Philippine Stock Prediction using Artificial NeuralNetworks

Pre-final Paper for the Project

Antonio, Daniel
Electronics and Communications Engineering Department
De La Salle University - Manila
Manila, Philippines

Tan, Sidney Sheldon Urlanda
Electronics and Communications Engineering Department
De La Salle University - Manila
Manila, Philippines
sidney_tan@dlsu.edu.ph

Eco, Lance Randall Lim
Electronics and Communications Engineering Department
De La Salle University – Manila
Manila, Philippines
lance_eco@dlsu.edu.ph

Zarzoso, Aaron John Pascua
Electronics and Communications Engineering Department
De La Salle University - Manila
Manila, Philippines
aaron_zarzoso@dlsu.edu.ph

***Abstract*—This paper represents the students' preliminary report on the project for the course LBYCP29. It contains the procedures done for the implementation of the stock price prediction in the Philippines through the use of an artificial neural network. The group outlined the design of the Neural Network model with its features parameters. A number of activation functions are implemented along with options for cross validation sets. The scaled conjugate gradient algorithm as well as the backpropagation algorithm were used to train the network. Testing of the Artificial Neural Network was done by using PLDT stock index dataset. There is also the use of preprocessing data and the definition of the error function which is to be minimized.**

Keywords—matlab; artificial neural networks; multi-layer neural network; prediction methods; stock markets; logistic regression; training; testing; optimization; accuracy;

I. Introduction

Neural network is an information processing paradigm that is influenced by the path organic nervous systems process data such as the mind of humans. It generally shown as systems of interconnected neurons which can exchange messages to each other. In addition, there connections have individual numeric weights that can be tuned based on experience which makes the neural nets adaptive to any inputs and have the capacity to learn.

Stock market remains one of the primary means of investment for investors for at least several decades. An attempt is made to predict the closing prices of stock market using neural networks. Many people tried on stock market to extract some useful patterns in predicting their movements. The Artificial Neural Networks have the capacity to determine the nonlinear relationship in the input data set without a previous assumption of knowledge of relation between the input and the output which is compatible in other models in predicting stock prices.

II. Background of the Study

A. Artificial Neural Networks

In this section the group describes the structure of Artificial Neurons and how they are connected to construct Artificial Neural Network.

a. Artificial Neurons

Artificial Neurons are very much similar to the neurons in our body which performs the task of the transmission of a signal from neuron to neuron through synapses. This process is considered to be a complex chemical process such that the substances that are transmitted are released from the sending side of the junction. This process is done in order to either raise or lower the electric potential inside the body of the receiving cell. Once this graded potential reaches its limit, then the neuron fires.

This actual behavior of the neurons in our body is what artificial neuron models attempt to reproduce. The neuron model as seen in Figure 1 is the one of the popularly used models of artificial neural networks however, there are only some few modifications on it.

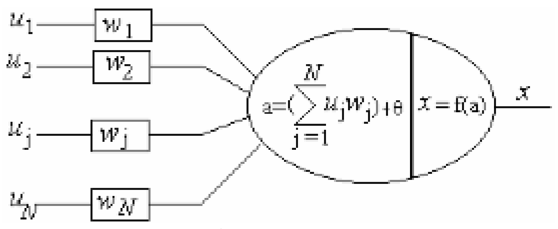

Fig. 1. Artificial Neuron Structure

As seen in the figure, N denotes the number of inputs so in this case it has N inputs and are represented as u_1, u_2, ...u_N. The lines which connect the inputs to the neuron each has weights which are equivalent to synaptic connections in actual neurons found in our body. The weights are denoted respectively as $w_1, w_2,.., w_N.$ Meanwhile, θ represents the threshold of the artificial neuron and the activation that is the counter part of the graded potential of actual neurons is represented by the formula:

$$a = \sum_{j=1}^{N} w_j u_j + \theta$$

The values of the inputs and weights are real. However, the values of the weights can be either positive or negative and this determines what type of connection the network has. For instance, if a weight has a negative value, then the connection is inhibitory which means that the artificial neuron is less likely to commence activation or in the case of biological neurons, it is less likely to generate action potential. On the other hand, if a weight has a positive value, then there is what is called as an excitatory connection in which case, the artificial neuron is more likely to commence activation or more likely to generate action potential in the case of actual neurons, as contrasted from having a negative weight. The threshold, denoted by θ is sometimes added or combined to the summation part through the assumption of an imaginary input such as $u_0 = +1$ and a connection weight $w_0 = \theta$. And therefore, in this case, the activation formula will become much simpler:

$$a = \sum_{j=0}^{N} w_j u_j$$

In comparison to actual neurons, the artificial neurons' output value is a function or a component of its activation just as biological neurons' firing is a result of its reached graded potential and has the formula:

$$x = f(a)$$

There are several functions used which includes Binary threshold, linear threshold, sigmoid, hyperbolic tan and Gaussian.

b. Artificial Neural Networks

Artificial neural networks are actually composed several connections of different neurons and thus they are called as a network. The main idea behind artificial neural networks is that the problem of having a single artificial neuron is its incapacity to implement some Boolean functions and so in order to overcome this problem, the output of some of these neurons are connected and fed as inputs to another neuron and thus constructing a neural network. In cases where many artificial neurons are connected to one another, it may be difficult which neurons distinguishes from each other and indices are assigned to the neurons in order to differentiate them. Meanwhile, to express the activation i^{th} iteration, the formula is modified as:

$$a_i = \sum_{j=1}^{N} w_{ji} x_j + \theta_i$$

where x_j could either be an output of another neuron which has been said earlier to be taken as an input to another neuron to construct an artificial neural network or it can also be an input which comes externally.

Artificial networks could be modeled through several architectures. For instance, feedforward neural networks are arranged in the form of layers such as the fundamental layers like the input layer, hidden layer and the output layer. The input layer basically is the initial layer of an artificial neural network and their inputs usually are external inputs since as what have been said they are the first layer. The hidden layer is the layer between the input and the output layer and they are fed with inputs coming from the output of the input layer. The output of the hidden layer is then taken as inputs to the last layer which is the output layer. It is called as the feedforward layer since neurons in a particular layer are fed with inputs coming from previous layers. Likewise, the neurons in that current layer then feed their outputs to the next layer which take them as input. In this case, neurons that are at the same layer or at the previous layers cannot be connected. Meanwhile, networks which only have a layer of input nodes and one layer of neurons comprising the output layer, then it is called as a single layer network. If a network has one or more hidden layers, then it is called as a multilayer network. Lastly, if connection to the same layer and previous layers are permitted then the network is known to be a recurrent network.

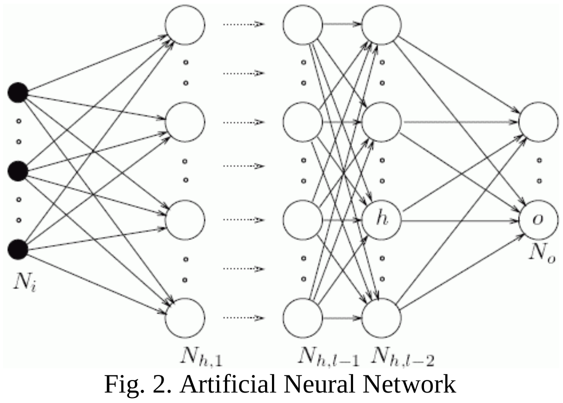

Fig. 2. Artificial Neural Network

B. *Learning Algorithm*

a. Back Propagation

Gradient descent algorithms, which expect to identity the minima/maxima of a given function in the process of iteratively moving toward the negative of the slope of the function to be minimized/maximized, is where the backpropagation algorithm categorizes. The primary target is to minimize the error function. The average error function to be minimized is given by:

$$\xi_{av} = \frac{1}{N}\sum_{n=1}^{N}\xi(n)$$

The weights are updated on a pattern by pattern basis until a complete epoch has been managed during this process. The modifications to the weights are made as per the individual errors processed for every example exhibited to the network. The arithmetic average of these individual weights over the whole training set is an approximation of the true change that would come about because of the adjustment of the weights due to the error function.

A gradient descent technique is utilized to minimize the error. The chain rule for differentiation turn out to be:

$$\frac{\partial\xi(n)}{\partial w_{ji}(n)} = \frac{\partial\xi(n)}{\partial e_j(n)}\frac{\partial e_j(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial v_j(n)}\frac{\partial v_j(n)}{\partial w_{ji}(n)}$$

After further simplifying:

$$\frac{\partial\xi(n)}{\partial w_{ji}(n)} = -e_j(n)\varphi'_j(v_j(n))y_i(n)$$

The final rule for updating weights becomes

$$\Delta w_{ji}(n) = \eta\delta_j(n)y_i(n)$$

where,

$$\begin{aligned}\delta_j(n) &= -\frac{\partial\xi(n)}{\partial v_j(n)} \\ &= -\frac{\partial\xi(n)}{\partial e_j(n)}\frac{\partial e_j(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial v_j(n)} \\ &= e_j(n)\varphi'_j(v_j(n))\end{aligned}$$

for the last layer and

$$\delta_j(n) = \varphi'_j(v_j(n))\sum_k \delta_k(n)\,w_{kj}(n)$$

for the intermediate hidden layers.

The batch learning plan is utilized for updating the weight– all the training samples are taken as inputs into the network and the adjustment in all weights is processed from every input samples. At that point toward the end where the weights are updated as indicated by the summation of all updates. One is equivalent to one iteration of inputting all the training samples.

For practicality, ANNs executing the backpropagation algorithm don't have an excess of layers, because of the time for training the networks increases in an exponential manner. Additionally, there are modifications to the backpropagation algorithm which permit a better learning.

Consequently, the algorithm above can be utilized to train an Artificial Neural Network (ANN) given the training information/input and the learning rate. The above network can usually have a subjective number of hidden layers and a discretionary number of hidden neurons in every layer. The number of input layer neurons is dependent on the quantity of input components in every example, and the quantity of output layer neurons is dependent on the quantity of output elements in the target values.

There are a couple of cons attributed to backpropagation learning also:

• The convergence acquired from backpropagation learning is slow.

• The convergence in backpropagation learning is not ensured.

• The outcome might merge to any local minimum on the error surface because stochastic gradient descent exists on a surface which is not flat.

• Backpropagation learning requires data scaling or normalization.

• Backpropagation requires the activation function utilized by the neurons to be differentiable.

b. Scaled Conjugate Gradient Algorithm

Scaled Conjugate Gradient Algorithm is used for solving numerical problems as well as optimization problems with systems of equations unitizing matrices which is positive-definite and symmetric. In neural networks, scaled conjugate gradient algorithm is used for training the data. It also allows gradient calculations for systems with large problems and is far more memory efficient than other algorithms. However, the down side of scaled conjugate gradient algorithm is that it requires the weight, net input, and transfer function to have derivative functions to train which is why the backpropagation is used to compute for the derivatives of the neural network's performance with respect to the weight and bias.

C. Ann Model Features

a. Activation Function

The user likewise has an alternative of three activation functions for the neurons:

- Unipolar sigmoid:

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

- Bipolar sigmoid:

$$f(x) = \frac{2}{1+e^{-\lambda x}} - 1$$

- Tan hyperbolic:

$$f(x) = \frac{e^{\lambda x} - e^{-\lambda x}}{e^{\lambda x} + e^{-\lambda x}}$$

- Radial basis function

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

The activation function utilized is common to every one of the neurons in the ANN.

b. Hidden Layers and Nodes

Neural networks has three types of layers namely: input layer, hidden layer, and output layer.

Every neural network contains exactly one input layer. The input layer is where the training data is and the number of neurons in this layer is equal to the number of features (columns) in the data. Often times, a node is added for the bias factor.

Similar to the input layer, every neural network has exactly one output layer. The number of neurons in the output layer can be determined by the chosen model configuration. Machine mode returns class label (used in statistics. example: premium or basic service) while regression mode returns value (example: price). If a neural network is a regressor then the output contains single node similarly a classifier neural network also contains single node except when softmax function (gradient-log normalizer) is applied for which the output layer now contains one node per class label.

Hidden layers contains the individual possibilities (hidden node/neuron) that a combination of inputs can result to. Hidden layers lies between the input and output layers. Unlike the input and output where both are only 1 layers, hidden layers could be more. Each addition of hidden layers improves the neural network's performance by a little; which is why one hidden layer often suffices for few data inputs and the amount of nodes in a hidden layer is often more than the number of nodes in the input and the output.

Hidden neurons are neither in the input or in the output layer. As explained in hidden layers, using additional layers of hidden neurons gives more processing power and flexibility but with the side effect of introducing more complexity within the training algorithm. Too many hidden neurons is similar to a system of equations with more equations than variables. The system is incapable of generalization due to being over specified. While too few hidden neurons prevent the system from fitting the input data thus reducing the robustness of the system.

Designing neural networks involves assigning values for the number of layer of each type as well as the number of nodes on each layer.

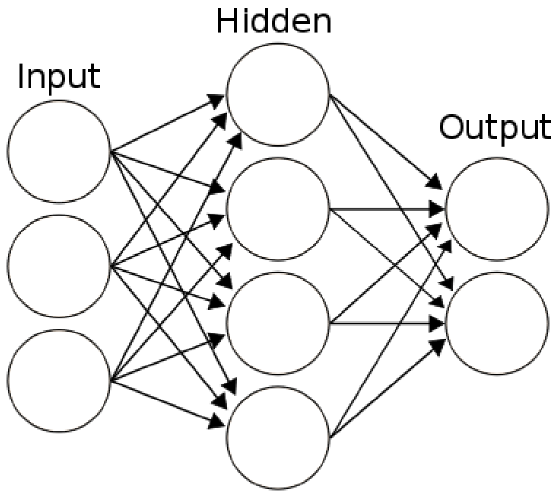

Fig. 3. The input, hidden, and output layer in an ANN.

The artificial neural system that the group trained for the prediction of the stock input has a discretionary number of hidden layers and discretionary number of hidden nodes in every layer, both of which the user chooses amid run-time.

c. Validation Error

Validation is used in measuring network generalization and to halt or stop the training when the generalization stops improving. It is usually set as 15% of the original data. However, it could be set as 20% or higher to test the data set accurately. This specific percentage is used to validate that the network is generalizing and to stop training before it overfits.

The training of the data set will continue until the validation error failed to decrease for a certain number of iterations and the validation will stop consecutively.

d. Error Calculation

The rms error between the target values and the actual outputs is the calculation of the error for convergence. The same error is used to report the performance of the algorithm on the test set.

e. Cross-correlation

This is an input function to acquire additional verification of the network's performance. It illustrates how the errors are correlated with the input sequence y(t). Moreover, all of the correlations should have a zero value to attain the perfect prediction model. It shows that the input is correlated with the error which will lead to the further improvement of the prediction. With this, all correlations concerning training set will fall to the boundary near zero value. The cross-validation set is autonomous of the training set and aides in a broader measure of the error and gives better results.

D. Data Pre-Processing

When training a network containing tapped delay lines, it is necessary to fill the delays with initial values of the inputs and outputs of the network. There is a toolbox command that facilitates this process - preparets. This function has three input arguments: the network, the input sequence and the target sequence. The function returns the initial conditions that are needed to fill the tapped delay lines in the network, and modified input and target sequences, where the initial conditions have been removed. The data for training and simulation are prepared after creating the external input of the nonlinear autoregressive network. This function prepares time series data for a certain network as well as shifting the time by the minimum number to be able to fill the state input and state layer. It allows the user to use the function to keep or hold the original time series without any alterations which can easily customize for the network with different amount of delay in open loop as well as in closed loop feedback modes. The function provided input and layer states in the early stage to be able to test the network as well as computing the network outputs which it can calculate several parameters such as errors, network outputs and overall performance.
The delay must be filled with initial values of the inputs and outputs of the network but the training must first contain tapped delay lines. There is a certain toolbox command that facilitates this particular process. This function has three input: network, input sequence and target sequence. The function returns the initial criteria that is necessary to congest the tapped delay lines in the network as well as modify the input and target sequences where the initial conditions have been removed.

III. Implementation

A. Data

A 4x503 matrix; 4 columns comprised of open, high, low, close with 503 samples.

B. ntstool

The group made use of the nnstart tool in matlab to produce the source code. The nnstart tool opens a window with launch buttons for neural network fitting, pattern recognition, clustering and time series tools. Found in the nnstart tool is the ntstool that opens the neural network time series tool and leads the user through solving a fitting problem using a two-layer feed-forward network.

C. NARX

The group used Nonlinear Autoregressive (NARX) as a solution for the time-series problem. NARX predicts series y(t) given d past values of y(t). a recurrent dynamic network, with feedback connections enclosing several layers of the network. The NARX model is based on the linear ARX model, which is commonly used in time-series modeling.

The defining equation for the NARX model is

$$y(t)=f(y(t-1),y(t-2),\ldots,y(t-ny),u(t-1),u(t-2),\ldots,u(t-nu))$$

Where the next value of the dependent output signal y(t) is regressed on previous values of the output signal and previous values of an independent (exogenous) input signal.

D. Target Time-series

Targets 'inputs' is a 4x503 matrix, representing dynamic data: 503 timesteps of 4 elements.

E. Training, Validation and Test Data

The group divided the data into three components; training, validation and testing. Training data is comprised of 60% (1308 target timesteps) of the input data, these are presented to the network during training, and the network is adjusted according to its error. Validation data is comprised of 15% (302 target timesteps) of the input data, these are used to measure network generalization, and to halt training when generalization stops improving. Testing data is comprised of 20% (402 target timesteps) of the input data, these have no effect on training and so provide an independent measure of network performance during and after training.

F. Network Architecture

The group would like to opt for a higher number of hidden layers to further refine our results, but due to memory restrictions our group used 2 but still managed to produce acceptable results, using more hidden layers requires more computation but will further improve the results of a neural network. The group used 2 hidden layers and 2 for the number of delays for the neural network. The network will be created and trained in open loop form. Training with open loop (single-step) prediction is more efficient than with closed loop (multi-step) prediction because it allows the group to supply the network with correct feedback inputs even as the group trains it to produce the correct feedback outputs.
Problem definition: y(t) = f(y(t-1), … y(t-d))

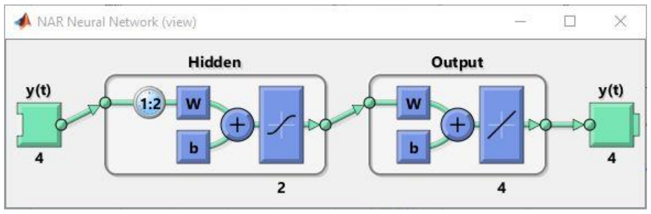

Fig. 4. Network View

G. Training and Testing the Network

The group used Scaled Conjugate Gradient as the training algorithm because it takes less memory. Training automatically stops when generalization stops improving, as indicated by an increase in the mean square error of the validation samples. Mean Squared Error is the average squared difference between outputs and targets. Lower values are better. Zero means no error.

IV. Data and results

A. User input data

• Inputs – the data of the specific company's stock index which is composed of the following: open, low, high, and close value.
• Output – the predicted close value.
• Percentage of training data (65%)
• Percentage of validation data (15%)
• Percentage of testing data (20%)
• Number of hidden layers, nH (2)
• Maximum number of epochs (100)
• Activation function (Bipolar Sigmoid)

B. Results

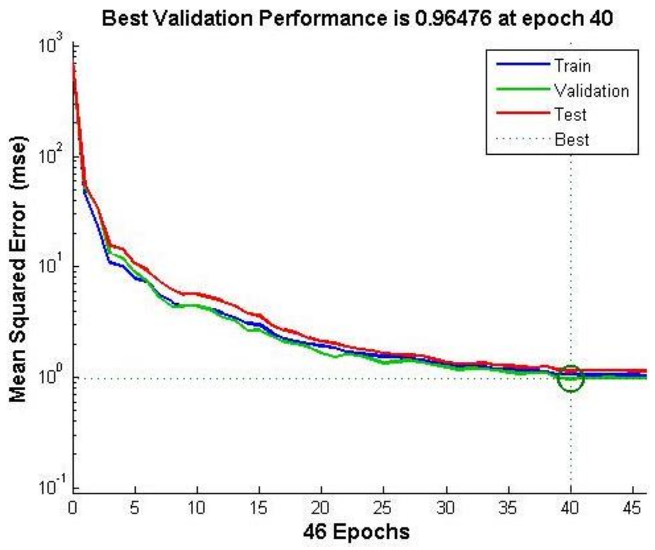

Fig. 5. MSE plot

The merging of the MSE error was quite smooth as the quantity of epochs increased. This figure shows that training, validation and testing errors all decreased until iteration 62. It does not appear that any overfitting has occurred, because neither testing nor validation error increased before iteration 62.

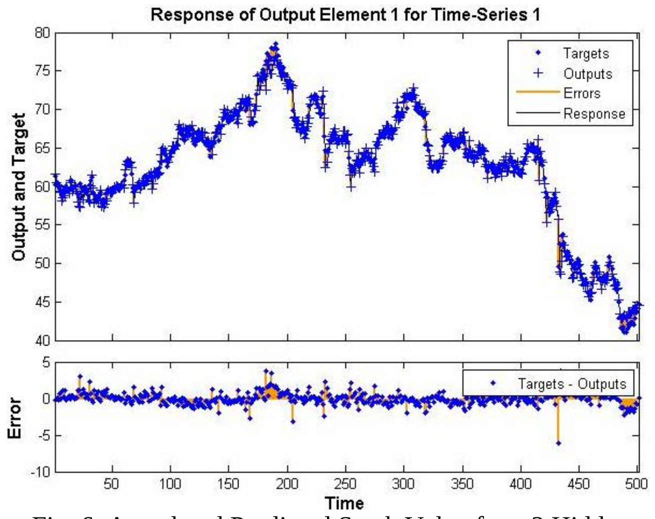

Fig. 6. Actual and Predicted Stock Value for a 2 Hidden Layer Network

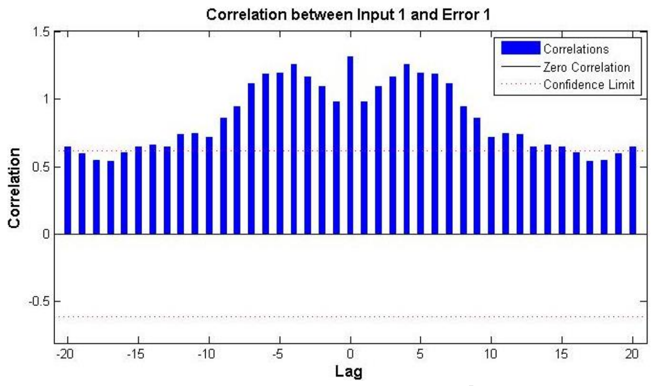

Fig. 7. Input-Error Cross-Correlation

This input-error cross-correlation function illustrates how the errors are correlated with the input sequence y(t). For a perfect prediction model, all of the correlations should be zero. If the input is correlated with the error, then it should be possible to improve the prediction, perhaps by increasing the number of delays in the tapped delay lines. In this case, most of the correlations fall within the confidence bounds (red dashed lines) around zero.

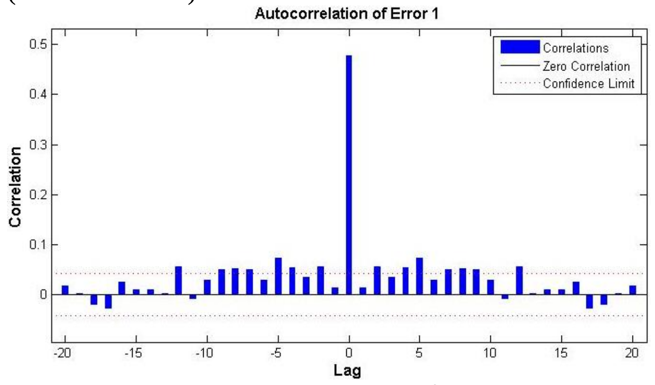

Fig. 8. Autocorrelation of Error

This error autocorrelation function describes how the prediction errors are related in time. For a perfect prediction model, there should only be one nonzero value of the autocorrelation function, and it should occur at zero lag. (This is the mean square error.) This would mean that the prediction errors were completely uncorrelated with each other (white noise). If there was 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. In this case, correlations, except for the one at zero lag, fall approximately within the 95% confidence limits around zero, so the model seems to be adequate. If even more accurate results were required, the group could retrain the network by clicking Retrain in ntstool. This will change the initial weights and biases of the network, and may produce an improved network after retraining.

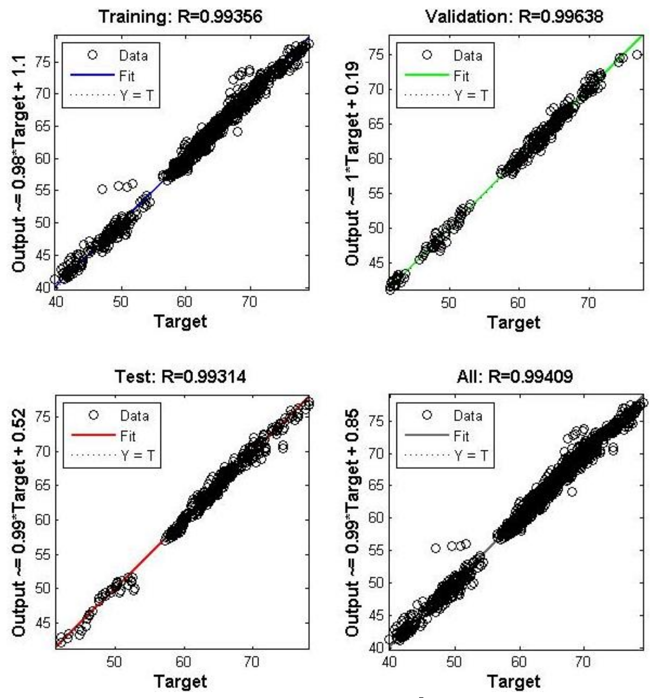

Fig. 9. Regression plots

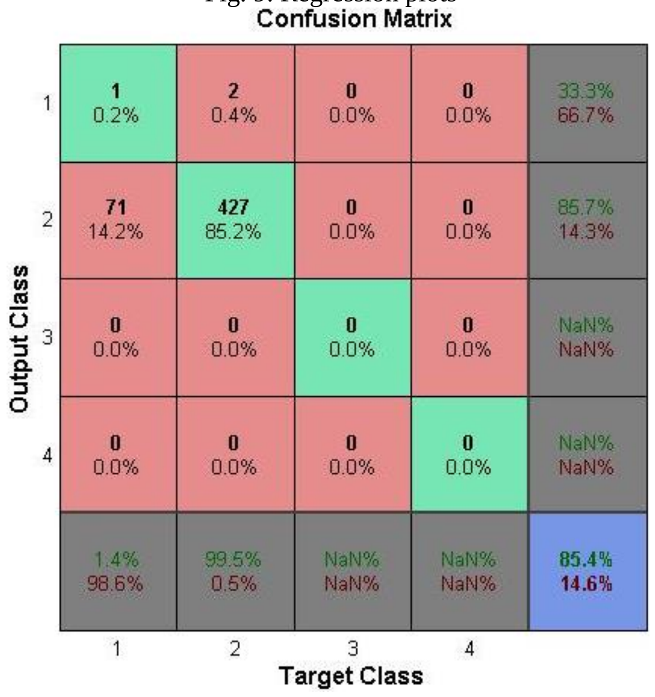

Fig. 10. Confusion Matrix

On the confusion matrix plot, the rows show the predicted class, and the columns show the true class. The diagonal cells show where the true class and predicted class match. The column on the right hand side of the plot shows the accuracy for each predicted class, while the row at the bottom of the plot shows the accuracy for each true class. The cell in the bottom right of the plot shows the overall accuracy. The overall accuracy achieved by our network is 85.4%

C. Notice/Update

The group was able to modify and optimize the system which resulted in a lower correlation thus increasing the overall accuracy of the neural network.

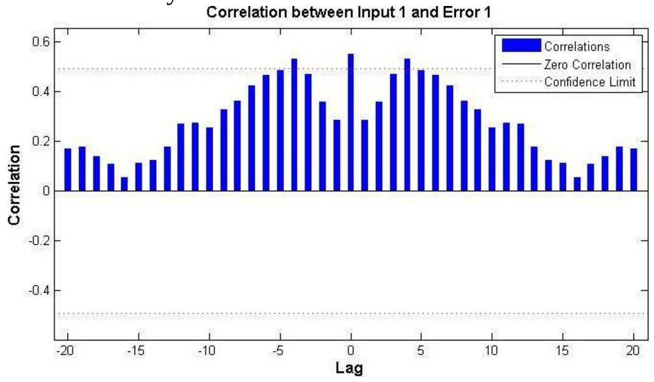

Fig. 11. Updated Input-Error Cross-Correlation

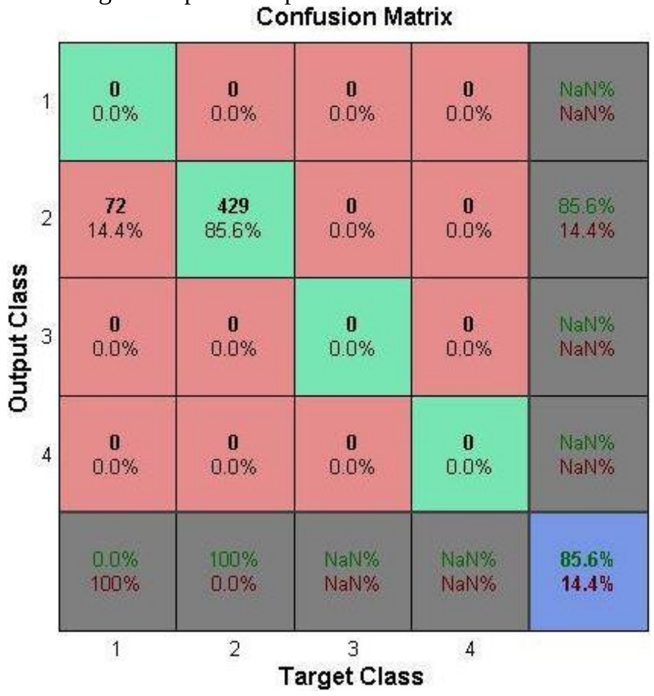

Fig. 12. Updated Confusion Matrix

The training, validation and testing of our program divides the samples by its corresponding percentage randomly. In our case, 20% is allotted for testing, the program will then choose 402 samples randomly because 402 samples is 20% of the 2012 samples available. This causes the accuracy to change for about +- 1.5 %. A part in determining accuracy is the confidence boundary in which predicted values exceeding the boundary are considered significant losses/errors. In our case the confidence boundary is at +-0.5, see figure 11 (correlation between input and error), is an acceptable confidence boundary in determining stock prices which would not be a significant loss when prediction fails as it is far lower than the SWIFT's collective reports "Adaptive Intelligent Systems" where 85% is the acceptance and 55% is confidence [16] as well as when compared to Patel's USD $1 to $5 (Php 47.03 to 235.13) minimum requirement as stated in the book “Stock Futures: Strategies for Profiting from Stock Futures” [17].

V. ANALYSIS AND CONCLUSION

In our study of the artificial neural network, a highly flexible non-linear modeling technique has been implemented to predict the stock prices of selected sectors. The input used are opening price, high, low, closing price as well as stock volume. The predicted outcome demonstrate that artificial neural network has been able to predict stock prices with better accuracy if the number of input data is increased. However, there is a considerable scope definitely to build on to attempt all possible ways to predict the stock prices with a higher accuracy rate.

Our data is represented by a 4x503 matrix in which the 4 rows are corresponding to the different values of the prices such as the opening price, high, low and closing price while the group gathered 503 samples which then corresponds to the number of rows of the matrix.

The main tool used is the nnstart tool in the Matlab software in order to produce the source code. The group also used the ntstool which can be found in the nnstart tool. This was used in order to utilize a two-layer feed-forward network in solving a fitting problem which is the time series. Also, in order to improve our predictions, the group used the Nonlinear Autoregressive with exogenous (external) input or more commonly known by the acronym NARX. This type of network is a recurrent network which was defined previously as a network that uses or connects to neurons within the same layer or previous layers. This is because the NARX predicts future values of y(t) from previous or past values of y(t). Moreover, out target inputs is a 4x503 matrix which means that the total number of target timesteps is 2012 since it was determined to have 503 timesteps of 4 elements.

Consequently, the group then divided the data into three sets which are the training set, validation set and the testing set, each of which are comprised of a percentage of target timesteps that covers the entire data. The division of the target timesteps to the three different sets are the following. 60% or 1308 target timesteps makes up the training data set. During training, these are presented to the network and it is adjusted according to its error. The validation data set is comprised of 15% or 302 target timesteps. Once generalization stops to improve, the training will be halted and these will be able to help measure the generalization of the network. Finally, the remaining 20% or 402 target timesteps goes to the testing data set. These has no effect on training and so it provides an independent measure of the performance of the network during and after training.

For our network architecture, as much as the group wanted to achieve the best results through the use of more hidden layers, it was found impossible to do so since memory limitations restricted the simulation and thus the group only used 2 hidden layers. However, the group was still able to produce acceptable results. The number of delays that used is 2 and these delays are used to feed the output of the NARX to the input of the network. The group created and trained our network in open loop form for the reason that it would allow us to supply the network with correct feedback inputs.

In the training and testing of our network, the group used Scaled Conjugate Gradient as the training algorithm since it uses less memory. Once the mean square error has increased, the generalization of the network stops to improve and thus training will also stop. This is because the increase in the mean square error indicates that the training of the data has reached its limit and the error cannot be further reduced.

Based from our results, it can be very evident that our predictions are considerably at par with the expected values since the convergence of the MSE (mean square error) from our plot seems to be smooth as the number of iterations or epochs increases which also implies that it is consistent. The training, validation and testing errors all decreased as well implied by the decreasing trend of the plot up until iteration 62. In addition, there seems to have been no problems of overfitting in our prediction since the trend is consistently downward and neither testing nor validation errors increased before iteration 62. Meanwhile, the group had identified the best validation performance to be 0.96476 at epoch 40 because it hits the best plot at that point.

Meanwhile, the plot of the time series response displays the inputs, targets and errors versus time. It also indicates which time points were selected for training, testing and validation.

Our input-error cross-correlation plot seems to indicate that there are inputs that are correlated to errors since a perfect prediction model would result to all correlations equal to zero however in our case it seems to be a bit far from zero and so it indicates that our prediction is not perfect but is very much close. In actuality, there can never really be an absolute perfect prediction and what the group can only do is to try to get as much as closer to the prediction as possible. However, it is possible to improve these predictions through the increase in delays in the tapped delay lines and then some of the correlation would at least fall beneath the confidence limits indicating a more favorable result.

The autocorrelation plot displays the relation of the error in time and is used to validate the networks performance. A perfect prediction model is indicated by a nonzero value of the autocorrelation function which is not occurring at the zero lag. This indicates that the prediction errors are uncorrelated with each other and that is how they measure the networks performance. Further, the predictions could also be improved by the same procedure of adding delays in the tapped delay lines and would result in the falling of the correlations to 95% of the confidence limits except of course for the correlation occurring at the zero lag.

Our regression plots on the different sets that the group used on our data which are the training, validation and testing prove to be of good results as well since most our data are able to comply with the trend of the plot. The group made regression plots for each of the sets and then combined them in one plot to see the overall trend.

Finally, the confusion matrix, show the predicted class or the output class on the rows. The columns show the true class. The diagonal cells show the true class and the predicted class match. The cells at the rightmost column shows the accuracy of each predicted class, likewise, the cells at the bottom row shows the accuracy of each true class. at the rightmost as can be seen from the figure. And lastly, the rightmost bottom cell shows the overall accuracy in which case the overall accuracy that was achieved was 85.4%. This accuracy seems to be very much considerable already since several techniques were used in order to improve it such as the use of the NARX network and the increasing of delays to get the prediction to be much closer to the expected values.

REFERENCES

[1] A. Kar. (2014, November 24). Stock Prediction using Artificial Neural Networks. [Online]. Available: http://www.cs.berkeley.edu/~akar/IITK_website/EE671/report_stock.pdf

[2] A. Karpathy. (n.d.). CS231n Convolutional Neural Networks for Visual Recognition. [Online]. Retrived from: http://cs231n.github.io/neural-networks-3/

[3] B.G. Malkiel. “A Random Walk Down Wall Street” (6th ed.). W.W. Norton & Company, Inc., 1973

[4] Beale, R. and Jackson, T.Neural Computing: An Introduction. Hilger, Philadelphia, PA, 1991.

[5] Doc.ic.ac.uk, 'Neural Networks', 2015. [Online]. Available: http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html#What is a Neural Network.

[6] H. Mizuno, M. Kosaka, H. Yajima, N. Komoda. “Application of Neural Network to Technical Analysis of Stock Market Prediction”. Studies in Informatic and Control, Vol.7, No.3, 1998, pages.111–120.

[7] Herculano-Houzel, Suzana. ”The human brain in numbers: a linearly scaled-up primate brain”. Frontiers In Human Neuroscience.doi:10. 3389/neuro.09.031.2009.

[8] J.E. Dayhoff, J.M. DeLeo, “Artificial neural networks: Opening the black box”. Cancer, 91 (8 Suppl.) (2001), pp. 1615–1635.

[9] M. Nedrich. (2014, June 24). An Introduction to Gradient Descent and Linear Regression. [Online]. Available: http://spin.atomicobject.com/2014/06/24/gradient-descent-linear-regression/

[10] MathWorks. (n.d.). *MATLAB -The Language of Technical Computing*. [Online]. Available: http://www.mathworks.com/products/matlab/

[11] MathWorks. (n.d.). Neural Network Time Series Prediction and Modeling. [Online]. Available: http://www.mathworks.com/help/nnet/gs/neural-network-time-series-prediction-and-modeling.html

[12] McCulloch, W. and Pitts, W. (1943). “A logical calculus of the ideas immanent in nervous activity”. Bulletin of Mathematical Biophysics, 7:115 - 133.

[13] P. McCollum. (n.d.). An Introduction to Back-Propagation Neural Networks. [Online]. Retrieved from: http://www.seattlerobotics.org/encoder/nov98/neural.html

[14] Seattlerobotics.org, 'An Introduction to Back-Propagation Neural Networks', 2015. [Online]. Available: http://www.seattlerobotics.org/encoder/nov98/neural.html.

[15] T. Kimoto, K. Asakawa,M. Yoda,M. Takeoka. “Stockmarket prediction systemwithmodular neural network”. Proceedings of the International Joint Conference on Neural Networks, 1990, pages 1–6

[16] SWIFT. (2014). Adaptive Intelligent Systems. Elsevier Science Publishers B.V.

[17] A. Patel. (2003). Stock Futures: Strategies for Profiting from Stock Futures. Harriman House Limited.