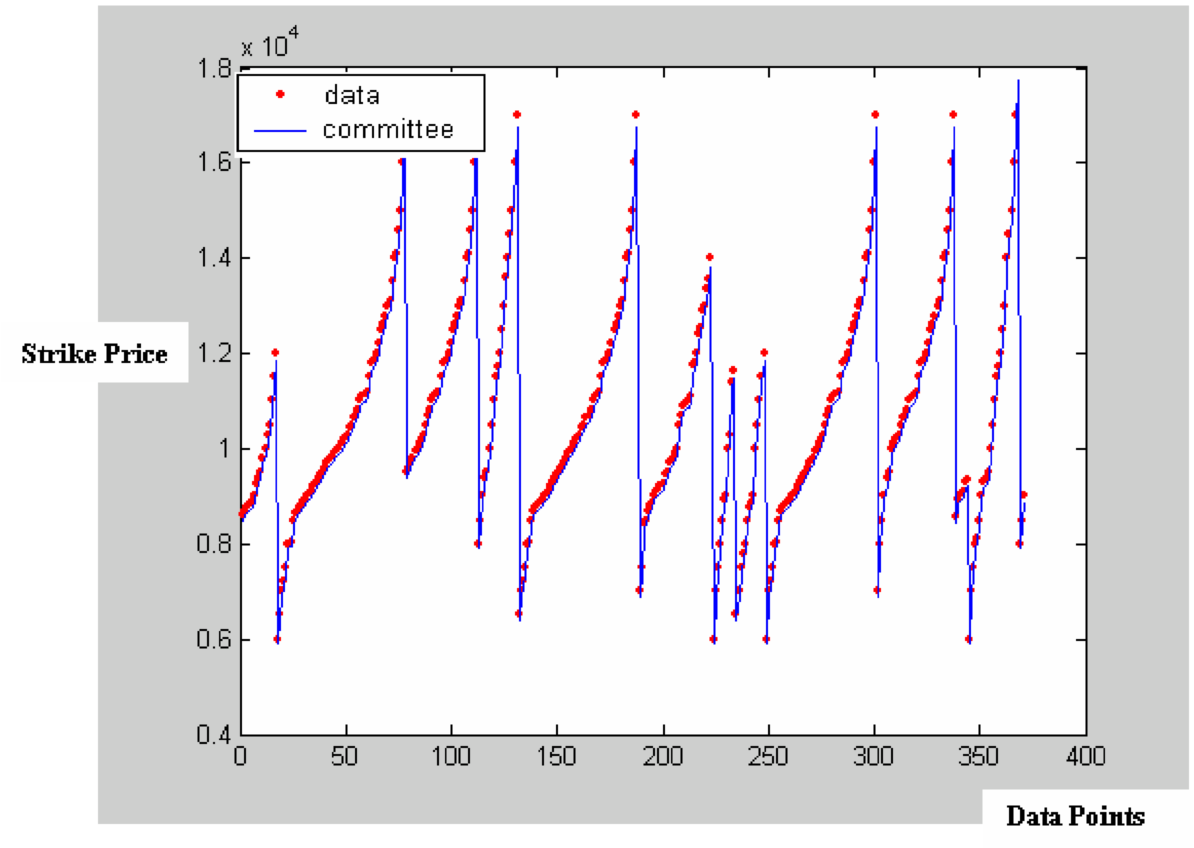

Figure 5-11 DCLN with SW weighting

5.4.3 Circular Committee Network (CCN)

The CCN method achieved much better results than the standard networks keeping in mind the fact that it took much less time computationally. Using equal weights for combining the networks, the resulting error was less than 0.06 % with a computational time of about 14 seconds. While the error was reduced even further by using the Standard Weighting (SW) method, the resulting error was just over 0.0002 %, with a similar computational time.

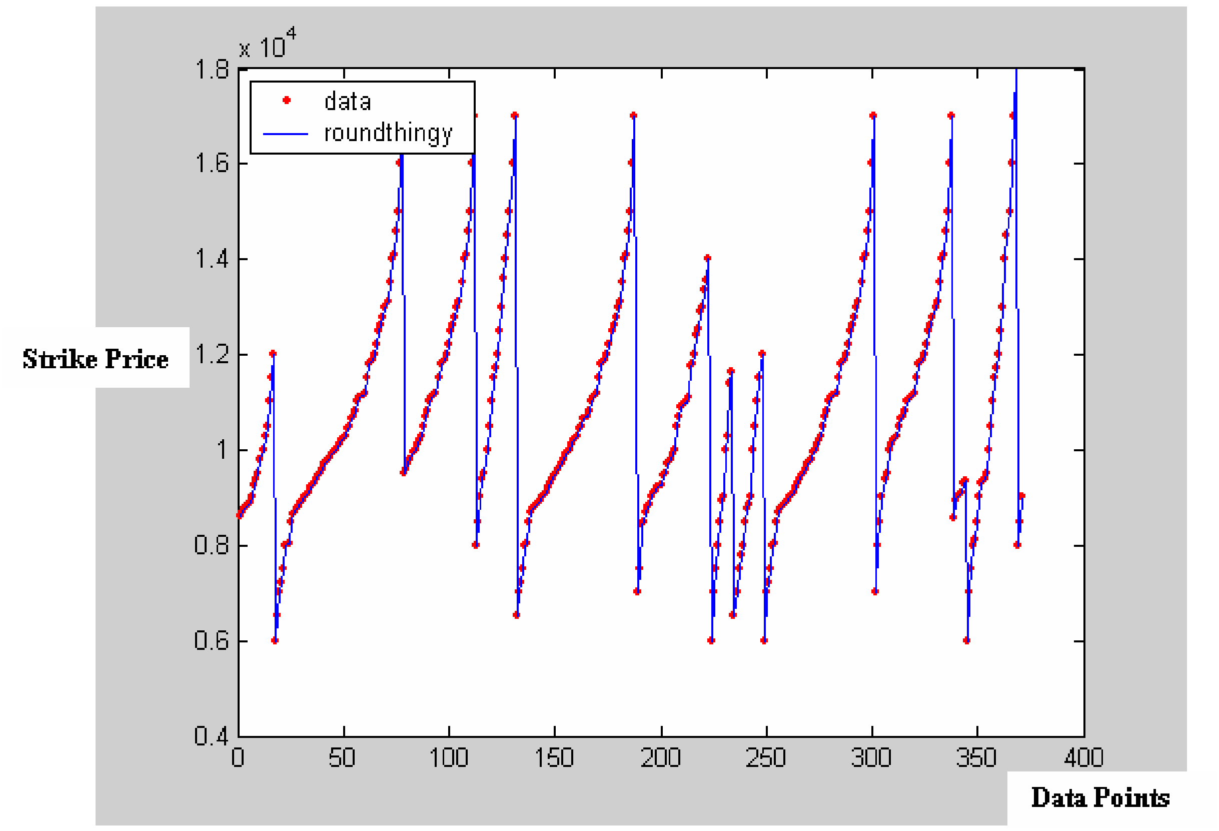

Figure 5-12 CCN with SW weighting

5.4.4 Genetic Algorithm with Metropolis-Hastings Algorithm (GAMHA) Committee Network

Finally, using the technique discussed in Chapter 4.2.4 for optimizing the committee of neural networks, the individual networks within the committee network were optimized in such way so as to improve the performance of the committee network. Again, as with ‘optimal architecture’ stand-alone networks, the computational time was unfeasible in any real world environment. To achieve good results the simulation had to run for several hours. In this particular simulation 20 MLP and 20 RBF networks were used to achieve an error of just under 0.08 % using the Standard Weighting (SW) method.

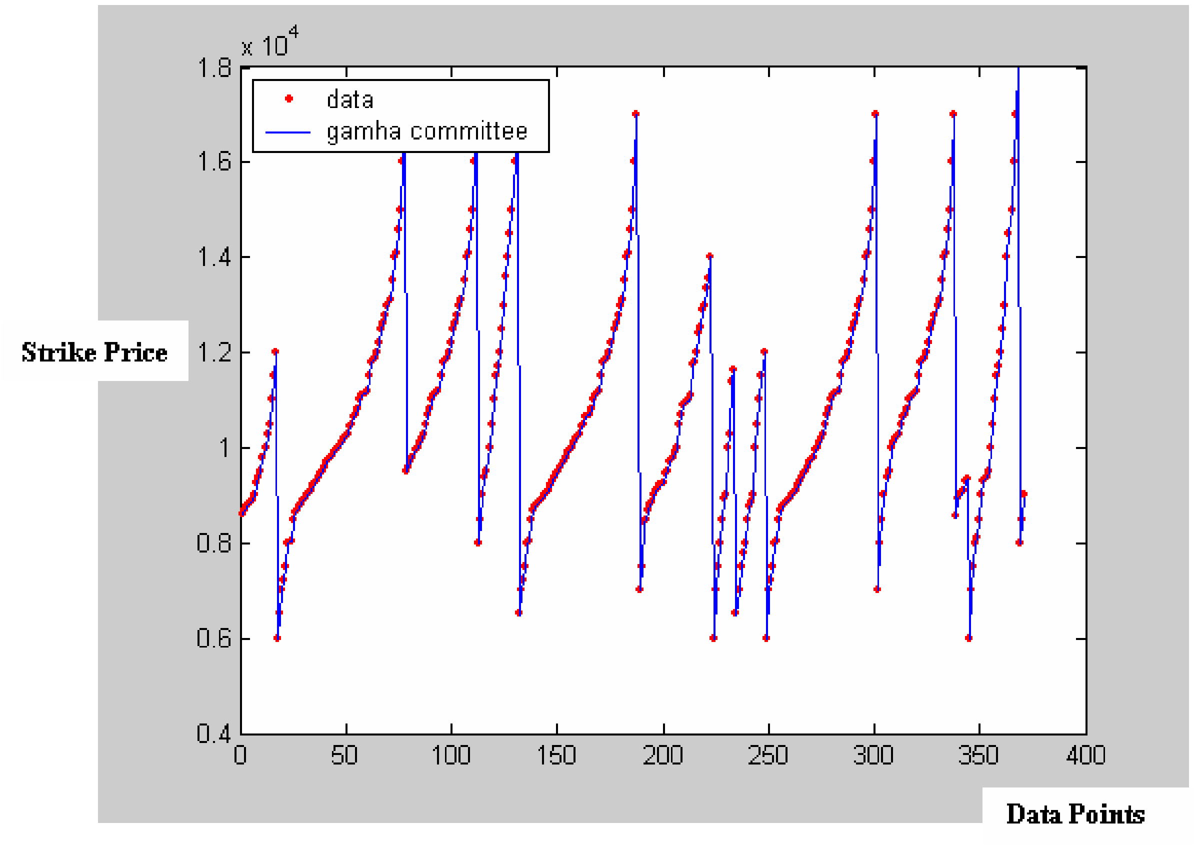

Figure 5-13 GAMHA

5.5 Conclusion

All the techniques introduced in previous chapters are now implemented using the methodologies developed as well as the Netlab software toolbox. Firstly the stand-alone MLP and RBF networks are implemented using architectures that are chosen by trial and error. The architectures of these stand-alone networks are then optimized using particle swarm optimization. Different committee networks are then introduced. The stand-alone networks within the committee network are combined using various techniques developed. These include the following methods; the committee with network integrator, the double layered committee network and the circular committee network. The results and conclusions are discussed further in the next chapter.

6. Discussion & Conclusion

The results in Chapter 5 are summarized in Table 6.1 and discussed below:

Type of Network Trained	Computational Time	Resulting Error
Stand-alone MLP 1	1.015 s	8.885 %
Stand-alone MLP 2	8.485 s	3.2653 %
Stand-alone RBF 1	0.235 s	8.2807 %
Stand-alone RBF 2	0.969 s	54.4137 %
Optimal MLP (with PSO)	Over 1 hour	0.0045 %
Optimal RBF (with PSO)	Over 1 hour	0.0046 %
Standard Committee (equal weights)	32.406 s	2.9258 %
Standard Committee (SW weighting)	32.535 s	0.6797 %
CNNI (MLP)	33.255 s	0.1411 %
CNNI (RBF)	32.955 s	0.00068 %
DLCN (equal weights)	57.425 s	1.3675 %
DLCN (SW weighting)	57.465 s	0.9255 %
CCN (equal weights)	13.715 s	0.0597 %
CCN (SW weighting)	13.735 s	0.00024 %
Optimized Committee (GAMHA)	Several hours	0.0783 %
Averages (excluding optimal networks)	23.6829 s	1.9013 %

Table 6-1 Table of results

Numerous conclusions can be drawn from the above results. Firstly, the computational time for the stand-alone networks are misleading, this is because good performing architectures need to be found by trial and error and this could lead to several networks being trained to eventually find good performing networks. The stand-alone RBF greatly outperforms the MLP with regard to computational time, the MLP, however, seems to be more a reliable network across all types of architectures.

Optimizing the stand-alone networks produce great results at the expense of a huge amount of computational time. Thus, these networks are unfeasible in a working environment where there are time constraints. The improvements of the results when compared to the unoptimized stand-alone networks, however, are huge.

The generalization and portability capabilities of committee networks are easily accessible with computational time that is proportional to the amount of networks trained. Using the Standard Weighting (SW) method, the committee achieves results comparable to the 'optimal architecture' networks. Since this network has major portable and generalization advantages over the 'optimal architecture' network, it seems the obvious choice when comparing the two.

The committee with neural network integrator method (CNNI) shows even more improvement over the above methods, with slightly more computational time. The one draw back of this particular method is that it could be over fitted to the validation/test data. It this respect it loses some of its generalization capabilities. The computational time is slightly misleading since a good architecture needs to be found for the final network. Here the RBF network outperforms the MLP network in the final layer.

The double layered committee network (DLCN) sees an increase in error as well as computational time. This method also included a lot of extra time to find architecture in the second layer that produced good results. Overall this network when compared to the CNNI and the standard committee doesn’t perform well.

Table 6.1 shows that the best performing network is the circular committee network (CCN) with SW weighting and since this network gives the best error result as well as the fact that its computational time is well below the average. The computational time with network is again misleading because good architectures needed to be found to achieve these results.

It is difficult to judge which is the best method overall. Each method has its advantages and disadvantages when comparing errors, computational time, generalization capability as well as portability capability.

Overall, since it has excellent generalization capability as well as portability, as well as the fact that it gives good results in just over average computational time, the standard committee with SW weighting seems a good choice for application to this type of highly non-linear data.

7. Bibliography

1. Z. Bodie, A. Kane, A. J. Marcus, 'Essentials of Investments', Homewood, Ill.: Irwin, 1992.
2. J. Hull, 'Options, Futures, and Other Derivatives', third edition, Upper Saddle River, New Jersey, Prentice-Hall, Inc., 1997.
3. S. Benninga, 'Financial modeling', second edition, Cambridge, Mass, MIT Press, 2000
4. F. Black and M. Scholes, 'The Pricing of Options and Corporate Liabilities', J. Political Economy, vol. 81, pp. 637-659, May/June 1973.
5. J. Yao, Y. Li and C. L. Tan, 'Option price forecasting using Neural Networks', Omega, vol.28, issue 4, pp 455-466, 1 August 2000.
6. K. Gurney, 'An Introduction to Neural Networks', 11 New Fetter Lane, London, UCL press 1997.
7. L Fausett, 'Fundamentals of Neural Networks', Prentice Hall, January, 1994.
8. L. Smith, 'An Introduction to Neural Networks', University of Stirling, http://www.cs.stir.ac.uk/~lss/NNIntro/InvSlides.html, last accessed – 22 June 2004.
9. C. M. Bishop, 'Neural Networks for Pattern Recognition', Oxford: Oxford University Press, 1995.
10. J. Duarte and Y. Ait-Sahalia, 'Nonparametric Option Pricing Under Shape Restrictions', Journal of Econometrics, vol. 116, issues 1-2, pp 9-47, September/October 2003.

11. M. Broadie, J. Detemple, E. Ghysels and O. Torres, 'NonParametric Estimation of American Options Exercise Boundaries and Call Options'. Unpublished Manuscript.

12. G. Montagna, M. Morelli, O. Nicrosini, P. Amato and M. Farina, 'Pricing Derivatives by Path Integral and Neural Networks', Statistical Mechanics and its Applications, vol. 324, issues 1-2, pp 189-195, 1 June 2003.

13. R. Gencay and M. Qi, 'Pricing and Hedging Derivative Securities with Neural Networks: Bayesian Regularization, Early Stopping, and Bagging', IEEE Transactions on Neural Networks, vol. 12, no. 4, pp 726-734, July 2001.

14. Anders, Ulrich, Korn, Olaf, Schmitt, 'Improving the pricing of options: A neural network approach', Journal of Forecasting, vol. 17, no. 5-6, pp 369-388, Sep-Nov 1998.

15. H. Ghaziri, S. Elfakhani, J. Assi, 'Neural networks approach to pricing options', Neural Network World, vol. 10, no. 1-2, pp 271-277, 2000.

16. P.R. Lajbcygier, J. T. Connor, 'Improved option pricing using bootstrap methods', IEEE International Conference on Neural Networks - Conference Proceedings, vol. 4, pp 2193-197, 1997.

17. M. P. Perrone, L. N. Cooper, 'When networks disagree: ensemble methods for hybrid neural networks', Artificial Neural Networks for Speech and Vision, Chapman and Hall, London, UK, 126-142.

18. T. Marwala, 'Fault Identification Using Neural Networks and Vibration Data', PhD thesis, Cambridge.

19. D. A. Coley, 'An Introduction to Genetic Algorithms for Scientists and Engineers', World Scientific Publishing Company; Bk&Disk edition (November 1, 1997)

20. http://www.faqs.org/faqs/ai-faq/genetic/part1 last accessed - 9th December 2004

21. J. Kennedy, R. C. Eberhart, '*Particle swarm optimization'*, International Conference on Neural Networks, IV, Perth, Australia, Piscataway, NJ, IEEE Service Center (1995), pages 1942-1948.

22. L. Haiming, Yen, G. Gary, 'Dynamic population strategy assisted particle swarm optimization', IEEE International Symposium on Intelligent Control - Proceedings, 2003, p 697-702

23. J. Kennedy, C. Russell, 'Swarm Intelligence', Morgan Kaufmann Publishers, 1st edition, March 2001.

24. F. Krogstad, 'Coding the Metropolis-Hastings Algorithm', UW Forestry, June 9, 1999.

25. J, Gasemyr, 'On a adaptive version of the Metropolis-Hastings alogorithm with independent proposal distribution.' Scand. J. Statist.,30, no. 1, 159-173.

26. JSE Securities Exchange of South Africa, http://www.safex.co.za last accessed – 14 June 2004.

27. I. Nabney, 'Netlab: Algorithms for Pattern Recognition', Springer-Verlag UK, November 2001.

Appendix A

A.1 PSO

Pseudo code:

```
Initialization:

No.of.particles

No.of.neighbours

Stopping.criteria

Dimension

Bestfunctionvalues                    (* an array of each particles best function value

For all particles [i]

        For dimensions [j]

                        Particle.next[j] = random

                        Particle.velocity = random(restricted)

        End

End

Particle.neighbours[i] = Getneighbours [i]    (*Getneighbours is a function that finds a

particular

                                              (* particle's neighbour

PSO loop:

While (best(Bestfunctionvalues)-worst(Bestfunctionvalues)) > Stopping.criteria


                For all particles [i]

                        For dimensions [j]
```

```
            Particle.current [j]  = Particle.next [j]
                        End
        Fitness = Test (particle.current)       (*  Test is a function that attains the function
value
        If  fitness > Bestfunctionvalue [i]
                    Bestfunctionvalue [i] = fitness
            Particle.best = particle.current
        End
        Fitneigh = getfittestneighbour              (* function to find a particle's fittest
neighbour
        Particle.velocity = findnewvelocity          (* function to find a particle's next
position
        Particle.next = Particle.current + Particle.velocity
        End
End
```

A.2 Optimal committee using GAMHA

1. generate a committee of networks:

   generate architectures of stand-alone networks from a random range of values

2. select parents:

   stochastic selection

3. crossover using real coded GA

   $x^{i+1} = \alpha_i x^i + (1-\alpha_i) y^i$
   $y^{i+1} = \alpha_i y^i + (1-\alpha_i) x^i$

   where $\alpha_i \in$ rand(-0.5,1.5)

4. mutation using real coded GA

   probability of x^i being chosen = 0.01.

   $x^i = x^i + \gamma\left(U^i - L^i\right)$

   γ=0.01

   U = upper limit

   L = lower limit

5. selection using MHA

   for all x^i

   draw x^* from($u :\ N(0,1) + \mu$) draw u from $u :\ U(0,1)$

if $u < \frac{f(x^*)}{f(x^i)}$

then $x^{i+1} = x^*$

else $x^{i+1} = x^i$

6. repeat step 2 – 5 until stopping criterion is met

Appendix B

Option Pricing Using a Committee of Neural Networks and Optimized Networks *

Zaheer A. Dindar
School of Electrical and Information Engineering
University of the Witwatersrand
P/Bag 3, Wits, 2050, South Africa
z.dindar@ee.wits.ac.za

Abstract[*] **-** *The derivative market has seen tremendous growth in recent times. We look at a particular area of these markets, viz. options. The pricing of options has its roots in stochastic mathematics since option pricing data is highly non-linear. It seems obvious to apply the training techniques of neural networks to this type of data. The standard Multi-Layer Perceptron (MLP) and Radial Basis Functions (RBF) were used to model the data; these results were compared to the results found by using a committee of networks. The MLP and RBF architecture was then optimized using Particle Swarm Optimization (PSO). The results from the 'optimal architecture' networks were then compared to the standard networks and the committee network. We found that, at the expense of computational time, the 'optimal architecture' RBF and MLP networks achieved better results than both un-optimized networks and the committee of networks.*

Keywords: Options, Multi-layer Perceptron (MLP), Radial Basis Functions (RBF), Particle Swarm Optimization.

1. Introduction

"Horror stories about large losses incurred by high-flying traders in the derivative markets such as those for futures and options periodically become a staple of the evening news. Indeed, there were some amazing loses to report in the last decade; several totaling hundreds of millions of dollars, and a few amounting more than a billion dollars. In the wake of these debacles, some venerable institutions have gone under…

…These stories, while important, fascinating, and even occasionally scandalous, often miss the point. Derivatives when misused can indeed provide a quick path to insolvency. When used properly, however, they are potent tools for risk management and control." [14]
Many investors, these days, are opting for securities such as mutual funds stocks and bonds [4]. Options are a security that was first traded in the 1970's, since then there has been a dramatic growth in these markets and they are currently being exchanged all over the world. Options are also traded in huge volumes by banks and other institutions [4].

Option markets attract many investors and if used properly could be very lucrative. The year 1973 saw great inroads being made in this market, the work of Black and Scholes, with an option pricing model being developed [2]. This particular model, though very good, depends on various assumptions which at times fail.

Many of the recent applications of Neural Networks to finance, especially option pricing, have just touched on the comparison of a single method to the Black/Scholes option pricing model.

In this study we intend to compare different neural network techniques in option pricing. Standard MLP and RBF networks are compared to the committee of networks approach, which consisted of a few MLP and RBF networks. We then used Particle Swarm Optimization to optimize the architectures of the MLP and RBF networks. The results for all the techniques were then compared. We show that network architecture plays a huge role in the models' accuracy, and that with the appropriate architecture near perfect results can be achieved.

[*] *IEEE* International Conference on Systems, Man and Cybernetics, The Hague, Holland, 2004.

2. Background

2.1 Options

An option is a contract giving the buyer, of this contract, the right but not the obligation, to buy or sell a particular underlying asset at a specific price on or before a certain date. Underlying assets include stocks, stock indices, foreign currencies, debt instruments, commodities, and future contracts [14].

There are two types of options, viz. a call and a put. A call option gives the holder the right to buy the asset by a certain date for a certain price. A put option gives the holder the right to sell the asset by a certain date for a certain price.

The two categories are the American option and the European option. An American option can be exercised at any time between the date of purchase up to the expiration date. A European option can only be exercised at the expiration date. The possibility of early exercise makes American options more valuable than otherwise similar European options; it also makes them more difficult to value. [12]

Modern option pricing techniques, with roots in stochastic calculus, are often considered among the most mathematically complex of all applied areas of finance [12]

In a Nobel Prize winning paper, Black and Scholes succeeded in solving their differential equation to obtain exact formulas for the prices of European call and put options, under certain assumptions [2]. The assumptions are:

- Geometric Brownian motion of stock price movements,
- Options have to be exercised at the expiration date (European),
- Constant interest rate,
- Continuous trading without dividends and tax applied to the stocks,
- The market is frictionless [7].

Although research shows that the Black/Scholes model outperforms other option pricing models, real data often violates the above assumptions. Because of this and since most options available in the market are American, other pricing methods need to be found [7].

2.2 Neural Networks

Neural networks are very sophisticated modeling techniques capable of modeling extremely complex functions. In particular, neural networks are non-linear. For many years linear modeling has been the commonly used technique in most modeling domains since linear models are simple to solve and also because non-linear models can be approximated by linear models. Where the linear approximation was not valid or the linear approximation was too localized to give globally valid results the models suffered accordingly [10].
Neural networks techniques learn by example. The neural network user gathers representative data, and then invokes training algorithms to automatically learn the structure of the data. Although the user does need to have some heuristic knowledge of how to select and prepare data, how to select an appropriate neural network, and how to interpret the results, the level of user knowledge needed to successfully apply neural networks is much lower than would be the case using some more traditional non-linear statistical methods [10].

Because of its non-linear modeling capability neural networks have recently been applied to option pricing data which is highly non-linear.

2.2.1 Multi-Layer Perceptron (MLP)

MLP's are feedforward neural networks. They are supervised networks, so they require a desired response to be trained. They learn how to transform input data into a desired response, so they are widely used for pattern classification. With one or more hidden layers, they can approximate virtually any input-output map [1]. MLP's are probably the most widely used architecture for practical applications. [3]

The network can be described as follows:

$$y(\vec{x}_k) = \sum_{j=1}^{M} w_{kj}^{(2)} \tanh\Big(\sum_{i=1}^{d} w_{ji}^{(1)} x_i + b_j^{(1)}\Big) + b_k^{(2)} \quad (1)$$

Where $\vec{x}_k$ is the input, $w_{ij}^{(1)}$ and $w_{kj}^{(2)}$ are the first and second layer matrices to be minimized, $b^{(1)}$ and $b^{(2)}$ are the bias parameters associated with the hidden units, d is the number of inputs and M is the number of hidden units. In this case the linear activation function was used for the output and the hyperbolic tangent function was used in the hidden layers [3]. Because of its efficiency, the scaled conjugate gradient method was the optimization technique used to train the networks.

2.2.2 Radial Basis Functions (RBF)

RBF's are a type of neural network employing a hidden layer of radial units and an output layer of linear units [3]. The activation of hidden units in a RBF network is given by a non-linear function of the distance between the input vector and a prototype vector [1]. The RBF network can be described as follows:

$$y(\bar{x}_k) = \sum_{i=1}^{j} w_{ij}\phi(\left|\bar{x}_k - x\right|) \qquad (2)$$

Again $\bar{x}_k$ is the input, $\phi(x)$ is the non-linear activation function and w_{ij} is the weight matrix to be minimized [3]. Because it is a distance, the absolute value between the prototype and input vectors is taken. RBF is characterized by its two stage training procedure which is considerably faster than the methods used to train the MLP. During the first stage the parameters of the basis functions are determined using fast unsupervised methods and the second stage determines the weights in the output layer [1].

RBF trains faster than a MLP. Another advantage is that the hidden layer is easier to interpret than the hidden layer in an MLP. Although the RBF is quick to train, when training is finished and it is being used it is slower than a MLP, so where speed is a factor a MLP may be more appropriate [8].

2.2.3 Committee Network

Modeling using neural networks often involves trying multiple networks with different architectures and training parameters in order to achieve acceptable model accuracy. Typically, one of the trained neural networks is chosen as best, while the rest are discarded [13]. The disadvantages of this approach are that effort involved in training is wasted and the network with the best performance on the training data might not have the best performance on test data [1].

Architectures of neural networks play a big role in capturing different aspects of data. Therefore different architectures are better approximators at different points in the data.

To capture these ideas, a committee of neural networks was used. The committee of neural networks, as implemented in Figure 1, was implemented with different architectures and averaged to obtain the output. An example of this approach was used by Peronne and Cooper [11] where they used the weighted average of outputs of individual networks.

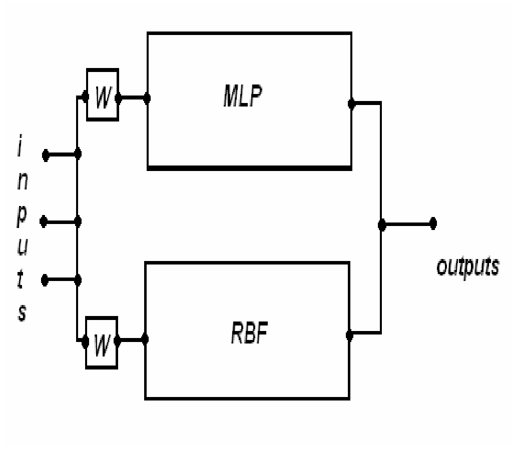

Figure 1. Committee network

This technique uses multiple MLP's, as well as multiple RBF's, each with a different number of hidden layers, in a network, so as to capture the different aspects of the data.

2.3 Particle Swarm Optimization (PSO)

To obtain the MLP and RBF architecture with the best results the MLP and RBF networks' architectures were optimized using Particle Swarm Optimization.

Particle swarm optimization is a technique used to globally optimize non-linear functions. This method is related to evolutionary programming and was discovered through simulation of a simplified social model such as the synchronized flocking of birds or the movement of a school of fish. "School of fish can profit from all discoveries and previous experiences of all members of the school during the search for food."[5]

As with most global optimization techniques particle swarm initializes a population of points, called particles, and function values either by selecting them randomly or at equal spacing across the space. At each iteration, this method influences the update of each particle according to the following:

- Each particle's best function value,
- The function value of its fittest neighbor,
- An element of randomness.

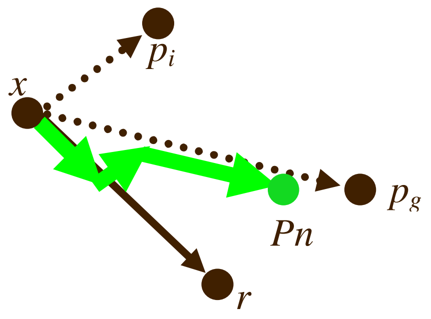

Figure 2. Pi = particle's best position, Pg = neighbor's best position, r = random direction, Pn = next move

Figure 2 shows how this method updates the particles, influenced by the above directions [5].

Compared to GA, the advantages of PSO are that PSO is easy to implement and there are few parameters to adjust. PSO does not have genetic operators like crossover and mutation. Particles update themselves with the internal velocity. They also have memory, which is important for the speed of the algorithm [9].

3. Results

The data set used was collected from the South African Foreign Exchange between January 2001 and December 2003 [6]. The data contained the underlying stock price, strike/option price, time to maturity, stock volatility, market to market price as well as the high and low prices.

The key parameters needed for option pricing are [4]:

- underlying asset price,
- volatility of the underlying asset,
- interest rate,
- time to maturity

Therefore these were the parameters used as inputs. The network output was the strike/option price.

Approximately two thirds of the data set was used for training and the rest for the testing.

There were 5 different training algorithms used, viz. standard MLP and RBF networks, the committee network and the 'optimal architecture' MLP and RBF networks.

The standard MLP and RBF were implemented using arbitrary values for the number of hidden layers and training cycles.

In the implementation, the committee network consisted of 10 MLP's and 10 RBF's, all with different architectures (i.e. number of hidden layers), which were averaged to attain the results.

Using PSO the number of hidden layers and training cycles were found that optimized the neural network architecture, for both the MLP and RBF. What these tests also showed the highly non-linear nature of option pricing.

The results are as follows:

Figure 3 and 4 shows the standard MLP and RBF respectively. The standard networks are characterized by relatively fast training and testing at the expense of accuracy, which was over 8% for both these networks.

The accuracy improved significantly when using the committee network (Figure 5). The error of this network was 3.0995%. The average computational time was over forty seconds.

The optimization of the MLP and RBF networks took several minutes but lead to great accuracy improvements (Figure 6 and Figure 7), surpassing even the committee of networks, with errors of less than 0.01%

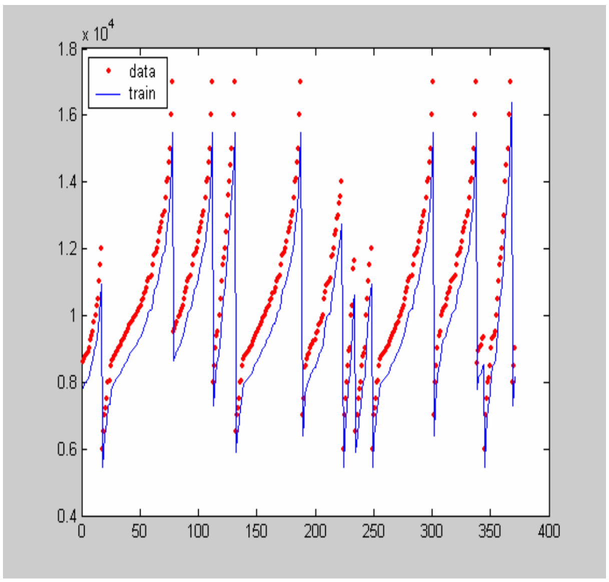

Figure 3. MLP network, error = 8.8850%

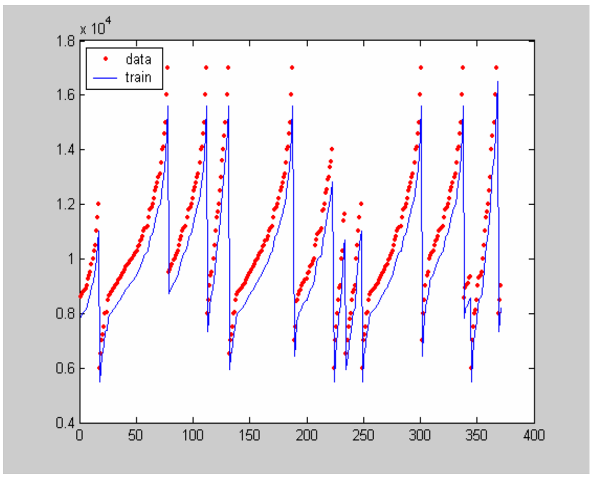

Figure 4. RBF network, error = 8.2807%

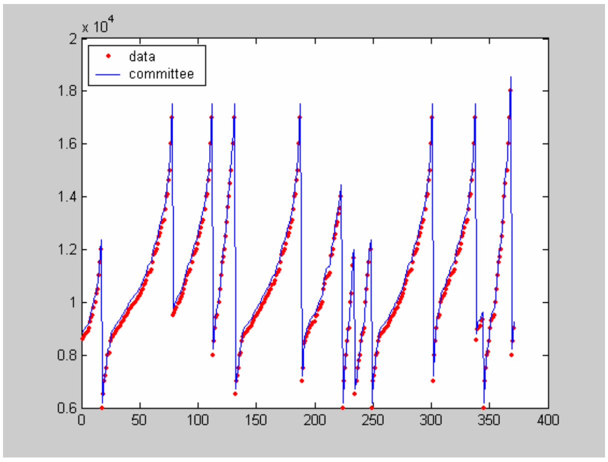

Figure 5. Committee network, error = 3.0995%

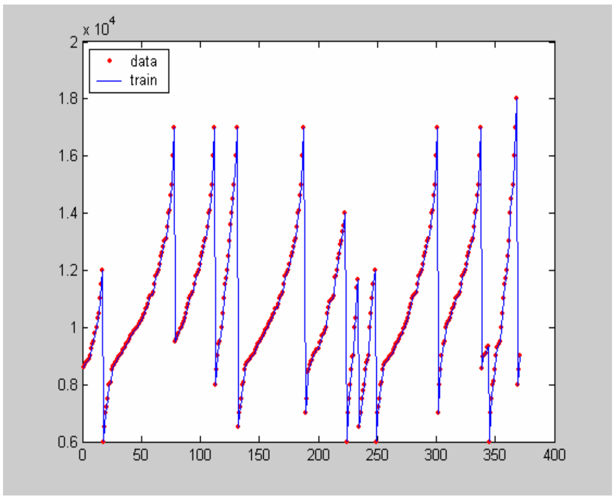

Figure 6. 'Optimal architecture' MLP network, error = 0.0045%.

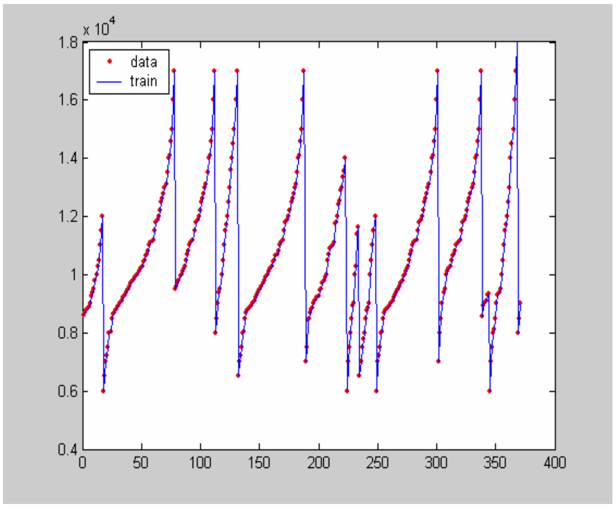

Figure 7. 'Optimal architecture' RBF network, error = 0.0046%.

4. Discussion

Even though the standard methods have larger errors, they prove useful when comparing computational time, since the average computational time was just under a second for MLP and under half a second for the RBF, this included training and testing.

The computational time for the committee network was on average about forty seconds which was considerably less than the computational time for the optimized networks, which took, at times several minutes.

Near perfect results were attained when using PSO to optimize the networks.

5. Conclusion

Since the options market's tremendous growth in recent times, there has been many works relating to these markets. Previous work has seen stand-alone networks being used, the methods of committee networks and optimized networks, using PSO, as adapted in this work, however, have not been used in this application.

Using optimization methods on the committee network could lead to even better results at the cost of computational time. The results prove that these methods are capable of estimating highly non-linear data.

6. References

[1] C. M. Bishop, 'Neural Networks for Pattern Recognition', Oxford: Oxford University Press, 1995.

[2] F. Black and M. Scholes, 'The Pricing of Options and Corporate Liabilities', J. Political Economy, vol. 81, pp. 637-659, May/June 1973.

[3] I. Nabney, 'Netlab: Algorithms for Pattern Recognition', Springer-Verlag UK, November 2001.

[4] J. Hull, 'Options, Futures, and Other Derivatives', third edition, Upper Saddle River, New Jersey, Prentice-Hall, Inc., 1997.

[5] J. Kennedy, C. Russell, 'Swarm Intelligence', Morgan Kaufmann Publishers, 1st edition, March 2001.

[6] JSE Securities Exchange of South Africa, http://www.safex.co.za last accessed – 14 June 2004.

[7] J. Yao, Y. Li and C. L. Tan, 'Option price forecasting using Neural Networks', Omega, vol.28, issue 4, pp 455-466, 1 August 2000.

[8] L Fausett, 'Fundamentals of Neural Networks', Prentice Hall, January, 1994.

[9] L. Haiming, Yen, G. Gary, 'Dynamic population strategy assisted particle swarm optimization', IEEE International Symposium on Intelligent Control - Proceedings, 2003, p 697-702

[10]L. Smith, 'An Introduction to Neural Networks', University of Stirling, http://www.cs.stir.ac.uk/~lss/NNIntro/InvSlides.html , last accessed – 22 June 2004.

[11]M. P. Perrone, L. N. Cooper, 'When networks disagree: ensemble methods for hybrid neural networks', Artificial Neural Networks for Speech and Vision, Chapman and Hall, London, UK, 126-142.

[12]S. Benninga, 'Financial modeling', second edition, Cambridge, Mass, MIT Press, 2000

[13]S. Hashem, B. Schmeiser, Y. Yih, 'Optimal linear combinations of neural networks', IEEE International Conference on Neural Networks - Conference Proceedings, v 3, 1994, p 1507-1512

[14]Z. Bodie, A. Kane, A. J. Marcus, 'Essentials of Investments', Homewood, Ill.: Irwin, 1992.

Appendix C

'Optimal Architecture' Stand-alone and Committee Neural Networks Applied to Option Pricing Data

Zaheer A. Dindar
School of Electrical and Information Engineering
University of the Witwatersrand
P/Bag 3, Wits, 2050, South Africa
z.dindar@ee.wits.ac.za

Abstract: *In this paper the multilayer perceptron (MLP), radial basis functions (RBF) as well as 'committee networks', are used to model highly non-linear data from the South African Foreign Exchange. Architecture plays a very important role in model accuracy of neural networks. When dealing with highly non-linear data, achieving good performing architectures becomes very difficult. We investigate two different optimization algorithms for finding 'optimal architecture' neural networks, namely, particle swarm optimization as well as a modified genetic algorithm. The Metropolis-Hastings Algorithm was used as the selection step in the genetic algorithm. Once optimized, the new 'optimal architecture' networks were compared to the standard networks. The results show huge improvements on the unoptimized networks at the expense of much larger computational times.*

Keywords: Multilayer Perceptron, Radial Basis Functions, Committee Networks, Particle Swarm Optimization, Genetic Algorithm, Metropolis-Hastings Algorithm.

1. Introduction

Artificial neural networks' non-linear modeling capabilities are being applied in many fields of study. With this comes a constant demand for better performing networks when dealing with, sometimes daunting, highly non-linear data. The approach taken in this paper is to explore the architectures of neural networks, by using optimization techniques, in an attempt to find 'optimal architecture' neural networks.

2. Neural Networks

A neural network is an interconnected assembly of simple processing elements, units or nodes, whose functionality is loosely based on the animal neuron [1]. The processing ability of the network is stored in the inter-unit connection strengths, or weights, obtained by a process of adaptation to, or learning from, a set of training patterns.

Neural networks are frequently applied to statistical analysis and data modeling, where it is used as a substitute to standard non-linear regression or cluster analysis techniques [2]. Therefore, they are used for classification, or forecasting. Some examples include image and speech recognition, character recognition, and domains and human expertise such as medical diagnosis, and financial market prediction. Neural networks fall within the sphere of artificial intelligence, so that neural networks are perceived as an alternative to the algorithmic techniques that have dominated in machine intelligence.

Possibly the largest users of neural networks are people using it to analyze badly understood data that arise in the workplace [1].

There are many different types of neural networks and here we look at two; viz. the Multi-layer perceptron and Radial Basis Functions:

2.1 Multi-Layer Perceptron (MLP)

MLP's are feedforward neural networks. They learn how to transform input data into a desired response. Feed forward neural networks provide a general framework for representing non-linear functional mappings between a set of input variables and a set of output variables. This is achieved by representing the non-linear function of however may variables in terms of compositions of non-linear functions of single variable, called activation functions [3].

Networks with just two layers of weights are capable of approximating any continuous functional mapping [3]. They are supervised networks, so they require a desired response to be trained.

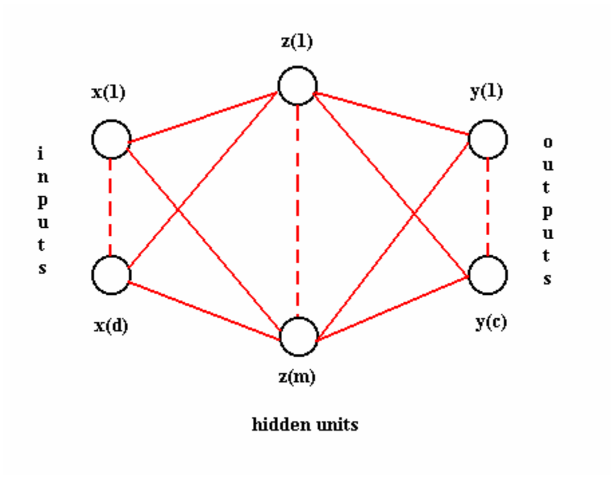

Figure 1 Multilayer perceptron with inputs x, hidden units z and outputs y

The above is an example of a layered network. This network has d inputs, M hidden units and c outputs. The output of this figure analytically is as follows: to attain the i^{th} hidden unit, a weighted linear combination of the d input values and adding a bias, is formulated to give:

$$a_j = \sum_{i=1}^{d} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \qquad [1]$$

Here w_{ij} represents a weight in the first layer, going from input i to hidden unit j and w_{j0} denotes the bias for hidden unit j the hidden unit can be incorporated into the weight matrix by including an extra input variable x_o. Hence the equation now becomes:

$$a_j = \sum_{i=0}^{d} w_{ij}^{(1)} x_i \qquad [2]$$

The activation of hidden unit j is then obtained by transforming the linear sum in the above equation using an activation function $g(g)$ to give:

$$z_j = g(a_j) \qquad [3]$$

The outputs of the network are obtained by transforming the activations of the hidden units using the second layer of processing elements. Thus for each output unit k, we construct a linear combination of the outputs of the hidden units of the form:

$$a_k = \sum_{j=0}^{M} w_{kj}^{(2)} z_j \qquad [4]$$

The activation of the k^{th} output unit with activation $z_0 = 1$ is obtained by transforming this linear combination using a non-linear activation function, to give:

$$y_k = \bar{g}(a_k) \qquad [5]$$

Combining all the above equations [3]:

$$y_k = \bar{g}\left(\sum_{j=0}^{M} w_{kj}^{(2)} g\left(\sum_{i=0}^{d} w_{ji}^{(1)} x_i \right)\right) \qquad [6]$$

Another explanation has its basis in thinking of the neural network as implementing a mathematical function of its inputs, and is especially pertinent if we are dealing with continuous input and output signals. For example if we wanted to make a forecast P_n of a particular stock value based on previous values of the stock price $P_{n-1}, P_{n-2}, \ldots P_{n-k}$ we want to train a network to discover the functional relation between quantities: this is to discover the underlying function $P_n = P_n(P_{n-1}, P_{n-2}, \ldots P_{n-k})$ [4].

A MLP can perform categorization of an arbitrary number of classes and with an arbitrary decision surface. All that is required is that we have a set of inputs and targets, and that we fix the number of hidden units that are going to be used.

2.2 Radial Basis Functions (RBF)

Radial basis functions are another major class of neural networks; here the distance between the input vector and the prototype vector determines the activation unit. Radial Basis Functions have their roots in techniques for performing exact interpolation of a set of data points in a multi dimensional space. The exact interpolation problem required every input vector to be mapped exactly onto the corresponding target vector [1].

The radial basis function approach introduces a set of N basis functions, one for each data point, which take the form $\phi\left(\left\|x-x^{n}\right\|\right)$ where $\phi(g)$ is some non-linear function. Thus, the n^{th} function depends on the distance $\left\|x-x^{n}\right\|$, between x and x^{n}. The output of the mapping is then taken to be a linear combination of the basis functions:

$$h(x)=\sum w_{n}\phi\left(\left\|x-x^{n}\right\|\right) \qquad [7]$$

The most common form of basis function is the Gaussian:

$$\phi(x)=\exp\left(-\frac{x^{2}}{2\sigma^{2}}\right) \qquad [8]$$

Where σ is a parameter that controls the smoothness properties of the interpolating function.

Generalizing this for several outputs: each input vector must be mapped exactly onto an output vector x^{n}, having components t^{n}. The equation becomes [3]:

$$h(x)=\sum_{n} w_{kn}\phi\left(\left\|x-x^{n}\right\|\right) \qquad [9]$$

The exact interpolating function for noisy data is typically a highly oscillatory function. Functions such as these are generally undesirable. The interpolation function that gives the best generalization is typically smoother and averages over the noise in the data. Another disadvantage of exact interpolation, is the fact that the number of basis functions are equal to the number of patterns in the data set, and when computing can become costly to evaluate [1].

There are a number of modifications to the exact interpolating procedure that provides a smooth interpolating function as well as the reduction of the number of basis functions, which are determined by the complexity of the mapping rather than by the size of the data set [1].

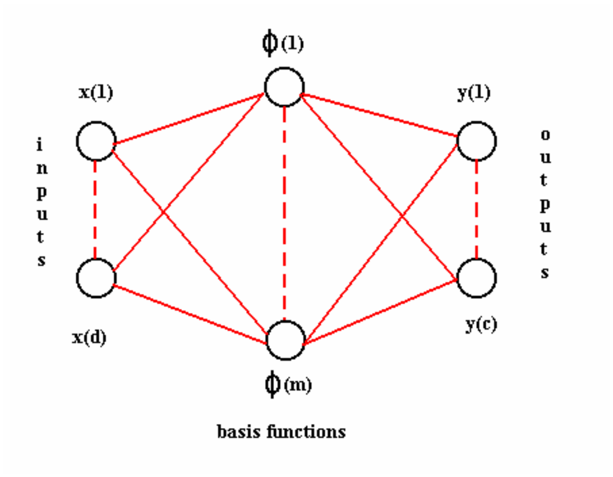

Figure 2 Radial basis functions with inputs x, basis functions ϕ and outputs y.

RBF's are usually two layer networks in which the first (hidden) layer is a series of RBF's and the second (output) layer is a set of linear units that can be thought of as computing a weighted sum of the evidence from each of the feature template RBF units.

3. Committee of Neural Networks

Modeling using neural networks often involves trying multiple networks with different architectures and training parameters in order to achieve acceptable model accuracy.

Selection of the best network is based on the performance of the network on an independent validation or test set (as will be discussed), and to keep only the best performing network and to discard the rest. There are two disadvantages with such an approach; first, all the effort involved in training the remaining neural networks is wasted, second, the networks generalization performance is greatly reduced.

These drawbacks can be overcome by combining the networks together to form a committee (Perrone and Cooper)[5] The importance of such an approach is that it could lead to significant improvements in the predictions on new data, while involving the training of a few additional networks. In fact performance of a committee can be better than the performance of the best stand-alone network used in isolation [3]. The committee of networks contains neural networks with different architectures and or different types of neural networks trained on the same training data set. It might even include different kinds of network models with a mixture of conventional models.

The error can never increase by using a committee of networks. Typically the error is reduced considerably by taking the average error of the combined networks [6].

$$E_{COM} \leq E_{AV} \qquad [10]$$

Another advantage of a committee of neural networks is the fact that it is more reliable than stand-alone networks.

The committee network can be represented by Figure 3:

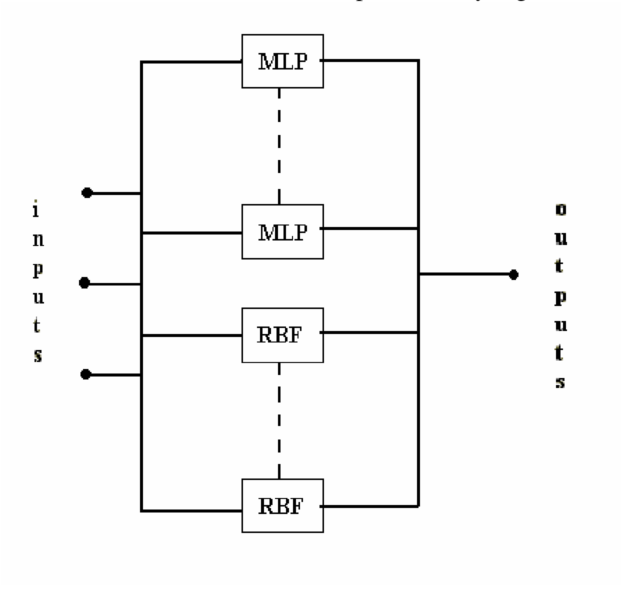

Figure 3 Committee Network

Architectures of neural networks, as well as performance, play a big role in capturing different aspects of data. Since the members of the committee network have different architectures, some will make better predictions than others.

4. Training

Training a neural network is usually done by presenting a training set to the network, and at each step of an iterative process, adjusting the weights of the network to bring its output closer to the desired output. This process of changing or adapting the weights is referred to as the learning rule of the network [1].

5. Cross-validation

The objective of training a neural network is to have a network that performs best on unseen data. A simple method to compare the performance of neural networks, is to test the errors of the networks using a separate validation/test data set. This is done by training many networks on a training set and comparing the errors of the networks on the validation set. The networks that performed best on the validation data set are then selected. This technique is called cross validation[1]

Another very important aspect of attaining good results from neural networks is choosing the correct architecture. The architecture as it is dealt with here refers to the number of hidden layers used to train the neural network.

6. 'Optimal Architecture': Neural Networks and Optimization Techniques

As mentioned before neural network architecture plays a big role in model accuracy. Finding the correct amount of hidden layers is crucial when dealing with highly non-linear data. Because of the non-linearity, the amounts of hidden layers required is almost counterintuitive, which leads, again, to high non-linearity with regard to the best architecture. In an attempt to find the best architectures that model our particular data, the techniques described below are proposed to find 'Optimal Architecture' networks. Particle Swarm Optimization is used to find stand-alone 'Optimal Architecture' networks, while Genetic Algorithm finds the optimal architecture for a committee of neural networks.

6.1 Optimization Techniques

Global optimization algorithms imitating principles of nature have been very useful in recent times and have been applied in various domains. Such phenomena can be found in annealing processes, central nervous systems and biological evolution, which in turn have lead to the field of Evolutionary Computation (EC). [7]

Evolutionary computation includes; Genetic algorithms, evolutionary programming, evolution strategies, classifier systems, genetic programming and numerous other problem solving approaches that are based on biological observations. These observations date back to Charles Darwin's 'theory of evolution', hence the term Evolutionary Algorithms [8].

This term ‘evolutionary algorithm’ refers to evolutionary processes used by computer-based problem solving systems.

6.1.1 Particle Swarm Optimization (PSO)

PSO is an algorithm proposed by James Kennedy and R. C. Eberhart in 1995, motivated by social behavior of organisms such as bird flocking and fish schooling. PSO as an optimization tool provides a population-based search procedure in which individuals called particles change their position with time. This method is related to evolutionary programming and was discovered through simulation of the simplified social model, namely, the synchronized flocking of birds or the movement of a school of fish. In a PSO system, particles fly around in a multidimensional search space. During flight, each particle adjusts its position according to its own experience, and according to the experience of a neighboring particle, making use of the best position encountered by itself and its neighbor. Thus, a PSO system combines local search methods with global search methods, attempting to balance exploration and exploitation.

Thus, PSO is a technique used to globally optimize non-linear functions. Since, a school of fish profits from previous experiences and discoveries of each member during the search for food [9].

As with most global optimization techniques particle swarm initializes a population of points, called particles, and function values either by selecting them randomly or at equal spacing across the space. At each iteration, this method influences the update of each particle according to the following: [10]

Each particle’s best function value, the function value of its fittest neighbor and an element of randomness.

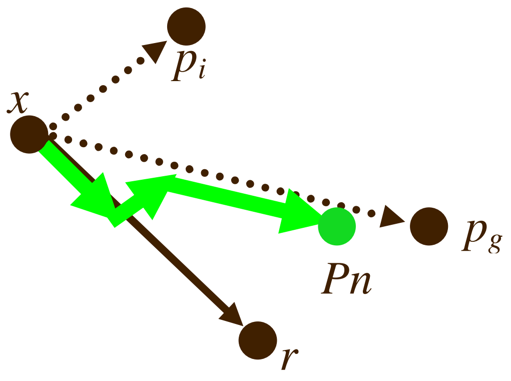

Figure 4. Pi = particle’s best position, Pg = neighbor’s best position, r = random direction, Pn = next move
Figure 1 shows how this method updates the particles, influenced by the above directions.

Algorithm:

- Create random particle population and assign to them random positions,
- Evaluate their function values at their current positions,
- Choose a set of neighbors for each particle,
- Update the particle population using the new direction, called the velocity,
- Repeat until a stopping criterion is met.

The MLP and RBF network architectures were optimized using PSO, to find ‘Optimal Architecture’ MLP and RBF networks.

6.1.2 Metropolis-Hastings Algorithm (MHA)

In mathematics and physics, the Metropolis-Hastings algorithm is an algorithm to generate a sequence of samples from the joint distribution of two or more variables. The purpose of such a sequence is to approximate the joint distribution. This algorithm is an example of a Markov chain Monte Carlo algorithm. The Metropolis-Hastings algorithm can draw samples from any probability distribution P(x), requiring only that the density can be calculated at x [11].

The key to the Metropolis-Hastings algorithm is to create a sampling strategy by which the probability of being in state a and moving to state b is the same as from b to a, subject to a few regularity conditions. This series of draws is accomplished by proposal and acceptance/rejection of candidate values x^*.

Algorithm:

```
for $j = 1:N$
draw $x^*$ from $q(x^* \mid x_j)$, draw $u$ : $U(0,1)$
if $u < \frac{\left[q(x_j \mid x^*) f(x^*)\right]}{\left[q(x^* \mid x_j) f(x_j)\right]}$
then $x_{j+1} = x^*$
else $x_{j+1} = x_j$
```

The proposal of a candidate value x^* is conducted through a proposal function $q(x^* \mid x_j)$ the form of which is quite arbitrary. In order to get a lot of x values, the values of x^* should not be rejected too often. The arbitrariness of the proposal function $q(x^* \mid x)$ and the lack of theory guiding our choice leave lots of room for experimentation [12]. The two functions that suggest themselves are the uniform and normal distributions [11]. The arbitrariness of the proposal function is supported in the acceptance/rejection step, which corrects for unlikely steps from x_j to x^* by accepting them only with the ratio of moving from one to the other.

6.1.3 Genetic Algorithm (GA)

The genetic algorithm is a model of machine learning, which derives its behavior from a metaphor of the processes of evolution in nature. This is done by the creation within a machine of a population of individuals represented by chromosomes. The individuals in the population then go through a process of evolution. Some variables or individuals are better or fitter than others. Those that are better are more likely to survive and regenerate their genetic material. Reproduction allows the creation of genetically radically different offspring that are still of the same general species. At the molecular level what occurs is that a pair of Chromosomes bump into one another, exchange chunks of genetic information and drift apart. This is the recombination operation, which genetic algorithms generally refer to as crossover, because of the way that genetic material crosses over from one chromosome to another. Crossover takes place when the selection of who gets to mate is made a function of the fitness of the individual. The genetic algorithm uses stochastic processes, but the result is distinctly non-random [7].

Firstly, an initial population of parent individuals (feasible solutions) is randomly created. Each individual is represented by a chromosome, a string of characteristic genes. Secondly, all the individuals are ranked with a fitness function appropriate to the problem at hand. The fittest of these, pass directly to the following generation; a process of 'elitism'. Thirdly, a breeding population is formed by selecting top-ranking individuals from those that remain. This is the natural selection step. Lastly, these selected individuals undergo certain transformation via genetic operators to reduce children for the next generation. Operators include recombination by crossover and mutation (a randomly generated gene that is somehow altered). Mutation ensures a genetic diversity in the population [7].

This process is repeated for a certain number of generations, or until some stopping criterion is met.

Algorithm:

1] Generate initial population
2] Generate offspring:
- selection: probability of being accepted according to fitness
- crossover: parents are paired to generate offspring
- mutation: each string has a very small chance of being mutated
- selection/rejection: selection or rejection of the new generation according to some criterion

3] Repeat step 2 until stopping criterion is met.

6.1.3 Genetic Algorithm with Metropolis-Hastings Algorithm

In the above algorithm the selection/rejection step of the new generated population is not specified, and many methods can be used to do this step. Metropolis-Hastings Algorithm (MHA) has been adopted in this text to make the selection or rejection of the new generation in the genetic algorithm.

The Genetic Algorithm Metropolis-Hastings Algorithm (GAMHA) method was then used to manipulate the architectures in the individual MLP and RBF networks so as to optimize the committee of networks.

7. Results

The data set used was collected from the South African Foreign Exchange between January 2001 and December 2003 [14]. The data contained the underlying stock price, strike price, time to maturity, stock volatility, market to market price as well as the high and low prices.

The inputs were underlying asset price, volatility of the underlying asset, interest rate and time to maturity. The network output was the strike price.

Approximately two thirds of the data set was used for training and the rest for the testing.

7.1 'Optimal Architecture' Multi-layer Perceptron (MLP) and Radial Basis Functions (RBF) using Particle Swarm Optimization (PSO)

Particle Swarm Optimization was used to find the optimal architecture of MLP and RBF networks by optimizing the error function; this was achieved by adjusting the number of hidden units and training iterations. The results from these 'Optimal Architecture' MLP and RBF networks are as follows:

The error was greatly reduced achieving near perfect results. The errors for both, MLP and RBF, 'optimal architecture' networks were less than 0.005%. The computational time was quite high, it sometimes in excess of one hour to run an 'optimal architecture' simulation. This was due to the amount of networks that had to be trained and the highly non-linearity of the data, which meant that finding a global minimum could take a really long time. When running the simulation for a shorter period, say a few minutes, the results would still be a great improvement on the stand-alone MLP and RBF networks. This shows that the 'optimal architecture' program would choose local minima as its output. Figure 5 and Figure 6 illustrate the standard MLP and RBF, respectively. The 'optimal architecture' MLP and RBF networks are illustrated in Figure 7 and Figure 8, respectively.

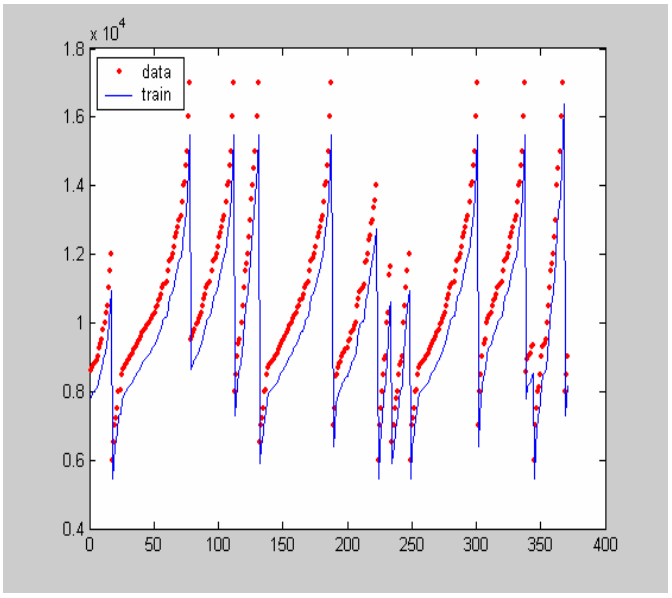

Figure 5 Standard MLP

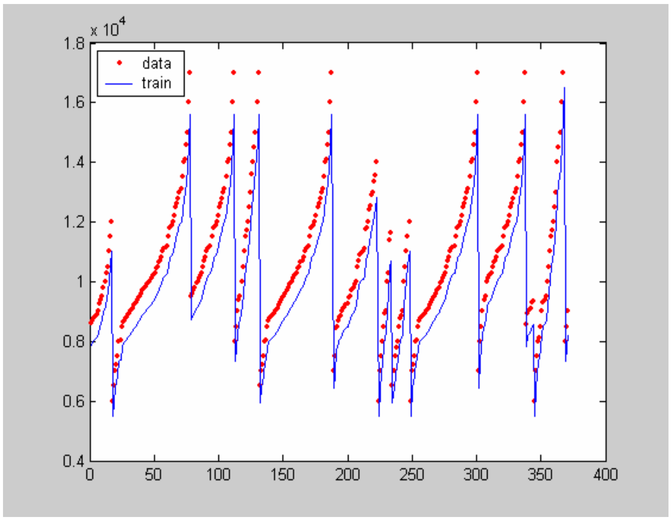

Figure 6 Standard RBF

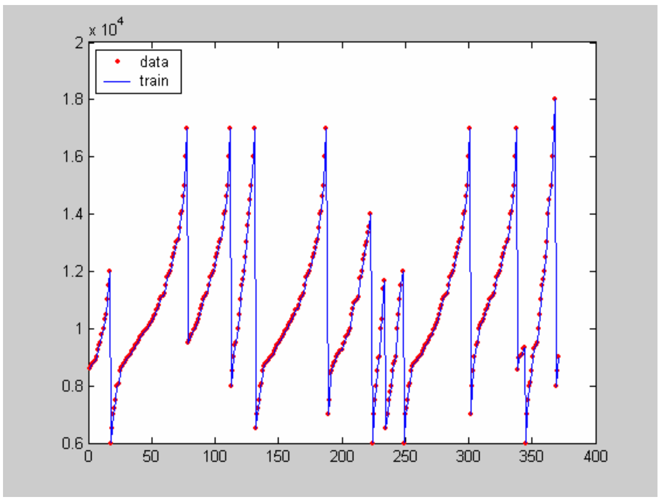

Figure 7 ‘Optimal Architecture’ MLP

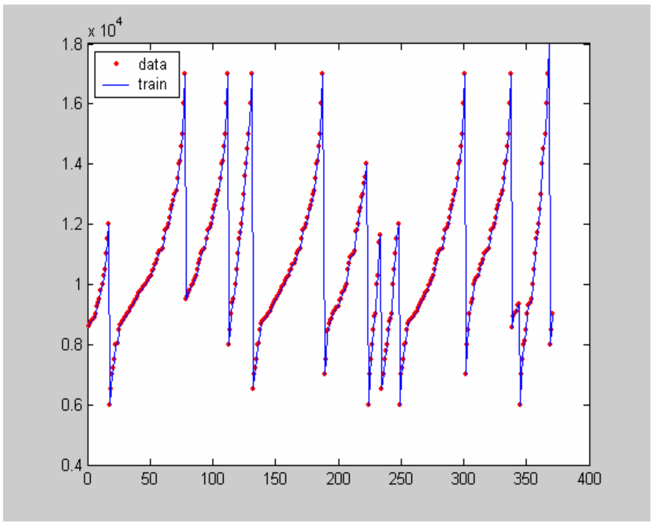

Figure 8 ‘Optimal Architecture’ RBF

7.2 Genetic Algorithm with Metropolis-Hastings Algorithm (GAMHA) Committee Network

Finally using the technique discussed in section 6.1.3 for optimizing the committee of neural networks, the individual networks within the committee network were optimized in such way so as to improve the performance of the committee network. To achieve good results the simulation had to run for several hours. In this particular simulation 20 MLP and 20 RBF networks were used to achieve an error of just under 0.08 %. Figure 9 illustrates the output of the standard committee network and Figure 10 illustrates the output for the ‘optimal architecture’ committee network.

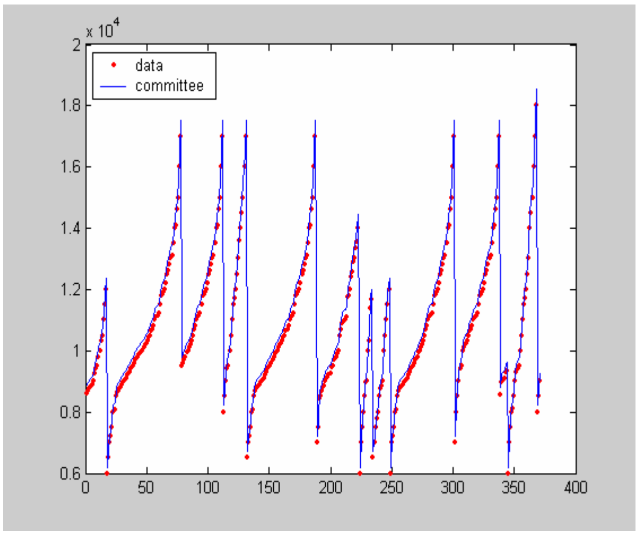

Figure 9 Standard Committee

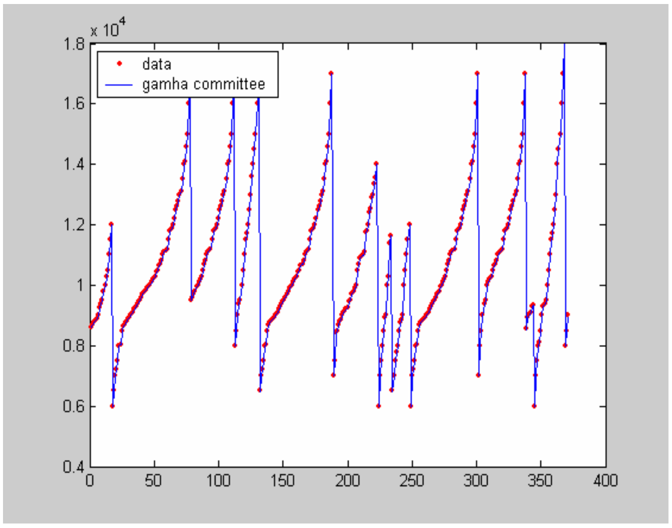

Figure 0 ‘Optimal Architecture’ Committee

8. Discussion & Conclusion

The results in Section 7 are summarized and discussed below:

Type of Network Trained	Computational Time	Resulting Error
MLP	1.015 s	8.885 %
RBF	0.235 s	8.2807 %
Optimal MLP (with PSO)	Over 1 hour	0.0045 %
Optimal RBF (with PSO)	Over 1 hour	0.0046 %
Committee Network	32.406 s	2.9258 %
Optimized Committee (GAMHA)	Several hours	0.0783 %

The computational time for the stand-alone networks are misleading, this is because good performing architectures need to be found by trial and error and this could lead to several networks being trained to eventually find good performing networks. The stand-alone RBF greatly outperforms the MLP with regard to computational time. Optimizing the stand-alone networks produce great results at the expense of a huge amount of computational time.

The generalization and portability capabilities of committee networks are easily accessible with computational time that is proportional to the amount of networks trained. Since this network has major portable and generalization advantages over the 'optimal architecture' network, it seems the obvious choice when comparing the two.

9. References:

1. K. Gurney, 'An introduction to neural networks', 11 New Fetter Lane, London, UCL press 1997.

2. L Fausett, 'Fundamentals of Neural Networks', Prentice Hall, January, 1994

3. C. M. Bishop, 'Neural Networks for Pattern Recognition', Oxford: Oxford University Press, 1995

4. I. Nabney, 'Netlab: Algorithms for Pattern Recognition', Springer-Verlag UK, November 2001.

5. M. P. Perrone, L. N. Cooper, 'When networks disagree: ensemble methods for hybrid neural networks', Artificial Neural Networks for Speech and Vision, Chapman and Hall, London, UK, 126-142

6. T. Marwala, 'Fault Identification Using Neural Networks And Vibration Data', PhD thesis, Cambridge.

7. D. A. Coley, 'An Introduction to Genetic Algorithms for Scientists and Engineers', World Scientific Publishing Company; Bk&Disk edition (November 1, 1997)

8. http://www.faqs.org/faqs/ai-faq/genetic/part1 last accessed - 9th December 2004

9. J. Kennedy, C. Russell, 'Swarm Intelligence', Morgan Kaufmann Publishers, 1st edition, March 2001.

10. L. Haiming, Yen, G. Gary, 'Dynamic population strategy assisted particle swarm optimization', IEEE International Symposium on Intelligent Control - Proceedings, 2003, p 697-702

11. F. Krogstad, 'Coding the Metropolis-Hastings Algorithm', UW Forestry, June 9, 1999.

12. J, Gasemyr, 'On a adaptive version of the Metropolis-Hastings alogorithm with independent proposal distribution.' Scand. J. Statist.,30, no. 1, 159-173.

13. A. Blanco, M.Delgado, M. C. Pegalajar, 'A real-coded genetic algorithm for training recurrent neural networks', Neural Networks, Volume 14, issue 1, January 2001, 93 – 105.

14. JSE Securities Exchange of South Africa, http://www.safex.co.za last accessed – 14 June 2004.

Appendix D

Committee Networks applied to option pricing data

Zaheer A. Dindar
School of Electrical and Information Engineering
University of the Witwatersrand
P/Bag 3, Wits, 2050, South Africa
z.dindar@ee.wits.ac.za

Abstract: *The importance of a neural networks' ability to generalize is relevant in most applications. A committee of neural networks has excellent generalization capabilities. Committee networks, consisting of multiple multilayer perceptrons and radial basis functions, are looked at in detail. These stand-alone networks are combined using the various techniques developed, and applied to data from the South African Foreign Exchange. The techniques include; different weighting functions to combine the stand-alone networks in the committee network and multiple layers of neural networks within a committee of networks. These different techniques were compared, to find that good modeling results could be found at relatively little extra computational time.*

Keywords: Multilayer Perceptron, Radial Basis Functions, Committee Networks, Committee with neural network integrator, Double layered committee network, Circular committee network.

1. Introduction

Generalization capabilities of neural networks are an important aspect to encompass within every neural network. That is, the neural networks ability to be used on data that is unseen. Committee networks are looked at it detail in this study. Various different techniques are used to combine stand-alone networks. Neural networks are also used to smooth the outputs of committee networks. These techniques were trained and tested using option pricing data.

2. Architecture

One of the most important aspects of neural network training is architecture. Architecture, as it is dealt with here, refers to the amount of hidden units and training cycles or iterations used. Choosing the correct architecture helps a great deal with regard to model accuracy.

3. Generalization and Overtraining

Generalization is a neural networks ability to be used on different data. This is a very important aspect that a trained neural network should have. A neural network with out this generalization property is said to be over fitted or over trained. Over trained neural networks model the data used to train the network very effectively, but this however, will perform very badly on a test data set. The example below illustrates generalization and overtraining [1].

The training patterns, in Figure 1, are shown by circular symbols and the two classes shown by open and filled symbols. The two lines represent the output of neural networks; where the solid line has been trained more than the dotted line. The solid line classifies all the training patterns correctly. The dotted line, however, misclassifies four of the eighteen training patterns, and it may appear at first sight that this network has performed poorly since there will be some residual error. Suppose that some previously unseen test patterns are presented, as shown by the square symbols, again filled and open squares correspond to the different classes.

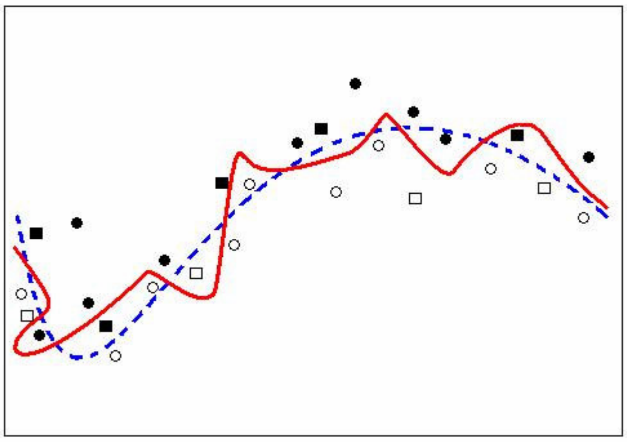

Figure 1 Generalization and overtraining

These have been classified correctly by the dotted line and the network is said to have generalized from the training data. This would seem to support the choice of using fewer training epochs since it may be that the two misclassified training patterns result from noisy data. In this case the network has implemented a good model of the data, which captures the essential characteristics of the data in pattern space.

Consider now the solid line. The training set is identical to that used in the previous example and each one has been successfully classified, resulting in a significantly smaller error. However, three of the nine test patterns have been incorrectly classified so that, even though the training data are all dealt with correctly, there may be many examples, especially those close to the decision boundary from each class, that are misclassified. The problem is that the network has too much freedom to choose its decision surface and has over fitted it to accommodate all the noise and intricacies in the data without regard to the underlying trends [1].

Generalization and over fitting is directly related to the architecture used in the neural network to model the data, since training iterations and the number of hidden units are key elements during the training of the network, and adjusting these elements could lead to great improvements in the networks modeling capability [2].

Figure 2 shows how increasing the training epochs eventually leads to greater error values on the validation set, and in turn leads to poor generalization, this illustrates the importance of finding the correct architecture for modeling accuracy [1].

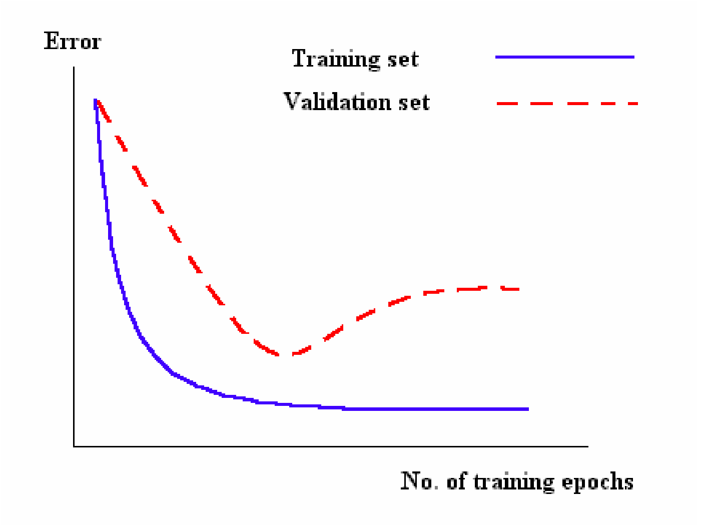

Figure 2 Cross validation behaviour [1]

4. Committee of Neural Networks

Modeling using neural networks often involves trying multiple networks with different architectures and training parameters in order to achieve acceptable model accuracy. Selection of the best network is based on the performance of the network on an independent validation or test set for instance, and to keep only the best performing network and to discard the rest. There are two disadvantages with such an approach; first, all the effort involved in training the remaining neural networks is wasted, second, the networks generalization performance is greatly reduced.

These drawbacks can be overcome by combining the networks together to form a committee (Perrone and Cooper) [4]. The importance of such an approach is that it could lead to significant improvements in the predictions on new data, while involving the training of a few additional networks. In fact performance of a committee can be better than the performance of the best stand-alone network used in isolation [3]. The committee of networks contains neural networks with different architectures and or different types of neural networks trained on the same training data set. It might even include different kinds of network models with a mixture of conventional models.

The error can never increase by using a committee of networks. Typically the error is reduced considerably by taking the average error of the combined networks [5].

$$E_{COM} \leq E_{AV} \qquad [1]$$

Another advantage of a committee of neural networks is the fact that it is more reliable than stand-alone networks.

The committee network can be represented by Figure 3:

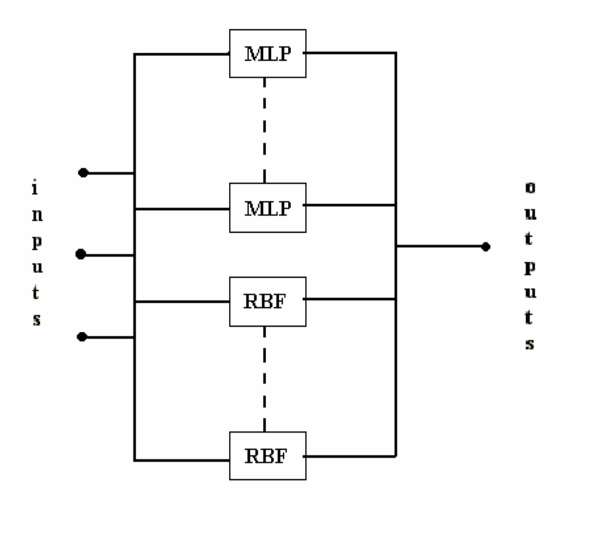

Figure 3 Standard Committee

Architectures of neural networks play a big role in capturing different aspects of data, as well as increased performance. Since the members of the committee network have different architectures, some will make better predictions than others; we expect to be able to reduce the error further if we give greater weight to the better performing committee members than to others. Thus, we consider different forms of committee networks given by weighted combinations of the member networks:

4.1 Weighted Averages

There are many possible ways, mathematically, to average the neural networks in a committee network:

1] One could divide the outputs of all the networks by the number of networks to attain an average, giving each network an equal weight, though not the most effective way to reduce the error, this method is easy to implement computationally.

2] Intuitively, one of the best ways to achieve minimum error from a committee of networks is to give greater weight to the stand-alone networks that give better results. This technique uses the following weighting function:

$$Y_{COM} = \alpha_1 y_1 + \alpha_2 y_2 + \ldots + \alpha_n y_n \qquad [2]$$

Where the α's are the weighting variables and,

$$\alpha_1 + \alpha_2 + \ldots + \alpha_n = 1 \qquad [3]$$

n Is the number of networks used in the committee. According to a theorem [5], there exists an optimal committee network that gives the least mean square error if the weight variables α are chosen to be:

$$\alpha_i = 1 \Big/ \sum_{i=1}^{n} \frac{\varepsilon(e_i^2)}{\varepsilon(e_j^2)} \qquad [4]$$

Where $\varepsilon(e)$ is the expected error. This method will be referred to as the standard weighting method (SW), in this text.

3] Another way of assigning weights to the outputs is to use priory based weighting. In this method weights are assigned according to the individual priority of the data points as well as satisfying the overall reduction in error.

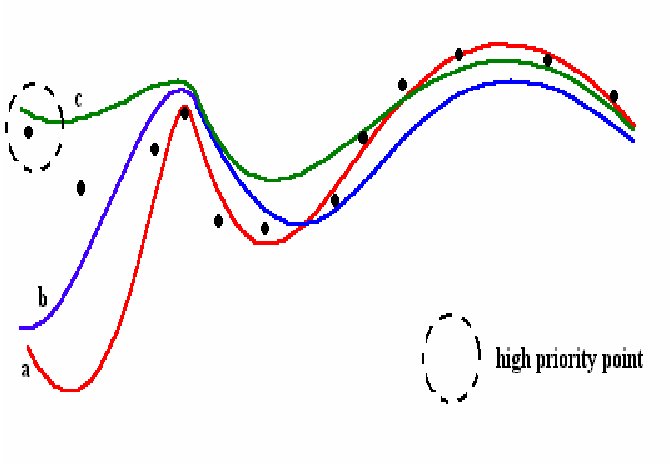

Figure 4 Priority based weighting

a, b and c are different approximations made by three different neural networks in a committee. As we can see from Figure 4 a, is the most accurate while c is the least accurate. c, however, does approximate the data better at a point that has high priority. High priority points are points that are more important to the modeler in a particular simulation. This weighting method satisfies both criteria, viz. better overall results as well as better approximations at points of high priority. It does this by using a strategy similar to the above weighting method, and by adding a priority variable:

$$Y_{COM} = (\alpha_1 + p_1)y_1 + (\alpha_2 + p_2)y_2 + \ldots + (\alpha_n + p_n)y_n \qquad [5]$$

Here

$$(\alpha_1 + p_1) + (\alpha_2 + p_2) + ... + (\alpha_n + p_n) = 1 \quad [6]$$

Where $$\sum_{i=1}^{n} p_i = k \quad [7]$$

And $$\sum_{i=1}^{n} \alpha_i = k - 1 \quad [8]$$

k is a value the modeler chooses according to the model. The value k is divided according to which neural network approximates the high priority points better, the better the approximation of the i^{th} network, the bigger the chunk of k, p_i gets according to some division rule.

There are a number of other ways to average the outputs in a committee network. Some employ non-linear techniques. Below are a few different ways to average the committee network using a neural network as a non-linear combiner.

4.2 Committee with Neural Network Integrator (CNNI)

This committee has the same architecture as the one mentioned previously, the technique uses either a MLP or a RBF to combine the networks in the committee to create a non-linear weighted average of the neural networks, Figure 5. The inputs for the integrator MLP or RBF are the outputs of the first layer of networks. This is the same output that would be combined using the methods discussed in the previous section. This integrator network serves to smooth the outputs of the first layer networks.

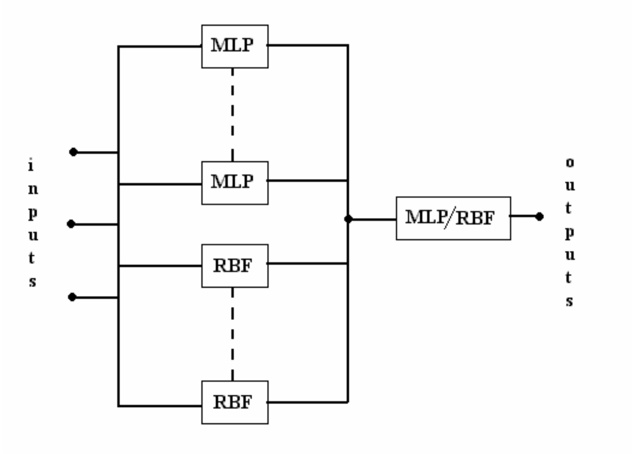

Figure 5 Committee with Neural Network integrator (CCNI)

With regard to computational time, the 'committee neural network integrator' method only requires one more network to be trained; this does not affect computational time much.

4.3 Double Layered Committee Network (DLCN)

This technique, illustrated in Figure 6, uses a smoothing neural network after each of the first layer networks. The output of the first layer is the same as the standard committee network. This output provides the input for the second layer. The second layer smoothes these inputs and the second layer outputs are now combined using the weighting methods mentioned in section 4.1.

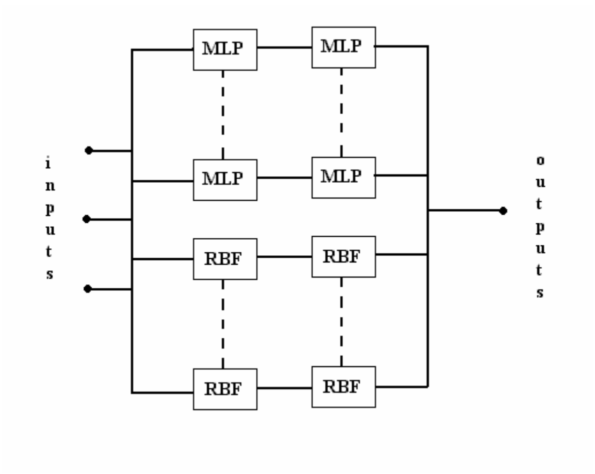

Figure 6 Double Layered Committee Network (DLCN)

With the 'double layered committee network' the computational time is doubled, since twice the amounts of networks have to be trained.

4.4 Circular Committee Network (CCN)

The 'circular committee network' (CCN) uses a jump technique, Figure 7. Firstly, the inputs to all the stand-alone networks are the same. During the next step the outputs are jumped to the neighboring neural networks to serve as inputs to the second step of training. The second training step serves to smooth the outputs of the networks. Thereafter the weighting techniques above are used to combine the outputs of this committee network.

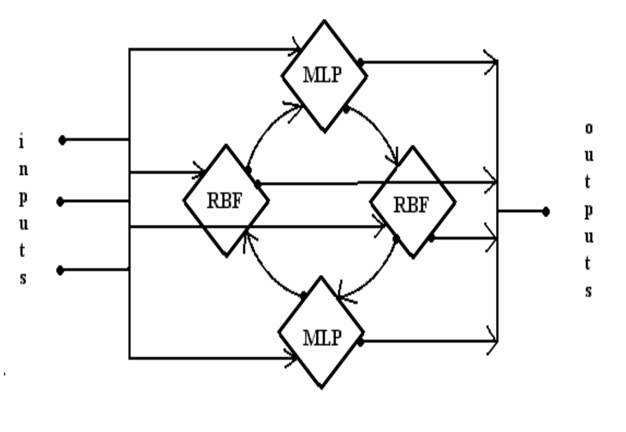

Figure 7 Circular Committee Network (CCN)

The computational time required by the (CCN) technique is much less than that required for all previous techniques as it requires only four neural networks being trained twice.

5 Results

The data set used was collected from the South African Foreign Exchange between January 2001 and December 2003 [6]. The data contained the underlying stock price, strike price, time to maturity, stock volatility, market to market price as well as the high and low prices.

The inputs were underlying asset price, volatility of the underlying asset, interest rate and time to maturity. The network output was the strike price.

Approximately two thirds of the data set was used for training and the rest for the testing.

When comparing committee networks, generalization capability, computational time as well as portability, always need to be considered. Committee networks in particular have excellent generalization capabilities, but intuitively take a much longer time to compute when compared to stand-alone networks. This is not always true, since the time taken during trial and error to find good performing stand-alone networks could take far longer than the computational time taken to train a committee network. Another very important aspect is portability. To be able to run the simulation on different data set without previous knowledge or understanding of the data is essential in a working environment. With regard to this attribute, the committee network outperforms any stand-alone network.

The standard committee network consists of 10 MLP and 10 RBF networks, each of which gets the same inputs. Each neural network in the standard committee had different architectures. The error from the individual stand-alone networks varied between 0.3 % and 13 %. The architectures were chosen in a random range of values.

Initially the committee networks were each given equal weights followed by the second weighting method mentioned above. The error was effectively reduced by using this technique to average the MLP and RBF networks. The error of the standard committee network with equal weights was just under 3 %, while the error when using the SW technique produced an error of only about 0.7 % with the computational time only increasing very slightly. The computational time both these committee networks were just under 33 seconds.

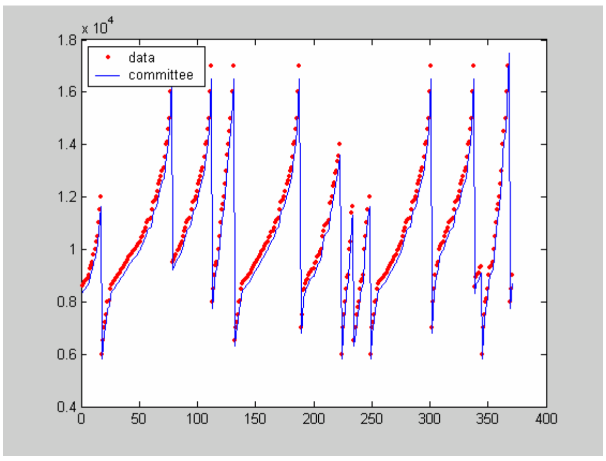

Figure 8 Standard Committee

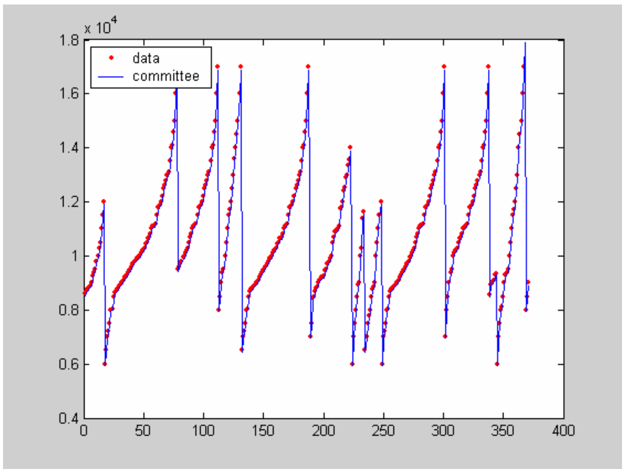

Figure 9 Standard Committee with SW weighting

5.1 Committee with Neural Network Integrator (CNNI)

The CNNI method uses a stand-alone MLP or RBF network to combine the outputs of the entire committee network. The final network in CCNI uses all the outputs of the first layer of networks as its input. When using a MLP network as the final network the results were again a great improvement on the standard committee with equal weights and it also performed better than the standard committee with the SW weighting method. The error for CNNI method was just over 0.1 seconds with a computational time of about 34 seconds.

When using a RBF network as the final network an even better improvement was achieved. The error was reduced to less than 0.001 while the computational time was about the same as using the MLP network.

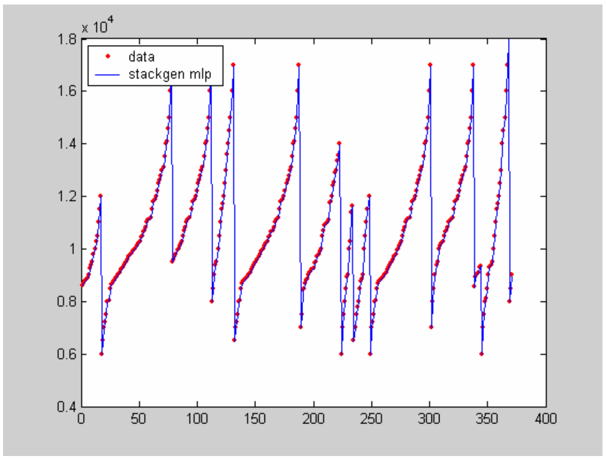

Figure 10 Committee with Neural Network integrator (CCNI)

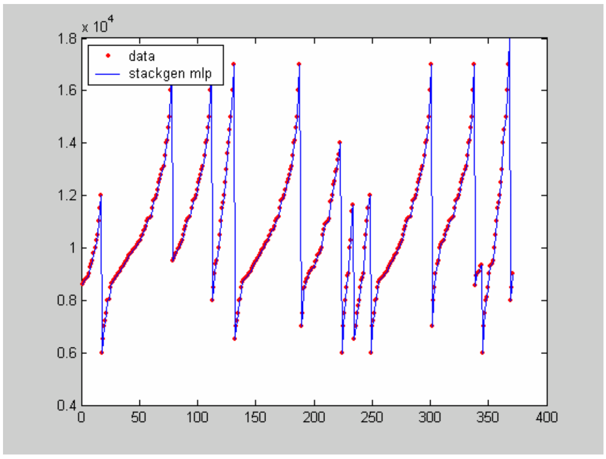

Figure 11 Double Layered Committee Network (DLCN)

It's worthwhile to note that the architecture of the final layer network also plays an important role in model accuracy. As in the first section, trial and error was used to find good performing final networks.

5.2 Double Layered Committee Network (DLCN)

Intuitively, this method should double the computational time since there is a smoothing neural network after each of the first layer networks, thus doubling the amount of networks to be trained. The architectures of the second layer networks were substantially different thus leading to extra computational and programming time outside of the simulation time. The results were an improvement on the standard committee with equal weights but failed to improve on the results for the SW weighting method. The resulting error was just over 1 % for the final layer being combined with equal weights and just under 1 % for final layer being combined by the SW weighting method. Thus, the big increase in computational time doesn't lead to better results. The computational time for the actual simulation without the extra setting up time was just under a minute.

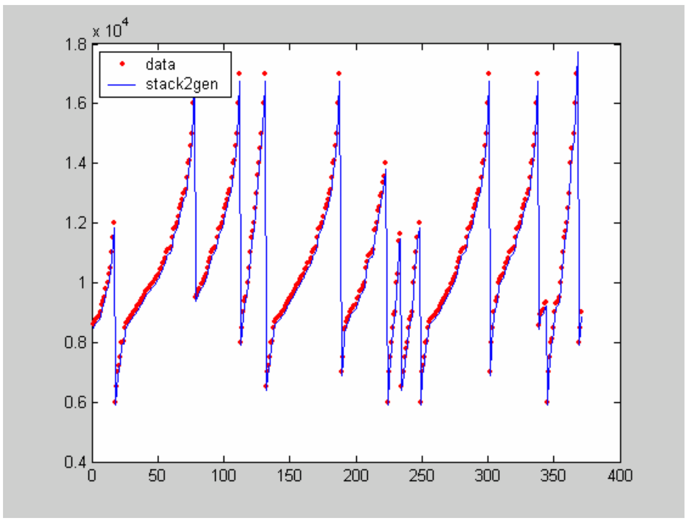

Figure 12 Circular Committee Network (CCN)

5.3 Circular Committee Network (CCN)

The CCN committee network achieved much better results than the standard networks keeping in mind the fact that it took much less time computationally. Using equal weights for combining the networks, the resulting error was less than 0.06 % with a computational time of about 14 seconds. While the error was reduced even further by using the SW weighting method, the resulting error was just over 0.0002 %, with a similar computational time.

6 Discussion & Conclusion

The results in section 5 are summarized in Table 1 and discussed below:

Type of Network Trained	Computational Time	Resulting Error
Standard Committee (equal weights)	32.406 s	2.9258 %
Standard Committee (with SW weighting)	32.535 s	0.6797 %
CNNI MLP	33.255 s	0.1411 %
CNNI RBF	32.955 s	0.00068 %
DCLN (equal weights)	57.425 s	1.3675 %
DCLN (with SW weighting)	57.465 s	0.9255 %
CCN (equal weights)	13.715 s	0.0597 %
CCN (with SW weighting)	13.735 s	0.00024 %

Table 1 Summary of Results

Numerous conclusions can be drawn from the above results:

The generalization and portability capabilities of committee networks are easily accessible with computational time that is proportional to the amount of networks trained. Using the above mentioned weighting method, the committee achieves excellent results.

The committee with neural network integrator method (CNNI) sees even more improvement over the above methods, with slightly more computational time. The one drawback of this particular method is that it could be over fitted to the validation/test data. It this respect it loses some of its generalization capabilities. The computational time is slightly misleading since a good architecture needs to be found for the final network. Here the RBF network outperforms the MLP network in the final layer.

The double layered committee network (DLCN) sees an increase in error as well as computational time. This method also included a lot of extra time to find architecture in the second layer that produced good results. Overall this network when compared to the CNNI and the standard committee doesn't perform well.

The table above shows that the best performing network is the circular committee network (CCN) with SW weighting and since this network gives the best error result as well as the fact that its computational time is well below the average. The computational time with network is again misleading because good architectures needed to be found to achieve these results.

Its is difficult to judge which method is the best, since each method has its advantages, comparing computational time as well as the generalization, portability and most importantly minimum error. Overall, since it has excellent generalization capability as well as portability, as well as the fact that it gives good results in just over average computational time, the standard committee with SW weighting seems a good choice for application to this type of highly no linear data.

7 References :

1. K. Gurney, 'An introduction to neural networks', 11 New Fetter Lane, London, UCL press 1997.

2. L Fausett, 'Fundamentals of Neural Networks', Prentice Hall, January, 1994

3. C. M. Bishop, 'Neural Networks for Pattern Recognition', Oxford: Oxford University Press, 1995

4. M. P. Perrone, L. N. Cooper, 'When networks disagree: ensemble methods for hybrid neural networks', Artificial Neural Networks for Speech and Vision, Chapman and Hall, London, UK, 126-142

5. T. Marwala, 'Fault Identification Using Neural Networks And Vibration Data', PhD thesis, Cambridge.

6. JSE Securities Exchange of South Africa, http://www.safex.co.za last accessed – 14 June 2004.