Available online at www.sciencedirect.com
ScienceDirect
Engineering Applications of Artificial Intelligence 20 (2007) 453–461
Engineering Applications of ARTIFICIAL INTELLIGENCE
www.elsevier.com/locate/engappai

Design and implementation of NN5 for Hong Kong stock price forecasting

Philip M. Tsang[a,*], Paul Kwok[a], S.O. Choy[a], Reggie Kwan[b], S.C. Ng[a], Jacky Mak[a], Jonathan Tsang[c], Kai Koong[d], Tak-Lam Wong[e]

[a]*The Open University of Hong Kong, HKSAR, China*
[b]*Caritas Francis Hui College, HKSAR, China*
[c]*Sierra College, USA*
[d]*University of Texas, Pan American, USA*
[e]*City University of Hong Kong, HKSAR, China*

Received 8 February 2006; received in revised form 25 September 2006; accepted 3 October 2006
Available online 21 December 2006

Abstract

A number of published techniques have emerged in the trading community for stock prediction tasks. Among them is neural network (NN). In this paper, the theoretical background of NNs and the backpropagation algorithm is reviewed. Subsequently, an attempt to build a stock buying/selling alert system using a backpropagation NN, NN5, is presented. The system is tested with data from one Hong Kong stock, The Hong Kong and Shanghai Banking Corporation (HSBC) Holdings. The system is shown to achieve an overall hit rate of over 70%. A number of trading strategies are discussed. A best strategy for trading non-volatile stock like HSBC is recommended.

Keywords: AI engineering application; NN5; SVM

1. Introduction

Forecasting stock prices and market indices is an important topic in finance and a good example to demonstrate the power of various AI techniques in teaching and learning. The successful prediction of prices, coupled with an appropriate trading strategy, would presumably result in substantial monetary rewards. However, stock markets are affected by many highly interrelated economic, political, and sentimental factors, which often interact with one another in a very complex manner. As such, it has always been very difficult to forecast the movements of stock prices and market indices. Indeed, in finance, the efficient market hypothesis (EMH) asserts that in an efficient market, stock price movements are random and unpredictable, which means that abnormal profiting from predicting price movements is impossible (Fama, 1965). Despite some empirical evidence supporting the EMH, there has been no consensus so far on its validity (Malkiel, 1999). Many investment professionals and market participants have met the EMH with skepticism and regard it purely as conservative academic opinion. They believe that mechanisms can be devised to predict market prices. The methods that had been used to predict market prices fall broadly into three categories—fundamental analysis, technical analysis and traditional time series forecasting.

Fundamental analysis (Ritchie, 1996) presumes that the price of a stock depends on its intrinsic value and anticipated return on investment. By analyzing the company's operations and the market in which the company is operating, it is possible to determine the company's intrinsic value and expected returns. Consequently, the stock price can be predicted reasonably well. Most people believe that fundamental analysis is a good method only on a long-term basis. However, for short- and medium-term speculations, fundamental analysis is generally not appropriate. It is also difficult to formalize fundamental analysis for automated decision support

*Corresponding author.
E-mail address: ptsang@ouhk.edu.hk (P.M. Tsang).

0952-1976/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.engappai.2006.10.002

because the interpretation of financial analysis is often highly subjective.

Technical analysis (Murphy, 1999) refers to the various methods that aim to predict future price movements using past stock prices and volume information. It is based on the assumption that history repeats itself and that future market directions can be determined by examining historical price data. As such, it is presumed that price trends and patterns exist that can be identified and exploited for profit. Most of the techniques used in technical analysis are highly subjective in nature and have been shown not to be statistically valid. Many of them even lack a rational explanation for their use (Coulson, 1987). Despite these criticisms, technical analysis is nevertheless used by about 90% of the major stock traders (Van Eyden, 1996).

Traditional time series forecasting (Box and Jenkins, 1976) techniques in statistics have also been applied to predicting stock price movements. Using techniques such as autoregressive integrated moving-average (ARIMA) or multivariate regression, it is possible to model historical price data as a nonlinear function using a recurrence relation. The derived recurrence relation is then used to forecast the future prices. A good example of using multivariate regression to predict the S&P 500 index and the Dow Jones Industrial Average (DJIA) is presented by Pesaran and Timmermann (1994). In general, time series forecasting is better suited to short-term forecasting, typically less than a year but it relies on a large amount of high-quality data.

In recent years, artificial neural networks (NNs) (Haykin, 1998) have become another important technique for predicting stock prices. A NN is an interconnected network of simple processing elements (artificial neurons) with a different weight associated with each connection. With a proper network topology and appropriate weights between the connections, a NN can be trained to approximate any function mapping between its input(s) and output(s) by using an appropriate learning algorithm such as backpropagation (BP) (Rumelhart et al., 1986a,b).

Following the premise of technical analysis that patterns exist in price data, it is possible in principle to use a NN to discover these patterns in an automated manner. Once these patterns have been discovered, future prices can be predicted. As such, numerous sources have been reported using NN for the purpose of predicting stock prices since the late 1980s (White, 1998; Kimoto et al., 1990; Yoon and Swales, 1991; Freisleben, 1992; Baestaens and van den Bergh, 1995; Yao and Poh, 1996; Yao et al., 1999) and the most commonly used networks are feed-forward BP networks (Wong, 1995).

In this paper, an empirical study on building a stock buying/selling alert system using a feed-forward BP NN (BPN) is presented. The NN, codenamed NN5, is built using the NeuralWorks Professional II/Plus software from NeuralWare. It is trained and tested with past price data from the Hong Kong and Shanghai Banking Corporation Holdings (HSBC, which has a stock ID of #5 in the HK Stock Exchange) over the period from January 2004 to December 2005. At the market opening, the system will produce a buy signal if it predicts the closing price of HSBC will be higher than the closing price of the previous trading day. Otherwise, it will produce a sell signal. The trading signals can be delivered to the user via numerous possible communication means, such as email, short message system (SMS), fax and ICQ. The objective is to determine whether NN5 can be used successfully as decision support in real world trading with a stock such as HSBC. The effect of transaction costs on the profitability of the system will also be evaluated.

2. Literature review

The study of stock prediction can be broadly divided into two schools of thought. One school focuses on computer experiments in virtual/artificial markets. This is often the case when researchers want to model the complex movements in market economics (Matsui et al., 2005). The other school focuses on stock prediction based on real-life financial data as exemplified in Zhora (2005), Yamashita et al. (2005), Wang et al. (2005) and Skabar and Cloete (2002). The study in this paper follows the latter school's approach. What differentiates this study from previous studies is that we applied the Keep It Simple and Short (KISS) principle and applied the BP algorithm of NN. As a comparison, we also subscribed to a commercial NN stock prediction service to give an indication of how our algorithm fares (Fig. 1).

2.1. BP learning and the BP network

Consider the basic structure of a feed-forward BPN with a single hidden layer as shown in Fig. 2. The network

Fig. 1. A snapshot of a commercial Online NN prediction service subscribed to by the authors as part of this experiment.

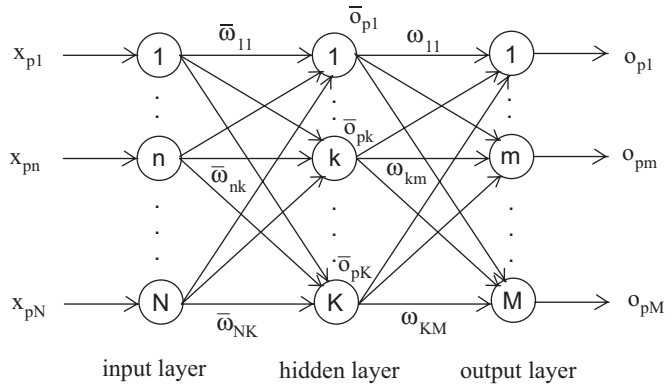

Fig. 2. Basic structure of a feedforward BPN.

consists of N input nodes, K hidden nodes and M output nodes. Let o_{pm} and $\bar{o}_{pk}$ be the output of output node m and hidden node k from input pattern p, respectively. Assume ω_{km} is the network weight for hidden node k and output node m, and $\bar{\omega}_{nk}$ is the network weight for input node n and hidden node k. Also, let x_{pn} be the input value in input node n for input pattern p, and t_{pm} be the target output value in the output node m for input pattern p. Note that the symbol Δ represents the difference between the current and the new value in the next iteration.

The standard BP algorithm is shown below:

(1) *Initialization*: Initialize all weights and refer to them as current weight $\omega_{km}(0)$ and $\bar{\omega}_{nk}(0)$. Set the learning rate μ and the momentum factor α to small positive values (e.g. 0.1). Set the error threshold E and the iteration number $i = 0$.

(2) *Forward Pass*: Select the input pattern $\boldsymbol{x}_p = \{x_{p1}, \ldots, x_{pN}\}$ from the training set and compute $o_{pm}(i)$ and $\bar{o}_{pk}(i)$ using the following equations :

$$o_{pm}(i) = f\left(\sum_{k=1}^{K} \omega_{km}(i)\, \bar{o}_{pk}(i)\right) \tag{1}$$

and

$$\bar{o}_{pk}(i) = f\left(\sum_{k=1}^{K} \bar{\omega}_{nk}(i) x_{pn}\right), \tag{2}$$

where the activation function is $f(x) = 1/(1+\mathrm{e}^{-x})$.

Use the desired target $\boldsymbol{t}_p = \{t_{p1}, \ldots, t_{pM}\}$ associated with $\boldsymbol{x}_p$ to compute the sum of the squared system error, $E(i)$, for all input patterns as follows:

$$E(i) = \frac{1}{2}\sum_{p=1}^{P}\sum_{m=1}^{M}[t_{pm} - o_{pm}(i)]^2. \tag{3}$$

If $E(i) \leqslant E$, then the algorithm is completed and the convergence is achieved; otherwise, go to step 3 (Backward Pass).

(3) *Backward Pass*: Compute the changes of the weights for the next iteration $\Delta\omega_{km}(i+1)$ and $\Delta\bar{\omega}_{nk}(i+1)$ using the following equations where μ is the learning rate and α is the momentum:

$$\begin{aligned}\Delta\omega_{km}(i+1) &= -\mu\frac{\partial E(i)}{\partial\omega_{km}(i)} + \alpha\Delta\omega_{km}(i)\\ &= \mu\sum_{p=1}^{P}\delta_{pm}(i)\bar{o}_{pk}(i) + \alpha\Delta\omega_{km}(i),\end{aligned} \tag{4}$$

$$\begin{aligned}\Delta\bar{\omega}_{nk}(i+1) &= -\mu\frac{\partial E(i)}{\partial\bar{\omega}_{nk}(i)} + \alpha\Delta\bar{\omega}_{nk}(i)\\ &= \mu\sum_{p=1}^{P}\bar{\delta}_{pk}(i)x_{pn} + \alpha\Delta\bar{\omega}_{nk}(i),\end{aligned} \tag{5}$$

where

$$\delta_{pm}(i) = (t_{pm} - o_{pm}(i))o_{pm}(i)(1 - o_{pm}(i))$$

and

$$\bar{\delta}_{pk}(i) = \bar{o}_{pk}(i)(1 - \bar{o}_{pk}(i))\sum_{m=1}^{M}\delta_{pm}(i)\omega_{km}(i).$$

Update the weights for the next iteration $\Delta\omega_{km}(i+1)$ and $\Delta\bar{\omega}_{nk}(i+1)$ and by considering the equations:

$$\omega_{km}(i+1) = \omega_{km}(i) + \Delta\omega_{km}(i+1), \tag{6}$$

$$\bar{\omega}_{nk}(i+1) = \bar{\omega}_{nk}(i) + \Delta\bar{\omega}_{nk}(i+1). \tag{7}$$

Set $i = i+1$ and go to step 2 (Forward Pass).

Due to its relative simplicity, the BPN has become the most widely used NN in recent years. There are further issues when using the BPN, including the appropriate number of hidden nodes and layers, selecting the training data, fine tuning the learning rate and momentum, avoiding overtraining, the use of threshold, and so on.

2.2. Application to the stock prediction

The major benefit of the BPN is that it is capable of learning the nonlinear mapping between the inputs and outputs by using an appropriate network topology and given a sufficient amount of training data. Once the underlying mapping has been learnt sufficiently well, the BPN can also produce an acceptable output for some unseen data. Obviously, this feature makes the BPN an attractive candidate for prediction tasks.

The major presumption of technical analysis is that history repeats itself and that trends and patterns exist in the price data. A traditional technical analyst tries to identify these patterns by examining price charts and technical indicators. If such patterns are identified, he or she can presumably predict the future price movements.

A multilayer perceptron (MLP) with a sufficient number of hidden nodes can, in principle, approximate *any* continuous nonlinear function. This means that if such

price patterns exist, a BPN can be trained to "discover" the patterns provided that:

- A sufficient amount of appropriate data is available to train the BPN.
- An appropriate network topology is selected.
- There is a sufficient number of hidden nodes.
- An appropriate learning rate and momentum are used.

These issues will be addressed in our design of NN5 and will be discussed in detail in the next section.

3. Methodology

The objective of this empirical study was to build a trading alert system (TAS) using a NN and to evaluate its hit rate (accuracy). The codename of the system is NN5. Initially, NN5 was implemented using the *NeuralWorks Professional II/Plus* software from *NeuralWare*. The software was made available for free use in the *T396 AI for Technology* course at The Open University of Hong Kong. Price history data from HSBC Holdings (HSBC) was used to train and test the NN5. The NN5 was then put into action in a computing environment such as a UNIX server.

Fig. 3 illustrates the architecture and the working of the TAS. First, a software agent is installed in the execution environment to fetch the HSBC stock price from an outside server. This software agent is scheduled to fetch the price every day right after the market opens. Second, the stock quote fetch agent feeds the NN5 the fresh price of HSBC. Third, the NN5 outputs a buy signal (+1) if it predicts that the subsequent closing price will be higher than the closing price on the previous trading day. Otherwise, NN5 outputs a sell signal (0). Fourth, another software agent is used to communicate with the user through whatever communication means is available and convenient to the user, including email, SMS, fax and instant messaging such as ICQ and MSN Messenger. The NN5 provides the trading decision message (buy signal or sell signal) for the agent to communicate with the user. Both the stock quote fetch agent and the communication agent are implemented in PHP. A UNIX *cron* job is used to schedule the fetch agent to be executed periodically.

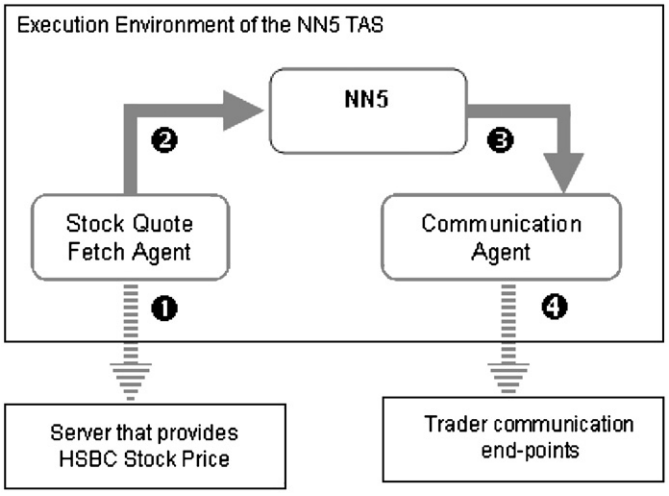

Fig. 3. The Working of the trading alert system.

In general, the following issues need to be considered during the design of a NN system:

- Defining the output goal.
- Determining the proper input data and necessary preprocessing.
- Choosing the network architecture and specifying the training algorithm.
- Determining the training and testing strategy.
- Determining the optimum network topology.
- Evaluating the results.

In the following, we discuss these issues for the specific case of NN5.

3.1. Output

A single output, which is the buy/sell trading signal as described above.

3.2. Input data

The basic input data includes raw data such as the daily open, high, low and close prices, and trading volumes of HSBC. In addition, the following may also be considered where available:

- Fundamental factors—P/E ratios and yield, among others.
- Technical indicators—n-day moving averages, RSI, MACD, and momentum, among others.
- International indices—DJIA, NASDAQ, among others.
- Interest rates—both long- and short-term; delayed data is typically used.
- Economic figures—unemployment rate, import, export, GDP growth, and the like.

Initially, eight inputs were chosen: O_i, O_{i-1}, H_{i-1}, L_{i-1}, C_{i-1}, C_{i-2}, V_i and M_{i-1} where:

O_i	the opening price of day i
O_{i-1}	the opening price of day i–1
H_{i-1}	the daily high price of day i–1
L_{i-1}	the daily low price of day i–1
C_{i-1}	the closing price of day i–1
C_{i-2}	the closing price of day i–2
V_i	the trading volume of day i
M_{i-1}	the 5-day momentum of day $i-1$ C_{i-1}–C_{i-6}

If NN5 does not converge satisfactorily with this set of inputs, then variants of the inputs can be considered. The inputs to the NN need to be scaled to fall within the range of −1 to +1. However, *NeuralWorks* can be configured to perform the scaling automatically. Therefore, it was not necessary to preprocess the data manually.

3.3. Network architecture and training algorithm

The BPN was used following the discussion in previous section. The delta rule and the sigmoid function were selected for training. The default values of the learning rate (0.35) and momentum (0.4) provided in *NeuralWorks* were adopted. If the BPN did not converge satisfactorily, then we would consider adjusting these values.

3.4. Training and testing strategy

The training and testing strategy is probably the most important issue in designing a NN. A *moving-windows* approach, as described in Skabar and Cloete (2002), was adopted for training and testing NN5.

Care must be taken in choosing the appropriate size of the training set. On one hand, if the training set is too small, NN5 will converge easily, but it will not predict with acceptable accuracy because the network is unable to learn the underlying patterns sufficiently well with such a limited amount of data. On the other hand, if the training set is too large, it will be more difficult for the network to converge. In fact, sometimes it fails to converge at all. Moreover, even if the network can converge, it may have learned some historical patterns that may no longer be effective because the market conditions have already changed too drastically.

The use of the moving-windows approach allows for the fact that the prediction model may change over time (Tsang et al., 2006a). That means the trained network that was optimal in the past may not be optimal any more (Tsang et al., 2006b).

After some preliminary experiments with the NN5 prototype, it was decided to use 24 cycles, which are listed in the following table (Table 1).

The testing period was from January 2004 to December 2005. The total number of testing patterns was 496.

Another issue in the training process concerns *over-training* of the network, which may happen when the network has been trained for too many epochs or the network is over-complex. When this happens, the network memorizes patterns and loses the ability to generalize and hence to predict. Overtraining can be avoided, or at least minimized, by monitoring the RMS error during training. When the RMS error of the training set begins to increase, overtraining has occurred. Consequently, the number of training epochs should be reduced.

In training, the *testing* data are not to be used in training. We prevented over-training by dividing the training data into training data and validation data. Training data were used to train the network. In each iteration, the trained network was applied to the validation data. After convergence, the model achieving the best performance with the validation data was chosen.

Table 1
Training and testing cycles used for NN5

Cycle	Training set		Testing set	
	Period	No. of patterns	Period	No. of patterns
1	Jul 03–Dec 03	126	Jan 04	19
2	Aug 03–Jan 04	123	Feb 04	20
3	Sept 03–Feb 04	123	Mar 04	23
4	Oct 03–Mar 04	125	Apr 04	19
5	Nov 03–Apr 04	122	May 04	20
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
18	Dec 04–May 05	121	Jun 05	22
19	Jan 05–Jun 05	121	Jul 05	20
20	Feb 05–Jul 05	120	Aug 05	23
21	Mar 05–Aug 05	126	Sept 05	21
22	Apr 05–Sept 05	126	Oct 05	20
23	May 05–Oct 05	126	Nov 05	22
24	Jun 05–Nov 05	128	Dec 05	20
	Total	2976	Total	496

3.5. Network topology

Following the Universal Approximation Theorem, it is possible to use only one hidden layer in NN5. So we started with just one hidden layer and changed to two only if necessary. Since we started with eight inputs and had only one output, the BPN of NN5 will be in the form 8-*K*-1, where *K* is the number of hidden nodes to be determined.

The BP algorithm does not provide any hint to determine the appropriate number of hidden layers and nodes. The manufacturer of *NeuralWorks* suggests the following formula for finding the total number of hidden nodes, *K*:

$$K = \frac{P}{10 \times (N + M)}, \tag{8}$$

where P is the amount of training patterns, N the number of inputs and M the number of outputs.

Using $P = 125$, $N = 8$, $M = 1$, we obtained $K = 1.39$ from Eq. (8). This result was not very useful because an 8-1-1 BPN does not seem much different from a single perceptron with 8 inputs.

Freisleben, 1992 suggested the following formula:

$$K = i \times N - 1, \tag{9}$$

where $i = 1,2,3,\ldots$

Following Eq. (9), NN5 should be of the form 8-7-1, 8-15-1, 8-23-1, and so on. In our implementation of NN5, we experimented with networks of the forms:

$$8-K-1, 8-(K\pm 1)-1, 8-(K\pm 2)-1, \ldots,$$

until an acceptable level of overall accuracy was achieved, where K is given by Eq. (9).

4. Measurement of success

In general, the performance of a NN is measured by its RMS error. During the training of NN5, we also aimed to minimize the RMS error while avoiding overtraining. However, the actual value or usefulness of NN5 is measured by its ability to make accurate predictions of future price movements.

The output of NN5 was categorized as a buy signal if it was greater than or equal to 0.5—otherwise it was categorized as a sell signal. We recorded the predictions of NN5 over the 24 months and compared them with the actual price movement directions. To be potentially useful for trading decision support, the overall hit rate (the percentage of accurate predictions) must be considerably higher than 50%. In addition to the overall hit rate, the hit rates on buying and selling signals were measured separately.

After considering the above issues, we are now ready to present the steps in designing and implementing NN5 below:

(1) Define the output and its categorization.
(2) Select the appropriate network architecture. BPN was selected primarily, but other network architectures such as RBF were also potential candidates.
(3) Determine the set of input data and preprocess the data if necessary.
(4) Select the appropriate learning algorithm and corresponding parameters. If a BPN is used, the delta rule is primarily chosen, but other choices such as ExtDBD and Quickprop are also possible. Other parameters such the learning rate and momentum can also be adjusted if necessary.
(5) Choose the appropriate network topology.
(6) Perform the training and testing for each cycle.
(7) If the network converges and produces acceptable hit rates for all cycles, go to step 8. Otherwise, go back to step 5 if there are other appropriate network topologies to try. Otherwise, go back to step 4 if there is a more appropriate learning algorithm or parameter to try. Otherwise, go back to step 3 if there is more appropriate data to add into the input data set. Otherwise, go back to step 2 to try a different kind of NN.
(8) Finish—record the results.

Another set of experiments was conducted to compare the effectiveness of NN5 with an existing work, namely, Support Vector Machine (SVM[1]) (Trafalis and Ince, 2000; Vapnik, 1995). SVM is a state-of-the-art machine learning technique that achieves very promising results in different areas such as classification, pattern reorganization and financial forecasting. We adopted the same experimental setup as described above. Instead of using NN5, we employed SVM to generate the buy and sell signals. We adopted the radial-basic function as the kernel function, which achieved the best performance for SVM in the experiments.

The empirical results are presented in the next section.

5. Results

The network that finally gave us an acceptable hit rate was an 8-14-1 BPN. The screen dump of using *NeuralWorks* to implement the network is shown in Fig. 4.

NN5 uses the delta rule of BP for learning and the sigmoidal function as the activation function. It is trained for 150,000 epochs for each training set. The default learning rate and momentum were used, which are 0.35 and 0.4 respectively. The following results were obtained (Table 2).

6. Discussion

Table 2 depicts the experimental results. Columns labeled NN5 and SVM show the results for NN5 and SVM in different runs respectively. From Table 2, it can be said that an average overall hit rate of 74.2% with a standard deviation of 11.0% is fairly good. Compared with SVM, which achieved an overall hit rate of 64.4% with a standard deviation of 10.0%, NN5 achieved more satisfactory results. For buy signals, the average hit rate of NN5 was 71.8% with a standard deviation of 18.0%, whereas SVM obtained an average hit rate of 46.2% with a standard deviation of 26.7%. NN5 is more promising and reliable than SVM. Although the average hit rate of SVM is slightly higher than NN5 in generating the sell signals, their performances are comparable. Overall, it is safe to say that NN5 can perform the specified prediction task quite well.

The implemented NN5 has shown the power of using NN, particularly the BPNs, for stock prediction tasks. Using only relatively few and simple input data and a simple BPN, the system is able to predict price movement directions fairly accurately. It is reasonable to expect that NN5 can be improved to make predictions with greater accuracy in the following ways:

- Including more selective fundamental factors, technical indicators, and broader market indices into the input data set.
- Fine-tuning the network topology and learning parameters.

[1] SVM^{light}, which is an implementation of SVM, was adopted in our experiments. SVM^{light} can be obtained from http://svmlight.joachims.org/.

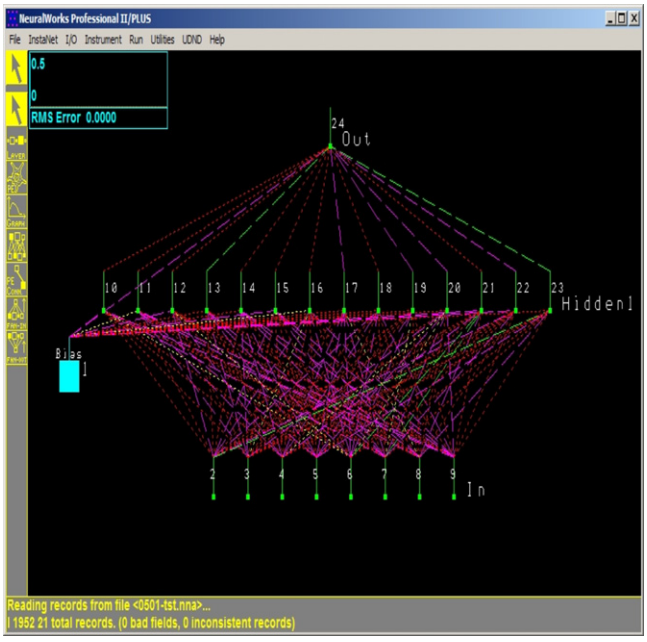

Fig. 4. NN5 implemented using an 8-14-1 BPN in *NeuralWorks*.

Table 2
Listing of the NN5 and SVM trading signal accuracies for HSBC

Trading period	Trading days	Buy hit rate		Sell hit rate		Overall hit rate	
		NN5 (%)	SVM (%)	NN5 (%)	SVM (%)	NN5 (%)	SVM (%)
Jan04	19	100.0	62.5	84.6	63.6	89.5	63.2
Feb04	20	72.7	77.8	88.9	63.6	80.0	70.0
Mar 04	23	62.5	100.0	100.0	72.2	87.0	78.3
Apr 04	19	50.0	66.7	81.8	46.2	68.4	52.6
May 04	20	58.3	22.2	75.0	81.8	65.0	55.0
Jun 04	21	75.0	77.8	76.9	75.0	76.2	76.2
July 04	21	50	16.7	76.5	86.7	71.4	66.7
Aug 04	22	70	40.0	75.0	91.7	72.7	68.2
Sep 04	21	100	0.0	65.0	100.0	66.7	61.9
Oct 04	19	100	16.7	81.3	100.0	84.2	73.7
.	.	.			.	.	
.	.	.			.	.	
.	.	.			.	.	
Aug 05	23	72.7	30.0	83.3	84.6	78.3	60.9
Sept 05	21	80.0	70.0	81.8	81.8	81.0	76.2
Oct 05	20	77.8	85.7	100.0	61.5	90.0	70.0
Nov 05	22	88.9	42.9	53.8	100.0	68.2	63.6
Dec 05	20	66.7	66.7	87.5	54.5	75.0	60.0
	496	71.8	46.2	75.5	80.1	74.2	64.4

- Exploring different kinds of networks such as RBF and SOM networks.

However, with the current level of prediction accuracy, can NN5 be used successfully as *decision support* in a real trading situation? Can the user really make profits by basing decisions *solely* on the trading signals of NN5?

In the following discussion, we assume the following specifications for simplicity:

- The user buys or sells one share of HSBC in each transaction.
- The user can buy exactly at the market opening price and closing price.
- The user can own only one share at any time.
- The user cannot sell short.
- The transaction cost of each transaction is 0.3% of the total amount. The commission rate varies and depends on the online brokerage service to which one subscribes. The choices of internet brokerage services depends on a number of factors including commission rate, efficiency, convenience and security (Laukkanen, 2006).
- The effect of a dividend will not be considered.

Before proceeding further, it is important to point out that NN5 only gives trading signals; it does not tell the user *how* to trade. The user of NN5 must first devise some trading strategy in order to benefit from the predictions of the system.

First of all, we need a baseline trading strategy to compare with the effectiveness of the trading strategies devised for NN5. Recall from the Introduction the discussion on the EMH. It was mentioned that according to the EMH, it is futile trying to make profit by predicting prices. Thus, the best strategy is to buy-and-hold. Let us use this as the baseline strategy.

Let us now consider the following trading strategies:

Strategy 1. Buy-and-hold. The user buys one share at the market opening on January 2, 2004, at the price of HK$123.0. The share is sold at the market closing on December 30, 2004, at the price of HK$124.4. The gross profit is HK$1.4 and the net profit is HK$1.03.

Strategy 2: This is a naïve strategy based directly on the output signals of NN5. The user buys one share at the market opening and sells it at the market closing on the same day, if NN5 produces a buy signal at the market opening.

Strategy 3: This is a refinement of Strategy 2 to minimize the number of transactions and hence the amount of transaction costs. If NN5 produces a buy signal at the market opening and the user does not own any share, he or she buys one share at the market-opening price. If NN5 produces a sell signal at the market opening and the user owns one share, he or she sells the share at the market open price. In this way, the user buys and holds a share until a sell signal is produced, instead of doing day-trading every time.

Table 3 shows the profits resulting from these strategies, trading over the period from January 2004 to December 2005.

It is obvious from Table 3 that transaction costs have a detrimental effect on the profitability of our trading strategies. It is also interesting to note that the baseline buy-and-hold strategy outperforms the other strategies that

Table 3
Trading results using Strategies 1, 2 and 3

	Before transaction costs	After transaction costs/at commission rate	
		0.30%	0.15%
Strategy 1	\$ 1.4	\$0.66 (gain)	\$ 1.03(gain)
Strategy 2	\$ 19.2	−\$ 111.6 (loss)	−\$ 71.1(loss)
Strategy 3	\$ 6.7	−\$ 71.2 (loss)	−\$ 45.3 (loss)

Table 4
Trading results using all the strategies mentioned

	Before transaction costs	After transaction costs at commission rate of 0.30%	After transaction costs at commission rate of 0.15%
Strategy 1	HK\$ 1.4	HK\$0.65 (gain)	HK\$ 1.03 (gain)
Strategy 2	HK\$ 19.2	−HK\$ 111.6 (loss)	−HK\$71.1 (loss)
Strategy 3	HK\$ 6.7	−HK\$ 71.2 (loss)	−HK\$ 45.3 (loss)
Strategy 4	HK\$ 91.0	−HK\$ 61.7 (loss)	−HK\$ 39.3 (loss)
Strategy 5	HK\$ 72.4	−HK\$ 25.6 (loss)	−HK\$16.3 (loss)

are based on the signals of NN5 in the real trading situation. It seems that the EMH is again valid in this case.

Now suppose we improved the hit rate of NN5 to 100%. Using this NN5-v2, we devise Strategies 4 and 5, which are exactly the same as Strategies 2 and 3 respectively, except that Strategies 4 and 5 are based on the output signals of NN5-v2.

Table 4 shows the profits resulting from all strategies mentioned, trading over the period from January 2004–December 2005.

From Table 4, it is observed that even increasing the prediction accuracy to 100% does not help to make the system profitable. This is because NN5 tries to predict price movement directions in the short term and the short-term price fluctuations are often not large enough to cover transaction costs, especially for a not very volatile stock such as HSBC. Moreover, the system produces signals depending on the market opening prices. Even if the system can achieve 100% accuracy, it still may not be profitable because the user can only buy accordingly after the market opens, and the difference between the closing price and the opening price may be smaller than the difference between the closing price and the previous closing price. It is also not uncommon that the opening price is higher than closing price, even if the closing price is higher than the previous closing price.

Our empirical results have shown that the BP network can produce reasonably accurate predictions without the need for extensive market data or knowledge. It is definitely worthwhile to pursue further development in this direction. And for non-volatile stocks like HSBC, the NN5 tends to suggest the EMH model works and it is better to take the buy-and-hold strategy.

Finally, we want to add that what differentiates this study from previous studies is that we applied the KISS principle and applied the well-known BP algorithm of NN in the search for the best strategy for stock trading.

We conclude that the KISS principle is a real asset and things do not have to be complicated. For example, when compared to the (apparently more sophisticated) SVM and commercial NN prediction software services, we note that our simple NN5 is better in terms of both cost and performance.

7. Conclusion

This paper reported an empirical study on the design and implementation of a trading alert system for HSBC holdings using a backpropagation network, and the evaluation of the usefulness of the NN5 for use as decision support in a real trading situation. The design issues of neural networks for prediction tasks were discussed in detail and the design of our trading alert system was presented. The empirical results showed that the implemented system was able to predict short-term price movement directions with an overall accuracy of over 74%. The system demonstrated that fairly good prediction accuracy can be achieved using a backpropagation network without the use of extensive market data or knowledge.

Acknowledgments

The authors wish to thank all the 2005-2006 tutors and students in the OUHK class of T396 *AI for Technology* for inspiring the write-up of this paper. Gratitude also extends to Dr. Rex Sharman, Educational Technology and Publishing Unit (ETPU), OUHK, for offering many constructive review comments.

References

Baestaens, D.E., van den Bergh, W.M., 1995. Tracking the Amsterdam Stock Index Using Neural Networks, Neural Networks in the Capital Markets. Wiley, New York (pp. 149–162, Chapter 10).

Box, G., Jenkins, G., 1976. Time Series Analysis: Forecasting and Control. Holden Day.

Coulson, D.R., 1987. The Intelligent Investor's Guide to Profiting from Stock Market Inefficiencies. Probus Publishing.

Fama, E.F., 1965. The behavior of stock market prices. Journal of Business 38, 34–105.

Freisleben, B., 1992. Stock market prediction with backpropagation networks. The Fifth International Conference Industrial and Engineering Applications of Artificial Intelligence and Expert System, Paderborn, Germany, 451–460.

Haykin, S., 1998. Neural Networks: a Comprehensive Foundation, second ed. Prentice-Hall, Englewood Cliffs, NJ.

Kimoto, T., Asakaya, K., Yoda, M., Takeoka, M., 1990. Stock market prediction system with modular neural networks. Proceedings of the IEEE International Joint Conference on Neural Networks, I1–I6.

Laukkanen, T., 2006. Customer–perceived value of e-financial services: a means-end approach. International Journal of Electronics Finance 1, 5–15.

Malkiel, B.G., 1999. A random walk down wall street, seventh ed. W.W. Norton & Company.

Matsui, H., Koyama, Y., Ishiuyama, K., 2005. A report of large-scale gaming simulation using a U-mart system in economic education. Proceedings of IEEE Third International Conference on Creating, Connecting and Collaborating through Computing, Washington, 179–184.

Murphy, J.J., 1999. Technical Analysis of the Financial Markets: a Comprehensive Guide to Trading Methods and Applications. New York Institute of Finance.

Pesaran, H.M., Timmermann, A., 1994. Forecasting stock returns: an examination of stock market trading in the presence of transaction costs. Journal of Forecasting 13, 335–367.

Ritchie, J.C., 1996. Fundamental Analysis: a Back-to-the-Basics Investment Guide to Selecting Quality Stocks. Irwin Professional Publishing.

Rumelhart, D.E., Hinton, G.E., Williams, R.J., 1986a. Learning internal representations by backpropagating errors. Nature 323 (99), 533–536.

Rumelhart, D.E., Hinton, G.E., Williams, R.J., 1986b. Learning internal representations by backpropagating errors. In: Rumelhart, D.E., McClelland, J.L., and the PDP Research Group, (Eds.), Parallel Distributed Processing, vol. 1, MIT Press, Cambridge, MA, pp. 318–362 (Chapter 8).

Skabar, A., Cloete, I., 2002. Neural networks, financial trading and the efficient market hypothesis. Australian Computer Science Communications 24 (1), 241–249.

Trafalis, T.B., Ince, H., 2000. Support vector machine for regression and applications to financial forecasting. Proceedings of IJCNN, 348–353.

Tsang, P., Ng, S.C., Ng, P., 2006a. Window-of-opportunity of HK jockey club game predication. DataVision.

Tsang, P., Ng, S.C., L Wong, T., 2006b. T396 Artificial Intelligent for Technology Project Guide OUHK 2006 Presentation.

Van Eyden, R.J., 1996. The Application of Neural Networks in the Forecasting of Share Prices. Finance and Technology Publishing.

Vapnik, V.N., 1995. The Nature of Statistical Learning Theory. Spinger, Berlin.

Wang, Y., Hung, G., Saratchandran, P., Sundararajan, N., 2005. Time series study of GGAP-RBF network: predictions of nasdaq stock and nitrate contamination of drinking water. Proceedings of the International Joint Conference on Neural Networks, Montreal, 3127–3132.

White, H., 1998. Economic prediction using neural networks: the case of the IBM daily stock returns. Proceedings of IEEE International Conference on Neural Networks, II451–II458.

Wong, B.K., 1995. A bibliography of neural network business applications research: 1988–September 1994. Expert Systems 12 (3).

Yamashita, T., Hirasawa, K., Hu, J., 2005. Application of multi-branch neural networks to stock market prediction. Proceedings of the International Joint Conference on Neural Networks, Montreal, 2544–2548.

Yao, J.T., Poh, H.-L., 1996. Equity forecasting: a case study on the klse index. Neural networks in financial engineering, Proceedings of the Third International Conference on Neural Networks in the Capital Markets. World Scientific, Singapore, pp. 341–353.

Yao, J.T., Tan, C.L., Poh, H.-L., 1999. Neural networks for technical analysis: a study on KLCI. International Journal of Theoretical and Applied Finance 2 (2), 221–241 World Scientific, Singapore.

Yoon, Y., Swales, G., 1991. Predicting stock price performance: a neural network approach. Proceedings of IEEE 24th Annual International Conference of System Sciences, 156–162.

Zhora, D., 2005. Data preprocessing for stock market forecasting using random subspace classifier network. Proceedings of the International Joint Conference on Neural Networks, Montreal, 2549–2554.