Expert Systems with Applications 38 (2011) 4741–4755

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Stock trading with cycles: A financial application of ANFIS and reinforcement learning

Zhiyong Tan[a], Chai Quek[a,*], Philip Y.K. Cheng[b]

[a] Centre for Computational Intelligence (Formerly the Intelligent Systems Laboratory), School of Computer Engineering, Nanyang Technological University, Blk N4 #2A-32, Nanyang Avenue, Singapore 639798, Singapore
[b] Australian Catholic University, Department of Business and Informatics, 7 Mount St., Sydney, Australia

ARTICLE INFO

Keywords:
Stock trading
Price cycles
Investment decisions
ANFIS
Reinforcement learning

ABSTRACT

Based on the principles of technical analysis, this paper proposes an artificial intelligence model, which employs the Adaptive Network Fuzzy Inference System (ANFIS) supplemented by the use of reinforcement learning (RL) as a non-arbitrage algorithmic trading system. The novel intelligent trading system is capable of identifying a change in a primary trend for trading and investment decisions. It dynamically determines the periods for momentum and moving averages using the RL paradigm and also appropriately shifting the cycle using ANFIS-RL to address the delay in the predicted cycle. This is used as a proxy to determine the best point in time to go LONG and visa versa for SHORT. When this is coupled with a group of stocks, we derive a simple form of "riding the cycles – waves". These are the derived features of the underlying stock movement. It provides a learning framework to trade on cycles. Initial experimental results are encouraging. Firstly, the proposed framework is able to outperform DENFIS and RSPOP in terms of true error and correlation. Secondly, based on the test trading with five US stocks, the proposed trading system is able to beat the market by about 50 percentage points over a period of 13 years.

1. Introduction

Numerous market gurus such as G. Soros, W. Henry II, and L. Hite have accumulated huge fortunes through the judicious application of technical analysis and its concepts. Despite the claims and refutes put forward by the proponents of the Efficient Market Hypothesis (Fama, 1970) and the Random Walk Hypothesis (Malkiel, 1973), technical analysis does have its place in the financial world of trading and investments. The objective of this paper is to propose an artificial intelligence model; which employs the Adaptive Network Fuzzy Inference System (ANFIS) and supplemented by reinforcement learning, as a non-arbitrage algorithmic trading system. Such a novel intelligent trading system is able to judiciously identify a change in a trend for investment decisions.

The origin of trend analysis can be attributed to the Dow Theory, named after its creator Charles Dow. Many of today's more sophisticated models are essentially variants of Dow's approach. The Dow Theory posits that there are three trends in stock prices; namely:

- The primary trend is the long-term movement of prices, lasting from several months to several years.
- The secondary or intermediate trends are the short-term deviations of prices from the underlying trend. These deviations are eliminated by corrections and prices revert back to the underlying trend.
- The tertiary or minor trends are daily fluctuations.

Our study attempts to identify the change of a primary trend or a broad movement. In our study, a primary up trend, followed by a primary down trend and a primary up trend again would constitute a cycle. The secondary and tertiary trends or short-term wave cycles are eliminated by a smoothing technique.

Artificial intelligent models based on neural networks, genetic algorithm and fuzzy neural techniques can learn to detect complex patterns inherent in the data. Mathematically, these AI techniques are universal non-linear function approximators capable of capturing and modelling almost any input–output relationships.

Artificial intelligence models also have advantages over statistical models. Relatively, they are more efficient and effective than statistical models in pattern recognition (cycles is a pattern) and prediction. Unlike most of the statistical models, they do not require any specific distribution of data or underlying theory. They are able to manage non-linear or complex relationships better and at the same time accommodate relatively larger number of variables. Another advantage of artificial intelligence models is that they can learn patterns from the past data, a feature that

* Corresponding author. Tel.: +65 67904926; fax: +65 6792 6559.
E-mail address: ashcquek@ntu.edu.sg (C. Quek).

cannot be matched by statistical models. But not all artificial intelligence models are the same. It is illustrated in this paper that, the Adaptive Network Fuzzy Inference System (ANFIS), when supplemented by the reinforcement learning paradigm, is a powerful and valuable tool in predicting stock cycles, and hence is a significant tool in stock investment decisions.

The rest of the paper is organized as follows. A brief literature review is presented in Section 2. Section 3 outlines the formalism for reinforcement learning as well as the processes of cycle determination and shifting in stock cycles. Experimental results demonstrating the derivation of stock cycles based on GM stock are also analyzed in Section 3. ANFIS-RL will be introduced to predict the next inflexion point together with a concise introduction to ANFIS in Section 4. Section 5 presents the mathematical description of a maximum reward reinforcement learning based trading system using ANFIS-RL. Extensive experiments were undertaken with dynamic asset switching based on the detection of peaks and troughs within a portfolio of stock counters and the results are presented in Section 5. A conclusion is presented in Section 6.

2. Literature review

Numerous literature employ trends to decide on the timing to trade. Research by Lee, Liu, and Chen (2006) and Kamijo and Tanigawa (1990) capitalised on candlestick patterns to trade. Lee et al. (2006) employed a knowledge based method by representing candlestick patterns with fuzzy time series while Kamijo and Tanigawa (1990) employed a recurrent neural network to recognize the triangular pattern after a series of candlesticks.

Ang and Quek (2006), Tan, Quek, and Yow (2007) Cheng, Quek, and Mah (2007) employ novel neuro-fuzzy systems in their design of such trading systems. Their trading system utilized technical indicators such as moving average crossover and Relative Strength Index. On the other hand, Kuo, Chen, and Hwang (2001), Moral-Escudero, Ruiz-Torrubiano, and Suarez (2006) and Huang, Pasquier, and Quek (2009) made use of genetic algorithms (GA) and fuzzy logic in their design of the trading systems. GA is used to determine the best set of trading rules or to address the combinatorial problem in asset allocation. Moody and Saffell (2001) and Moody and Wu (300) proposed a trading system that made use of direct reinforcement learning to learn to trade a portfolio of assets and to optimize the portfolio by making use of the Sharpe ratio and the downside deviation. There are also several research efforts devoted to stock selection, where Zargham and Sayeh (1999) used weighted fuzzy rules to select stocks based on some rule of thumb in investing and Fan and Palaniswami (2001) employed support vector machine to select stocks. These trading systems generally have subjective rules or suffer from black box characteristic or involved in unnecessary trades.

One of the classic problems, optimal option pricing had been attempted by Tsitsiklis and Roy (1999). Recent work by Quah and Quek (2005) and Quah, Quek, and Leedham (2005) proposes an actor-critic model that made use of Fuzzy Input Takagi–Sugeno–Kang (FITSK) (Quah & Quek, 2006) and maximum reward reinforcement learning to approximate the optimal stopping time, which achieved a superior performance to Tsitsiklis and Roy (1999) in terms of reward and input dimension.

There are several interesting studies on business cycles. Gallegos (2004) utilises time series theory to identify market cycles in Sweden while Plummer (2005) adopts a different approach by using the price momentum in the decision process. Generally, the values for the parameters are subjective and the cycles found are not well-defined. Lucas, Dijk, and Kloek (2002) tried out several stock selection styles and found that the best performance are derived from investment using the business cycle. Sarantis (2001) employed the STAR model (smooth transition autoregressive model) to investigate the cyclical properties for seven countries and Blackman (2004) investigated the possibility of investment based on business cycles in mutual funds. However many of these research efforts do not provide much empirical evidence and the actual applicability.

3. Cycle finding and shifting

In this section, reinforcement learning is applied to judiciously determine cycles. These cycles are fed to the proposed framework in Section 4, as training data. Cycles of stock prices are often periodic and coincidently reflect major events such as Gulf war and Great Depression (Plummer, 2005; Sarlan, 2001). Some of the earliest studies were undertaken by Jugular and Kitchin and in these studies, cycles of different length are found. A long cycle of 9–11 years was discovered by Jugular and on the other hand, a shorter cycle was found by Kitchin (1923), which is around 3–4 years. In a recent study by Plummer (2005), a relationship between the cycles had been observed. The observed relationship demonstrates that a long cycle is composed of three shorter cycles. In the case for Jugular and Kitchin, the Jugular cycle is made up of three Kitchin cycles. In this paper, in order to operate within a shorter investment period, the period of the cycle to be used in this paper is one-third of the Kitchin cycle, which is around 300 days or 1 year.

3.1. Mathematical definitions in reinforcement learning (RL)

Reinforcement learning (RL) is a trial and error learning process, whereby the agent learns through its interaction with the environment, very much akin to a form of learning (in psychology) exhibited by human in discovering new knowledge through a system of reward and punishment based on the outcome of an action undertaken by the human (Skinner, 1953). The agent does not have any idea as to what the actual correct action would achieve but it attempts to evaluate the different actions from the feedback provided by the environment. The feedback in this case is a reward and does not quantify whether the action is correct or otherwise. After several trials, the agent will have gained sufficient experience that it could exploit should it encounter similar situation.

In general, a reinforcement learning system consists of three main elements; namely: the policy, the reward function and the value function (Sutton, 1998). The policy is the mapping from the state of the environment to the probabilities of deciding its action. The state of the environment is a form of representation of the environment at that time. It can be as simple as the actual representation of the environment or it can be some form of preprocessed data, which is able to represent the current environment better. This mapping can be represented with using a lookup table or any other suitable forms of representation. This lookup table with the expected rewards and actions is the experience of the agent and will be called upon during exploitation.

In this section, a brief introduction to the mathematical formulation in RL paradigm will be provided. These formulations will be applied to all the RL applications in this paper for updating the experience during exploration and for generating the expected reward during exploitation. The reward function is the expected reward for the agent. The mathematical definition of the expected return in a state is described in Eq. (3.1)

$$Q(a) = \frac{r_1 + r_2 + r_3 + \cdots + r_k}{k}, \tag{3.1}$$

where k is the number of times action a is chosen and $r_{1\ldots k}$ are the individual reward for each time.

Eq. (3.1) can be modified as a recursive form, for easy updating and is stated in Eq. (3.2)

$$Q_{k+1} = \frac{1}{k+1}\sum_{i=1}^{k+1} r_i = \frac{1}{k+1}\left(r_{k+1} + \sum_{i=1}^{k} r_i\right) = \frac{1}{k+1}(r_{k+1} + kQ_k + Q_k - Q_k) = Q_k + \frac{1}{k+1}(r_{k+1} - Q_k). \tag{3.2}$$

The return function over the long run is usually defined as the accumulated reward over the future time steps. It is defined in Eq. (3.3)

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \tag{3.3}$$

where γ is the discount rate.

The discount rate, γ, determines the present value of the future rewards that can be achieved over time. It can be interpreted as follows:

- When the discount rate is high, the agent will take the future rewards more seriously.
- When the discount rate is small, the agent will only be concerned with the current reward.

With the Markov property, one is able to determine the probability of the next state based on only the current state and action. This is formulated in Eq. (3.4)

$$P^a_{SS'} = \Pr\{s_{t+1} = S' | s_t = S, a_t = a\}, \tag{3.4}$$

where s_t is the current state and a_t is the action taken in the current state.

With the current state, current action and the next state, one is able to compute the expected value of the next state. This is formulated in Eq. (3.5)

$$R^a_{SS'} = E\{r_{t+1} | s_t = S, a_t = a, s_{t+1} = S'\}, \tag{3.5}$$

where E is the expectation function of the next reward, s_t is the current state, s_{t+1} is the next state and a_t is the current action.

Eq. (3.3) can then be updated as follows in Eqs. (3.6) and (3.7)

$$V_t(s) = E\{R_t | s_t = s\} = E\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s\right\}, \tag{3.6}$$

where γ is the discount rate.

Under policy π,

$$V^\pi(s) = E_\pi\{R_t | s_t = s\} = E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s\right\} = E_\pi\left\{r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \middle| s_t = s\right\} = \sum_{\forall a \in A(s)} \pi(s,a) \sum P^a_{ss'}\left[R^a_{ss'} + \gamma V^\pi(s')\right], \tag{3.7}$$

where E_π is the expectation function for the policy π, a is an action that belongs to the set of actions, $P^a_{ss'}$ is the probability from s to s', and γ is the discount rate and $R^a_{ss'}$ is the future reward.

Given the current state and action taken, one can also formulate the action-value function in a recursive manner. This is formulated in Eq. (3.8).

Under policy π,

$$Q^\pi(s,a) = E_\pi\{R_t | s_t = s, a_t = a\} = E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s, a_t = a\right\} = E_\pi\{r_{t+1} + \gamma V^\pi(s') | s_t = s, a_t = a\} = \sum_{\forall s' \in S'} P^a_{ss'}\left[R^a_{ss'} + \gamma V^\pi(s')\right], \tag{3.8}$$

where E_π is the expectation function for the policy π, s' is a state that belongs to the set of states, $P^a_{ss'}$ is the probability from s to s', γ is the discount rate and $R^a_{ss'}$ is the future reward.

An optimal policy will be the policy that has the highest state-value and is defined in Eq. (3.9)

$$V^*(s) = \max_\pi V^\pi(s) \quad \text{for all } s \in S. \tag{3.9}$$

Likewise to represent an optimal state-value function, Eq. (3.7) has to be updated as shown in Eq. (3.10). This is also known as the Bellman optimality equation for V^*.

Under optimal policy,

$$V^*(s) = \max_\pi V^\pi(s) = \max_a Q^{\pi*}(s,a) = \max_a \sum_{\forall s' \in S'} P^a_{ss'}\left[R^a_{ss'} + \gamma V^*(s')\right], \tag{3.10}$$

where γ is the discount rate, s' is the next state, a is the action taken, $P^a_{ss'}$ is the probability from s to s' and $R^a_{ss'}$ is the future reward.

3.2. Formulation of rewards and states in cycle finding

This section discusses the formulation and the implementation of the tuning of the two parameters, momentum period and the moving average period using RL. The momentum or the rate of change of the stock price represents the market sentiments on the trading of the stock. As the rate of change decreases, it is assumed that the stock is overbought and vice versa. The moving average, on the other hand, is used to remove the short-term trend and noise in the cycle. The use of these two parameters promises well-defined cyclical patterns, if available.

3.2.1. Reward

The cycles used in this paper, has a period from 260 days (1 year = 52 weeks) to 340 days (1/1/4 year). These cycles are generated by tuning these two parameters; namely: momentum and moving average. The detection of trough is mathematically formulated in Eq. (3.12) using Eq. (3.11)

$$Trend_x = Cycle_t - Cycle_{t+x}, \tag{3.11}$$

where t is the current day and $x \in \{40, 80\}$

$$Trough = t\left|\begin{cases} Trend_{-40} < 0,\ Trend_{-80} < Trend_{t-40}, \\ Trend_{40} < 0,\ Trend_{80} < Trend_{t+40}, \\ Cycle_t < Cycle_{t-1},\ Cycle_{t+1} > Cycle_t, \end{cases}\right\} \tag{3.12}$$

where t is the current day.

Similarly, the detection of peak is formulated in Eq. (3.13)

$$Peak_x = t\left|\begin{cases} Trend_{-40} > 0,\ Trend_{-80} > Trend_{t-40}, \\ Trend_{40} > 0,\ Trend_{80} > Trend_{t+40}, \\ Cycle_t > Cycle_{t-1},\ Cycle_{t+1} < Cycle_t, \end{cases}\right\} \tag{3.13}$$

where t is the current day.

Using Eq. (3.12), the period of a cycle can be formulated as shown in Eq. (3.14)

$$Period_x = Trough_{x+1} - Trough_x, \tag{3.14}$$

where x is the index of *Period*.

As such, the reward for the agent to determine a single cycle will be formulated as described in Eq. (3.15)

$$r_x = \min(340 - Period_x, Period_x - 260), \tag{3.15}$$

where x is the index for *Period*.

However, Eq. (3.15) is a biased reward function. Consider the case, where the period is 300 days, the agent will receive a reward of 40. This is unjustified as any period within the range should be acceptable. To remove the bias, zero is included into Eq. (3.15).

Eq. (3.16) shows an updated equation and Figs. 3.1 and 3.2 show the reward functions for Eqs. (3.15) and (3.16) respectively. By analyzing Fig. 3.2, it can be observed that the same reward value is given to the agent for any cyclical period between 260 and 340 days

$$r_x = \min(340 - Period_x, Period_x - 260, 0). \tag{3.16}$$

Thus, the reward for the agent using a particular set of parameters is described in Eq. (3.17):

$$R(mom, ma) = \frac{1}{N}\sum_{n=1}^{n=N} r_n, \tag{3.17}$$

where N is the total number of cycles found, *mom* is the momentum period used and *ma* is the moving average period.

However, Eq. (3.17) does not assure the agent of the number of cycles found. Thus a better formulation is required and is described in Eqs. (3.18) and (3.19)

$$MinN = \frac{TotalDays - 2 * 340}{340}, \tag{3.18}$$

where *MinN* is the minimum number of cycles and *TotalDays* is the number of days in the financial times series

$$MaxN = \frac{TotalDays - 2 * 260}{260}, \tag{3.19}$$

where *MaxN* is the maximum number of cycles and *TotalDays* is the number of days in the financial times series.

Eqs. (3.18) and (3.19) are used to determine the acceptable minimum and maximum number of cycles. The reduction is needed to consider the case where there are non-cycles at the head and tail of the financial time series. Thus, the final reward formulation can be mathematically described by Eq. (3.20)

$$R(mom, ma) = \begin{cases} 260 * (N - MinN) + R(mom, ma) & N < MinN, \\ 260 * (MaxN - N) + R(mom, ma) & N > MaxN, \\ R(mom, ma) & MaxN \geqslant N \geqslant MinN, \end{cases} \tag{3.20}$$

where N is the number of cycles found, *mom* is the momentum period used, *ma* is the moving average period used, *MaxN* and *MinN* are the maximum and minimum number of cycles.

3.2.2. States

A state is a representation of the environment. In this case, the environment is the stock data and to represent it efficiently, each stock is being quantified by the percentage standard deviation of the time series, see Eq. (3.22) and the correlation to an index, see Eq. (3.21). As such, similar stocks are being grouped together and are presented as a state. There are two benefits. Firstly, the agent will be exploring (learning) more intelligently by averaging the past and new experience together. Secondly, the exploitation will be more efficient as the state-reward table is smaller

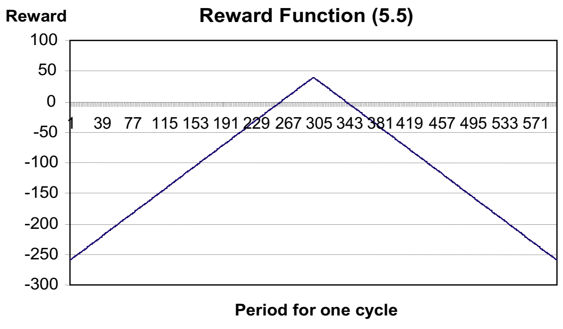

Fig. 3.1. Biased reward function.

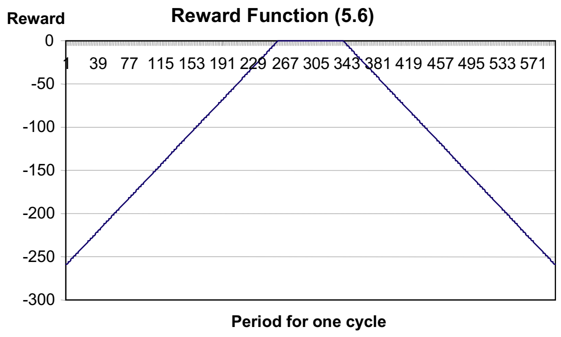

Fig. 3.2. Unbiased reward function.

$$Corr(X, Y) = \frac{\sum XY - \frac{\sum XY}{N}}{\sqrt{\left(\sum X^2 - \frac{(\sum X)^2}{N}\right)\left(\sum Y^2 - \frac{(\sum Y)^2}{N}\right)}}, \tag{3.21}$$

where X is the financial time series of the stock data, Y is the time series of the Dow Jones Index and N is the number of data

$$Percent\ StdDev = round\left\{\left[\sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2}/\bar{x}\right] \times 10\right\}, \tag{3.22}$$

where N is the number of stock data, x_i is the closing price, $\bar{x}$ is the mean of the stock price and *round*{} is to round it to the nearest integer

$$State_{Stock} = (Corr(Stock, DowJones), Percent\ StdDev). \tag{3.23}$$

Eq. (3.23) shows the formulation of the state.

3.2.3. Flow of the algorithm

The investor will attempt different combinations of the moving average period and the momentum period to achieve the respective reward. It follows the flow in Fig. 3.3.

The state of the stock is computed using Eq. (3.23). If a past experience exists, generate a random number to decide whether the agent should exploit the experience by getting the highest reward in the state-reward or to explore (learn) the new experience. To explore, the cycle data is computed with a set of parameters (moving average period and momentum period). Using the cycle data as input for Eq. (3.12), the trough is determined and is used to compute the reward using Eq. (3.20). The reward is then updated to the state table with Eq. (3.2). This exploration process will continue till all the possible combinations of the parameters have been attempted. The parameters will be bounded from 60 to 260 days. The minimum and maximum bounds are justified as our main concern is on the longer trend of the stock. The tuning of parameters is undertaken in steps of 5.

3.3. Experiments on cycle finding

3.3.1. Singapore market

In the Singapore market, the macroscopic aspect of the market is evaluated first. The indices of the various industries from the year 2000 to 2006 are used as the input data. Fig. 3.4 shows the cycle found for the hotel index and Fig. 3.5 shows the cycles found for different industries.

The 'Close' in Fig. 3.4 is the closing stock price/index value on that particular day and 'Tuned' is the cycle data identified in Section 3.2.3. These parameters are used throughout Figs. 3.7, 3.8, 3.9 and 3.10. From Fig. 3.5, it can be seen that the cycles for the different industries approximately coincides with one another.

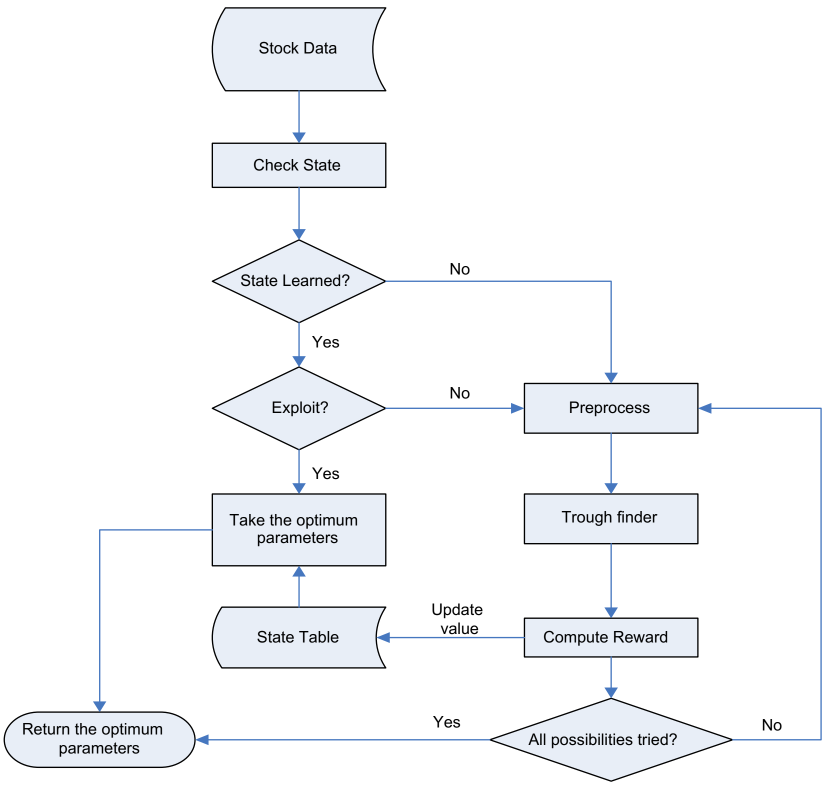

Fig. 3.3. Flowchart of the cycle tuning module as a reinforcement learning process.

However, some industries have leading cycles and others have lagging cycles. In order to apply the proposed novel trading strategy, it was hoped that uncorrelated cycles could be identified. As such, the most leading and lagging industries; namely the finance and hotel were selected in the microscopic analysis. In each industry, two stocks from the year 2000 to 2006 were selected based on their market capitalization. By doing so, it was hoped that uncorrelated cycles could be found.

Fig. 3.6 shows the four companies selected from the two industries. DBS and Kim Eng Securities are in the finance industry and Hotel Plaza and Hotel Negara are in the hotel industry. By analyzing Fig. 3.6, cycles that are completely out of sync can possibly be determined and a trader can capitalise on these cycles to buy Kim Eng Securities at the trough as indicated and sold it in the peak in year 2004 before switching to Hotel Plaza at the trough. This observation demonstrates that the proposed trading strategy is plausible and highly applicable.

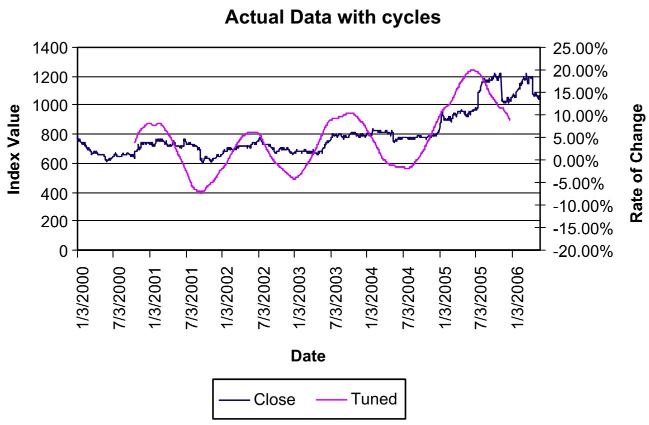

Fig. 3.4. Cycles for hotel industry.

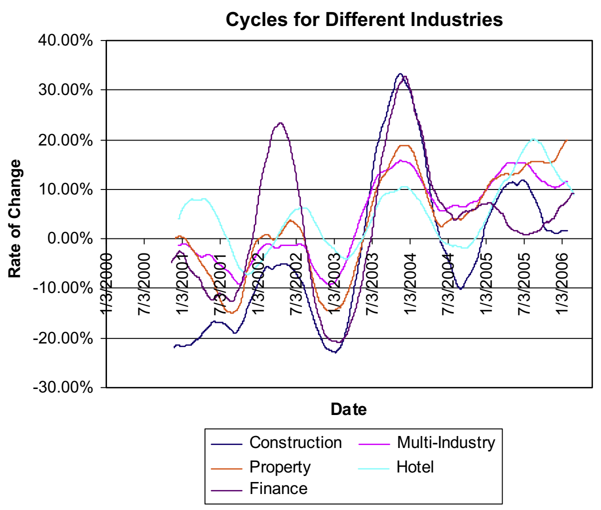

Fig. 3.5. Cycles for various industries.

3.3.2. US market

In this experiment, the applicability and scalability of the algorithm were tested on 20+ stocks from the US market. These stocks have at least 12 years of history from the year 1994 to 2006. The system will utilize the data from the first 6 years to determine the parameter values and subsequently attempts to use these sub-optimal parameters for generating the cycle for the following 6 years. These sub-optimal parameters and the cycles are then compared against the optimal parameters and the optimal cycle.

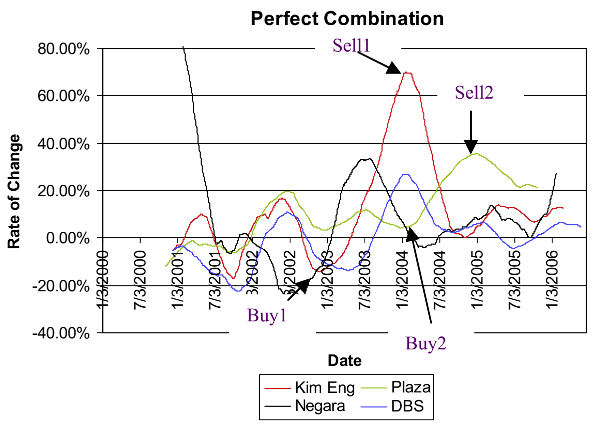

Fig. 3.6. Perfect combination for trading.

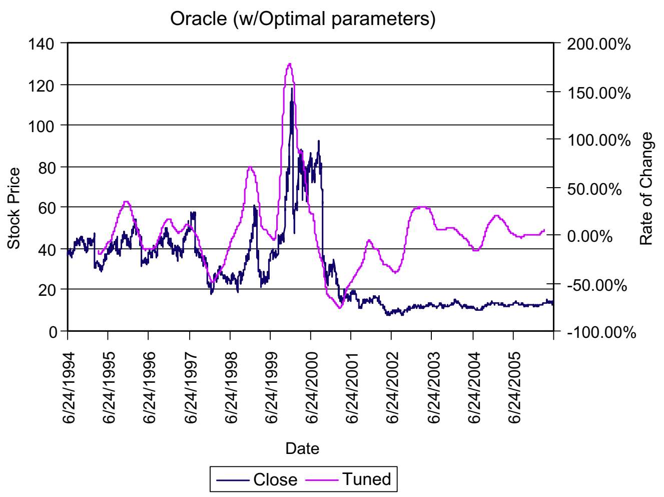

Fig. 3.9. Optimum cycles for Oracle.

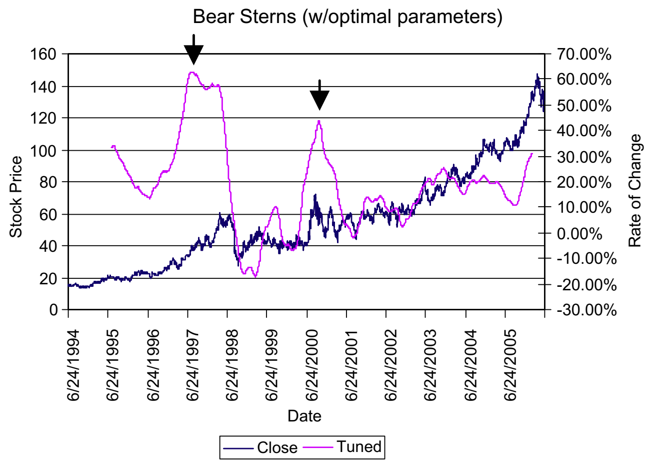

Fig. 3.7. Optimum cycles for Bear Sterns.

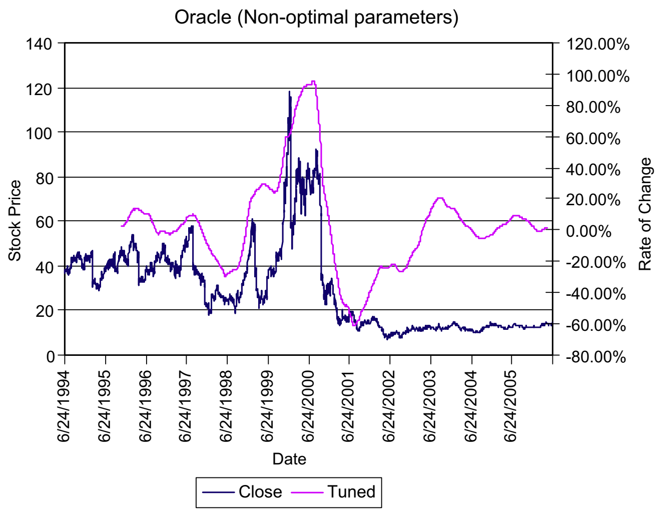

Fig. 3.10. Cycles for Oracle using sub-optimal parameters.

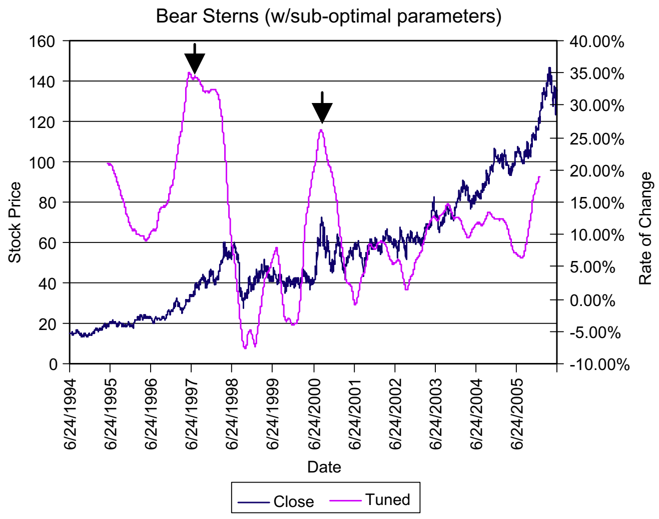

Fig. 3.8. Cycles for Bear Sterns using sub-optimal parameters.

From the experimental results, it can be observed that there are some differences between the parameters found for the 6-year and 12-year period. Most of them only differ from the optimal parameters by at most 10 days. However, there are a few stocks that have significant difference in their parameters. The analysis of two such stocks, Bear Sterns and Oracle, are shown below.

The sub-optimal parameters for Bear Sterns are 135 and 210 days for the momentum and moving average period respectively. On the other hand, the optimal parameters are 220 and 130 days for the momentum and moving average period. There is a difference of approximately 70 days for each parameter. Fig. 3.7 shows the cycle for Bear Sterns using the optimal parameters and Fig. 3.8 shows the cycle for Bear Sterns that made use of the sub-optimal parameters. Statistically, these cycles have a correlation coefficient of 0.9354 and graphically, significant inflexion points as indicated on both figures are similar to each other as indicated in the figures.

Another stock that has wide differences in the parameters is Oracle. The sub-optimal parameters for Oracle are 155 and 230 days for the momentum and moving average period respectively. On the other hand, the optimal parameters are 160 and 85 days for the momentum and moving average period. There is a difference of at most 145 days in the moving average period. Fig. 3.9 shows the cycle for Oracle using the optimum parameters and Fig. 3.10 shows the cycle for Oracle that made use of the sub-optimal parameters. These cycles have a correlation coefficient of 0.7209.

From these analyses, it can be observed that generally the parameters found in the first 6 years, can be applied to the next 6 years. But it will be advisable if the tuning can be done periodically for every 3–4 years to ensure that the cycles are well-defined.

3.4. Formulation of cycle shifting using reinforcement learning

The goal for a potential investor is to determine the shift that can fit the tuned cycles to the actual price movements. Therefore the reward for the agent will be the Pearson's linear correlation coefficient that is computed using the shifted price movement with the tuned cycles. The reward is mathematically formulated in Eq. (3.24)

$$Reward = \frac{\sum XY - \frac{\sum XY}{N}}{\sqrt{\left(\sum X^2 - \frac{(\sum X)^2}{N}\right)\left(\sum Y^2 - \frac{(\sum Y)^2}{N}\right)}}, \tag{3.24}$$

where X is the tuned cycle and Y is the actual price movement.

Fig. 3.11 shows the flow of the algorithm. Initially, the size of the entire tuned cycle will be found and this size will be the window size for the actual stock data. The window will initially end at the last stock data and will gradually be shifted to the left. On every shift, the reward (correlation) will be computed based on the window and the cycle.

The amount of shift depends on the moving average period (Gallegos, 2004) and can be mathematically formulated in Eq. (3.25). As the agent steps through, this window will be shifted by a step size of 5 days. After all iterations, the shift that achieves the highest correlation coefficient will be considered the optimal shift

$$No\ of\ iterations = \left(\frac{MAperiod}{2}\right) \Big/ 5, \tag{3.25}$$

where *MAperiod* is the moving average period.

3.5. Experiments on cycle shifting

Experiments were performed on the same basket of stocks used in Section 3.3.2. One of the stocks, General Motors, is being demonstrated and analyzed in this section.

Fig. 3.12 shows the tuned cycles for General Motors. The parameters 'Close' is the closing price of the stock on that day and 'Not Shifted' or 'Shifted' are the cycles before being shifted and after shifted respectively. From the indications in the figure, it can be observed that the peak of the identified cycle lags the peak of the actual price by around 80 days.

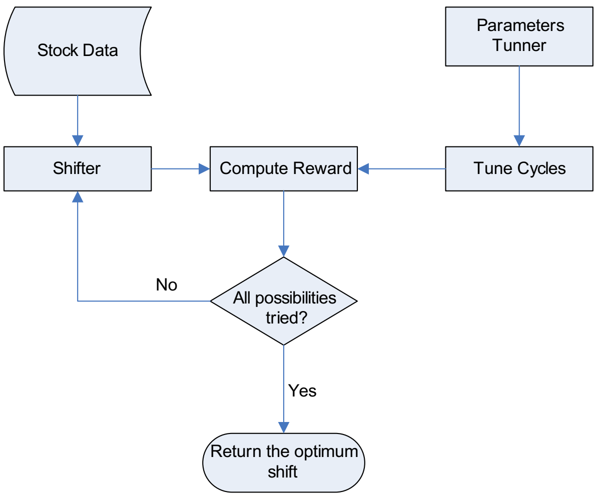

Fig. 3.11. Flow of the shift tuning algorithm.

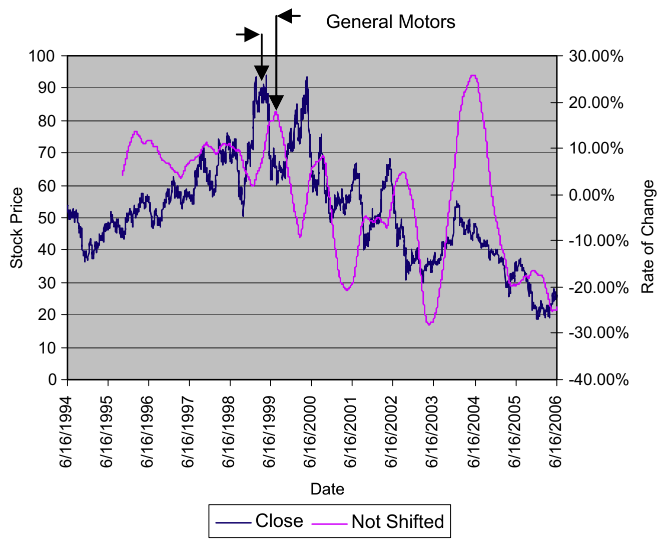

Fig. 3.12. Un-shifted cycle of General Motors.

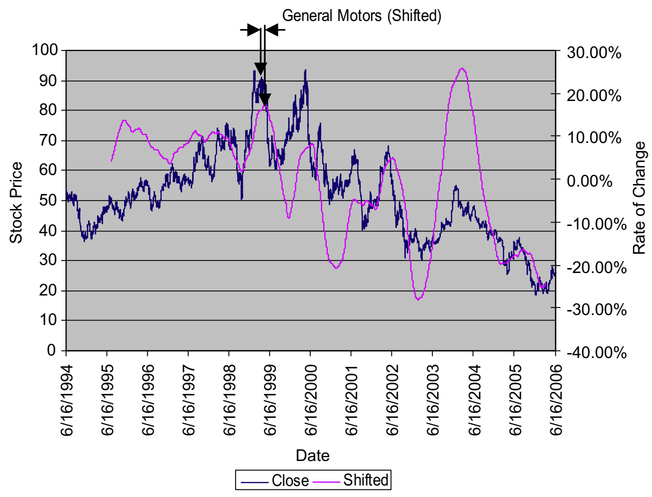

Fig. 3.13. Shifted cycle of General Motors.

Fig. 3.13 shows the cycle after the automatic shifting achieved through reinforcement learning. The automated shifting algorithm manages to identify a shift of 65 days. It can be observed that the cycle had been shifted nearer to the actual inflexion point, lagging from the actual inflexion by only 15 days. Thus, it can be observed that the shifted cycle is able to better reflect the actual turning points fairly accurately as illustrated with the indicators.

4. ANFIS-RL*

ANFIS-RL* is a framework that employs ANFIS as a function approximator. This framework attempts to optimize the performance of ANFIS in terms of accuracy and correlation. In this framework, three different forms of learning can be observed, from unsupervised learning of the data to supervised learning of the network and then finally to reinforcement learning of the different models. The function approximator will subsequently be employed in Section 5 to identify inflexion points for trading.

4.1. ANFIS

Artificial neural network is a network of artificial neurons that mimics the signaling and processing of the human brain. They have very strong pattern recognition ability but suffer from the inherent problem of being a black box, as they are not able to explain the causal relationship between the inputs and the results (outputs). On the other hand, fuzzy systems model human reasoning ability in an environment of uncertainty. These systems employed the use of fuzzy IF-THEN rules that are similar to human reasoning. These fuzzy logics are easy to comprehend and they tend to be tolerant to imprecise data. There are two main models for fuzzy systems; namely: the Mamdani and the Sugeno. Their difference lies in the way their outputs are interpreted. Mamdani model generates a fuzzy logical system that has a highly interpretable consequent; while Sugeno model generates a consequent that is a crisp function of the inputs. The latter is computationally more accurate but not as interpretable as the former.

Neuro-fuzzy systems are hybrid systems, integrating neural networks with fuzzy systems. These systems do not suffer from the limitation of a black box and is able to perform in an environment of uncertainty. Neuro-fuzzy systems such as Pseudo Outer-Product based Fuzzy Neural Network (POPFNN) (Quek & Zhou, 1999, 2006), eFSM (Tung & Quek, 2010) and SeroFNN (Javan & Quek, 2010) – are examples of Mamdani model, ANFIS (Jang, Sun, & Mizutani, 1997), MS-TSKFnn (Wang, Quek, & Ng, 2004), GA_TSKFnn (Tang, Quek, & Ng, 2005) and FITSK (Quah & Quek, 2006) – examples of Sugeno model are gaining recognition in several different areas such as finance, security and health care, as their performance and interpretability are superior to other systems such as radial basis function (RBF) networks.

The *Adaptive Neuro-Fuzzy Inference System* (ANFIS) implements the TSK fuzzy model. The inference process based on the TSK fuzzy model is shown in Fig. 4.1.

The ANFIS network has a five-layered structure as illustrated in Fig. 4.1. For the ANFIS network, the inputs and outputs are not considered part of the network structure. Moreover, the network structure is predetermined by the user prior to commence of training. The training cycle of the ANFIS network thus tunes the parameters of the network (known as *parameter learning*) but do not modify the connectionist structure of the network. In this introduction, the input nodes to the ANFIS network are denoted as L_i, where $i \in \{1 \ldots n1\}$. The label $n1$ refers to the number of inputs to the ANFIS network. The vector $X = [X_1, \ldots, X_i, \ldots, X_{n1}]^T$ denotes the numerical inputs presented to the ANFIS network. The output is denoted as f. Here, only the multiple-inputs-and-single-output (MISO) system is considered. This is because a multiple-inputs-and-multiple-outputs (MIMO) system can be readily decomposed into several MISO systems.

With reference to Fig. 4.2, layer 1 essentially consists of the linguistic terms (fuzzy sets) of the input nodes to the ANFIS network. The jth linguistic term of the ith input is denoted as $IL_{i,j}$. The label T_i denotes the number of linguistic terms that input L_i has. Each input node L_i may have different number of linguistic terms T_i. Hence, the number of nodes in layer 1 is $\sum_{i=1}^{n1} T_i$. Layer 2 of the ANFIS network is the fuzzy rule base that models the underlying characteristics of the numerical training data. The rule nodes are denoted as R_k, where $k \in \{1 \ldots n3\}$. There are $n1$ nodes (fuzzy terms) from layer 1 (one from each input variable) feeding into an arbitrary node R_k in layer 2. The label $n3$ refers to the number of fuzzy rules in the ANFIS network. Layer 3 is the *normalization layer* of the ANFIS network. There is *full connectivity* between the nodes of layer 2 and layer 3. The number of nodes in layer 3 (denoted as $n4$) is determined by the number of fuzzy rules in the ANFIS network. That is, $n4 = n3$. The functionality of each of the layer 3 nodes is to perform normalization of the firing strength of the fuzzy rule it represents. Subsequently, the computation of the rule consequents is performed at layer 4 of the ANFIS network. Since the ANFIS network adopts the TSK fuzzy model, the consequents of the rules are functions of the inputs. These functions are denoted as C_m, where $m \in \{1 \ldots n5\}$. The label $n5$ refers to the number of output functions in layer 4 and is again determined by the number of fuzzy rules. Hence, $n5 = n3$. Each output function C_m may be interpreted as Eq. (4.1):

$$C_m : (X_1, \ldots, X_i, \ldots, X_{n1}) \rightarrow \Re, \tag{4.1}$$

where X_i is the ith numerical input to the ANFIS network and $\Re$ is the set of real numbers.

Since each rule would compute an inferred output (crisp for ANFIS) based on the input stimulus $X = [X_1, \ldots, X_i, \ldots, X_{n1}]^T$, the final network output is the aggregation of all the computed inferred outputs. Hence, the function of the last ANFIS layer (layer 5) is to aggregate all the inferred outputs of the rules through summation and presents the computed value as the network output. This output is denoted as f. During the training cycle of the ANFIS network, the numerical training data set S consisting of

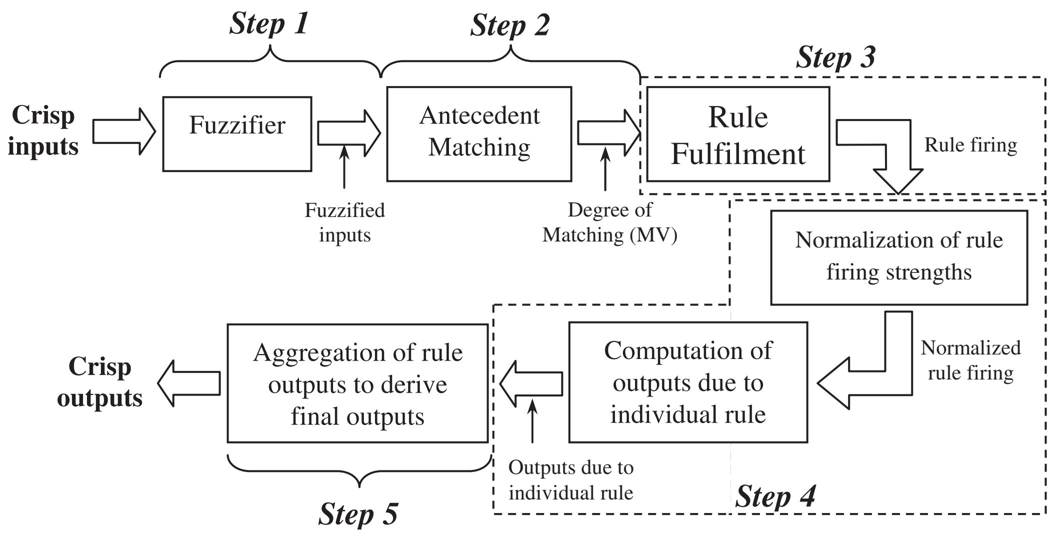

Fig. 4.1. Inference process of the TSK fuzzy model.

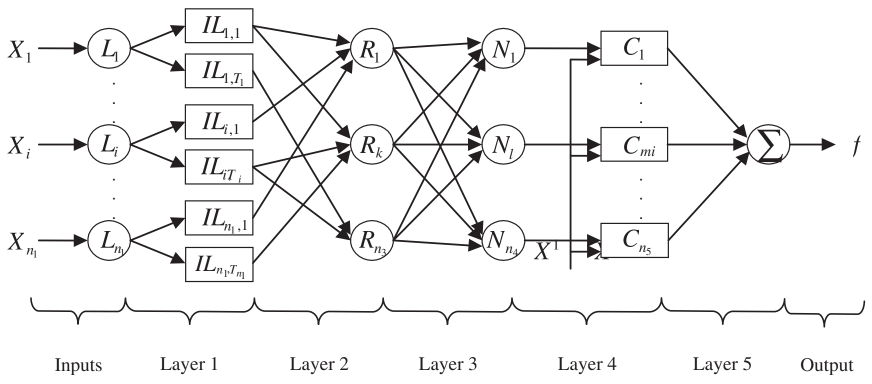

Fig. 4.2. Structure of the ANFIS network.

the desired input–output pairs $(X^{(p)}, Y^{(p)})$ (where $p \in \{1 \ldots P\}$ and P denotes the number of training instances) is fed into the ANFIS network from the input and output layers. The parameters of the ANFIS network can be subsequently tuned either using the *negative-gradient-descent*-based back-propagation algorithm (Rumelhart, Hinton, & Williams, 1986) or the hybrid learning algorithm proposed by Jang et al. (1997).

With reference to Figs. 4.1 and 4.2, one can easily deduce the correspondence between the steps of the TSK inference process and the functions of the ANFIS network. Step 1 of the TSK inference is implicitly performed by the input nodes of the ANFIS network. The input nodes functioned as *singleton fuzzifiers* to the inference process. The function of layer 1 nodes of the ANFIS network is readily mapped to step 2 of the inference process. The membership values of the numerical inputs with respect to the fuzzy sets are computed. Layer 2 of the ANFIS network implements step 3 of the inference process where the firing strengths of the fuzzy rules are determined. Step 4 of the TSK inference process is performed by the nodes in layer 3 and layer 4 of the ANFIS network. Layer 3 nodes compute the normalized fuzzy rule strengths while layer 4 derives the inferred output. Layer 5 of the ANFIS network subsequently aggregates the inferred outputs to derive the required crisp output as stated by step 5 of the TSK inference process.

ANFIS is a fuzzy inference system implemented in the framework of an adaptive network, where the membership parameters are automatically tuned. The fuzzy inference system can be generated in two ways, either by clustering or non-clustering. Clustering is a form of unsupervised learning, where unknown data is grouped into several clusters that are associated to different patterns. Using this method to generate the fuzzy inference system is able to reduce the curse of dimensionality and can easily generate the fuzzy inference system without the need to specify the membership functions. The tuning of the membership parameters will be done via supervised learning of the input–output pairs that are given to the system as training data. This employs a hybrid learning algorithm that synergizes both the back propagation method and the least squares estimate to improve the learning performance.

4.2. Design and formulation

In this paper, ANFIS is used as the function approximator to predict the gradient to the next inflexion point. Initially, a fuzzy inference system is generated via subtractive clustering. Subsequently, they undergo tuning through training of the network. Reinforcement learning will be then be applied to determine the optimum input dimension and radius of influence by evaluating the network whereby the radius of influence is a parameter for subtractive clustering of the fuzzy inference system.

Over fitting is a problem where the network is trained and evaluated using similar data. This is often due to limited data or due to situation when the validation data is coincidently similar to the training data. In these cases, it is not uncommon to have 100% correct classification on the validation data, which is overly optimistic. The model selected will be incorrect and may subsequently perform badly using out-of-sample data. In this paper, K-fold cross validation will be extended and employed in this framework. The past data will be split into K parts, with $K - 1$ parts as the training set and 1 part as the testing set. This validation is then executed K times, trying out all different parts. The advantage of this strategy is that all the past data have the chance to be trained and evaluated, giving the true error shown in Eq. (4.3) of the model

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\bar{y}_i - y_i)^2}, \tag{4.2}$$

where n is the number of outputs predicted, $\bar{y}_i$ is the predicted output and y_i is the expected output

$$TrueError = \frac{1}{K}\left(\sum_{k=1}^{K} RMSE_k\right), \tag{4.3}$$

where K is the number of folds and k is the index of each fold.

However the true error is often insufficient to gauge the performance of the model and, the correlation as shown in Eq. (3.24), is often used to complement the $RMSE$ for benchmarking of the performance. In order to adopt this measure in K-fold cross validation, Eq. (3.24) have to be updated to Eq. (4.4)

$$MeanCorr_x = \frac{1}{K}\sum_{k=1}^{K} Corr(y, \tilde{y}), \tag{4.4}$$

where x is the model used, K is the number of folds used, y is the expected value and $\tilde{y}$ is the predicted value.

Thus, a performance measure employing these two measures is proposed in this study. This is illustrated in Eq. (4.5)

$$Reward_x(R_x) = \frac{1}{(TrueError_x)} \times MeanCorr, \tag{4.5}$$

where x is the model used and $MeanCorr$ is the mean correlation coefficient.

The tuning of the two parameters will be done simultaneously to determine the model that yields the highest reward which is expressed in Eq. (4.5). After the optimum parameters are found, the entire past data set will be used to train the network, ANFIS.

Fig. 4.3 shows the flow chart of the learning process for the training and evaluation of ANFIS. The inflexion point detector is a combination of both the trough and peak finders which are expressed in Eqs. (3.12) and (3.13). With the identified inflexion points from the inflexion point detector, the inflexion point sorter module will then sort them accordingly a trough to peak, peak to trough sequence.

The tuned cycle will then be preprocessed to generate the training set. The design of the prediction model employs the past gradients of the tuned cycle as inputs. These past gradients are spaced exponentially and can be mathematically formulated as shown in Eq. (4.6)

$$Input_x = \frac{Cycle_{t-(2^x \times 5)} - Cycle_t}{(2^x \times 5)}, \tag{4.6}$$

where x is the number of input dimensions and t is the current day.

On the other hand, with the inflexion points identified in the inflexion point detector, the next valid inflexion point can be determined. The gradient from the current day to the next inflexion point can be mathematically formulated as in Eq. (4.6) and will be used as the expected output data, see Eq. (4.7)

$$Output_t = \frac{Cycle_t - Cycle_T}{T - t} \tag{4.7}$$

where $T \in \{\text{Inflexion Pts}\}$.

After the data is processed, it will be split into three parts, training, checking and testing data. ANFIS will make use of the training data to tune the membership parameters. ANFIS will then be evaluated thru reinforcement learning and the reward as expressed in Eq. (4.5) will be feedback to the framework. The parameters for the most rewarding model will then be remembered with respect to the state of the stock. Finally, this model will then be trained with 95% of the past data and the remaining 5% of the data is used for validation.

4.3. ANFIS-RL*: experimental setup

Stock data from five companies, namely, Citigroup, General Motors, Walmart, Wyeth and IBM are used to generate their cycles from 1986 to 1994. These cycles are then used as the training data in the following experiments. The number of folds, K, used for K-fold cross validation in the framework is set as 4. Lastly, the number of training episodes for all the models is 200.

4.4. Model comparison using K-fold cross validation

In this experiment, two neuro-fuzzy models, Dynamic Evolving Neural Fuzzy Inference System (DENFIS) (Kasabov & Song, 2002) and Rough Set-Based Pseudo Outer-Product (RSPOP) (Ang & Quek, 2006) are benchmarked against ANFIS_RL*. DENFIS and RSPOP are based on two different fuzzy models; namely: Sugeno and Mandami respectively. DENFIS have better prediction capability as compared to ANFIS (Jang et al., 1997) and RSPOP has better interpretability than Sugeno type model. Default parameters for DENFIS are set and RSPOP will be using three labels for the fuzzy inference system.

Table 4.1 shows the mean true error, correlation coefficient and the proposed performance measure. ANFIS-RL* has the smallest mean true error and highest mean correlation. However, considering the worse case in the true error measure, where the standard deviation is taken into consideration, it is observed that RSPOP with four input dimensions has the best performance. On the other hand, DENFIS has the highest mean correlation. However, if the overall performance measure is considered, it can be observed that ANFIS-RL* is able to consistently achieve the highest correlation to

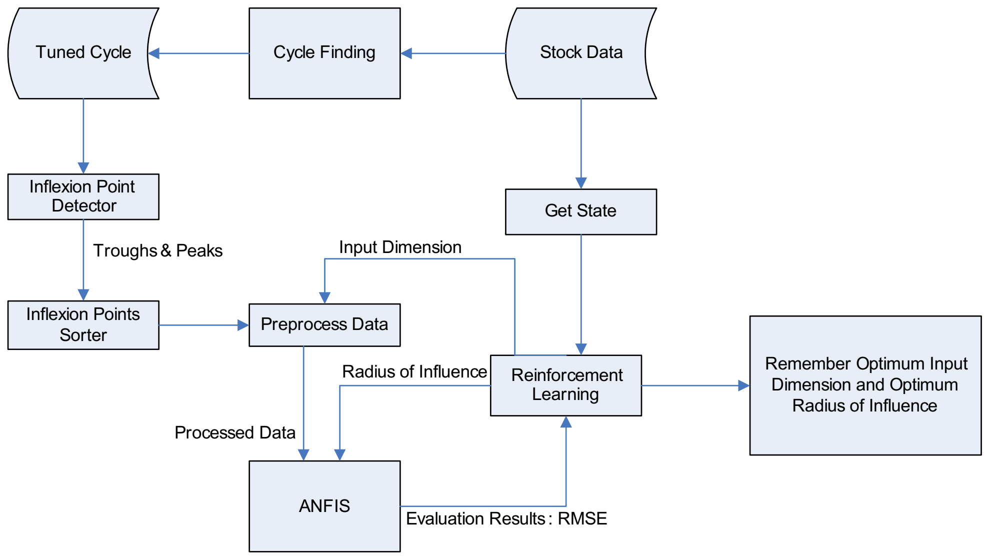

Fig. 4.3. Framework of ANFIS-RL*.

Table 4.1
Overall performance measures.

Model	Input dim.	True error		Correlation		Overall performance
		Mean	Std. deviation	Mean	Std. deviation	
DENFIS	4	0.1423	0.0406	0.6556	0.1060	4.6089
DENFIS	5	0.1598	0.0478	0.5733	0.1137	3.5884
RSPOP	4	0.1399	0.0381	0.6281	0.1533	4.4884
RSPOP	5	0.1547	0.0512	0.5945	0.1261	3.8419
ANFIS-RL*	Variable	0.1363	0.0511	0.6638	0.1078	4.8696

true error ratio. Thus, the proposed framework is able to optimize the performance of the network through reinforcement.

4.5. Prediction evaluation

In this subsection, simulation benchmark is based on the two different performance measures and the final model will then be evaluated with the entire dataset. The two different performance measures are the proposed performance measure as expressed in Eq. (4.5) – reward and Eq. (4.3) – true error a function of *RMSE*. The objective of this experiment is to determine the impact of the proposed performance measure (R_x).

From Figs. 4.4 and 4.5, it can be observed that using the proposed performance measure, ANFIS is able to provide a more stable predicted output than using the performance measure. This is illustrated in the indicated area whereby the prediction, using the proposed performance measure – (R_x), does not generate large fluctuation. Such fluctuations are intolerable in certain applications.

4.6. Detection of inflexion region

Even though, there is a prediction to the next inflexion point but one is still unable to identify an inflexion point. In this section we shall formulate the identification mathematically.

From Eq. (4.7), it can be observed that as the cycle approaches the inflexion points, the value for $Cycle_t$ will be very close to $Cycle_T$. This implies that $Output_t$ will approach near to zero when the current day is near to the inflexion point. To know whether it is a trough or peak, the trading system needs to know the past trend. We can mathematically formulate the past trend in Eq. (4.8)

$$Past\ Trend = Cycle_t - Cycle_{t-X}, \tag{4.8}$$

where t is the current day and X is a parameter value that is found via reinforcement learning.

With the past trend and the predicted gradient from ANFIS-RL*, the inflexion point can be identified via Eqs. (4.9) and (4.10)

$$(Past\ Trend < 0) \wedge (Predicted \approx 0) \rightarrow Trough, \tag{4.9}$$

$$(Past\ Trend > 0) \wedge (Predicted \approx 0) \rightarrow Peak. \tag{4.10}$$

Using Eq. 4.11, Eqs. (4.9) and (4.10) can be updated as Eqs. (4.12) and (4.13)

$$Offset \times Predicted > -Precision \wedge Offset \times Predicted < Precision \rightarrow Inflexion, \tag{4.11}$$

where *Offset* and *Precision* are parameters that are identified via reinforcement learning

$$(Past\ Trend > 0) \wedge Inflexion \rightarrow Peak, \tag{4.12}$$

$$(Past\ Trend > 0) \wedge Inflexion \rightarrow Trough. \tag{4.13}$$

The parameter, *Precision* in Eq. (4.11) marks out the region where a value will be deemed as an inflexion point. The other parameter,

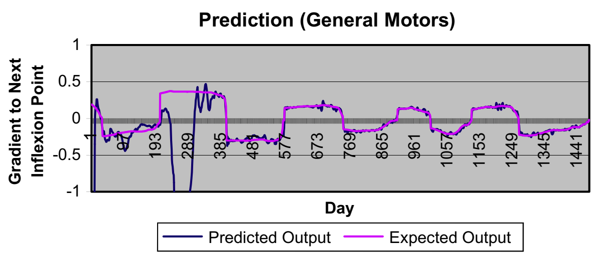

Fig. 4.4. Prediction based on true error (Eq. (4.3)) as reward.

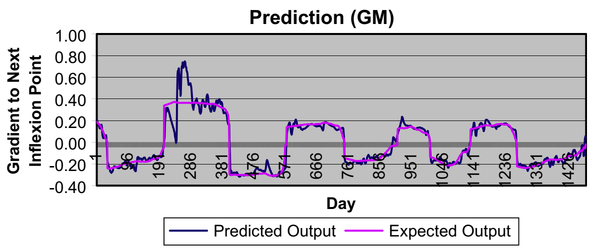

Fig. 4.5. Prediction based on proposed performance measure (R_x) (Eq. (4.5)) as reward.

Offset is a multiplicative factor to reduce the lagging effect of the tuned cycle.

5. Maximum reward reinforcement learning in trading

Maximum reward reinforcement learning is claimed in Quah and Quek (2005) to be more appropriate in applications, where the use of cumulative rewards is unjustifiable. Stock trading is one such application where the profit is non-cumulative. In addition to this, maximum reward reinforcement learning had shown superior performance in finding optimal solution without being hindered by sub-optimal solutions (Quah & Quek, 2005).

5.1. Mathematical definitions for maximum reward reinforcement learning

In this subsection, equations for reinforcement learning will be modified to implement the maximum reward reinforcement learning paradigm. The reward function,R_t, is modified as shown in Eq. (5.1)

$$R_t = \max(r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots) = \max\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}\right), \tag{5.1}$$

where γ is the discount rate.

Using Eq. (5.1), Eqs. (3.7), (3.8) and (3.10) are updated as 5.2, 5.3 and 5.4 respectively.

Under policy π,

$$\begin{aligned} V^\pi(s) &= E_\pi\{R_t|s_t = s\} = E_\pi\left\{\max_{\forall k \in [0,\infty]}(\gamma^k r_{t+k+1})\middle| s_t = s\right\} \\ &= E_\pi\left\{\max(r_{t+1}, \gamma \max_{\forall k \in [0,\infty]}(\gamma^k r_{t+k+2}))\middle| s_t = s\right\} \\ &= \sum_{\forall a \in A(s)} \pi(s,a) \sum_{\forall s' \in S'} P^a_{ss'}[\max(R^a_{ss'}, \gamma V^\pi(s'))], \end{aligned} \tag{5.2}$$

where E_π is the expectation function for the policy π, γ is the discount rate, a is an action, $P^a_{ss'}$ is the probability from s to s' and $R^a_{ss'}$ is the future reward.

Under policy π,

$$\begin{aligned} Q^\pi(s,a) &= E_\pi\{R_t|s_t = s, a_t = a\} \\ &= E_\pi\left\{\max_{\forall k \in [0,\infty]}(\gamma^k r_{t+k+1})\middle| s_t = s, a_t = a\right\} \\ &= E_\pi\left\{\max(r_{t+1}, \gamma \max_{\forall k \in [0,\infty]}(\gamma^k r_{t+k+2})\middle| s_t = s, a_t = a\right\} \\ &= \sum_{\forall s' \in S'} P^a_{ss'}[\max(R^a_{ss'}, \gamma V^\pi(s'))], \end{aligned} \tag{5.3}$$

where E_π is the expectation function for the policy π, γ is the discount rate, s' is a state that belongs to the set of states, $P^a_{ss'}$ is the probability from s to s' and $R^a_{ss'}$ is the future reward.

Under optimal policy,

$$V^*(s) = \max_\pi V^\pi(s) = \max_a Q^{\pi*}(s,a)$$
$$= \max_a \sum_{\forall s' \in S'} P^a_{ss'} \left[\max\left(R^a_{ss'}, \gamma V^*(s')\right)\right], \quad (5.4)$$

where γ is the discount rate, s' is the next state, a is the action taken, $P^a_{ss'}$ is the probability from s to s' and $R^a_{ss'}$ is the future reward.

5.2. Learning to trade

The proposed learning will be different from Moody & Saffell's (2001) in terms of concept and technique. Conceptually, the agent will learn to trade at potential states (troughs or peaks) based on actual price actions to get potential high returns. In terms of technique, the agent will learn to trade in a value based approach. This agent will be termed as Max-Reward trading agent (the "agent") subsequently.

The inexperienced agent will undergo an evaluative learning session prior to actual trade. To learn when to buy, the selling day is first fixed at the peaks of each cycle. The agent will then retrieve the state of the current day which composed of three processed components; they are computed from the historical price data. These are mathematically formulated in Eqs. (5.5) and (5.6)

$$Past\ Trend_T = round\left(\frac{price_{current} - price_{current-T}}{T} \times 10\right), \quad (5.5)$$

where $T \in \{30, 70, 90\}$.

The timing intervals, {30,70,90}, are arbitrary values to represent the past short, medium and long term trend. As the gradient is a continuous value, it will be rounded to the nearest discrete value. Since the state is represented by three components, it can be expressed as in Eq. (5.6)

$$State_{current} = (Past\ Trend_{30}, Past\ Trend_{70}, Past\ Trend_{90}). \quad (5.6)$$

With knowledge of the current state, the agent will buy on that day and sell at the next peak. The agent will then evaluate his action by using the multiplicative profit (Moody & Saffell, 2001) as a form of reward. The reward formulation is shown in Eqs. (5.7) and (5.8)

$$\text{Multiplicative Profit} = (W_t - W_o)/W_o, \quad (5.7)$$

where W_t is the current wealth and W_o is the initial wealth.

The wealth of the trading system is formulated in Eq. (5.8)

$$W_t = W_o \prod_{t=1}^{T} \{1 + R_t\}, \quad (5.8)$$

where W_t is the current wealth, W_o is the initial wealth and

$$\{1 + R_t\} = \{1 + (z_t/z_0 - 1)\}\{1 - 2\delta\},$$

where z_0 is the initial price, z_t is the current price of the securities and δ is the transaction rate.

In Eq. (5.8), the transaction rate is fixed at 0.5%, to simulate as the brokerage charges and miscellaneous handling fees. After the reward is given, the agent will update his experience using Eq. (3.2). This trial and error learning will continue until there is no more peak for the agent to learn to buy. This process is exactly the same in learning to sell, except that the trough is used for the buying and the selling will be done on every day until there is no more peak left.

5.3. Decision making process

During trading, the agent will exploit his experience to decide on the trading signals. The agent will attempt to retrieve the current state using Eq. (5.6) and will decide on its actions. Prior to decision making, Eqs. 5.2, 5.3 and 5.4 are modified as undiscounted equations as shown in Eqs. 5.9, 5.10 and 5.11. This modification will allow the agent to take the future values seriously.

Under policy π,

$$V^\pi(s) = \sum_{\forall a \in A(s)} \pi(s,a) \sum_{\forall s' \in S'} P^a_{ss'} \left[\max\left(R^a_{ss'}, V^\pi(s')\right)\right], \quad (5.9)$$

where a is an action that belongs to the set of actions, $P^a_{ss'}$ is the probability from s to s' and $R^a_{ss'}$ is the future reward.

Under policy π,

$$Q^\pi(s,a) = \sum_{\forall s' \in S'} P^a_{ss'} \left[\max\left(R^a_{ss'}, V^\pi(s')\right)\right] \quad (5.10)$$

where s' is a state that belongs to the set of states, $P^a_{ss'}$ is the probability from s to s' and $R^a_{ss'}$ is the future reward.

Under optimal policy,

$$V^*(s) = \max_a \sum_{\forall s' \in S'} P^a_{ss'} \left[\max\left(R^a_{ss'}, V^*(s')\right)\right], \quad (5.11)$$

where s' is the next state, a is the action taken, $P^a_{ss'}$ is the probability from s to s' and $R^a_{ss'}$ is the future reward.

In the decision making process, the agent will compare the current state value and the future value. The future value can be computed using Eq. (5.11) with the number of look ahead states set as 5. If the future value is lower than the current state-value, then the agent will decide to buy, as there is no other days that has better opportunity other than the current day. On the other hand, if the reverse occurs, the agent will decide to hold. However this control is still inadequate in some cases. Consider the case where the agent reaches a state that has a reward of 5% and it is higher than the future value, then should he buy? Considering the fact that the proposed investing style is a long term investing, it will be better if the expected reward is high enough. These stopping criteria can be mathematically formulated as shown in Eq. (5.12)

$$\tilde{\tau} = \min\{t | r_t \geqslant \widetilde{V}(s_t), r_t > Y\}, \quad (5.12)$$

where t is the current time and $\widetilde{V}(s_t)$ is the expected future value and Y is the expected reward that is identified via RL.

On the other hand, the agent will decide to sell based on the current profit. Then the agent will get the expected future value by using Eq. (5.11). If the current reward is higher than the future value, the agent will decide to sell, since the current reward is higher than any expected future rewards. This optimal stopping time is formulated in Eq. (5.13)

$$\tilde{\tau} = \min\{t | r_t \geqslant \widetilde{V}(s_t)\}, \quad (5.13)$$

where t is the current time and $\widetilde{V}(s_t)$ is the expected future value.

Fig. 5.1 shows the design of the trading system. It consists of three modules, cycle tuner, ANFIS and the trading agent that are generated in Sections 3–5 respectively. The cycle tuner, in Section 3, is used to generate the cycle data. This output data is then used by ANFIS, in Section 4, to predict the next inflexion point of the cycle. The prediction is employed by the trading agent (Section 5) as an indicator that the stock pricing is now within the inflexion region. The agent will then exploit its trading experience to decide on the trading action. With the trading decision, the portfolio will be updated with the new holdings and capital.

5.4. Experimental setup

Five stocks, having 20+ years of history, from different industries are randomly selected from the US market, for trading simulation and analysis. These stocks are; namely: Citigroup (Finance), General Motors (Automotive), Wyeth (Healthcare), IBM (Information Technology) and Walmart (Variety Stores). By selecting from different industries, the stock data will be highly uncorrelated from

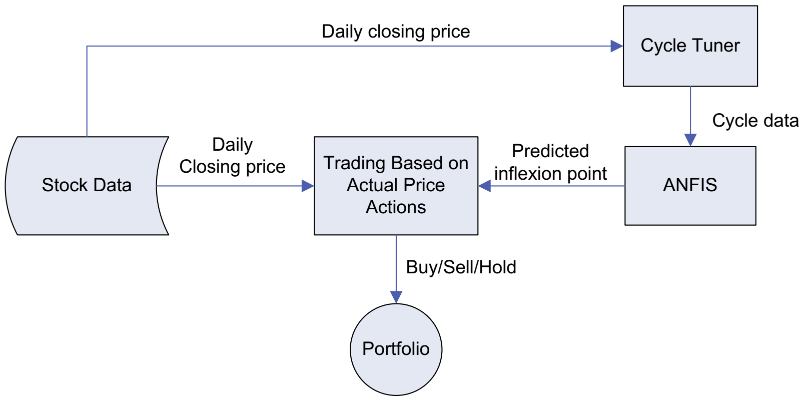

Fig. 5.1. Design of the trading system.

one another and thus will allow the proposed trading system to switch funds easily. Table 5.1 shows the correlation between the individual stocks.

In the following experiments, historical data from each stock from 1986 to 1994 was fed to the cycle finding module to determine the cycles of each stock. These cycles were then used to train the inflexion point predictor, ANFIS-RL* (see Section 4) and the Max-Reward trading agent. Reinforcement learning is then used to tune the parameters in the trading system (see Section 5). The simulations span a period of 13 years, starting from 24th August 1994 to 30th August 2006. The trading system will start off with an initial capital of US$100,000 and each buying or selling will incur a transaction cost of 0.5%.

5.5. Experimental results

Table 5.2 shows the trading performance for trading each individual stock. It shows that the trading system is able to achieve an average of 95% winning trades for the five counters. It can be observed that the average actual return over a year is around 36.91% based on the holding period. If it is based on the simulation period of 13 years, then the average yearly return is around 13.05%.

Table 5.1
Correlation ratio of stocks.

Correlation	IBM	Walmart	Citigroup	Wyeth	General Motors
IBM	1.000	0.224	−0.002	0.406	0.566
Walmart	0.224	1.000	0.320	0.273	−0.190
Citigroup	−0.002	0.320	1.000	0.028	0.179
Wyeth	0.406	0.273	0.028	1.000	0.312
General Motors	0.566	−0.190	0.179	0.312	1.000

Table 5.2
Trading performance for each stock.

Company	Profit (%)	Wining trades (%)	No. of trades	Total holding period in year	Profit (% per holding year)
IBM	212.76	75.00	4	4.86	43.73
Walmart	110.69	100.00	4	5.09	21.74
Citigroup	131.68	100.00	3	3.55	37.09
Wyeth	189.30	100.00	6	4.01	47.23
General				Motors	203.95
	6	5.87	34.75		
100.00					
Average	169.68	95.00	4.6	4.67	36.91

This analysis shows that the trading performance is not optimized yet. To optimize it, the waiting period (peak to trough) must be reduced. Furthermore, by analyzing the average return and average trades, it can be observed that for every trade, the investor can expect around 36.88% (169.68%/4.6) return.

5.5.1. Trading activity on IBM

In this section, the trading activity of one of the stocks, IBM is being demonstrated. Fig. 5.2 shows the trading activity on the counter: IBM. The parameter ‘Close’ is the closing price of the stock, ‘Trading Signal’ is the trading signal (low to high -> buy; high to low -> sell) and ‘Cycle’ is the cycle being generated as described in Section 3. Table 5.3 shows the detailed trading activity. Generally, from Fig. 5.2, it can be observed that the trading is done near the inflexion points of the generated cycle and 3 out of 4 trades are profitable. On further analysis of the losing trade, it can be observed that the trading loss is almost negligible compared to the profits made and most importantly, the system is still trading in the inflexion region. From Fig. 5.2 and Table 5.3, it can be observed that most of the 13 years is used to wait for a new buying opportunity. In Section 5.5.3, the trading system will utilize all the five stocks for trading, attempting to maximize the number of buying opportunities and thus reducing the waiting period.

5.5.2. Performance benchmark

The return from each individual stock is compared against the stock market (as proxied by the market indices) returns as well as the individual stock return, see Table 5.4. This provides an effective evaluation of the performance of the proposed long

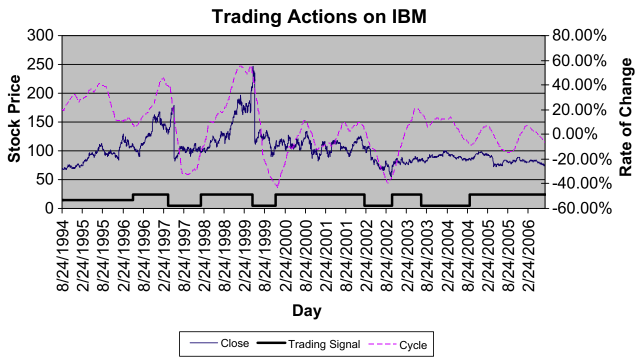

Fig. 5.2. Trading actions on IBM.

Table 5.3
Trading activities on IBM.

Date bought	Price bought	Date sold	Price sold	Profit (%)	Holding period (days)	Wealth
22-May-96	$110.37	7-April-97	$132.50	20.05	220	$118785.59
28-January-98	$97.00	4-May-99	$212.00	118.56	318	$257135.86
2-December-99	$105.27	7-February-02	$103.91	−1.29	546	$251238.79
15-October-02	$68.48	7-July-03	$86.09	25.72	181	$312743.02
17-September-04	$85.74	NA	NA	NA	NA	NA
Total profit	$212743.02					
Total profit	212.74%					

term trading method. Table 5.4 shows the return of two market indices (Dow Jones Index and the S&P 500 index) and the five stocks prices for the period from 1994 to 2006. Comparing the returns from the indices with Table 5.2, it can be observed that the average total return of trading a single stock is 8.63% lesser than the S&P Index and 26.23% lesser than Dow Jones Index. However, by comparing the yearly return of the indices with the returns per holding year in Table 5.2, it can be observed that the proposed investing style is capable of beating the market by at least around 21.84%. In addition to the analysis above, by analyzing the trading performance of a buy-and-hold strategy for the entire trading period, it can be observed that the expected yearly return of trading a stock using this strategy is −6.8%. The highest return is from IBM, having a total of 37.06%. However achieving a 37.06% return after a 13-year period is not significant as it implies only a 2.85% yearly return only. Comparing these results with Table 5.2, it is apparent that the proposed trading system is able to filter downside periods and capture the upside potential.

5.5.3. Trading with dynamic asset switching strategy

In Section 6, the trading system is traded using only one stock and hence, is unable to truly reflect the actual performance of the proposed trading strategy and system. In this section, dynamic asset switching strategy is used to switch investment funds to another stock as soon as there is a buying opportunity (trough). By doing so, the entire 13 years of trading period will be better optimized as there are more trading opportunities. Table 5.5 shows the trading record for all the trading done for this 13-year period with these five stocks. Each row shows the price and day of trade for the stock, together with the multiplicative wealth and the holding period for that particular investment.

Comparing the results between Tables 5.5 and 5.2, the total percentage profit has increased from an expected return of 167% to 240%, which is around 70% increment. It can be observed that all the trades are profitable trades. It is apparent that the improvement in the trading performance is due to the reduction in the waiting time (peak to trough) by applying dynamic asset allocation. This allows the trading system to judiciously identify more trading opportunities within the 13 years. Comparing this trading performance with the market return, it is apparent that the proposed trading system has convincingly outperformed them by around 50%.

Table 5.4
Returns on Dow Jones Index and S& P 500 Index.

Index/stock	Performance	
	Return (%)	Yearly return (%)
S&P (1994–2006)	178.31	13.72
Dow Jones (1994–2006)	195.91	15.07
IBM	37.06	2.85
Walmart	−29.87	−2.30
Citigroup	−42.51	−3.27
Wyeth	20.56	1.58
General Motors	−19.26	−1.48

6. Conclusions

To many, technical analysis is a valuable and profitable tool in trading and investment decisions. Relying on the principles of technical analysis, our study proposes a non-arbitrage trading system that is built from an optimized Adaptive Neuro-Fuzzy Inference System, (ANFIS) and supplemented by reinforcement learning. Reinforcement learning is used to formalize an automated process for determining stock cycles by tuning the momentum and the average periods. The initial experimental results based on five US stocks are promising. On average, the total returns from the five stocks are able to beat the market by about 50 percentage points.

Admittedly, more rigorous empirical testing, further refinements of the model in identifying the change in trends are required for the proposed trading system to be more reliable and acceptable. As an extension of the model, further development work would also be undertaken for portfolio composition from a given pool of stocks.

Table 5.5
Trading performance with dynamic asset allocation strategy.

Company	Day bought	Price bought	Day sold	Price sold	Wealth	Holding period (days)
General Motors	1	$50.38	696	$58.00	$113973.80	695
Wyeth	840	$75.00	922	$94.56	$142261.18	82
Wyeth	942	$49.00	1130	$59.25	$170299.70	188
Walmart	1244	$44.00	1434	$55.38	$212201.94	190
Walmart	1523	$49.75	1894	$62.98	$265946.41	371
Citigroup	1963	$40.71	2285	$46.99	$303902.07	322
Wyeth	2344	$39.65	2740	$44.85	$340320.50	396
Total profit	$240,320.50					
Total profit	240.32%					

References

Ang, K. K., & Quek, C. (2006). Stock trading using RSPOP: A novel rough set-based neuro-fuzzy approach. *IEEE Transactions on Neural Networks, 17*(5), 1301–1315.

Blackman, M. (2004). *Understanding cycles – The key to market timing*.

Cheng, P., Quek, C., & Mah, M. L. (2007). Predicting the impact of anticipatory action on US stock market – An event study using ANFIS (a neural fuzzy model). *Computational Intelligence, 23*(2), 117–141.

Fama, E. (1970). Efficient capital markets: A review of theory and empirical work. *Journal of Finance, 25*, 383–417.

Fan, A., & Palaniswami, M. (2001). Stock selection using support vector machines. In *IJCNN'01, Proceedings of the international joint conference on neural networks, Washington, DC* (Vol. 3, pp. 1793–1798).

Gallegos, A. d. l. T. (2004). *Stock market cycles in Sweden*. Universidad De Sevilla.

Huang, H. M., Pasquier, M., & Quek, C. (2009). Financial market trading system with a hierarchical co-evolutionary fuzzy predictive module. *IEEE transactions on Evolutionary Computation, 13*(1), 56–70.

Jang, J. S. R., Sun, C. T., & Mizutani, E. (1997). Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence. *IEEE Transactions on Automatic Control, 42*(10), 1482–1484.

Javan, Tan., & Quek, C. (2010). A BCM-theory of meta-plasticity for online self-reorganizing fuzzy-associative learning. *IEEE Transactions on Neural Networks., 21*(6), 985–1003.

Kamijo, K.-I., & Tanigawa, T. (1990). Stock price pattern recognition: A recurrent neural network approach. In *International joint conference on neural networks, IJCNN* (Vol. 1, pp. 215–221).

Kasabov, N., & Song, Q. (2002). DENFIS: Dynamic evolving neural-fuzzy inference system and its application for time-series prediction. *IEEE Transactions on Fuzzy Systems, 10*(2), 144–154.

Kitchin, J. (1923). Cycles and trends in economic factors. *The Review of Economic Statistics, 5*(1), 10–16.

Kuo, R. J., Chen, C. H., & Hwang, Y. C. (2001). An intelligent stock trading decision support system through integration of genetic algorithm based fuzzy neural network and artificial neural network. *Fuzzy Sets and Systems, 118*(1), 21–45.

Lee, C.-H. L., Liu, A., & Chen, W. S. (2006). Pattern discovery of fuzzy time series for financial prediction. *IEEE Transactions on Knowledge and Data Engineering, 18*(5), 613–625.

Lucas, A., Dijk, R. v., & Kloek, T. (2002). Stock selection, style rotation, and risk. *Journal of Empirical Finance, 9*, 1–34.

Malkiel, B. (1973). *A random walk down wall street*. W.W. Norton & Company Inc.

Moody, J., & Wu, L. (1997). Optimization of trading systems and portfolios. In *Proceedings of the IEEE/IAFE on computational intelligence for financial engineering (CIFEr), New York City, NY* (pp. 300–307).

Moody, J., & Saffell, M. (2001). Learning to trade via direct reinforcement. *IEEE Transactions on Neural Networks, 12*(4), 875–889.

Moral-Escudero, R., Ruiz-Torrubiano, R., & Suarez, A. (2006). Selection of optimal investment portfolios with cardinality constraints. In *IEEE congress on evolutionary computation, CEC2006, Sheraton Vancouver Wall Centre Hotel, Canada* (pp. 2382–2388).

Plummer, T. (2005). *Forecasting financial markets: the psychology of successful investing* (5th ed.). Kogan Page.

Quah, K. H., & Quek, C. (2005). Maximum reward reinforcement learning: Application to optimal stopping in financial derivative pricing. *Expert Systems with Applications, 31*(2), 351–359.

Quah, K. H., Quek, C., & Leedham, G. (2005). Reinforcement learning combined with a fuzzy adaptive learning control network (FALCON-R) for pattern classification. *Pattern Recognition, 38*, 513–526.

Quah, K. H., & Quek, C. (2006). FITSK: Online local learning with generic fuzzy input Takagi-Sugeno-Kang fuzzy framework for nonlinear system estimation. *IEEE Transactions on Systems, Man and Cybernetics, 36*(1), 166–178.

Quek, C., & Zhou, R. W. (1999). POPFNN-AARS: A pseudo outer-product based fuzzy neural network. *IEEE Transactions on Systems, Man and Cybernetics, Part B, USA, 29*(6), 859–870.

Quek, C., & Zhou, R. W. (2006). Structure and learning algorithms of a nonsingleton input fuzzy neural network based on the approximate analogical reasoning schema. *Fuzzy Sets and Systems, 157*(13), 1814–1831.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart et al. (Eds.). *Parallel distributed processing* (Vol. 1(8)). Cambridge, MA: MIT Press.

Sarantis, N. (2001). Nonlinearities cyclical behavior and predictability in stock markets: international evidence. *International Journal of Forecasting, 17*, 459–482.

Sarlan, H. (2001). Cyclical aspects of business cycle turning points. *International Journal of Forecasting, 17*, 369–382.

Skinner, B. (1953). *Science and human behavior*. New York: MacMillan.

Sutton, R. S. (1998). *Reinforcement learning: An introduction*. MIT Press.

Tang, A. M., Quek, C., & Ng, G. S. (2005). GA-TSKFNN: Parameters tuning of fuzzy neural network using genetic algorithms. *Expert Systems with Applications, 29*(4), 769–781.

Tan, A., Quek, C., & Yow, K. (2007). Maximising winning trades using a novel RSPOP fuzzy neural network intelligent stock trading system. *Applied Intelligence, 29*(2), 116–128.

Tsitsiklis, J. N., & Roy, B. V. (1999). Optimal stopping of Markov processes: Hilbert space theory, approximation algorithms, and an application to pricing high-dimensional financial derivatives. *IEEE Transactions on Automatic Control, 44*(10), 1840–1851.

Tung, W. L., & Quek, C. (2010). eFSM – A novel online neural-fuzzy semantic memory model. *IEEE Transactions in Neural Networks., 21*(1), 136–157.

Wang, D., Quek, C., & Ng, G. S. (2004). Novel self-organizing Takagi-Sugeno-Kang fuzzy neural networks based on ART-like clustering. *Neural Processing Letters, 20*(1), 39–51.

Zargham, M. R., & Sayeh, M. R. (1999). A web-based information system for stock selection and evaluation. *Advance Issues of E-Commerce and Web-Based Information Systems*, 81–83.