TIME SERIES FORECASTING WITH NEURAL NETWORK: A CASE STUDY OF STOCK PRICES OF INTERCONTINENTAL BANK NIGERIA

[1]Akintola K.G., [2]Alese B.K. & [2]Thompson A.F.
[1]Computer Science Department, University of Houston-Victoria, Victoria, Texas 77901 United States of America.
[2]Computer Science Department, Federal University of Technology Akure, Ondo State Nigeria
[1]Email: akintola2087@yahoo.com

ABSTRACT
In today's investment in stock markets, buyers are concerned about the economic meltdown and are afraid in investing their hard earned funds on stocks. Shares are very good financial instrument and have been found useful in times of needs as a reliable way of preparing for the future. However the sudden downward trends in stock prices movement and huge loss of capital can discourage the potential investors from buying shares. The aim of this paper is to develop a neural network solution that can be used to predict the values of shares so that buyers especially short time operators can know which share to acquire or sell at the appropriate time. The error back propagation algorithm is used to train a feed-forward neural Network in order to be able to predict stock prices for short time duration. Stock Prices of Intercontinental Bank Nigeria were used as a case study. Intercontinental Bank stock market prices were collected for the period of a year and three months and grouped into average weekly prices. Normalization of the stock prices is done which gives ranges from [0 .. 1] in order to standardize the input data. A neural network with four inputs and two hidden layers with four neurodes each and a single (neurode) output layer [4:4:4:1] is trained to learn the data. The neural net was written in C Programming Language. The result obtained is presented in this paper. This work will be very useful for buyers to make decisions on acquiring and selling their stocks at the appropriate time and will help in minimizing losses. The remaining sections are organized as follows. Section 1 introduces some background of Time series forecasting and neural network. Section 2 deals with Neural Network model development for time series forecasting. In Section 3, we discussed the result obtained using neural network to forecast Stock Prices time series data of Intercontinental Bank Nigeria. Section 4 presents the possible explanations of the observations. In Section 5, we conclude.

Keywords: *Time Series, Neural Networks, Normalization*

1. BACKGROUND

1.1 Time Series Forecasting

Time series forecasting, or *time series prediction*, takes an existing series of data to predict future values. The goal is to observe or model the existing data series to enable future unknown data values to be forecasted accurately. Examples of data series include financial data series (stocks, indices, rates, etc.), physically observed data series (sunspots, weather, etc.), and mathematical data series (Fibonacci sequence, integrals of differential equations, etc.). The phrase "time series" generically refers to any data series, whether or not the data are dependent on a certain time increment. Eric A. Plummer (2000).

A lot of projects using different techniques abound in the literature on stock prices prediction. In 2004, Karl Nygren made one-day predictions (using daily data) of some companies such as Swedish stock index and also one week predictions (using weekly data) of Ericsson B and Volvo B. The network used is Error Correction Neural Network (ECNN). For all predictions the following four time series were used as raw input: Closing price y (price of the last fulfilled trade during the day), Highest price paid during the day, yH , Lowest price paid during the day, yL Volume V (total amount of traded stocks during the day) [Karl Nygen 2004]. The prediction was found to be very good.

1.2 Basic Concepts of Artificial Neural Networks

Artificial neural network (ANN) is a widely used pattern-recognition methodology for machine learning. ANN is an emulation of biological neural network, which is composed of many interconnected neurons. However, it only utilized a very limited set of concepts from its biological counterpart.

An ANN could have one or more layer of neurons. They could be fully or partially connected. Each connection between two nodes has a weight, which encapsulate the "knowledge" of the system. By processing existing cases with inputs and expected outputs, these weights would be adjusted based on differences between actual and expected outputs. Because of the nonlinear fashion of ANN, they could be used in a lot of business applications.

The general ANN learning process has three continuous steps:
1. Compute temporary outputs
2. Compare outputs with desired targets
3. Adjust the weights and repeat the process

1.3 Neural Networks for Time Series Prediction

Time Series is a sequence of vectors (or scalars) which depend on time. In this project scalar values are used { x(t0), x(t1), · · · x(ti−1), x(ti), x(ti+1), · · · }. It's the output of some process P that we are interested in: Time series may be discrete or continuous.
Discrete Phenomena
• Intercontinental Bank stock Average closing value each day.
• currency exchange values each day.
Sometimes data have to be aggregated to get meaningful values. Like births per minute might not be as useful as births per month. [Dave Touretzky and Kornel Laskowsk 2006].

1.4 The Problem of Predicting the Future.

Extending backward from time t, we have time series {x[t], x[t −1], · · ·}. From this, we now want to estimate x at some future time ˆx[t+s] = f(x[t], x[t − 1], · · ·). where s is called the horizon of prediction. To predict just one time sample into the future, s = 1. This is a function approximation problem.
Here's how we'll solve it:
1. Assume a generative model.
2. For every point x[ti] in the past, train the generative model with what preceded ti as the Inputs and what followed ti as the Desired output.
3. Now we run the model to predict ˆx[t+s] from {x[t], · · ·}.
[Dave Touretzky and Kornel Laskowsk 2006].

1.5 Embedding

Time is constantly moving forward. Temporal data is hard to deal with... If we set up a shift register of delays, we can retain successive values of our time series. Then we can treat each past value as an additional spatial dimension in the input space to our predictor. This implicit transformation of a one-dimensional time vector into an infinite-dimensional spatial vector is called embedding.
The input space to our predictor must be finite. At each instant t, truncate the history to only the previous d samples. d is called the embedding dimension.

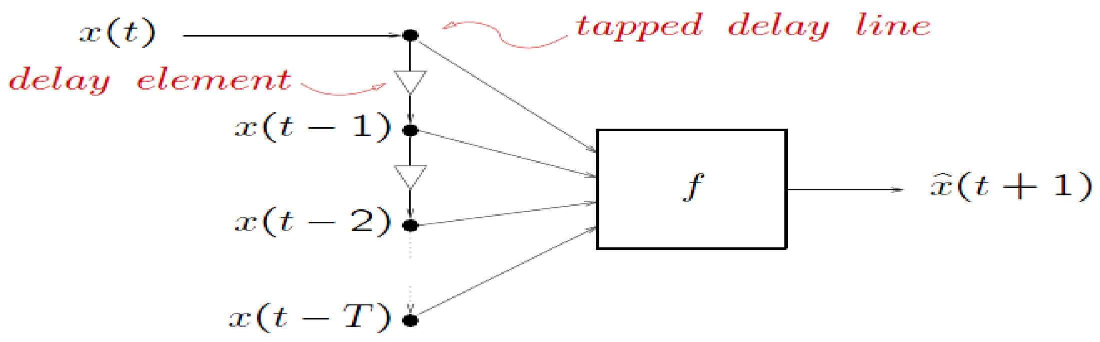


Figure 1 Source [Dave Touretzky and Kornel Laskowsk 2006]

We're interested in modeling a particular process, for the purpose of predicting future inputs.
We have three classes of possible linear process models:
• Autoregressive (AR[p]) models
• Moving Average (MA[q]) models
• Autoregressive Moving Average (ARMA[p, q]) models

2. MODEL DEVELOPMENT

We develop a neural network model to train the network. Figure 2 shows the architecture of the neural network developed. The data collected online from a Nigerian stockbroking company CashCraft Nigeria Limited on Nigeria stock Exchange Summary Trading on Quotation (N) on Intercontinental Bank PLC, Stock prices listed from January to March 2009 is used for the modeling.

Data preparation.
The data is transformed to weekly data using average since we are interested in making weekly predictions. For instance average calculation for first week January is given by. 31st December to 4th January January 1st Week – 40.60+39.06+39.06 + 40.50 = 159.22/4 = 39.805 = 39.81. This is repeated for all the data collected.
The data is first transformed into a standard form for the network to learn easily using the formula
(each data -Min(Data))/ (Max(data)-min(data). This transforms the data to between [0..1].
To forecast one future price based on 4 past stock prices, we have for the time series:
0.85335, 0.86613, 0.88117, 0.84558, 0.87641, 0.88268, 0.88243 ,0.89371, 0.92153, 0.94836, 0.96766, 0.95939, 0.96992, 0.99599, 0.99599, 0.98997, 1.00000.

Train	**Target**
0.85335, 0.86613 , 0.88117, 0.84558	0.87641
0.86613 , 0.88117, 0.84558, 0.87641	0.88268
0.88117, 0.84558, 0.87641, 0.88268	0.88243
…..... …....…... …....	…..….
0.96992 ,0.99599, 0.99599, 0.98997	1.00000.

The number of inputs is four. The network has two hidden layers each layer having four neurodes each.
The back propagation Algorithm.
This is a supervised learning algorithm in which the input vectors are supplied together with the desired output.

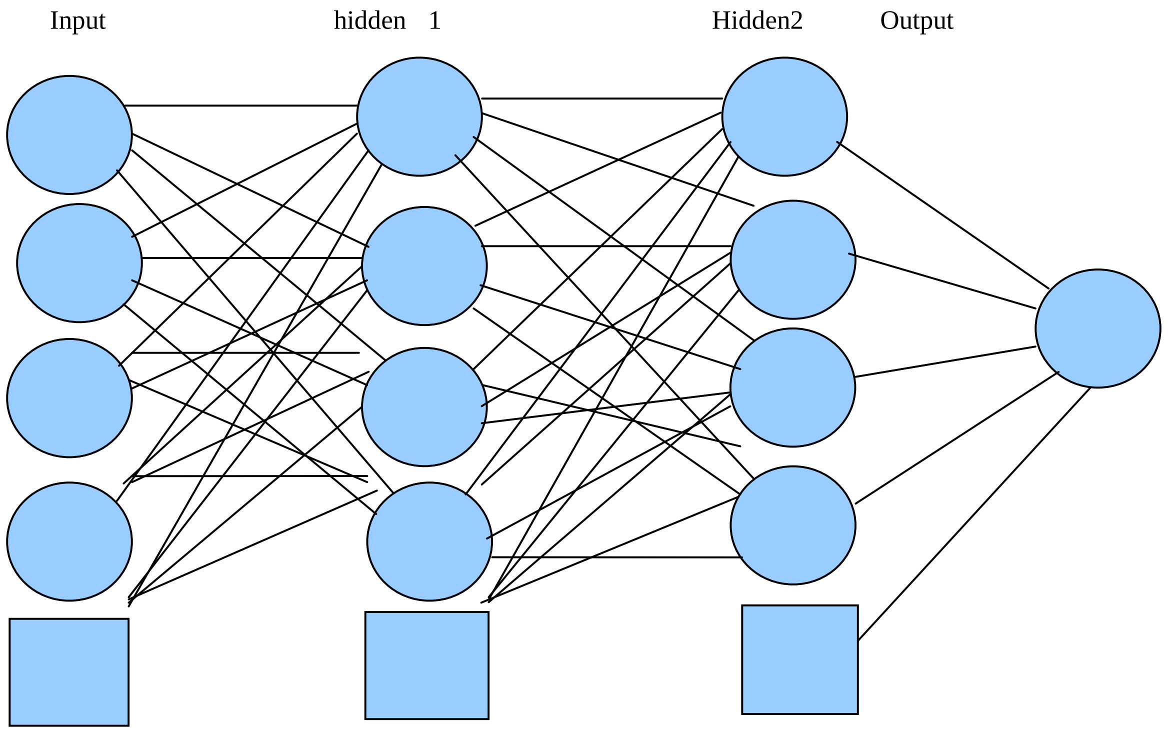


Figure 2 The Neural Network model for the stock price forecasting.

2.1 The Learning algorithm.

2.1.1 The back propagation algorithm.

The BPN learns during a training epoch, you will probably go through several epochs before the network has sufficiently learnt to handle all the data you've provided it and the end result is satisfactory. A training epoch is described below: For each input entry in the training data set:

- feed input entry data into the network (feed forward).
- Initialized weights
- check output against desired value and feed back error (back-propagate)

Where back-propagation consists of:

- calculate error gradients
- update weights (in our case the weights between output and hidden layers hidden and hidden layers and input and hidden layers were updated in that order)

2.1.2 Network Parameters

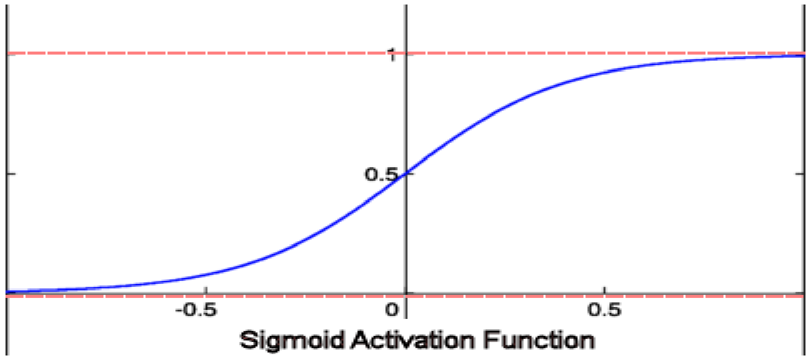
For our system model the ANN model (MLP 4: 4: 4: 1) as shown above were used. The number of epochs used is 500, with the momentum of 0.5 and learning rate of 0.3. The initial weights were randomly initialized to small random numbers less than 1 using a random number generators.

Neuro computation

At each of the Neuron of the Hidden and the output layers, The weighted sums are calculated, and passed through the activation function F(X), in our case its F(wSum – T) where wSum is the weighted sum of the inputs and T is a threshold or bias value. To take care of this biased, it is passed to the network as a node indicated by the rectangular node in the network above. The value is 1. The weighted sum (wSum) calculated at each neurode is given below.

$$wSum = \sum_{i=1}^{n} weight_i \times input_i$$

For the activation function F, there are various functions that can be used for F; the most common ones include the step function and the sigmoid function. We used the sigmoid function in this project at the hidden and the output layers. The sigmoid function and its derivative are defined as:



$$\sigma(x) = \frac{1}{1+e^{-x}}$$

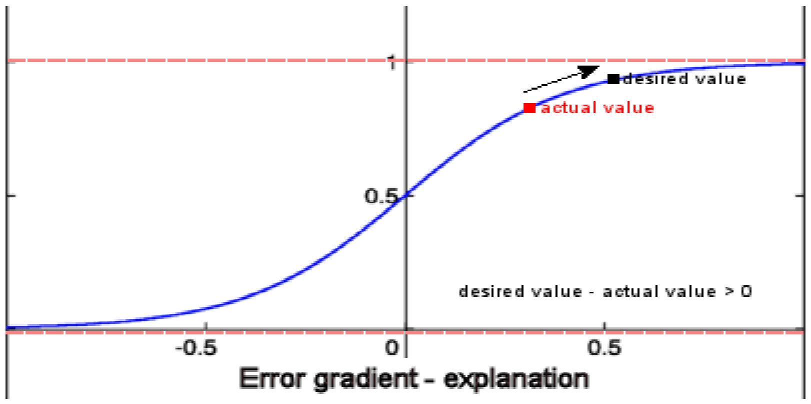
$$\frac{d\sigma(x)}{dx} = \sigma(x) \times (1 - \sigma(x))$$

2.1.3 Updating the Weights

The Neuron Error Gradients

The activation function used at the two hidden layers and the output layer is the sigmoid function. We updates the weights in our neural network to give the correct output at the output layer. This forms the basis of training the neural network. We use back-propagation for these weight updates. This just means input is fed in, the errors calculated and filtered back through the network making changes to the weights to try reduce the error. (Bobby 2008).

The weight changes are calculated by using the gradient descent method. This means we follow the steepest path on the error function to try and minimize it. That is we take the error at the output neurons (Desired value – actual value) and multiplying it by the gradient of the sigmoid function. If the difference is positive we need to move up the gradient of the activation function and if its negative we need to move down the gradient of the activation function (Bobby 2008).



The formula to calculate the basic error gradient for each output neuron k is :

$$\delta_k = y_k(1-y_k)(d_k-y_k)$$
where y_k is the value at output neuron k
and d_k is the desired value at output neuron k

(Bobby 2008).
There is a difference between the error gradients at the output and hidden layers. The hidden layer's error gradient is based on the output layer's error gradient (back propagation) so for the hidden layer the error gradient for each hidden neuron is the gradient of the activation function multiplied by the weighted sum of the errors at the output layer originating from that neuron. i.e using this formula:

$$\delta_j = y_j(1-y_j)\sum_{k=1}^{n} w_{jk}\ \delta_k$$

(Bobby 2008)
The Weight Update
The final step in the algorithm is to update the weights, this occurs as follows:

$$w_{ij} = w_{ij} + \Delta w_{ij} \ and \ w_{jk} = w_{jk} + \Delta w_{jk}$$

$$\text{where } \Delta w_{ij}(t) = \alpha . inputNeuron_i . \delta_j$$
$$and \ \Delta w_{jk}(t) = \alpha . hiddenNeuron_j . \delta_k$$

$$\alpha - learning\ rate$$
$$\delta - error\ gradient$$

The alpha value is the learning rate, this is usually a value between 0 and 1. It affects how large the weight adjustments are and so also affects the learning speed of the network. This value need to be careful selected to provide the best results, too low and it will take ages to learn, too high and the adjustments might be too large and the accuracy will suffer as the network will constantly jump over a better solution and generally get stuck at some sub-optimal accuracy.(Bobby 2008)

2.1.4 Stopping Conditions

These are some commonly used stopping conditions used for neural networks: desired accuracy , desired mean square error and elapsed epochs. In this project we used the 500 elapsed epochs to train the Network.

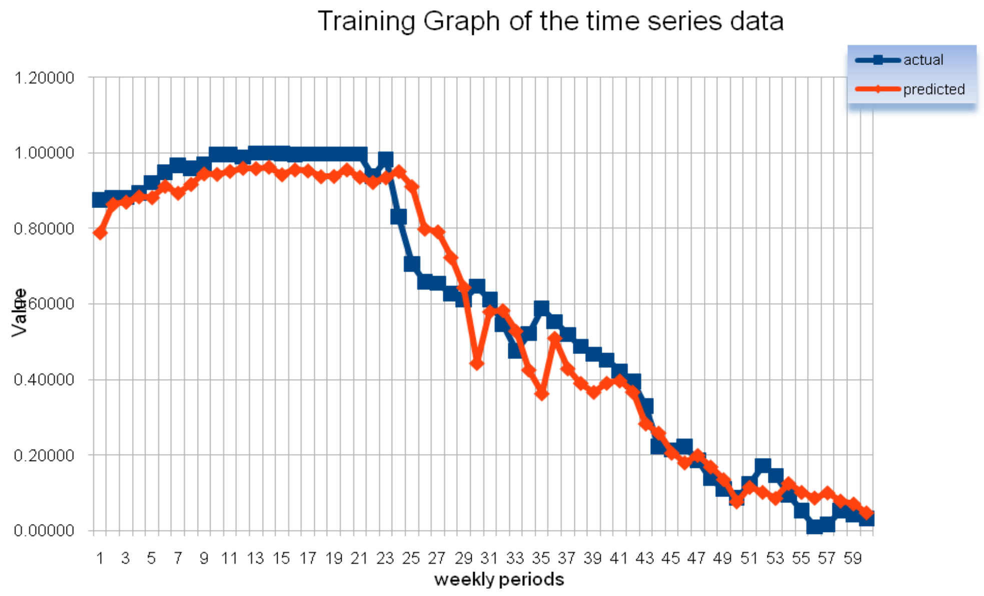


Figure 3. Training of the Neural Network

3. FORECAST RESULT

ANN models' performances can be measured by the mean relative percentage error. It measures the accuracy of prediction through representing the degree of scatter. Eq.1 was utilized to calculate the relative error for each case in the testing set. Then, the calculated values were averaged and factored by 100 to express in percentages. (actual)-(predicted)/actual *100%.

This neural network is trained to forecast one period ahead. So for the data : 0.043870 the forecast is 0.032756 Error in forecast is 25% so the accuracy is 75% and for the next data 0.026320 the forecast 0.032799 Error 24.61 the forecast accuracy is 75.49%. The Accuracy of the forecast can be improved further by training the network with more data.

4. DISCUSSIONS

We made several attempts before we got the architecture that was suitable for this network. We tried several combinations of layers, momentum, learning rate, epochs and so on. Due to the nature of time series forecast, we need past data to predict future, this network can be used to predict one period ahead. The predictions could be done recursively to get more predictions ahead.

5. CONCLUSION

The application of neural network in forecasting stock prices is studied in this paper. From Our predictions, we realized we might need more data to train the network to be able to give a better prediction. Also time series data of stock can be erratic at times and give a sudden increase or decrease in price. So at times predictions could be further from the actual prices experienced in the market. Future work can be done on this study by using different network systems such as recurrent network system and by considering more than one time series variables with the hope of investigating the performance of Neural network in time series forecasting. The result shows that Neural network is a good method for predicting time series data.

REFERENCES

[1]. Bobby (2008) **Basic Neural Network Tutorial –Theory** http://takinginitiative.net/2008/04/03/basic-neural-network-tutorial-theory/
[2]. Dave Touretzky and Kornel Laskowsk (2006) Lecture Notes Neural Networks for Time Series Prediction 15-486/782: Artificial Neural Networks Fall 2006
[3]. Eric A. Plummer (July, 2000) A thesis submitted to the Department of Computer Science, and The Graduate School of The University of Wyoming in partial fulfillment of the requirements for the degree of Master of Science Computer Science.
[4]. He Yan (2009) Time Series Forecasting with Neural network CS545 Assignment 8.
[5]. Karl Nygren (2004) Stock Prediction – A Neural Network Approach Master Thesis Royal Institute of Technology, KTH.
[6]. R. Samsudin, A. Shabri and P. Saad A (2010) Comparison of Time Series Forecasting Using Support Vector
[7]. Machine and Artificial Neural Network Model Journal of Applied Sciences Asian Network for scientific Information.