Short Term Forcasting of Financial Market Using Adaptive Learning in Neural Network

Hong Li
Department of Computer Systems Technology
New York City College of Technology - The City University of New York
New York, New York, USA

Abstract - *This paper presents a short term forecasting of stock market using Feed Forward Multilayer Perceptron (FMP) with an adaptive learning algorithm. Business people often attempt to anticipate the market by interpreting external parameters, such as economic indicators, public opinion, and current political climate. However, the neural network is used in purpose of discovering trends in data that humans might not notice, and successfully use these trends in their predictions. Traditionally, the neural network training process takes trial and error for different values of learning factors. In work [1, 2], the analysis of convergence of learning process based on the Backpropagation algorithm leads to conditions that the learning factors satisfy to guarantee the convergence. In this paper, the conditions are further extended to a feasible formula that can be calculated to define an adaptive learning factor at iteration of learning process. The result of simulations using stock market data sourced from Yahoo! demonstrates that errors steadily decrease in training with the adaptive learning factor starting at different initial value and errors behave volatile with constant learning factors with different values. Once the neural network is trained, it provides predication for future market performance. The network is modified or retrained for every future data set starting with trained weights and learning factor. Performance of such network learning with adaptive learning factors are presented and demonstrated that the adaptive learning factor enhance the performance of training while avoiding oscillation phenomenon.*

Keywords: Neural network, financial market, stability analysis, adaptive learning

1. INTRODUCTION

The idea to predicate stock market is not new. It has been for many years the focus of many researchers to seek methods that achieve accurate forecasting using historical data [8, 9, 10]. Business people often attempt to anticipate the market by interpreting external parameters, such as economic indicators, public opinion, and current political climate. The question is, though, if neural networks can discover trends in data that humans might not notice, and successfully use these trends in their predictions. The Artificial Neural Networks (ANNs) have been proven in many real world applications to be useful in various tasks of modeling nonlinear systems, such as signal processing, pattern recognition, optimization, weather forecasting, to name a few. It has drawn many researchers in financial market because neural networks may be easy to use once the network is set up, but the setup and training of the network requires skill, experience, and patience. It's not all hype, though; neural networks have shown success at prediction of market trends. There are number of commercial software as well as free software that offer neural network simulation.

The ANN is a set of processing elements (neurons or perceptrons) with a specific topology of weighted interconnections between these elements and a learning law for updating the weights of interconnection between two neurons. The FMP networks have been shown to obtain successful results in system identification and control [3]. The Lyapunov function approach was used to provide stability analysis of Backpropagation training algorithm of such network in [4-7]. However, the training process can be very sensitive to initial condition such as number of neurons, number of layers, and value of weights, and learning factors which are often chosen by trial and error. This paper presents a detailed analysis of the FMP architecture and its stability. The Backpropgation algorithm is used for learning – that is, weight adjusting. This is a simple back-propagation network of three layers, and it is trained and tested on a high volume of historical market data. The challenge here is not in the network architecture itself, but instead in the choice of variables and the information used for training. In this research, historical data were chosen from previous six days open, close, high and low price as well six previous Nasdaq open, close, high and low.

The Least Square error function is defined and verified that it satisfies Lyapunov condition so that it guarantees the stability of the system. In the work [1], the analysis carries out a method that defines a range for value of learning factor at iteration which ensure the condition for stability are satisfied. In simulation, instead of selecting a learning factor by trial and error, author defines an adaptive learning factor which satisfies the convergence condition and adjust

connection weight accordingly. The simulation results are presented to demonstrate the performance.

2. BASIC PRINCIPLE OF FMP NETWORK

A system identification problem can be outlined as follow: a set of data is collected from the system including input data and corresponding output data observed, or measured as target output of the identification problem. The set is often called "training set". A neural network model with parameters, called weights, is designed to simulate the system. When the output from neural network is calculated, an error representing the difference between target output and calculated output from the system is generated. The learning process of neural network is to modify the network to minimize the error.

Consider a system with N inputs $X = \{X_1, \dots, X_N\}$ and M output units Y $=\{Y_1, \dots, Y_M\}$. A recurrent FMP network combines number of neurons, called nodes, feed forward to next layer of nodes, illustrated in Figure 1. Suppose N_l is number of nodes in lth layer, each output from the l-1th layer will be used as input for next layer. A system of a single layer with M outputs can be expressed in form of

$$Y_{jp} = f(Z) = f\left(\sum_{i=1}^{N} w_{ij} X_{ip} + \sum_{i=1}^{D} v_{ij} Y_{j(p-i)}\right) \quad (1)$$

where w_{ij} is called connection weight from input X_i to output Y_j;vij is called connection weight of local feedback at jth node with ith delay; $f(\,)$ is a nonlinear sigmoid function

$$f(Z) = \frac{1-e^{-\theta Z}}{1+e^{-\theta Z}} \quad (2)$$

with constant coeficient θ, called slope; p = 1, …, T, T is number of patterns, D is number of delay used in local feedback.

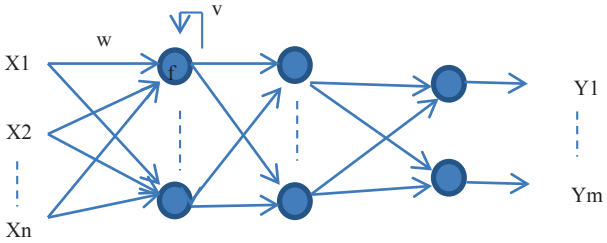

Figure 1. Feed-Forward Multi-Layer Network

The back-propagation algorithm has become the standard algorithm used for training feed-forward multilayer perceptron. It is a generalized the Least Mean Square algorithm that minimizes the mean squared error between the target output and the network output with respect to the weights. The algorithm looks for the minimum of the error function in weight space using the method of gradient descent. The combination of weights which minimizes the error function is considered to be a solution of the learning problem. A proof of the Backprpagation algorithm was presented in [10] based on a graphical approach in which the algorithm reduces to a graph labeling problem.

The total error E of the network over all training set is defined as

$$E = \frac{1}{T}\sum_{k=1}^{N_L}\sum_{p=1}^{T} e_k^2(p) \quad (3)$$

where $e_k^2(p)$ is the error associated with *pth* pattern at the *kth* node of output layer,

$$e_k^2(p) = (\, d_k(p) - Y_k^L(p))^2 \quad (4)$$

where $d_k(p)$ is the target at kth node and $Y_k^L(p)$ is the output of network at the kth node. The learning rule was chosen following gradient descent method to update the network connection weights iteratively,

$$\Delta W_j = -\mu \frac{\partial E}{\partial W_j}; j = 1, \dots, M \quad (5)$$

$$\Delta v_j = -\mu \frac{\partial E}{\partial v_j}; j = 1, \dots, D \quad (6)$$

where $W_j = (w_{1j}, \dots, w_{Nj})$ *and* $v_j = (v_{j1}, \dots, v_{Dj})$ are weight vectors in jth node; μ is a constant called learning factor.

3. ADAPTIVE LEARNING FACTOR FROM STABILITY ANALYSIS

There are no inclusive general concepts of stability for nonlinear systems. The behavior of a system may depend drastically on the inputs and the disturbances. However, Lyapunov theory has been used in many researches to examine the stability of nonlinear systems.

The definition of the Lyapunov function and the Lyapunov theorem are quoted below [11]:

Definition 1 (Lyapunov function): A scalar function V(x) is a Lyapunov function for the system

$$x(t+1) = f(\,x(t)), \; f(0) = 0 \quad (7)$$

if the following conditions hold:

1. $V(0) = 0$ and $V(0)$ is continuous in x
2. $V(x)$ is positive definite, that is, $V(x) \geq 0$ with $V(x) = 0$ only if $x = 0$

3. $\Delta V(x) = V(f(x(t)) - V(x(t))$ is negative definite, that is, $V(f(x(t)) - V(x(t)) \leq 0$ with $\Delta V(x) = 0$ only if $x = 0$;

Theorem 1 (Lyapunov Theorem): The solution $x(t) = 0$ for the system given by (7) is asymptotically stable if there exists a Lyapunov function of the system in x.

The stability of the learning process in an identification approach leads to better modeling and a convergent process. According to the Lyapunov theorem, determination of stability depends on selection and verification of a positive definite function. For the systems defined in (1) – (2), assume that the Backpropagation learning rule is applied and the error function and weights updating rule are defined in (5) - (6), then define

$$V(t) = \frac{1}{N_L T} \sum_{j=1}^{N_L} \sum_{p=1}^{T} e_k^2(t) \quad (8)$$

The proof is given in the following theorem that the V (t) satisfies the Lyapunov conditions.

Theorem 2: Assume that the nonlinear sigmoid function $f(\,)$ defined in equation (2) is continuous and differentiable, the network is defined in (1)-(2) with learning rule (5) - (6), then the system is stable under the condition:

$$\mu < \frac{TM}{\sum_{j=1}^{M} \sum_{p=1}^{T} \left(\left\| \frac{\partial Y_p}{\partial W_j} \right\|^2 + \left\| \frac{\partial Y_p}{\partial v_j} \right\|^2 \right)} \quad (9)$$

Proof: It is give in [1].

Theorem 3. Assume that the system with one hidden layer can be represented in the form of:

$$Y_{jp} = f(Z_{jp}^1) = f(\sum_{i=1}^{N} w_{ij}^o X_{ip}^1 + \sum_{i=1}^{D} v_i^o Y_{j(p-i)}) \quad (10)$$

$$X_{jp}^1 = f(Z_{ip}) = f(\sum_{i=1}^{N} w_{ij}^h X_{ip} + \sum_{i=1}^{D} v_i^h X_{j(p-i)}^1) \quad (11)$$

the gradient descent rule

$$\Delta W_o^j = -\mu \frac{\partial E}{\partial W_o^j}, \ j = 1, \dots, M \quad (12)$$

$$\Delta v_o^j = -\mu \frac{\partial E}{\partial v_o^j}, \ j = 1, \dots, D \quad (13)$$

$$\Delta W_h^j = -\mu \frac{\partial E}{\partial W_h^j}, \ j = 1, \dots, M \quad (14)$$

$$\Delta v_h^j = -\mu \frac{\partial E}{\partial v_h^j}, \ j = 1, \dots, D \quad (15)$$

where

$$W_o^j = (w_{o1}^j, \dots, w_{oN}^j)^T, \ v_o^j = (v_{o1}^j, \dots, v_{oD}^j)^T$$

$$W_h^j = (w_{h1}^j, \dots, w_{hH}^j)^T, \ v_h^j = (v_{h1}^j, \dots, v_{hD}^j)^T$$

are weight vectors in jth node in output layer and hidden layer respectively. H is the number of nodes in hidden layer. The system is stable when the learning factor in (18) - (21) satisfies the condition given below:

$$\mu < \frac{TM}{2\sum_{j=1}^{M} \sum_{p=1}^{T} \left[\left\| \frac{\partial Y_{pj}}{\partial W_o^j} \right\|^2 + \left\| \frac{\partial Y_{pj}}{\partial v_o^j} \right\|^2 + \left\| \frac{\partial Y_{pj}}{\partial W_h^j} \right\|^2 + \left\| \frac{\partial Y_{pj}}{\partial v_h^j} \right\|^2 \right]} \quad (16)$$

Proof: Similarly, given in [1].

In general simulation, the learning factor was predefined constant whose value was selected by trial and error. The simulation performance differs from different values of learning factor. The learning process may converge or may not reach a satisfactory threshold with different learning factors. From the result of above theorem, the convergence is guaranteed if an adaptive learning factor is selected at iteration of the learning which satisfies the stability condition. For purpose of simplifying the simulation, instead of calculating all $\frac{\partial Y_{pj}}{\partial W^l}$ and $\frac{\partial Y_{pj}}{\partial v^l}$ for l = 1, … L; j = 1, … N_L, the following corollary will provide a more restrictive but easier calculated condition.

Consider infinite norm notation for any vector $X = \{x_1, x_2, \dots, x_n\}$ that $\|X\|_\infty = \max_{1<i<n} \{x_i\}$, for simplicity, use notation $\|*\|$ in this paper representing $\|X\|_\infty$). Applying infinite norm in (16) and notation $|v_j^l| = \sum_{d=1}^{Dv} v_{jd}^l$, calculation of $\left\| \frac{\partial Y}{\partial W^o} \right\| \ and \ \left\| \frac{\partial Y}{\partial v^o} \right\|$ lead to

$$\left\| \frac{\partial Y}{\partial W^o} \right\| \leq \frac{\theta}{2} \left[1 + |v_j^o| \left\| \frac{\partial Y}{\partial v^o} \right\| \right]$$

$$\left\| \frac{\partial Y}{\partial v^o} \right\| \leq \frac{\theta}{2} \left[1 + |v_j^o| \left\| \frac{\partial Y}{\partial v^o} \right\| \right]$$

then

$$\left\| \frac{\partial Y}{\partial v^o} \right\| \leq \frac{\theta}{(2 - \theta |v_j^o|)}, \text{ and } \left\| \frac{\partial Y}{\partial v^o} \right\| \leq \frac{\theta}{(2 - \theta |v_j^o|)} \quad (17)$$

Further calculation of $\left\| \frac{\partial Y}{\partial W^h} \right\| \ and \ \left\| \frac{\partial Y}{\partial h} \right\|$ lead to

$$\left\| \frac{\partial Y}{\partial W^h} \right\| \leq \frac{\theta}{2} (\sum_{j=1}^{N} w^o \left\| \frac{\partial Y^h}{\partial W^h} \right\| + |v_j^o| \left\| \frac{\partial Y}{\partial v^h} \right\|)$$

$$\leq \frac{\theta}{2} (|w^o| \frac{\theta}{(2 - \theta |v_j^o|)} + |v_j^o| \left\| \frac{\partial Y}{\partial v^h} \right\|)$$

$$\left\| \frac{\partial Y}{\partial W^h} \right\| \leq \frac{\theta^2 |w^o|}{(2 - \theta |v_j^o|)^2}, \text{ and } \left\| \frac{\partial Y}{\partial v^h} \right\| \leq \frac{\theta^2 |w^o|}{(2 - \theta |v_j^o|)^2} \quad (18)$$

Add the (26) and (27), and result in the following Corollary.

Corollary 1: The system defined in (16) – (17) converges if the learning factor in (18) – (21) satisfies the following conditions:

$$2 - \theta\,|v^o| > 0\,, \tag{19}$$

$$\mu < \frac{\left(2-\theta\left|v_j^o\right|\right)^2}{4\theta(2+\theta(|w^o|-\left|v_j^o\right|} \tag{20}$$

4. SIMULATION

Many researches provide analysis that what historical data are adequate for training of the neural network system as the market can be influenced by many business or political factors. However, the research of neural network learning is to build a system that learned from only previous data and required less human inputs. Human never fully understand the complicated correlation of factors that might have influenced the market a high accuracy. This simulation mainly demonstrates that the enhanced learning algorithm may avoid many trial and error for selection of learning factors.

In traditional neural network training, the initial weights are randomly selected, a learning factor is predefined. The performance of the learning can sometime very volatile due to the selection of the learning factor. To find the optimal fit, the trial and error is common practice that runs the simulation with different values of learning factors. In this research, an upper boundary of learning factors (20) is derived from the theory of convergence. At iteration of network training, the norm of weights is calculated following (17), and a learning factor is defined to satisfy the convergence condition (20).

A three layer neural network structure was selected with 24 inputs, 10 and 7 nodes in the hidden layer, and two outputs. The data are sourced from Yahoo! Finance. The daily price of stock IBM including open, high low and close price in 2015 is selected for demonstration. Two outputs are daily high and low. 24 inputs are selected as follows: high and low of the IBM stock from previous six days; high and low of Nasdaq from previous six days. Input and output data are normalized to range from 0 to 1.

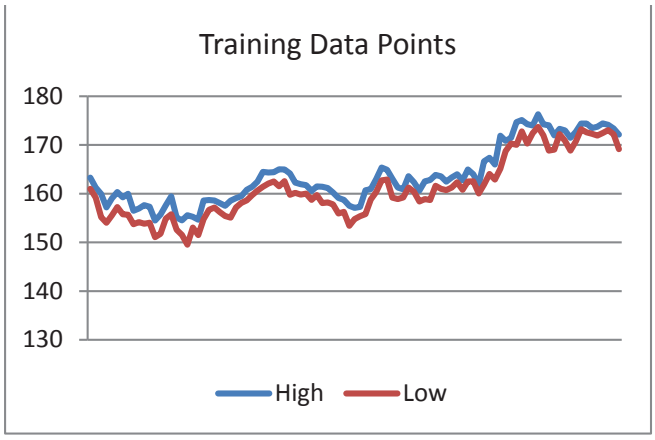

Figure 2. IBM Stock Daily High and Low Price as Training Points

With the constant learning factor, several values were used for the learning trials: 0.2, 0.15, 0.1, and 0.05. After number of attempts, with slope set as 0.7, learning factor set as constant 0.05, momentum term set as 0.1, and random generated initial weights, the system reached to absolute error 0.028 after 100000 iterations. The error behaviors are shown respectively in Figure 3 – Figure 6.

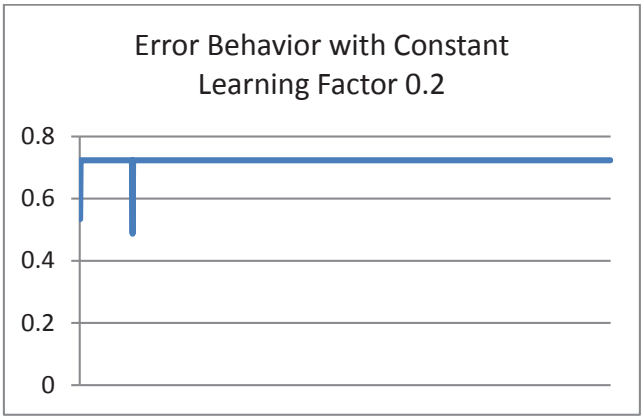

Figure 3. Error Behavior of Neural Network Training with Constant Learning Factor 0.2

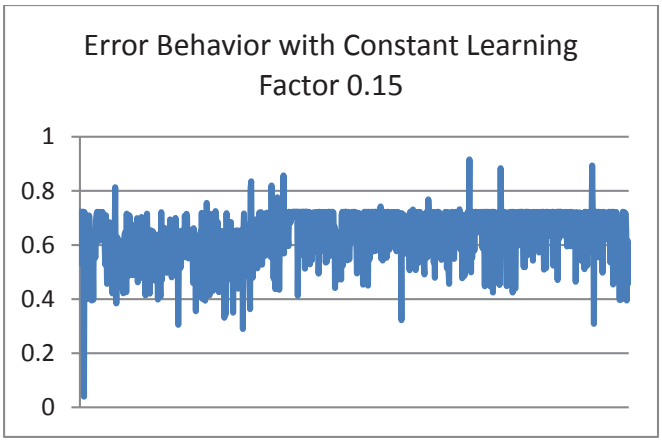

Figure 4. Error Behavior of Neural Network Training with Constant Learning Factor 0.15

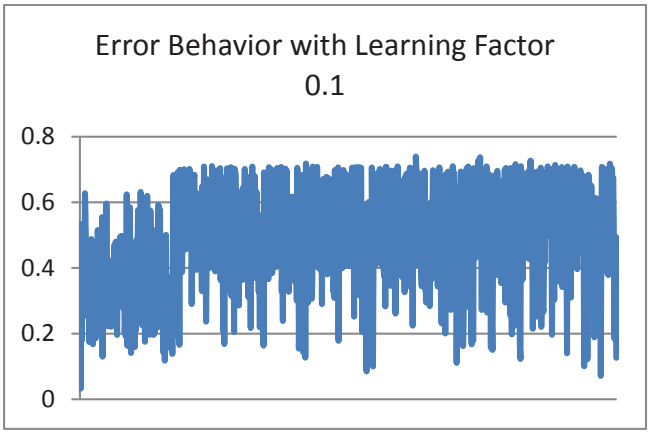

Figure 5. Error Behavior of Neural Network Training with Constant Learning Factor 0.1

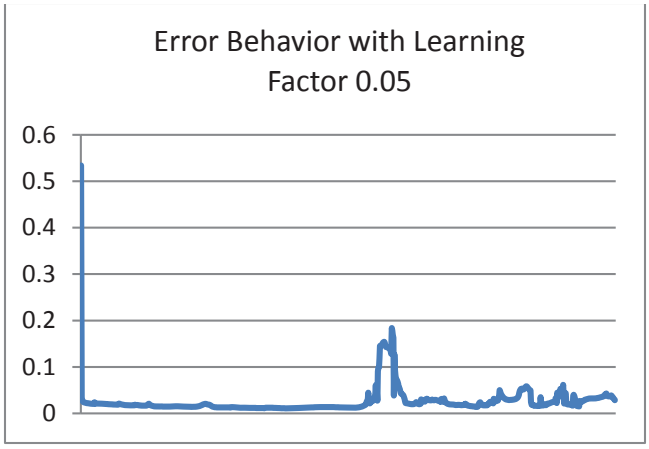

Figure 6. Error Behavior of Neural Network Training with Constant Learning Factor 0.05

With initial learning factor 0.2, 0.15, 0.1 and 0.05, the errors steadily decreases at iteration when an adaptive learning factor was applied at iteration, Figure 7 – Figure 10 demonstrate the error behaviors of learning with initial learning factors, 0.2, 0.15, 0.1 and 0.05 respectively. It is observed that error behaviors do not differ with different initial values of learning factors. However, the constant learning factor could cause volatile performance of training.

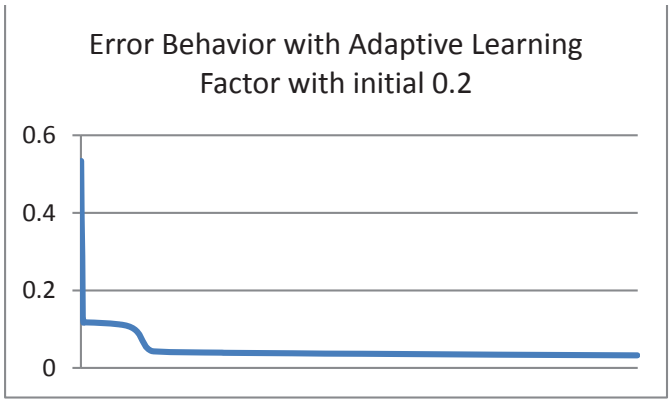

Figure 7. Error Behavior of Neural Network Training with adaptive Learning Factor with initial 0.2

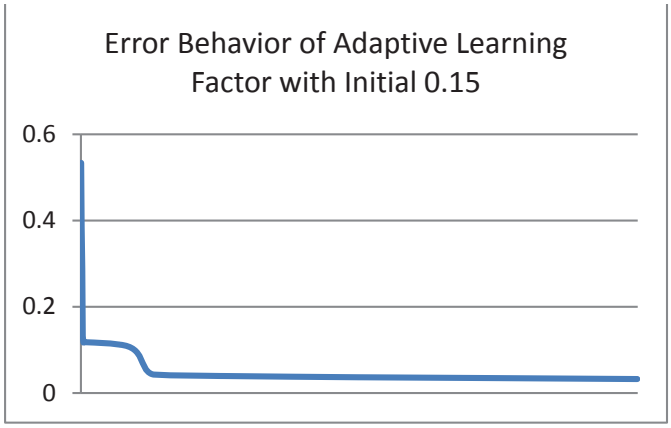

Figure 8. Error Behavior of Neural Network Training with adaptive Learning Factor with initial 0.15

The Figure 11 presents the comparison of IBM daily high and ANN model of 100 days and 9 days prediction of the trained neural network. Figure 12 presents comparison of IBM daily low and ANN model of 100 days and 9 days prediction of low from the trained neural network.

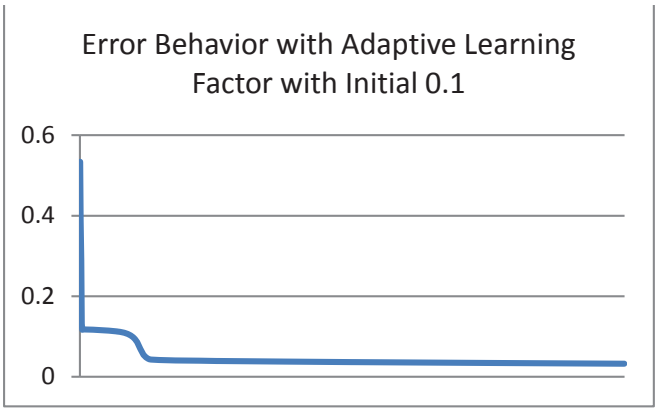

Figure 9. Error Behavior of Neural Network Training with Adaptive Learning Factor with initial 0.1

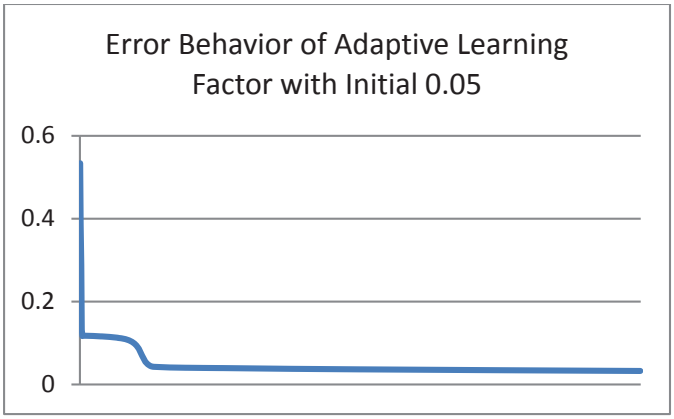

Figure 10. Error Behavior of Neural Network Training with adaptive Learning Factor with initial 0.05

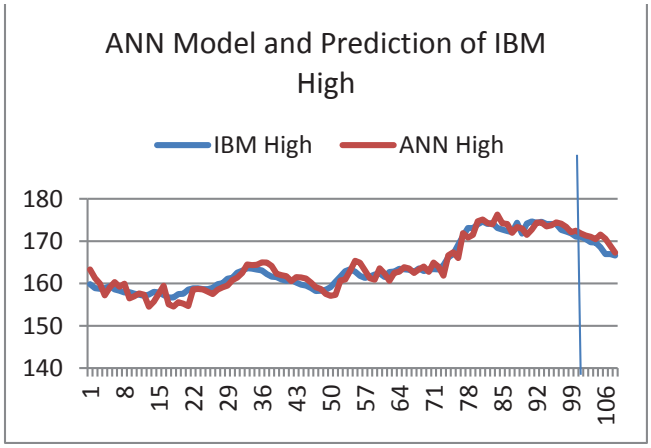

Figure 11. ANN Model and predication of IBM high

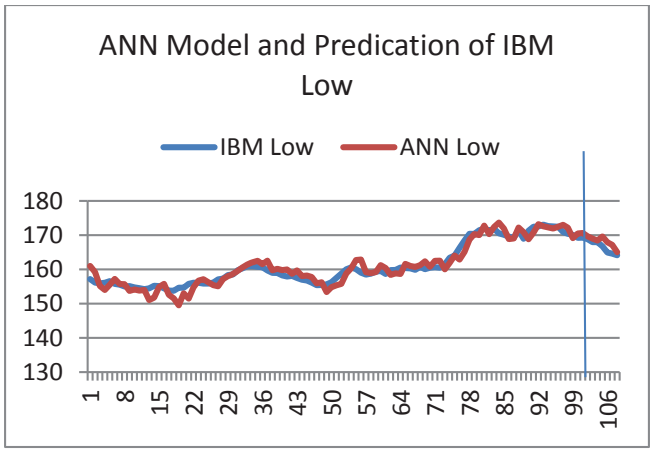

Figure 12. ANN Model and predication of IBM low

5. SUMMARY

This research focused on improvement of learning algorithm. A condition was derived from the proof of convergence of neural network system learning process using Backpropagation algorithm. The condition provides an upper boundary of the learning factor. Instead of select a constant learning factor by trial and error, an adaptive learning factor is calculated at iteration satisfying the convergence condition. Furthermore, a more simplified condition was used to provide a feasible implementation of the adaptive learning factor. At iteration of the learning process, an adaptive learning factor was selected satisfying the stability condition to avoid unstable phenomena. Simulation results of the IBM stock prices demonstrated that a learning factor arbitrarily chosen out of the predefined stability domain leads to an unstable identification of the considered system; however, an adaptive learning factor satisfying the conditions chosen for this study ensures the stability of the identification system.

6. REFERENCES

[1] Hong Li, Adaptive Learning Factor of Backpropagation in Feed Forward Neural Networks, International Journal of Modern Engineering (IJME), Vol. 14, No 2, Spring/Summer 2014, pp 47 – 53.

[2] Hong Li and Ali Setoodehnia, Convergence Analysis of Adaptive Recurrent Neural Network, International Journal of Engineering Research and Applications (IJERA), Vol. 4, Issue 6(Version 1), June 2014, pp.48-53

[3] Yu, W., Poznyak, A. S., & Li, X. (2001). Multilayer Dynamic Neural Networks for Non-linear

System Identification. International Journal of Control, 74(18), 1858-1864.

[4] Korkobi, T., Djemel, M. & Ctourou, M. (2008). Stability Analysis of Neural Netorks-based System Identification. Modeling and Simulation in Engineering, Volume 2008, Article ID 343940

[5] Zhu, E. (2010). Stability Analysis of Recurrent Neural Networks with Random Delay and Markovian Switching. Journal of Inequalities and Applications

[6] Akea, H., Alassar,R., & Covachev, V. (2005). Stability of Neural Networks with Time Varying Delays in the Presence Impulses. Advances in Dynamical Systems and Application,. 1(1), 1-15.

[7] Kolla, Sri R. (2012). Comparison of Time Delay Controllers for A class of Networked Control System, International Journal of Modern Engineering, Fall/Winter, 13(1), 51-56.

[8] Tarsauliya, A. Analysis of Artificial Neural Network for Financial Time Series Forecasting, International Journal of Computer Applications (0975 – 8887) Volume 9– No.5, November 2010.

[9] Andrew Skabar, A. Cloete, I Neural Networks, Financial Trading and the Efficient Markets Hypothesis the Twenty-Fifth Australasian Computer Science Conference (ACSC2002), Melbourne, Australia.

[10] Choon Ki Ahn, H. (2012) Stability Conditions for Fuzzy Neural Networks, Advances in Fuzzy Systems Volumn Articale ID 281821

[11] Zhao, H. (2003). Global Stability of Neural Network with Distributed Delays. Joural of Neural, Parallel & Scientific Computations, 11(3), 237-252.

[12] Liang, G. (2010). Global Asymptotically Stability of Cellular Neural Networks with Time-varying Delay, 2010 8th World Congress on Intelligent Control and Automation , (pp 5031-5036)