International Journal of Soft Computing 10 (6): 391-399, 2015
ISSN: 1816-9503

Volatility Prediction Model for Option Pricing: A Soft Computing Approach

[1]Vijayalaxmi, [2]Chandrashekara S. Adiga, [3]H.G. Joshi and [4]S.V. Harish
[1, 2]Department of Electrical and Electronics Engineering, [3]Department of Commerce,
[4]Department of Computer Science and Engineering,
Manipal University, Manipal, Karnataka, India

Abstract: Volatility is an important factor in the world of financial derivatives. Prediction of market volatility is very important for accurate valuation of stocks. This is required to calculate expected market return. Prediction of volatility is very much crucial in option pricing. Basically there are two main approaches to predict the volatility. They are historical approach and implied volatility approach. The main problem with the historical approach is that it pre assumes that future volatility will not change and that history will exactly repeat itself. Implied volatility claims that volatility on any day can only be estimated during trading on that day itself. In this study a sincere effort is made to predict and determine historical volatility using past data. Model works satisfactorily with minimum possible error.

Key words: Prediction, volatility model, option pricing, artificial neural networks, determine

INTRODUCTION

Financial gain is the most basic motivation to predict stock market prices is. Any system that can consistently pick winners and losers in the dynamic market would make the owner of the system very wealthy. Thus, many individuals including researchers, investment professionals and investors are continually looking for the superior system which will yield them high returns.

Since, the origin of the stock exchange, shares trading have gained importance in the financial markets. Today the relationship between the shares and its explanatory variables an appropriate forecasting model to predict the stock market, continues to be a subject of extensive research. The main reason for this is the availability of vast amounts of historical data along with enormous processing power of computers. This has enabled the use of automated systems to assist in complex decision making environments.

Today neural networks are one of the most innovative analytical tools in the financial area. Dealing with uncertainty in finance primarily involves recognition of patterns in data and using these patterns to predict future events. Artificial Neural Networks (ANNs) handle this better than other techniques because they deal well with large noisy data sets. ANNs learn from previous samples of data in much the same way that a r would teach a child to recognize shapes, colors, alphabets, etc. The ANN builds an internal representation of the data and by doing so, creates an internal model that can be used with new data that it has not seen before.

The ability of neural networks to discover non-linear relationships in input data makes them ideal for modelling non-linear dynamic systems such as the stock market. The goal of this work is to develop a Neural Network Model which can be used to predict the volatility in S&P (Standard and Poor) 500 index. This work also emphasizes the optimization of the model to produce more accurate results.

Literature review: Back in 1990, Hawley, Johnson and Raina identified various potential uses of neural networks in corporate finance, financial institutions and investments. In 1995, Boritz and Kennedy showed that the performance of neural networks is sensitive to the choice of variables selected and that the networks cannot be relied upon to evaluate and focus on the most important variables. Neural networks have been employed with success to make stock market predictions and stock selection. The networks have been used to determine optimal buy and sell timing for an equity index (Kimoto *et al*., 1990). In 1997, Gonzalez Miranda and Burgess have used the networks to predict intraday volatilities for the Spanish stock market. Volatility was predicted for the Austrian stock market and it was found that neural networks outperform Auto Regressive Conditional Heteroskedasticity (ARCH) Models. This study emphasizes on the fact that volatility predictions using neural networks are superior to Generalized ARCH (GARCH) Models.

This study focuses on the use of artificial neural networks For time series forecasting. This method

Corresponding Author: Vijayalaxmi, Department of Electrical and Electronics Engineering, Manipal University, Manipal, Karnataka, India

analyses past data and estimates the future data values in the time series which will be good approximations of the actual values. The traditional time series forecasting provides reasonable accuracy over short periods of time but the accuracy of time series forecasting diminishes sharply as the length of prediction increases. Neural networks have an advantage over traditional systems because they can extract rules without having them explicitly formalized. In a highly chaotic and only partially understood environment such as the stock market this is an important factor. Traditional systems are only good within their domain of knowledge and do not work well when there is missing or incomplete information. Neural networks handle dynamic data better and can generalize and make educated guesses. Thus, neural networks are more suited to the stock market.

The predictability of market volatility is very important for accurate valuation of stocks to calculate expected market return. Prediction of volatility is crucial in option pricing. There are two main approaches to estimate and predict the volatility. They are historical approach and the implied volatility approach. An obvious problem with the historical approach is that it assumes that future volatility will not change and that history will exactly repeat itself. Implied volatility claims that volatility on any day can only be estimated during trading on that day itself, i.e. in real time (Malliaris and Salchenberger, 1996).

The option markets depend on a large number of variables. As the economic environment changes, the option markets dynamically follow different, changing rules; sometimes a rule is valid for only a short period. The empirical financial studies in the GARCH Model imply that the training history length 'L' should not be very large. Additional research in the financial community has also demonstrated that financial data can be considered stationary only during a moderately short span of time (Liang *et al*., 2009). Three different measures are employed to quantify the implicit volatility of the Hong Kong financial market. They are namely: historical volatility, implied volatility model-based volatility. The volatility modelling and forecasting performances of evolving Fuzzy Semantic Memory (eFSM) are encouraging when subsequently benchmarked to several well-known other computational intelligence based modelling techniques (Tung and Quek, 2011).

Neural network is a black box which takes some variables as an input and gives an output. The network consists of neurons that are connected to each other. A neuron transforms the input information according to some rule and propagates the result further to other neurons. The neural network can be considered as an algorithm for an approximation of unknown functions. One of the major advantages of neural networks is that theoretically they are capable of approximating any continuous function and thus the researcher does not need to have any hypothesis about the underlying model. Rather, the model has a capacity for adaption, based on the features extracted from the data (Haoffi *et al*., 2007). Hajizadeh *et al*. (2012) stress the importance of Generalized ARCH Models in finance.

Artificial neural network is a valuable tool for time series forecasting. In the case of performing multi-periodic forecasting with artificial neural networks, two methods, namely iterative and direct can be used. In iterative method, first subsequent period information is predicted through past observations. Afterwards, the estimated value is used as an input; thereby the value of the next period is predicted. The process is carried on until the end of the forecast horizon. In the direct forecast method, successive periods can be predicted all at once. Hence, this method yields better results as only observed data is utilized in order to predict future periods (Hamzacebi *et al*., 2009).

Many neural network models have generalization ability, i.e., after training they can recognize new patterns even if they haven't been in the training set. Since, in most of the pattern recognition problems, predicting future events (unseen data) is based on previous data (training set), the application of neural networks would be very beneficial (Naeini *et al*., 2010). Forecasting foreign exchange rates using artificial neural networks and Particle Swarm Optimization (PSO) is highlighted by Georgios Sermpinis. In his study, he stresses on the risk of getting trapped into local optima and the final solution is assured to be optimal for a subset of the training set when ANN is used along with PSO instead of only PSO (Sermpinis *et al*., 2013).

Design of neural networks: In finance, volatility is a measure for variation of price of a financial instrument over time. Historic volatility is derived from time series of past market prices. An implied volatility is derived from the market price of a market traded derivative.

Neural networks model mathematical relationships between inputs and outputs. Based on the architecture of the human brain, a set of processing elements or neurons (nodes) are interconnected and organized in layers. These layers of nodes can be structured hierarchically, consisting of an input layer, an output layer and middle (hidden) layers. Each connection between neurons has a

numerical weight associated with it which models the influence of an input cell on an output cell. Positive weights indicate reinforcement; negative weights are associated with inhibition. Connection weights are 'learned' by the network through a training process as examples from a training set are presented repeatedly to the network. Each processing element has an activation level, specified by continuous or discrete values.

An artificial neural network consists of a collection of interconnected processing units or nodes. These are input nodes, output nodes and hidden nodes. Input nodes receive input signals or values from an external source, output nodes transmit the result of the neural network processing and hidden nodes make up the internal layers between input and output node layers. Connections between nodes are weighted that is they have a value that represents the strength of the connection. Different activation functions are possible, including the threshold function, piecewise linear, sigmoid and Gaussian.

Types of artificial neural networks: Feed forward artificial neural networks consist of an architecture where there are no connections that loop back to nodes that have already propagated their output signal. They have the property of being static, producing only one output pattern for each input pattern. In an implementation where all nodes are numbered in ascending sequence this means that there are no connections from higher numbered nodes to lower numbered nodes. Single-layer feed forward networks consist of input and output layers only.

Multi-layer feed forward networks contain at least one hidden layer of nodes that receives connections from the previous adjacent layer of nodes. Connections between nodes in adjacent layers are common but short cut connections that circumvent one or more hidden layers may also exist. Artificial neural networks with an architecture that includes feedback connections are called recurrent or feedback neural networks. As a result they are dynamic systems, entering more than one state for each new input pattern. Figure 1 is a simple neural network model with feed forward paths.

Training artificial neural networks: The purpose of neural network training is to produce appropriate output patterns for corresponding input patterns. It is achieved by an iterative learning process that updates the neural network weights based on the neural network response to a set of training input patterns. Learning can be categorized as supervised, reinforcement, unsupervised or hybrid. Supervised learning occurs when the correct output pattern is known and used during training. In reinforcement learning the correct output is not known but a measure of the correctness of a network output response can be computed. Unsupervised learning does not require a correct output to be available during training. Hybrid learning combines supervised and unsupervised approaches. Different learning rules form the basis of different training algorithms. The applicability of these rules is dependent on the neural network architecture and the learning category being used. Error-correction learning rules in supervised learning use the difference between the correct and actual output patterns to adjust connection weights with the aim of reducing this difference (error). The most popular supervised training algorithm for multi-layer feed forward neural networks is back propagation. Based on the

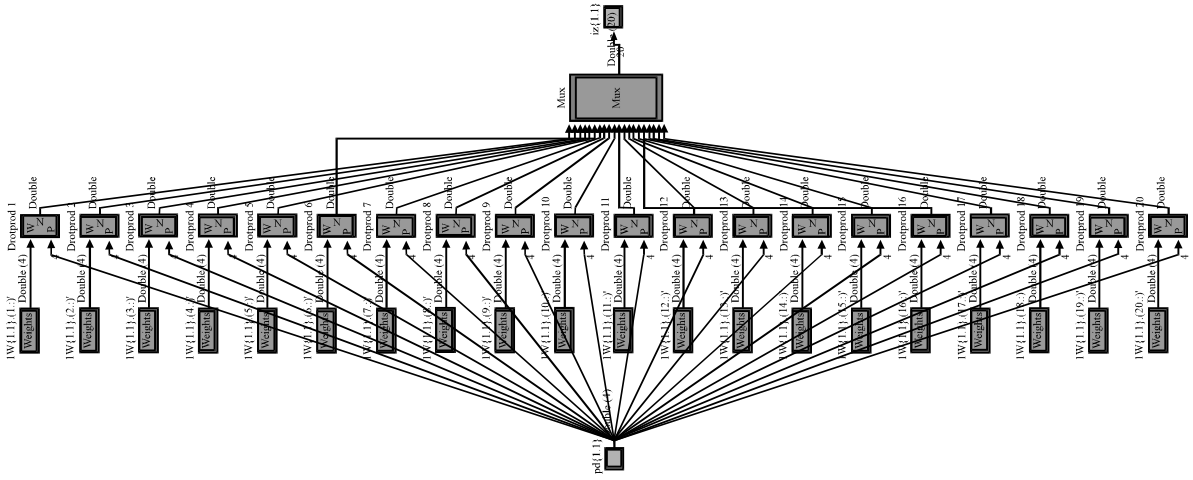

Fig. 1: Weights of the neural network

error-correction learning rule, back propagation is a gradient descent algorithm that updates connection weights by computing the benefit of the update in terms of reducing output error. Supervised learning in neural networks generally involves three data sets:

- A training set which consists of input patterns used during training
- A test set which consists of input patterns used to test the network after training is complete
- A validation set which includes input patterns used to determine when to stop training to prevent the network from becoming too specific to the training data

This is achieved by testing the network on the validation set for every n iterations of training on the training data. Training is stopped when the validation set error is as minimum as possible.

In neural networks, over-fitting is an another problem. These networks have enough flexibility in their structures to perform well on training data. They are characterized by a very high accuracy on training data and low accuracy on test data. Smaller, less complex neural networks that perform well, overcome this problem of over-fitting.

MATERIALS AND METHODS

Mathematical equations: In an artificial neural network, the output of non-input nodes can be described by the following equation:

$$y_i = f_i\left(\sum_{j=1}^{n} w_{ij}x_j - t_i\right)$$

Where:

y_i = The output of the node i
x_j = The jth input to the node
w_{ij} = The connection weight between the node and input x_j
t_i = The threshold (or bias) of the node
f_i = The node activation function

The most commonly sigmoid activation function is denoted by the logistic function:

$$1/(1+e^{-bx})$$

Where:

b = The slope parameter
x = The result of the weighted sum of the node inputs

The error-correction learning rule for back propagation uses a sum of squared error calculation generally given by the following equation:

$$E = \sum_{t=1}^{T}\sum_{t=1}^{n}(Y_i(t)-Z_i(t))^2$$

Where:

T = The number of training patterns
n = The number of output nodes
$Y_i(t)$ and $Z_i(t)$ = The actual and expected outputs of node i for pattern t

Volatility: The generalized volatility σT for time horizon T in years is expressed as:

$$\sigma_T = \sigma\sqrt{T}$$

If the daily logarithmic returns of a stock have a standard deviation of σSD and the time period of returns is P, the annualized volatility is:

$$\sigma = \frac{\sigma_{SD}}{\sqrt{P}}$$

In this work, the S&P 500 data is downloaded from Yahoo Finance. This data is used to calculate percentage change in closing prices and hence the volatility for the corresponding days. The input vectors and target vectors are divided into three sets as follows:

- The first set is used for training
- The second is used to validate that the network is generalizing and to stop training before over fitting
- The last is used as a completely independent test of network generalization (testing)

Validation vectors are used to stop training early if the network performance on the validation vectors fails to improve or remains the same for max_fail epochs in a row. Test vectors are used as a further check that the network is generalizing well but do not have any effect on training. We would like to predict future values of a time series y(t) from past values. This form of prediction is called nonlinear autoregressive with exogenous (external) input or NARX. The defining equation for the NARX Model is:

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

This NARX Model is realized using the Neural Network Time Series tool in MATLAB. The standard

NARX network is a two-layer feed forward network with a sigmoid transfer function in the hidden layer and a linear transfer function in the output layer. This network uses tapped delay lines to store previous values of the x(t) and y(t) sequences. The output of the NARX network, y(t) is fed back to the input of the network (through delays), since y(t) is a function of y(t-1), y(t-2), ..., y(t-d). However, for efficient training this feedback loop can be opened. The toolbox function (close loop) can be later used for converting NARX network from the series-parallel configuration (open loop) which is useful for training to the parallel configuration (closed loop) which is useful for multi-step-ahead prediction.

The Levenberg-Marquardt back propagation algorithm appears to be the fastest method for training moderate-sized feed forward neural networks. It also has an efficient implementation in MATLAB Software because the solution of the matrix equation is a built-in function, so its attributes become even more pronounced in a MATLAB environment.

The inputs and corresponding targets are fed to the network model for different numbers of hidden neurons, delays and number of training to find the best performing neural network. The model is then tried with different sets of data inputs of different time durations for further optimization.

Implementation

Neural network model: Following steps have been used to forecast volatility from a back propagation neural network:

- Selection of input variables
- Pre-processing the input data
- Specifying a neural network
- Training the network and forecasting

Selection of input variables: Kean suggests around 10% of the number of data observations. Thus, if there are 300 days of observations to train a network, Kean recommends thirty variables. The selection of input variables will depend on the knowledge of what affects the target variable.

Pre-processing the input data: Neural networks need properly transformed data to be able to process them and generate sound forecasts. Transformation, normalization and data smoothing are three common ways of preprocessing the data. Through transformation we can coalesce a few input variables to form a single input category. Methods include taking differences between inputs or ratios of inputs. Reducing the inputs may help the network learn better. However, it is not necessary to transform data before feeding into a network. Normalization makes the statistical distribution of each input and output data roughly uniform. The values are scaled to match the range that the input neurons use. Data normalization methods which include simple linear scaling and statistical measures of central tendency and variance, remove outliers and spread out the distribution of the data.

Data smoothing filters out noise in the data. Smoothing techniques suggested are simple and exponential moving averages and polynomial regression. Data smoothing serves two purposes. First, the network has been given useful information at a reasonable level of detail. Second, the noise entering the data is reduced.

Some of the networks have limited built-in preprocessing capabilities like scaling and randomly rearranging data to remove serial dependence. However, these networks cannot transform or smooth data. If we need to transform or smooth data, we have to do that before feeding the data into the system.

Pre-processing has two parts: arranging the data in the form that the neural network can read and scaling the data so that the maximum and minimum of each variable falls in a range of 1 and -1 (or 0 and 1 depending on the type of transfer function specified) respectively and the other values are scaled accordingly.

Specifying a neural network: Since, a typical back propagation network should have at least three-layers, we specify a three-layered network.

Appropriate specification of number of layers is an art. It needs experimentation. The countless combinations of layers and neurons that we can make and the time it takes to train a network after each specification is an arduous exercise. A single or two-layer network would be rather inadequate in capturing the complex interrelationships between market variables. If the number of layers specified is three, then it is such that it is not too few and not too many. Four layers can also be used but that would make the training time prohibitive. The resulting improvement in forecast accuracy may not be worth the extra time. However, a back propagation network should have at least three layers (Hamid and Iqbal, 2004).

Training the network and forecasting: The initial input and target data used are 1x4001 cell arrays of 1x1 matrices, representing dynamic data: 4001 time steps of closing prices and volatility. This is divided as follows:

- 70%, i.e., 2801 target time steps for training: these are presented to the network during training and the network is adjusted according to its error
- 15%, i.e., 600 target time steps for validation: these are used to measure network Fig. 1. Weights of the neural network generalization and to halt training when generalization stops improving
- 15%, i.e., 600 target time steps for testing

These provide an independent measure of network performance during and after training. There can be various combinations possible to obtain the closest results. The default number of hidden neurons is set to 10. The default number of delays is 2. The network is trained multiple times with different number of neurons and delays till the performance is optimum. If there is significant correlation in the prediction errors then it should be possible to improve the prediction, perhaps by increasing the number of delays in the tapped delay lines. If the performance on the training set is good but the test set performance is significantly worse which could indicate over fitting then reducing the number of neurons can improve the results. The network is trained using Levenberg-Marquardt back propagation. It automatically stops when generalization stops improving. Training multiple times will generate different results due to different initial conditions and sampling. A multi-layer perceptron with one hidden layer for volatility forecasting is as shown in Fig. 1. The input consists of previous values of volatility and other market data available on time t. The output of the network is a value of expected volatility. The input-error cross-correlation function illustrates how the errors are correlated with the input sequence x(t). For a perfect prediction model, all of the correlations should be zero (Fig. 2):

- If the network performance is not satisfactory, one could try any of these approaches: Increasing the number of training vectors
- Increasing the number of input values, if more relevant information is available

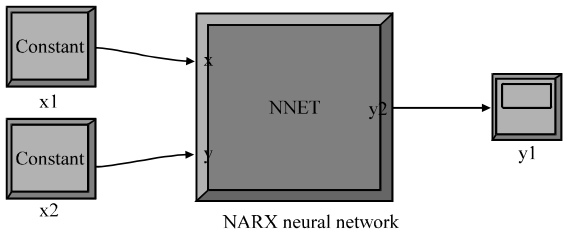

Fig. 2: Masked NARX Model

- Trying a different training algorithm
- Resetting the initial network weights and biases to new values with init and training again
- Increasing the number of hidden neurons or the number of delays

RESULTS AND DISCUSSION

This section includes various graphs which give the details pertaining to the designed neural network with different number of hidden layers and delays.

Figure 3 gives the input error cross co relation which is highest at zero lag. This is what is desired for a good neural network.

Figure 4 shows error auto co relation graph which is within the confidence limit. Here the confidence limit is 0.025%. Figure 5 is time series response which has two graphs, actual one and predicted one. This graph shows the error in the design too. It contains error curve also. Minimum error has been occurred in this network.

Figure 6 is the error histogram of the designed network. From this graph, we can infer that error is maximum at zero lag and reduces in the two sides. Figure 7 is the predicted volatility out put. At any point oftime volatility can be calculated using this greaph.

Figure 8 are the other outputs auch as gradient and mu. These outputs indicate the following. Figure 9 has

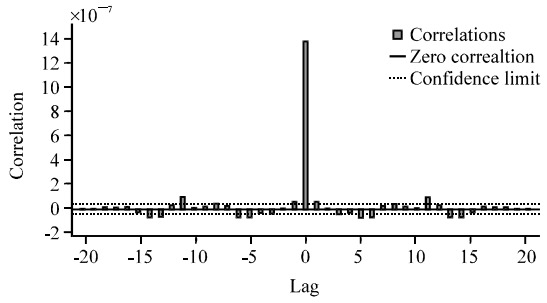

Fig. 3: Input error cross correlation

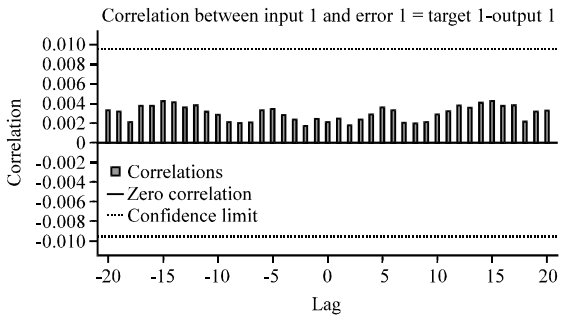

Fig. 4: Error auto correlation

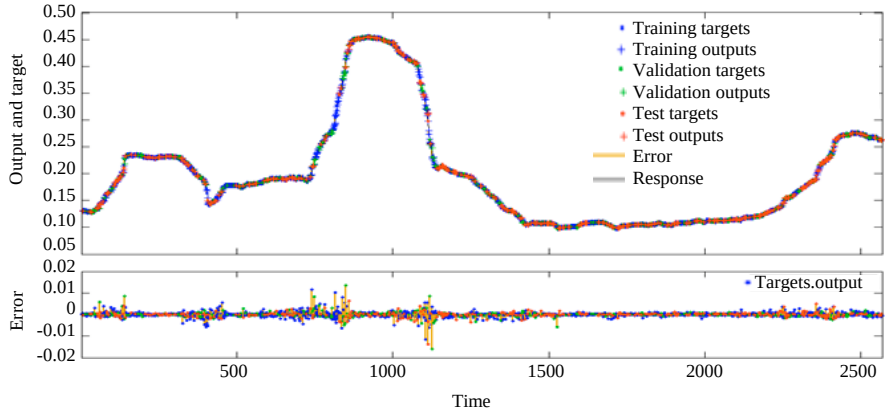

Fig. 5: Time series response (response of output element 1 for time-series 1)

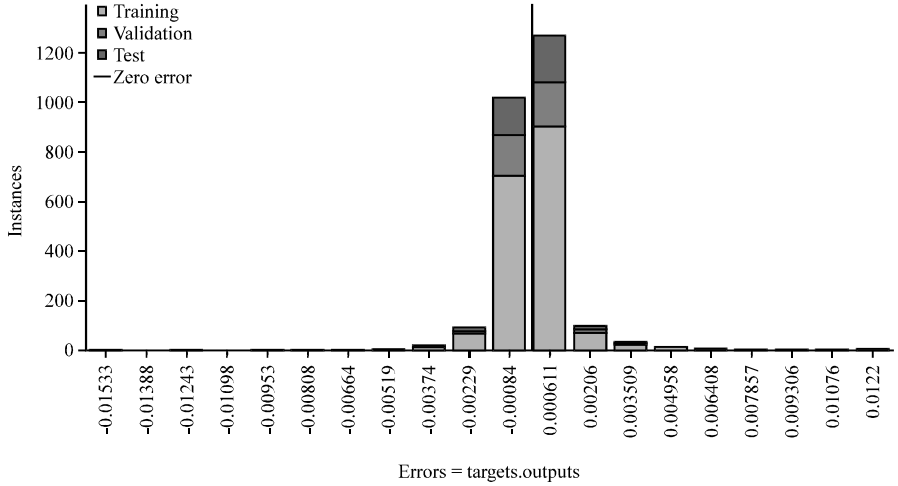

Fig. 6: Histogram (error histogram with 20 bins)

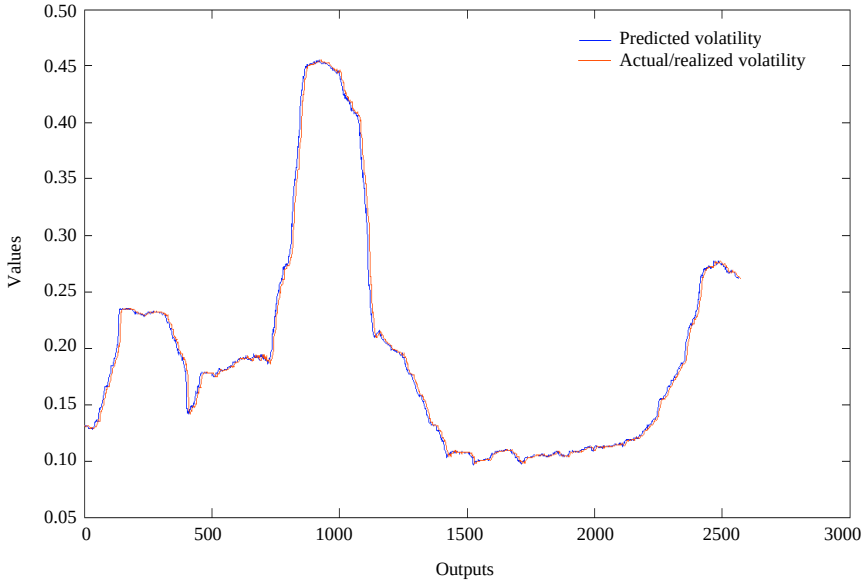

Fig. 7: Predicted output

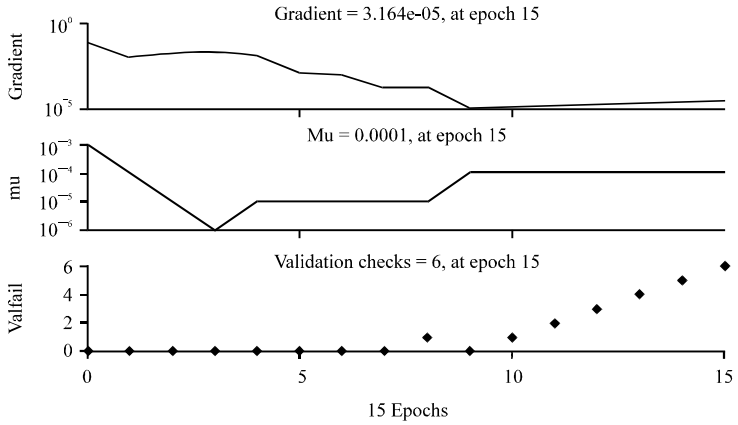

Fig. 8: Other outputs

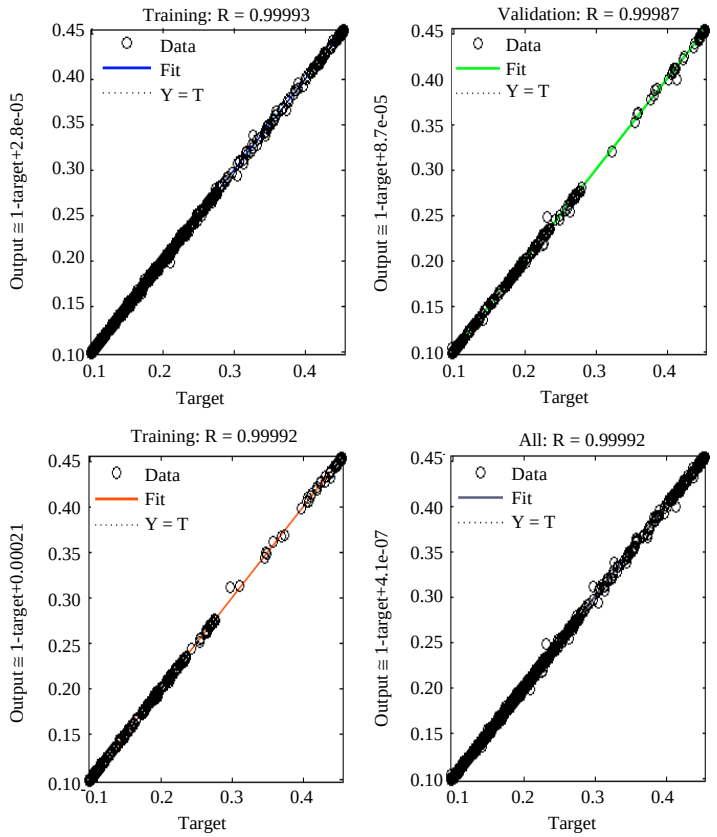

Fig. 9: Regression models

four regression models, training, validation, test and overall. Overall regression co efficient is 0.9992 which very well indicates that it is the best fit curve.

CONCLUSION

In this research, neural network has been used to develop a model for volatility. The developed model works using the historical data. To develop this model neural network time series tool box of MATLAB 2013b has been used.

The model works satisfactorily and this volatility will be used as one of the inputs to the next work. The optimization of Non-linear Autoregressive with exogenous (external) input (NARX) is critical to the performance. It involves changing the network size, the

size of the input and test data and training algorithms. Using NARX was beneficial as it allowed us to alter input taps as well as the number of neurons. Simulations have revealed that an adaptive feed forward network that varies the number of neurons alone would not have performed as well as the NARX System.

REFERENCES

Hajizadeh, E., A. Seifi, M.F. Zarandi and I.B. Turksen, 2012. A hybrid modeling approach for forecasting the volatility of S&P 500 index return. Expert Syst. Appl., 39: 431-436.

Hamid, S.A. and Z. Iqbal, 2004. Using neural networks for forecasting volatility of S&P 500 Index futures prices. J. Bus. Res., 57: 1116-1125.

Hamzacebi, C., D. Akay and F. Kutay, 2009. Comparison of direct and iterative artificial neural network forecast approaches in multi-periodic time series forecasting. Expert Syst. Appl., 36: 3839-3844.

Haoffi, Z., X. Guoping, Y. Fagting and Y. Han, 2007. A neural network model based on the multi-stage optimization approach for short-term food price forecasting in China. Expert Syst. Applic., 33: 347-356.

Kimoto, T., K. Asakawa, M. Yoda and M. Takeoka, 1990. Stock market prediction system with modular neural networks. Proceedings of the International Joint Conference on Neural Networks, June 17-21, 1990, IEEE Press, California, pp: 1-6.

Liang, X., H. Zhang, J. Xiao and Y. Chen, 2009. Improving option price forecasts with neural networks and support vector regressions. Neurocomputing, 72: 3055-3065.

Malliaris, M. and L. Salchenberger, 1996. Using neural networks to forecast the S&P 100 implied volatility. Neurocomputing, 10: 183-195.

Naeini, M.P., H. Taremian and H.B. Hashemi, 2010. Stock market value prediction using neural networks. Proceedings of the International Conference on Computer Information Systems and Industrial Management Applications, October 8-10, 2010, Krackow, Germany, pp: 132-136.

Sermpinis, G., K. Theofilatos, A. Karathanasopoulos, E.F. Georgopoulos and C. Dunis, 2013. Forecasting foreign exchange rates with adaptive neural networks using radial-basis functions and particle swarm optimization. Eur. J. Oper. Res., 225: 528-540.

Tung, W.L. and C. Quek, 2011. Financial volatility trading using a self-organising neural-fuzzy semantic network and option straddle-based approach. Expert Syst. Appl., 38: 4668-4688.