# A Multi-agent Q-learning Framework for Optimizing Stock Trading Systems

Jae Won Lee[1] and Jangmin O[2]

[1] School of Computer Science and Engineering, Sungshin Women's University, Seoul, Korea 136-742
jwlee@cs.sungshin.ac.kr

[2] School of Computer Engineering, Seoul National University, Seoul, Korea 151-742
jmoh@bi.snu.ac.kr

**Abstract.** This paper presents a reinforcement learning framework for stock trading systems. Trading system parameters are optimized by Q-learning algorithm and neural networks are adopted for value approximation. In this framework, cooperative multiple agents are used to efficiently integrate global trend prediction and local trading strategy for obtaining better trading performance. Agents communicate with others sharing training episodes and learned policies, while keeping the overall scheme of conventional Q-learning. Experimental results on KOSPI 200 show that a trading system based on the proposed framework outperforms the market average and makes appreciable profits. Furthermore, in view of risk management, the system is superior to a system trained by supervised learning.

## 1 Introduction

The number of investors in stock market is increasing everyday in this century and intelligent decision support systems aiding them to trade are keenly needed. Many of technical indicators such as moving averages have been developed by researchers in economic area [1]. Also statistical and other computer aided machine learning approaches are prevalent. But many of them, based on supervised learning, have a limitation that they are optimized to prediction criteria without considering trading policies in a unified framework. Recently, as an alternative, reinforcement learning is adopted to optimize trading systems [2][3]. The objective of reinforcement learning is not the minimization of the *sum-of-squares* error which is actually the objective of conventional supervised learning but the acquisition of an optimal policy under which the learning agent achieves the maximal average reward from the environment.

This paper proposes a reinforcement learning framework with multiple cooperative agents to integrate prediction criteria with trading policies more effectively. The agents for buy and sell signals use a matrix named *turning-point structure* in order to model the long-term dependencies of stock prices. To utilize intraday price swings, the agents for ordering executions optimize the short-term policies. Q-learning with neural networks is adopted for training the agents to get optimal policies. In addition, the value approximator is trained using a regularizing technique for the prevention of divergence of the parameters. We demonstrate a stock trading system implemented using the proposed framework can significantly outperform the market average and the one implemented by conventional supervised learning algorithm.

The paper is organized as follows. In section 2, we briefly summarize the concept of reinforcement learning and some previous researches on stock market analysis. In section 3, we introduce the overall structure of the framework and explain how the cooperative agents communicate with others. Also, the detailed learning algorithm is presented. In section 4, we describe the experimental setup and results. Finally, in section 5 we conclude with a few future directions.

## 2 Backgrounds

Reinforcement learning is a computational approach for understanding and automating goal directed learning and decision making. We introduce reinforcement learning following the notations of Sutton [4]. In the reinforcement learning framework, especially in Markov decision process (MDP), there are an agent and an environment interacting with each other at discrete time steps $t = 0, 1, 2, \ldots, T$. The agent selects an action $a_t \in \mathcal{A}$ from its policy $\pi$ based on the state $s_t \in \mathcal{S}$ of the environment. If certain action is taken by the agent, the environment changes its state to $s_{t+1}$ responding to action $a_t$ and also gives rise to a reward $r_{t+1} \in R$.

If one-step state transition probabilities and one-step expected reward models were available, the environment could be completely described as:

$$p^a_{ss'} = \Pr\{s_{t+1} = s' | s_t = s, a_t = a\} \tag{1}$$

$$r^a_s = \Pr\{r_{t+1} | s_t = s, a_t = a\}, \tag{2}$$

for all $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}$. The objective of the agent is to learn an optimal policy, a mapping from states to actions, that maximizes the expected discounted future reward from state-action pair $(s, a)$, called action-value function $Q^\pi(s, a)$. Given an episodic task which is defined as the history of interactions from the starting state $s_0$ to the terminal state $s_T$ , $Q^\pi(s, a)$ is defined as:

$$Q^\pi(s, a) = E_\pi\{r_t + \gamma r_{t+1} + \cdots + \gamma^{T-t-1} r_T | s_t = s, a_t = a\}, \tag{3}$$

where $0 \le \gamma \le 1$ is a discount-rate. Then optimal policy, denoted $Q^*$, can be defined as:

$$Q^*(s, a) = \max_\pi Q^\pi(s, a), \tag{4}$$

for all $s$ and $a$. Though there are many kinds of algorithms for learning optimal policy, they could be grouped into 3 categories according to how their backups are made up. First, dynamic programming uses full backup and can always converge but requires an exact model of the environment. Second, Monte Carlo technique uses only sample backups of the entire episode. It doesn't require any model but needs tremendous episodes for its convergence. Last, the temporal difference (TD) method is a compromise form of both algorithms, It uses both $n$-step samples and bootstrapping, namely currently learned value model [4], [7]. In practice, TD method is widely used for its easiness of handling but much caution is needed for its convergence [5].

neste parágrafo o autor fala de trabalhos anteriores

Researches on applying reinforcement learning to stock market analysis have a short history. Neuneier formulated financial market as MDP under some assumptions and simplifications about the market's characteristics [2]. He also modified Q-learning by adding preference to the risk avoiding tendency in [11]. He focused on the asset allocation, that is, how to change one's position to either DAX or DM. Moody et al. formulated a reinforcement learning framework similar to recurrent neural networks [9][10]. Within their framework the policy is directly updated by back propagation through time which is the famous learning technique of recurrent neural networks. He showed that the trained asset allocator could make a profit by changing successfully its position to either S&P 500 or T-Bill market. Xiu et al. also proposed a portfolio management system using Q-learning [8]. Since they used two performance functions, absolute profit and relative risk-adjusted profit, two networks were used in the training process.

Our main interest is not the asset allocation between two markets, but trading stocks in a single stock market. While most reinforcement learning formulations of stock market have one agent, we formulate reinforcement learning with multiple agents.

## 3 Proposed Framework and Algorithm

In this section, we describe the overall structure of the proposed framework and the details of the Q-learning algorithm for this framework.

### 3.1 Overall Structure

We propose a multi-agent framework as shown in Fig. 1. This framework aims to maximize the profits of investments by considering not only global trends of stock prices but also intraday price movements of stocks. Each agent has its own goal to achieve and interacts with others to share episodes in the learning process:

- *Buy signal agent* performs prediction by estimating the long-term and short-term information of the states of stocks to produce buy signals.
- *Buy order agent* determines prices for *bids* by estimating the short-term information of the states of stocks. A bid is an order to buy at a given price.
- *Sell signal agent* performs prediction by estimating the long-term and short-term information of the states of stocks and the current profits, to produce sell signals.
- *Sell order agent* determines the prices for *offers* by estimating the short-term information of the states of stocks. An offer is an order to sell at a given price.

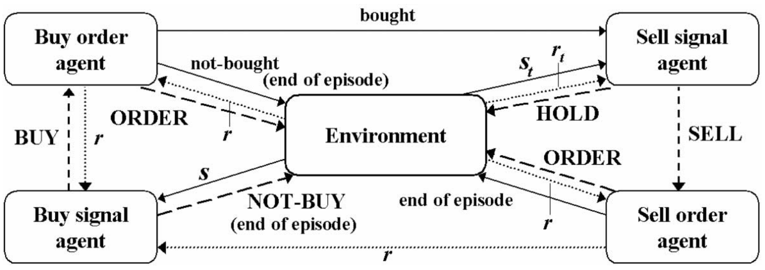


**Fig. 1.** Structure of the framework

The sell signal agent differs from the buy signal agent in that it also concerns in the policy as well as the prediction for selling. The buy/sell order agents do not perform prediction because their only goal is to provide a policy for optimal order executions in order to increase the success rate of trades.

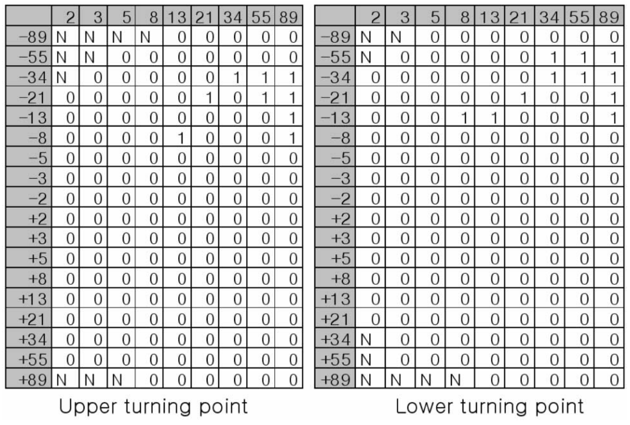
## 3.2 States, Actions, and Rewards

One of the most important keys to achieve reasonable performance in machine learning is the representation of the input space. Specially reinforcement learning is an art of the design of state, action and reward. Those in the proposed framework are as follows.

We adopt a binary matrix, the turning-point structure, summarizing the long-term dependencies based on both upper and lower turning points of 5-day moving averages. Fig. 2 shows an example of this matrix. The columns represent the displacement of the day of a turning point from the reference day, and the rows represent the displacement of the price of a turning point from the reference day. The value 'N' means that this slot can not be hit since the price of stocks in KOSPI never reaches that entry[1]. The total number of bits required to represent both matrices for the turning points is 305 with 'N' entries eliminated. In addition to these temporal structures, the buy signal agent shares the short-term technical indicators shown in Table 2, with the ordering agents. After all, the state of the buy signal agent consists of 353 binary bits. The sell signal agent has a few bits more, which represent the current profit rate during a holding period of a stock as shown in Table 1. We confine arbitrarily the profit range as $+30\% \sim -20\%$. So while holding the stock, if its profit rises above $+30\%$ or falls down below $-20\%$, the sell signal agent is compelled to sell the stock forcedly.

The signal agents can take only two kinds of actions. The buy signal agent takes NOT-BUY or BUY and the sell signal agent takes HOLD or SELL. The calculation of reward is as follows. The buy signal agent is given zero reward while it takes NOT-BUY. If it takes BUY the calculation of its reward is postponed until the sell order agent sells the stock. The rate of daily changes of stock prices are given while the sell signal agent takes HOLD. But when it takes SELL, zero

[1] In KOSPI, the restriction of price range for one day is $(+15\%, -15\%)$.



| | 2 | 3 | 5 | 8 | 13 | 21 | 34 | 55 | 89 |
|---|---|---|---|---|---|---|---|---|---|
| −89 | N | N | N | N | 0 | 0 | 0 | 0 | 0 |
| −55 | N | N | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| −34 | N | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| −21 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| −13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| −8 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| −5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| −3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| −2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| +2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| +3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| +5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| +8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| +13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| +21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| +34 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| +55 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| +89 | N | N | N | 0 | 0 | 0 | 0 | 0 | 0 |

Upper turning point

| | 2 | 3 | 5 | 8 | 13 | 21 | 34 | 55 | 89 |
|---|---|---|---|---|---|---|---|---|---|
| −89 | N | N | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| −55 | N | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| −34 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| −21 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| −13 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| −8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| −5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| −3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| −2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| +2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| +3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| +5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| +8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| +13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| +21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| +34 | N | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| +55 | N | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| +89 | N | N | N | N | 0 | 0 | 0 | 0 | 0 |

Lower turning point

**Fig. 2.** An example of turning-point sturucture. 'N' means *not available*

**Table 1.** Additional bits for the sell signal agent to represent the current profit, which is, 100×(closing price of the current day - buy price)/buy price

| Profit | Coding | Profit | Coding |
|---|---|---|---|
| $+(0 \sim 5)$ | 00000001 | $-(0 \sim 3)$ | 00010000 |
| $+(5 \sim 12)$ | 00000010 | $-(3 \sim 7)$ | 00100000 |
| $+(12 \sim 20)$ | 00000100 | $-(7 \sim 12)$ | 01000000 |
| $+(20 \sim 30)$ | 00001000 | $-(12 \sim 20)$ | 10000000 |

reward is given because the signal means that it exits the market. When the sell price is determined by the sell order agent, the buy signal agent receives the profit rate, considering the transaction cost, as the reward.

Ordering agents share the state and possible actions shown in Table 2. For a given state, the buy order agent tries to make an optimal bid, that is, the bid at lowest price in the range of stock price changes of the next day. When trading stocks with very high volatility, the buy price may significantly affect to the final profit of the trade. Fig. 3 shows an example of this. Here the bid at 'Low' can decrease the risk of stop-loss selling. The reward of the buy order agent is defined as:

$$r = \begin{cases} e^{-100*d/Low} & \text{bid price} \geq Low, \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

where $d$ = bid price $- Low$[2]. If $d$ is 0, the agent receives the maximal reward. Similarly the reward of the sell order agent is defined using $High$[3]:

$$r = \begin{cases} e^{100*d/High} & \text{offer price} \leq High, \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

where $d$ = offer price $- High$.

[2] The lowest price of the next day.

[3] The highest price of the next day.

**Table 2.** Coding of state and actions for ordering agents

| State | Bits | Action | Coding |
|---|---|---|---|
| Disparity(5) | 8 | −12% | 0000001 |
| Disparity(10) | 6 | −7% | 0000010 |
| Disparity(20) | 6 | −3% | 0000100 |
| MA-Grad(5) | 8 | 0% | 0001000 |
| MA-Grad(10) | 6 | +3% | 0010000 |
| MA-Grad(20) | 6 | +7% | 0100000 |
| Body | 8 | +12% | 1000000 |

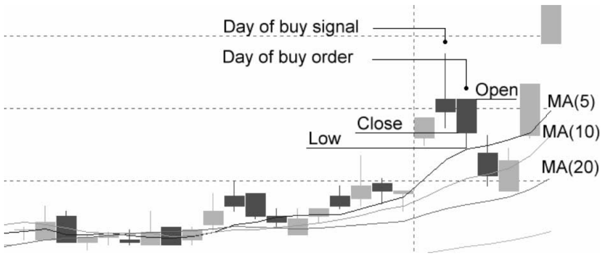


**Fig. 3.** An example of daily price changes

## 3.3 Learning Algorithm

An episode starts with a stock at a certain day randomly chosen by the environment. If the buy signal agent takes NOT-BUY as a response to the state of the given stock, the episode ends and a new episode starts. But if the buy signal agent takes BUY, it invokes the buy order agent. The buy order agent refers the state of the stock at the same day and determines the bid price. If the stock can not be purchased, the episode ends and invokes the buy signal agent with 0 as the reward. The whole algorithm including the selling side is shown in Fig. 4. Each agent has equations for reward, delta and Q-value, but all of them are expressed in the same notation for the brevity. Time index is also abbreviated for the same reason. If the sell order agent fails to sell the stock with its offer price, it sells the stock at close, the last traded price of the day. The update rules and function approximation are based on the ordinary Q-learning.

Since the search space of state-action pairs is too large, i.e. $2^{353}$ for the buy signal agent, to maintain the Q-values in a table, we use neural networks as Q-value approximators. Theoretically there are the situations where simple linear approximators of Q-table can diverge. Baird discussed some theoretical directions to prohibit the divergence of approximators [5]. In this paper, to make the problem simple, we introduce the regularized gradient descent as:

$$\nabla_{\theta_i} \widetilde{Q}(s,a) = \begin{cases} \nabla_{\theta_i} Q(s,a) & \text{if } \theta_i = \text{bias}, \\ \nabla_{\theta_i} Q(s,a) + \nu * \theta_i & \text{otherwise}, \end{cases} \tag{7}$$

```
Buy signal agent :
1. Environment produces $s$.
    $a \leftarrow \begin{cases} \arg\max_a Q(s,a) & \text{with prob. } 1-\epsilon, \\ \text{random selection} & \text{with prob. } \epsilon. \end{cases}$
    If $a$ = NOT-BUY Episode ends and goto 1.
    Else Invoke the buy order agent and wait for invocations from other agents.
    If the invocation is from the buy order agent
        $r \leftarrow 0$
    Else
        $r \leftarrow ((1-TC) * SP - BP)/BP$
    $\delta \leftarrow r - Q(s,a);\quad \theta \leftarrow \theta + \eta * \delta * \nabla_\theta Q(s,a)$
    Episode ends and goto 1.

Buy order agent :
2. Environment produces $s$.
    $a \leftarrow \begin{cases} \arg\max_a Q(s,a) & \text{with prob. } 1-\epsilon, \\ \text{random selection} & \text{with prob. } \epsilon. \end{cases}$
    $d = MA(5) + \frac{a}{100} * MA(5) - Low$
    If $d \geq 0$
        $r \leftarrow e^{-100*d/Low};\quad \delta \leftarrow r - Q(s,a);\quad \theta \leftarrow \theta + \eta * \delta * \nabla_\theta Q(s,a)$
        Invoke the sell signal agent with $BP$, where $BP = MA(5) + \frac{a}{100} * MA(5)$.
    Else
        $r \leftarrow 0;\quad \delta \leftarrow r - Q(s,a);\quad \theta \leftarrow \theta + \eta * \delta * \nabla_\theta Q(s,a)$
        Invoke the buy order agent.

Sell signal agent :
3. Environment produces $s$.
    $a \leftarrow \begin{cases} \arg\max_a Q(s,a) & \text{with prob. } 1-\epsilon, \\ \text{random selection} & \text{with prob. } \epsilon. \end{cases}$
    If $a$ = HOLD
        $s' \leftarrow Action(s,a);\ r \leftarrow RC;$
        $\delta \leftarrow r + \gamma * \max_{a'} Q(s',a') - Q(s,a);\ \theta \leftarrow \theta + \eta * \delta * \nabla_\theta Q(s,a)$
        Goto 3.
    Else Invoke the sell order agent.

Sell order agent :
4. Environment produces $s$.
    $a \leftarrow \begin{cases} \arg\max_a Q(s,a) & \text{with prob. } 1-\epsilon, \\ \text{random selection} & \text{with prob. } \epsilon. \end{cases}$
    $d = MA(5) + \frac{a}{100} * MA(5) - High$
    If $d \leq 0$
        $r \leftarrow e^{100*d/High};\quad \delta \leftarrow r - Q(s,a);\quad \theta \leftarrow \theta + \eta * \delta * \nabla_\theta Q(s,a)$
        Invoke the buy signal agent with $BP$ and $SP(MA(5) + \frac{a}{100} * MA(5))$.
    Else
        $r \leftarrow 0;\quad \delta \leftarrow r - Q(s,a);\quad \theta \leftarrow \theta + \eta * \delta * \nabla_\theta Q(s,a)$
        Invoke the buy signal agent with $BP$ and $SP(Close)$.
```

**Fig. 4.** Q-learning algorithm for the proposed framework. $TC$ is the transaction cost, $SP$ is the sell price, $BP$ is the buy price, and $RC$ is the rate of change of the closing price

where $\nu$ is the constant that controls the degree of weight decay. The regularized gradient descent is a popular technique in the neural network society, and can discourage a component of parameters from increasing indefinitely.

## 4 Experiment

We compare a stock trading system built according to the proposed framework with another trading system trained by supervised learning using a neural network with the same input space described in section 3. For convenience, the former system is called $MAQ$ and the latter $SNN$ in this section.

Table 3 shows the data set. The network structure of MAQ arbitrarily chosen as two hidden layers with $40 \times 20$ neurons. The strategy of shrinkage of the learning rate is not adopted, but the learning rate is fixed $\eta = 0.005$ for all the agents. The weight decay constant is fixed as $\nu = 0.2$ after some preliminary trials. The discount factor $\gamma$ is set to 0.95 and the exploration factor $\epsilon$ is set to 0.1. After each of 20,000 episodes is experienced, the system is verified on the validation set.

**Table 3.** Partitions and specification of the data set

| Partition | Period | Size |
|---|---|---|
| Training set | Jan-1999 ~ Dec-2000 | 32,019 |
| Validation set | Jan-2001 ~ May-2001 | 6,102 |
| Test set | Jun-2001 ~ Oct-2001 | 6,213 |

Fig. 5 shows the tendency of the training performance of MAQ. Before 2,800,000 episodes are experienced, the trades are conducted about 50 times from 20,000 episodes in the training set and never in the validation set. This means that the system is trying to buy and sell through only random exploration of $\epsilon$-policy. In this first period, trades lead to losses, about -1.2% ~ -0.1%[4]. After 2,800,000 episodes, the number of trades and the profit begin to increase in both data set. This means that the system begins to buy and sell stocks by its greedy policy and make profits from those trades. But after 5,900,000 episodes there is, though less significant, degradation of the average profit in the validation set. So the system is stopped to train at that point and is applied to the test set.

Both trading systems, MAQ and SNN, have five subtraders which individually trade their assets according to the strategy of the system they belong to. The trading system evaluates stocks and gathers the upper ranked candidates. From the candidates pool, the target stock is randomly distributed to each subtrader. After some steps of trading, the invest moneys of subtraders are merged and redistributed equally and trades are continued. SNN buys and sells stocks at opening prices while MAQ follows the decision of the ordering agents.

[4] The transaction cost is 0.5% in Korean stock market.

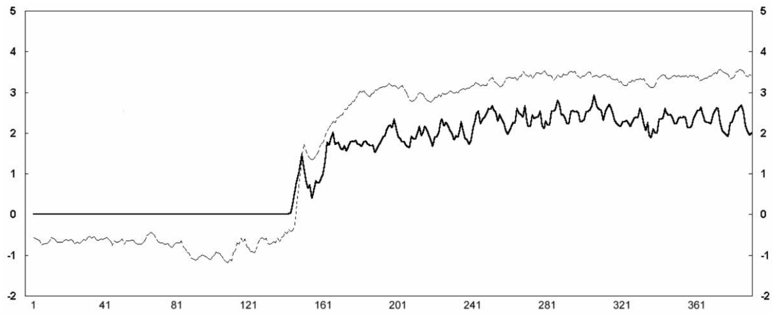


**Fig. 5.** Average profit rate induced during every 20,000 episodes

Fig. 6 shows the performance of both SNN and MAQ on the test set. KOSPI is 612 point on the starting day of the test period. On the whole, it can be said that MAQ outperforms both the market index and SNN. At the end of the test period, the asset of MAQ increases 28.26% to 785 while that of SNN increases 8.49% to 664.

In view of risk management, MAQ is also superior to SNN. There are two shocks in KOSPI during the test period. One is the internal shock between June to July, that is, the market is in a normal bear trend. The profit of SNN is severely degraded from this shock but MAQ endures the shock with a relatively small loss. The other is an external shock, the tragedy of September 11, 2001. For a week after the tragedy, The Korean stock market suffered from panic selling. Both systems lead to severe losses for this short period. But after the end of September, the market changes to a bull market. In this period, the profit of MAQ is increased steadily while that of SNN somewhat fluctuates.

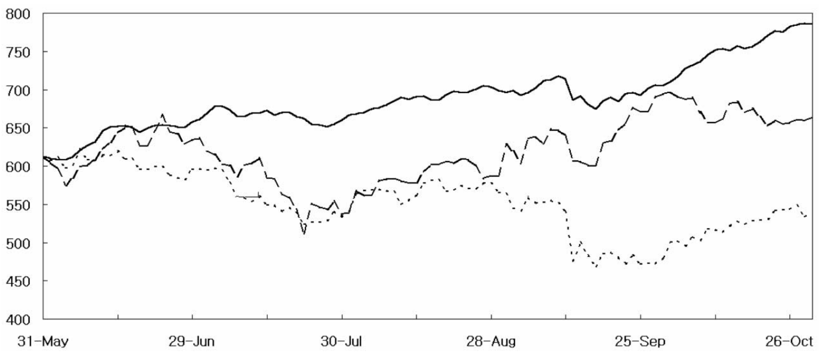


**Fig. 6.** Profit ratio of SNN (dashed) and MAQ (solid). Each system starts to trade with the asset 612 equivalent to the starting value of KOSPI (dotted) for comparison

## 5 Conclusion

We proposed a framework for building stock trading systems with multiple cooperative agents formulated under reinforcement learning. Also we designed a data structure for summarizing the historic information of price changes for a long period. The system achieves a higher profit and reasonable risk management compared with the one trained by supervised learning in the Korean stock market. However, there are several additional realistic considerations to be incorporated in a stock trading system. These include the number of portfolios, the distribution of the asset to each portfolio, and adaptation of the trend of the stock market. Reinforcement learning is promising but incorporation of these considerations makes the problem more complex. So making formulations of reinforcement learning with those considerations tractable is the future work.

## References

1. Kendall, S. M., Ord, K.: Time Series. Oxford, New York. (1997)
2. Neuneier, R.: Enhancing Q-Learning for Optimal Asset Allocation. Advances in Neural Information Processing Systems 10. MIT Press, Cambridge. (1998) 936–942
3. Lee, J.: Stock Price Prediction using Reinforcement Learning. In Proceedings of the 6th IEEE International Symposium on Industrial Electronics. (2001)
4. Sutton, R. S., Barto, A. G.: Reinforcement Learning : An Introduction. MIT Press, Cambridge. (1998)
5. Baird, L. C.: Residual Algorithms: Reinforcement learning with Function Approximation. In Proceedings of Twelfth International Conference on Machine Learning. Morgan Kaufmann, San Fransisco. (1995) 30–37
6. Bengio, Y., Simard, P., Frasconi, P.: Learning Long-Term Dependencies with Gradient is Difficult. IEEE Transactions on Neural Networks 5(2). (1994) 157–166
7. Jaakkola, M., Jordan, M., Singh, S.: On the Convergence of Stochastic Iterative Dynamic Programming Algorithms. Neural Computation, 6(6). (1994) 1185–2201
8. Xiu, G., Laiwan, C.: Algorithm for Trading and Portfolio Management Using Q-learning and Sharpe Ratio Maximization. In Proceedings of ICONIP 2000, Korea. (2000) 832–837
9. Moody, J., Wu, Y., Liao, Y., Saffell, M.: Performance Functions and Reinforcement Learning for Trading Systems and Portfolios. Journal of Forecasting, 17(5-6). (1998) 441–470
10. Moody, J., Saffell, M.: Learning to Trade via Direct Reinforcement. IEEE Transactions on Neural Networks, 12(4). (2001) 875–889
11. Neuneier, R., Mihatsch., O.: Risk Sensitive Reinforcement Learning. Advances in Neural Information Processing Systems 11. MIT Press, Cambridge. (1999) 1031–1037