

A Hybrid Time Lagged Network for Predicting Stock Prices

S.C. Hui, M.T. Yap and P. Prakash

*School of Applied Science,
Nanyang Technological University
Singapore
Email:{asschui, asmt yap}@ntu.edu.sg*

Abstract

Traditionally, technical analysis approach, that predicts stock prices based on historical prices and volume, basic concepts of trends, price patterns and oscillators, is commonly used by stock investors to aid investment decisions. Advanced intelligent techniques, ranging from pure mathematical models and expert systems to neural networks, have also been used in many financial trading systems for predicting stock prices. In this paper, we propose the Hybrid Time Lagged Network (HTLN) which integrates the supervised Multilayer Perceptron using temporal back-propagation algorithm with the unsupervised Kohonen network for predicting the chaotic stock series. This attempts to combine the strengths of both supervised and unsupervised networks to perform more precise prediction. The proposed network has been tested with stock data obtained from the main board of Kuala Lumpur Stock Exchange (KLSE). In this paper, the design, implementation and performance of the proposed neural network are described.

Keywords: Neural networks, hybrid approach, multilayer perceptron, Kohonen networks, stock prediction.

1. Introduction

Recently, with the rapid growth in the Internet and the possibility of on-line stock trading [1,2,3], intelligent tools are needed to help investors to predict stock movement over the Internet. Traditionally, technical analysis approach [4], that predicts stock prices based on historical prices and volume, the Dow theory, basic concepts of trends, price patterns and oscillators, is commonly used by stock investors to aid investment decisions. Advanced intelligent techniques ranging from pure mathematical models and expert systems [5,6,7] to neural networks [8,9,10,11] have also been used in many financial trading systems for stock prediction.

In this paper, we propose the Hybrid Time Lagged Network (HTLN) which integrates the supervised Multilayer Perceptron using temporal back-propagation algorithm [12,13] with the unsupervised Kohonen network [14] for predicting the chaotic stock series. The performance of the HTLN algorithm has been tested with a stock database that contains ten years of stock information from the main board of Kuala Lumpur Stock Exchange (KLSE). The rest of the paper is organised as follows. Section 2

describes the neural trading systems for stock prediction. Section 3 discusses the HTLN networks. The stock prediction process is then described in Section 4. The performance analysis of the HTLN networks is presented in Section 5. Finally, Section 6 concludes the paper.

2. Neural Trading Systems

Neural trading systems [8,9,10,11] provide an automated way of trading in a financial market based on neural networks. Neural networks are a collection of mathematical techniques for curve fitting, clustering and signal processing. They are simple models of neural interconnections in brains, adapted for use on digital computers. They learn from a training set and generalise patterns inside it for classification and prediction. Neural networks can generalise or interpolate between known values in a very high dimensional input/output space. Back-propagation neural networks are most successfully used for time series prediction [15,16,17] and provide a reasonably good performance.

One of the main advantages of neural networks is their wide applicability. Owing to their widespread utility, tools supporting neural networks are available from multiple vendors on a wide variety of platforms. Neural networks are also interesting because they detect patterns analogous to human thinking. Another advantage is the capability of the neural networks to approximate any function, any distribution and be consistent in their results. A lot of these factors are dependent on the quality of learning data provided. However, neural networks have two major problems. The first is the difficulty to understand the models that they produce. There are tools available to delve into the underlying neural network structure

to make them more explainable. But even then it has not been possible so far to provide a complete explanation for the results obtained by the neural network. The second problem that they face is their sensitivity to the format of input data. Different data representations can trigger different results. Therefore, setting up data is a significant part of the effort in using them.

3. Hybrid Time Lagged Networks

The Hybrid Time Lagged Networks attempt to utilise the strengths of the supervised Multilayer Perceptron using temporal back-propagation algorithm and unsupervised Kohonen network. It uses the Kohonen network as a primary layer for input processing, and Multilayer Perceptron to process the input signal treating the output of the Kohonen network as a guidance factor to predict the future values. The use of the unsupervised Kohonen network in the HTLN network enables the capturing of fine grain data series behaviour of the Multilayer Perceptron. However, the training algorithm for such a network is more complex than the Multilayer Perceptron and Kohonen networks, since it involves training these two different networks one after the other.

The HTLN networks are trained in a sequential fashion, starting from the input end [17]. The unsupervised Kohonen network is trained first, without concerning the top layer supervised Multilayer Perceptron network. It learns to classify the input data vectors into separate classes (twenty of them), and trains its slope parameters to predict the next point $g(n)$ used as a guidance factor by the Multilayer Perceptron network. The *guidance factor* $g(n)$ is calculated using the slope parameter for the pattern. The y-intercept is calculated from the data input vector, and is used in the

straight-line equation for the prediction. The supervised network is trained after the training of the unsupervised layer is completed. The supervised layer uses the guidance factor $g(n)$ generated by the lower layer and compares it with the desired

output, adjusting its weights so as to minimise the error function. Fig. 1 shows the mechanism of the HTLN network that contains Short Term Memory at the input-processing unit, to provide the tap delay filter to the hybrid network.

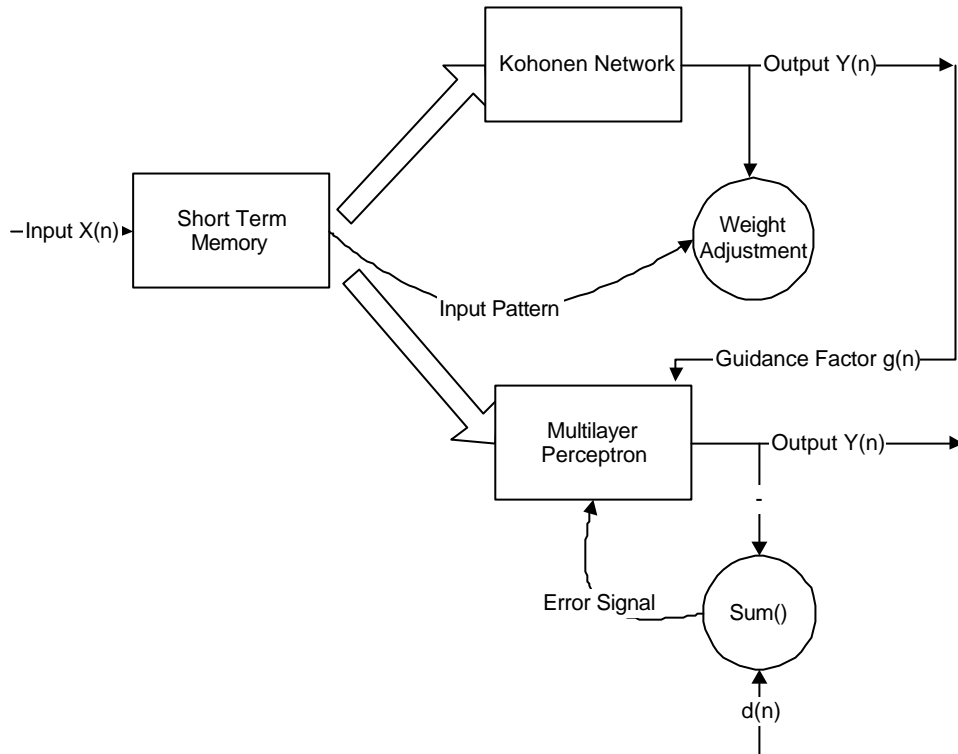


Fig. 1. The HTLN network.

3.1 Input Layer

The input vector $x(n)$ at the short term memory, is used by the Kohonen network to produce the guidance factor $g(n)$ which in turn is used by the Multilayer Perceptron network to predict the output. The order of the memory p is the magnitude of the short term memory to be provided to the system. It also performs all the necessary data pre-processing such as normalisation before the data can be fed into the system. Normalisation is necessary, as the activation function used in the network is a bounded function. The input vector to the system at iteration n can be represented as:

$$X(n) = \sum_{n=0}^p x(n)z^{-n}$$

where z^{-1} is the unit delay operator.

3.2 Unsupervised Kohonen Network

The unsupervised Kohonen network is used to filter the data into patterns and thus, simplify the learning of the Multilayer Perceptron at the top. It also provides the Multilayer Perceptron with a guidance parameter that helps it in obtaining a higher degree of accuracy in prediction. It has only

20 neurons, so as to reduce the time taken in training the network. Although it is now able to classify the input series into only 20 different patterns, it results in considerable gains in both training and execution time of the network.

In this network, the weights of all the neurons are initialised to a common value $\frac{1}{\sqrt{2}} [1,1,\dots,1]^T$ in accordance with the Heitch-Nielson [18] scheme of initialisation. The learning factor is initialised to a high value of 0.5 and is slowly decreased as the weight vectors of the network spread out in the weight hyperspace to approximate the input vector patterns. The Kohonen algorithm implements winner takes all strategy to train the winning neuron. The criterion for victory is the induced local field or the synaptic input. The induced local field at neuron j at iteration n is defined by:

$$v_j(n) = W_j(n) \cdot X(n)$$

where $W_j(n)$ is the weight vector of the j th neuron at iteration n . This equation can be further refined as follows:

$$v_j(n) = \sum_{i=0}^p w_{ji} x_j(n) z^{-i}$$

To decide the winning neuron, the network calculates the synaptic input to each neuron and the maximum synaptic input is calculated. In case of a tie between two or more neurons, the victor is the neuron whose synaptic input was last calculated. This helps the neuron weights in spreading apart, since in the beginning, all of them are initialised to the same value. The vector approach in the weight/input hypersphere is used as weight adjustment policy as shown in Fig. 2.

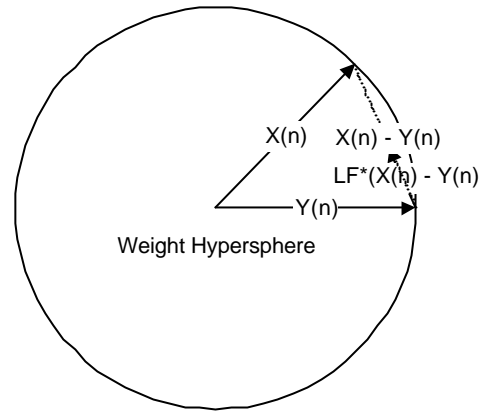


Fig. 2. Weight adjustment.

The weight adjustment is carried out only for the winning neuron in accordance with the winner takes all strategy. As seen in the figure, the weights are adjusted for the winning neuron j using the following expression:

$$\begin{aligned} W_{j(n+1)} &= W_j(n) + \mathbf{h} (X_j(n) - W_j(n)) \\ &= W_j(n) + \mathbf{h} \sum_{i=0}^p (x_j(n) z^{-i} - w_{ji}(n)) \end{aligned}$$

The weight adjustment factor \mathbf{h} is initialised to an initial value of 0.5 and is decreased in an *iterative decay* over iteration count n .

$$\mathbf{h}(n+1) = \mathbf{h}(n) - \frac{\mathbf{h}_{init} - \mathbf{h}_{final}}{n}$$

In such a decaying scheme for learning factor adjustment, the value of \mathbf{h} approaches the final value but never reaches it, resulting in a gradual learning curve.

Making predictions based on such a network would require some form of logic to evaluate the next point in the input vector. The Kohonen layer uses the concept of *curve fitting* to extrapolate the current information to calculate the future values. The curve fitting algorithm is the *slope extrapolation*

method. The algorithm assumes that the data vector for the time buffered by the short term memory at the input can be approximated by a straight line. It predicts the next point based on the equation of the line calculated from the data vector. The equation of a line can be given in the standard form as:

$$y = mx + c$$

where the parameter m describes the slope of the line and c denotes the y-intercept. The slope m is the parameter that offers the potential to the network to perform predictions. It is introduced as a state information variable in each neuron of the network. The Kohonen layer specialises the neurons into classifying different input patterns, and refines the slope parameter of the winning neuron to the slope of the input vector.

To predict the next data point, the algorithm first convolutes the input signal with the short term memory to produce the input vector of length p . Then, the input vector is processed in the unsupervised layer to find the winning neuron. The winning neuron then calculates c (y-intercept), using two points in the input vector, and predicts the next point using the straight-line equation.

3.3 Supervised Multilayer Perceptron

The supervised Multilayer Perceptron of the HTLN network consists of two hidden and one output layers. The first hidden layer has twenty neurons, and takes its input from the unsupervised Kohonen network. The output of the first hidden layer goes to the second hidden layer with 15 nodes, which in turn presents its output to the single neuron in the output layer. All the layers use unipolar sigmoidal function as the *transfer* (activation) function. The function of this network is to mine the input series for

patterns, assimilating characteristics of the mined series in terms of neuronal weights. Using the information stored in weights and the guidance factor generated by the unsupervised Kohonen network, it predicts the future behaviour of the time series.

Given an input signal $x(n)$ and p past values $x(n-1)...x(n-p)$ stored in a delay line in memory of order p , the Multilayer Perceptron adjusts the free parameters to minimise the cost function $\mathbf{x}(n)$. The first hidden layer together with the tapped delay line at the input layer forms the *temporal processing unit*. The tapped delay line memory captures the temporal information contained in the input signal and the hidden layer neuron embeds that information in its own synaptic weights. It is called a *focussed neuronal filter* because the entire memory structure is located at the input end of the network. The output of the filter is given by:

$$y(n) = \mathbf{j} \left(\sum_{l=0}^p w_j(l)x(n-l) + b_j \right)$$

where $\mathbf{j}(\cdot)$ is the activation function of the neuron j , the $w_j(l)$ are the synaptic weights, and b_j is the bias. The input to the activation function comprises the bias plus the convolution of the input sequence with the synaptic weights of the neuron.

The output of the second hidden layer is given by the following expression:

$$\begin{aligned} y(n) &= \sum_{j=0}^m w_j y_j(n) \\ &= \sum_{j=0}^m w_j \mathbf{j} \left(\sum_{l=0}^p w_j(l)x(n-l) + b_j \right) + b \end{aligned}$$

and the output of the Multilayer Perceptron is given by the expression as follows:

$$\begin{aligned}
y(n) &= \sum_{j=0}^m w_j y_j(n) \\
&= \sum_{j=0}^m w_j \mathbf{J} \quad \sum_{j=0}^m w_j \mathbf{J} \\
&\left(\sum_{l=0}^p w_j(l) x(n-l) + b_j \right) + b
\end{aligned}$$

This is the output $o(n)$ of the network. The back-propagation algorithm is used to train the Multilayer Perceptron and produce the predictions.

4. Stock Prediction Process

Neural networks need to be adapted for use with stock data for prediction. To do this, past stock data are first collected and pre-processed before being fed into the Hybrid Time Lagged Network for both training and prediction purposes. Stock data (in RM - Ringgit) from the main board of Kuala Lumpur Stock Exchange (KLSE) for the past ten years are used as the data source to the HTLN algorithm.

4.1 Data Format and Characteristics

The stock database is in Metastock file format. It contains inter-day stock data that indicates Intra-Day High, Intra-Day Low and Close prices, and the Volume of the stock ticker with the corresponding date. These are described as follows:

- Intra-Day High: This indicates the maximum price of a stock ticker during the intra-day trading.
- Intra-Day Low: This indicates the minimum price of a stock ticker during the intra-day trading.

- Close Price: This indicates the final price of a stock ticker during the intra-day trading, i.e. the price at close of the market at the end of the day.

- Volume: This indicates the number of units traded on the floor each day. It provides a measure of the volatility of the market in general and the stock ticker in particular. The higher the volume, the more volatile the market and vice-versa.

The Intra-Day High, Intra-Day Low and Close Price attributes appear to be quite independent of each other after a superficial view. However, a deeper view reveals that these three attributes are rather closely interrelated and follow each other. For example, each of these three attributes defines different financial status for the same ticker, within the same day. Hence, there is not going to be a wide variation in general (barring the cases of extreme volatility such as a market crash), and any one of these three attributes can be chosen to represent the ticker's price. The volume information is also related to the price information in general, with a greater number of units being traded on days of high market volatility, indicated by larger intra-day price fluctuations. For this research, the Close Price of a ticker is chosen to represent the price of a ticker. Hence, the time series fed into the neural network algorithm contains the close price of the ticker sorted by date.

4.2 Normalisation

Due to the chaotic nature of the data, there are no limits to the ticker price, which can shoot up to great heights or dip down to near zero level, all within a very short period of time, depending on the market volatility and company performance. This poses a great difficulty to neural network algorithm, which can get disturbed by the large fluctuations in the price, to lose the essence

of the series. Furthermore, the activation function used in the neural network algorithm is a bounded function and will not be capable of exceeding its upper bound, which the stock data can easily overshoot. This can cause inconsistencies in both the training and prediction phases, and confuse the learning mechanism.

To avoid this pitfall, all the data in the input series is normalised to a value between zero and one, which allows the neural network algorithm to comprehend the data more intelligently and make valid deductions from the input series. The data retains its inherent characteristics, but is only in more manageable bounds of the algorithm.

4.3 Time Windowing and Signal Sampling

The input is in the form of an ordered series which contains the close prices of the stock ticker to be predicted. Each unit in the series consists of the close prices of the ticker and the date identifying the temporal

position of the unit in the series [19]. The series shown in Fig. 7 illustrates the data organisation.

The column on the left of Fig. 3 is the time series that is to be fed to the neural network algorithm. Each entry in the series on the left-hand side is a time series unit that includes information about the date and the value (close price) of the ticker to be processed. The neural network algorithm does not process the full series at one go. It processes the series in a sequence of patterns [20], the length is being determined by the *Window Size* of the time series. The window size determines the amount of time series visible to the neural network algorithm in one processing cycle.

There are a few important considerations to be taken into account before fixing the window size, as it can have substantial effect on the performance of the algorithm. These include:

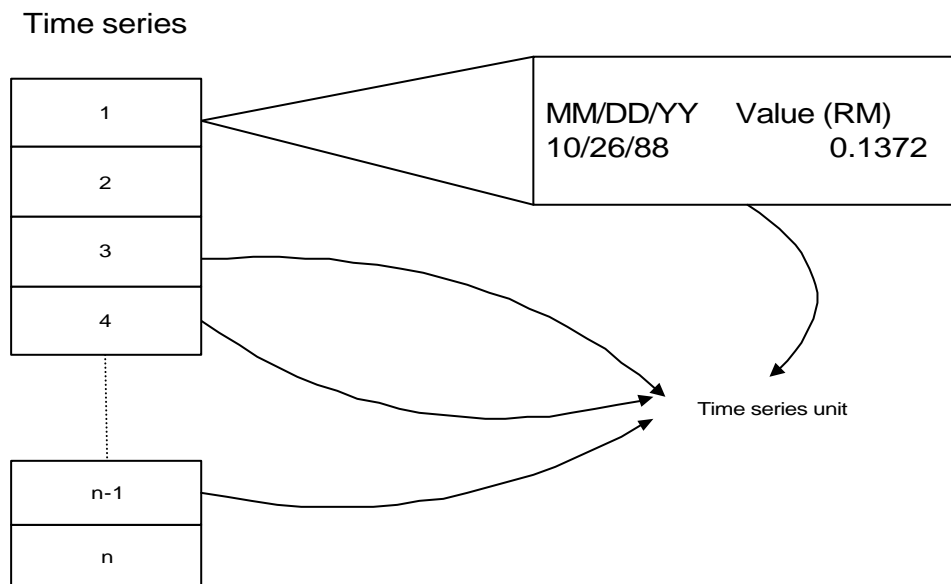


Fig. 3. Data organisation in the time series.

- The amount of data visible to the neural network algorithm directly affects the pattern recognition capability of the algorithm, as it limits the size of the patterns

that can be recognised and assimilated by the algorithm. The smaller the window size, the lesser the amount of information visible to the algorithm, and it can be confused between different patterns simply because they have the same sub-patterns within them. Having too large a window size also has its own hazards, as there can be more than one pattern within a single window, leading to confusion in the prediction algorithm.

- Another important consideration is the significance of the financial data within a window. As mentioned earlier, the financial data is such that every unit of the time series is the close price of the stock ticker. This implies that there is one unit per day in the time series. On an average, there are five working days in a week and twenty-two days in a month. Hence, this provides us with two logical grouping sizes, according to the weekly and monthly behaviours.

- System resources pose another limitation on the size of the window that can be selected for processing. The size of the window and the network size increase the

demand on resources in a geometric progression, especially memory. The system poses memory problems if the size of the input window is increased beyond twenty.

Given all these considerations, the *window size* is arbitrarily set to twenty, which gives the best compromise between system resource limit and the logical group size. This size allows the neural network algorithm to deduce valid monthly approximate training patterns and maximise the use of system resources.

Training constitutes the first phase of the neural network algorithm. The algorithm has to be trained over the series that they are going to predict. Fig. 4 shows the price attributes of Renong on KLSE main board during 1987-97. As shown, the input series is divided into two series: the training series and testing series. The training series is used to train the network, which goes on to predict the testing series, thus attempting to predict the portion of the series previously unknown to the network.

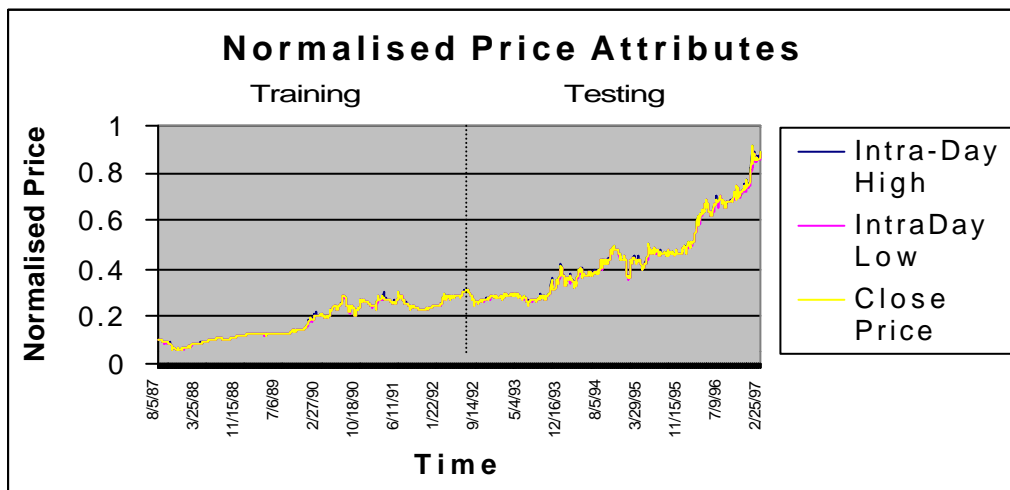


Fig. 4. Training and testing sets of the input series.

4.4 Training

The training process involves a sliding window with size of twenty data points representing twenty consecutive trading days of the stock ticker (as shown in Fig. 5). The training starts from the beginning of the series and follows the series till the midpoint. Once it reaches the midway point, training is deemed complete for one cycle. Next training cycle is then started from the beginning of the series and continues till the boundary of the training set, i.e. the midpoint of the input time series. This goes on till the neural network algorithm decides that the training has been sufficient for the input series and it can confidently predict the series further. On an average, it requires several thousand iterations before the neural network algorithm declares the training to be completed, and requires overnight training on a Pentium™ 200MHz machine.

4.5 Prediction

Once the training phase of the algorithm is completed, it is ready to predict the series on which it is trained upon. The prediction process is very similar to the training cycle, except for the evaluation and parameter adjustment steps. Prediction can be *one-point* or *m-point* in time (as shown in Fig. 6). *One-point* prediction implies that the algorithm will predict the point just after the end of the sliding window that forms the input to the algorithm. *M-point* prediction implies that the algorithm will attempt to predict m points in advance, i.e. m points after the end of the input sliding window. However, the prediction mode is dependent on the training mode, which should be the same. Hence, for *m-point* prediction, the training should have been carried out in the same mode. The algorithm is capable of performing both *one-point* and *m-point*

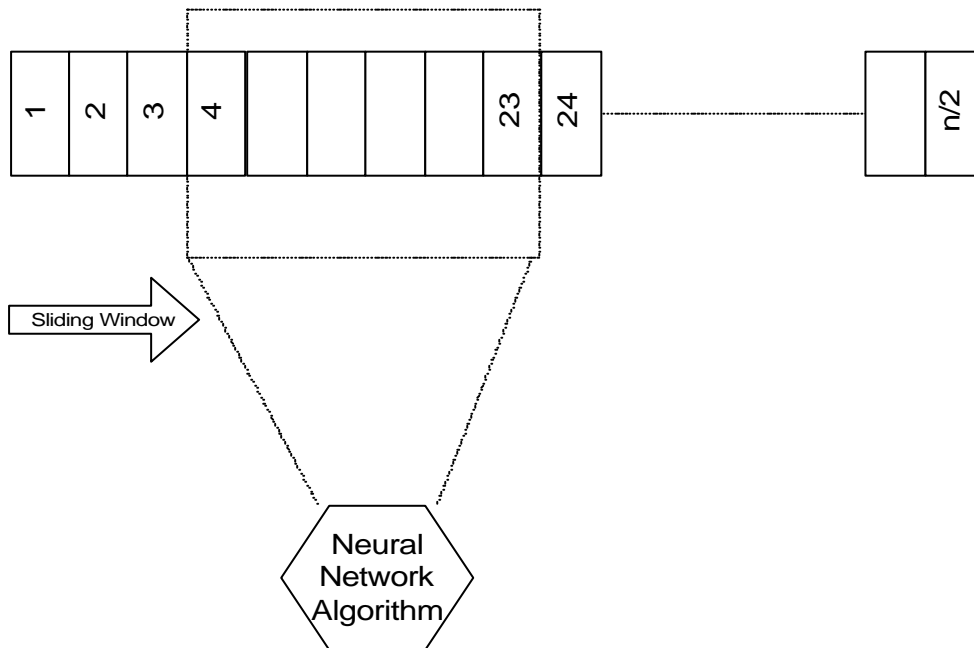


Fig. 5. The sliding window training of the neural network algorithm.

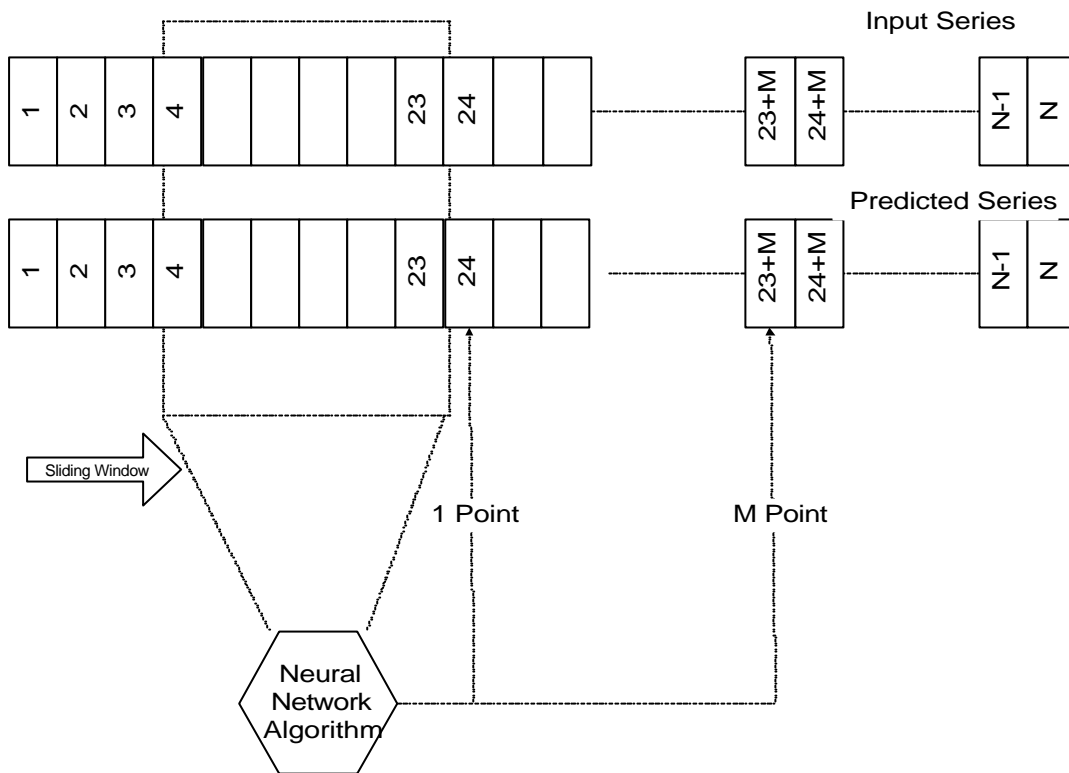


Fig. 6. One point and m-point prediction by the neural network algorithm.

predictions, although the performance varies. One-point prediction in general gives the best performance and the error factor increases geometrically with m , the degree of prediction attempted.

5. Performance Analysis

The Hybrid Time Lagged Network (HTLN) has been tested with stock series of various companies listed on the main board of the Kuala Lumpur Stock Exchange to analyse their behaviours with respect to the varying degree of chaos in the input series [21]. In addition, the performance of HTLN is compared with two standard networks for stock predictions. They are the supervised Multilayer Perceptron network known as

Time Lagged Feed-forward Network (TLFN) and unsupervised Kohonen network known as Highly Granular Unsupervised Time Lagged Network (HGUTLN) [21]. As the algorithms intend to be used by traders for stock trading, the performance analysis is carried out for the three neural network algorithms based on the stock trading performance.

In this section, the trading performance of *Diversif* (Diversified Resources Berhad) and *Carlsberg* (Carlsberg Brewery Malaysia Berhad) is used as examples to illustrate the performance of the three neural networks. The reasons for showing these two stocks over the others is the interesting nature of its stock price series including a few steep turns and lots of different turning points in the price series.

5.1 Experiment

In this experiment, an initial amount of 10,000 Ringgit (RM) is used for investment on each of the *Diversif* and *Carlsberg* stocks. The aim is to make as much profit as possible with the aid of the prediction algorithms. The trader can easily follow the decisions to buy or sell the stock. Another important factor to be considered is the decision-making policy of buying or selling the stock. This can also affect the performance of the prediction algorithms. The investment strategy used in this experiment is a watered down version of what one would expect to encounter in a real-world trading environment. The prediction algorithms instruct the trader to buy when it expects the future price to go up, i.e. the predicted price is greater than the current real price. If it expects the prices to fall in the future, it instructs the trader to sell the stock at the current high price. All these sell and buy transactions take place in a fixed quantum of one lot, to enable gradual investment and withdrawal from the market. This quantum size, however, can be modified according to the needs of the investment before running the program.

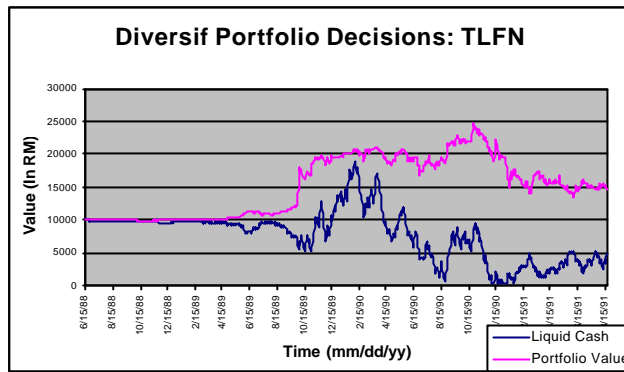
5.2 Performance Results

Fig. 7 and Fig. 8 show the performance of the TLFN, HGUTLN and HTLN networks based on the trading of *Diversif* and *Carlsberg* stocks with initial starting value of 10,000 RM. In the figures, two types of curves are shown. *Liquid Cash* indicates the cash values and *Portfolio Value* indicates the cash and stock values. As shown in Fig. 7 and Fig. 8, it can be seen that the hybrid

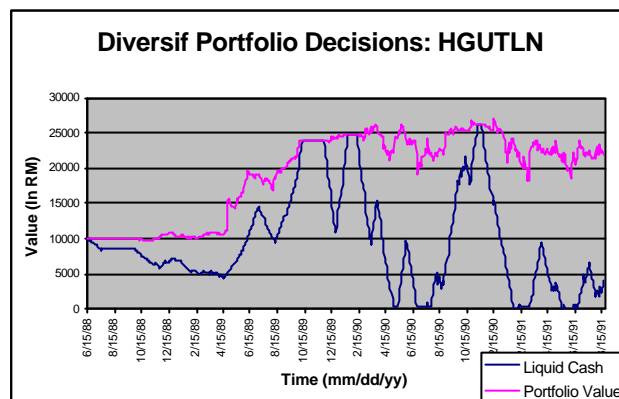
HTLN network performs the best in terms of both the prediction quality and the amount of profits generated. It gives a very stable performance and is not disturbed by the chaotic nature of data. The supervised TLFN network performs with reasonable amount of accuracy in terms of prediction. However, it can be thrown off balance if the input series is very chaotic in nature. The unsupervised HGUTLN network performs the worst of the three in terms of prediction quality. It has a lagging behaviour with respect to the input temporal series.

6. Conclusion

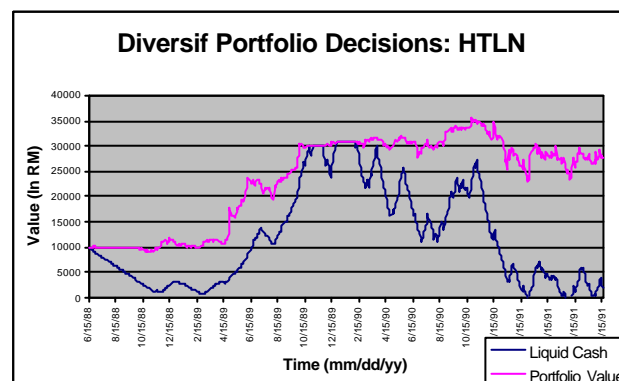
This paper has proposed the Hybrid Time Lagged Network which integrates the supervised multilayer perceptron with unsupervised Kohonen network for stock prediction. The performance of the network has been compared with two standard neural networks, namely, multilayer perceptron and Kohonen network. All these algorithms have a very good absolute performance, i.e. the quality and accuracy of prediction are very high, and proves that they can extract and model a temporal series with substantial performance. Among them, the hybrid HTLN network benefits from the stability of the unsupervised networks and the ability to recognise unlimited number of patterns from the supervised networks, making it the best performer. However, the trade-off is the speed, which is considerably slower from the other algorithms. Currently, the HTLN network is being investigated for integration with other intelligent techniques such as genetic algorithm [22] and fuzzy logic [23] for enhancing the accuracy of the prediction.



(a) Based on the TLFN network.

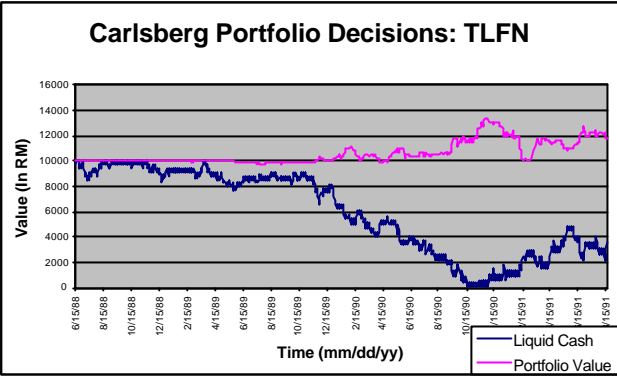


(b) Based on the HGUTLN network.

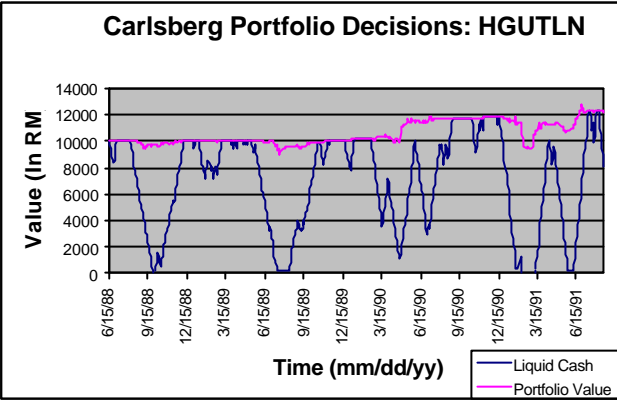


(c) Based on the HTLN network.

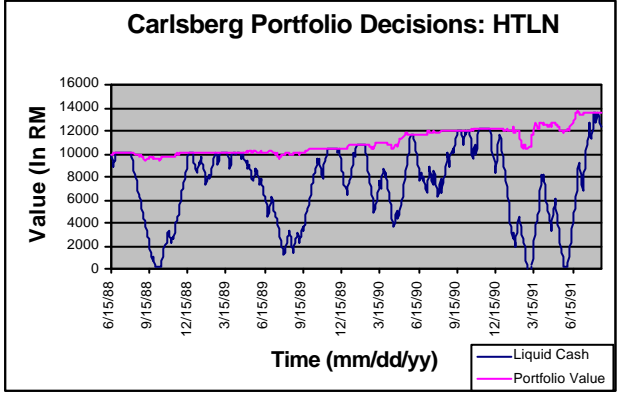
Fig. 7. Performance of *Diversif*.



(a) Based on the TLFN network.



(b) Based on the HGUTLN network.



(c) Based on the HTLN network.

Fig. 8. Performance of *Carlsberg*.

References

- [1] Phillip's On-line Electronic Mart System. Available online at URL:<http://www.poems.com.sg/>.
- [2] DBS Investor Online. Available online at URL:<http://iol.dbs.com.sg/>.
- [3] G.K. Goh. Available online at URL:<http://www.gkgoh.com.sg/>.
- [4] Murphy J.J. *Technical Analysis of Financial Markets – A Comprehensive Guide to Trading Methods and Applications*, New York Institute of Finance, 1999.
- [5] Han J., Lu H., Feng L. "Stock Movement Prediction and N-dimensional Inter-Transaction Association Rules". *Proc. of 1998 SIGMOD'96 Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'98)*, Seattle, Washington, June 1998, pp. 12:1-12:7.
- [6] John G.H., Miller P. "Stock Selection using RECON". *Neural Networks in Financial Engineering*, 1996, pp. 303-316.
- [7] John G.H., Miller P. "Stock Selection Using Rule Induction". *IEEE Expert*, October 1996, pp. 52-58.
- [8] Yoon Y., Swales G. "Predicting Stock Price Performance: A Neural Network Approach". *Proceedings of the IEEE 24th Annual Conference on System Science*, 1991, pp.156-162.
- [9] Choi J., Lee M., Rhee M. "Trading S&P 500 Stock Index Futures using a Neural Network". *Proceedings of the 3rd Annual Conference on AI Applications on WS*, 1995, pp. 63-71.
- [10] Gately E. *Neural Networks for Financial Forecasting*, New York: Wiley, 1996.
- [11] Drossu R., Obradovic Z. "Rapid Design of Neural Networks for Time Series Prediction". *IEEE Computational Science & Engineering*, 1996, pp. 78-89.
- [12] Works G.A. "Neural Network Basics". *SME Technical Paper MS89-738*, 1989.
- [13] Fu L. *Neural Networks in Computer Intelligence*, McGraw Hill, 1994.
- [14] Teuvo K. *Kohonen Self-Organising Maps*, Prentice Hall, Englewood Cliffs, NJ, 1989.
- [15] Hansen J.V., Nelson R.D. "Neural Networks and Traditional Time Series Methods: A synergistic Combination in State Economic Forecast". *IEEE Transactions on Neural Networks*, Vol. 8, No. 4, July 1997.
- [16] Austin M., Looney C., Zhuo J. "Security Market Timing using Neural Network Models". *The New Review of Applied Expert Systems*, Vol. 3, 1997, pp. 3-14.
- [17] Potts M.A.S., Brromhead D.S. "Time Series Prediction with a Radial Basis Function Neural Network. Adaptive Signal Processing", *SPIE Proceedings* Vol. 1565, 1991, pp. 255-266.

- [18] Hecht-Nielson R. *Neurocomputing*, Addison-Wesley, Reading, MA, 1990.
- [19] Han J., Gong W., Yin Y. “Mining Segment-Wise Patterns in Time Related Databases”. *Proc. of 1998 International Conference on Knowledge Discovery and Data Mining (KDD'98)*, New York City, NY, Aug. 1998, pp. 214-218.
- [20] Agarwal R., Srikant R. “Mining Sequential Patterns”. *Proceedings of the 20th Conference on Very Large Databases*, September 1994, pp. 478-499.
- [21] Prakash P. “Data Mining for Stock Selection. Technical Report” H33/99, School of Applied Science, Nanyang Technological University, 1999.
- [22] Kandal A. *Fuzzy Mathematical Techniques with Applications*, Addison Wesley, Reading, MA, 1986.
- [23] Van Rooij A.J.F., Jain L.C., Johnson R.P. *Neural Network Training using Genetic Algorithms*, World Scientific, 1996.
-