A Q-Learning Based Approach to Design of Intelligent Stock Trading Agents

J. W. Lee[1], E. Hong[1], J. Park[2]
[1] School of Computer Science & Engineering, Sungshin Women's University, Seoul Korea
[2]Department of Industrial Engineering, Seoul National University, Seoul Korea

Abstract— **The portfolio management for trading in stock market poses a challenging stochastic control problem of significant commercial interests to finance industry. To date, many researchers have proposed various methods to build an automated portfolio management system that can recommend financial decisions for daily stock trades. However, most previous attempts to predict future returns of stocks have been largely unsuccessful due to the high complexity of integrating price prediction results with trading strategies. Motivated by this, this paper presents a new stock trading system, named Q-trader, that can overcome the limitations of the existing approaches through the joint optimization of the prediction results and the corresponding trading strategies. Specifically, the proposed framework employs four cooperative Q-learning agents that adaptively interact with their environments by use of feedforward neural networks. In order to achieve computational efficiency for global trend prediction, a novel data structure is proposed for compact representation of long-term dependencies of stock price changes. Experimental results on KOSPI 200 show that the proposed approach outperforms the trading systems trained by supervised learning both in profit and risk management.**

Keywords—**Intelligent agents, stock trading, automated portfolio management**

I. INTRODUCTION

Stock prediction is one of the hottest topics that draw a lot of attention both from academia and industry in recent years. Owing to its popularity, there have been numerous research results that addressed fundamental issues as well as technical issues of portfolio management for trading in stock market in attempts to maximize expected returns.

In the mean time, the famous efficient market hypothesis (EMH) has marginalized those efforts for a long time. EMH states that it is not possible to predict the market only by use of historic data [1]. Although recent advancements in machine learning community reported that their proposed models could outperform the market average, it is still widely accepted among the researchers that stock market is highly non-stationary and that there are long-term correlations among prices.

That is, the trend of stock price series is affected by not only the series of past few days but also the coupled relations between the series of the distant past and those of recent days. Simple feed forward neural networks (FFNN) can hardly model the non-stationarity and the long-term dependencies of stock market. Similar problems have been reported when recurrent neural networks with memory architecture and dynamics are used to model the long-term dependencies [2].

On the other hand, reinforcement learning is a different paradigm from statistical data analysis and supervised learning like FFNN. Its objective is to compute the optimal policy under which a learning agent can achieve maximal average rewards from the environment whereas the objective of the supervised learning is to generalize the mapping from inputs to outputs. Accordingly it is expected that that the reinforcement learning is more suitable than the supervised learning for the stock market analysis since the problem is goal-directed and highly dynamic.

Motivated by the above remarks, the proposed framework, named Q-trader, employs Q learning algorithm, one of the well-known reinforcement learning methods under the assumption that the process of stock price changes follows Markov decision process (MDP). Q-trader defines four agents so that they can learn the sub-tasks of the entire trading task in a unified framework. Specifically, two agents, named buy / sell-signaling agents, perform global trend predictions, aiming to produce optimal trading signals. The other two agents, named buy / sell-ordering agents, carry out intra-day ordering executions. Cooperation of these agents facilitates learning of efficient policies to increase the profit while managing risk effectively.

One of the important issues that must be addressed when designing a reinforcement learning algorithm is the representation of states. In particular, the problem of maintaining the whole raw series of stock price data to compute the long-term correlations becomes intractable as the size of time window grows large. To address this problem, we propose a novel idea, named the turning-point structure that can summarize the historic information of price changes over the last 230 days. The turning point structure is essentially a binary matrix, and it is used for state representation of the signaling agents. Furthermore, short-term moving averages and Japanese candlestick charting techniques are used by the ordering agents.

In the following section, we present the architecture of the proposed framework, and describe how cooperation of stock trading agents in Q-trader is defined. Section 3 presents the learning algorithms employed by the participating agents. Experimental setup and results are presented in section 4. Section 5 concludes the paper with discussion on future research directions.

II. THE PROPOSED FRAMEWORK

Fig. 1 shows the reinforcement learning framework of Q-Trader. The framework aims to maximize profits of investments by considering the global trends of stock prices as well as the intra-day stock price changes. Each agent has its own goal and interacts with other agents to share episodes in the learning process.

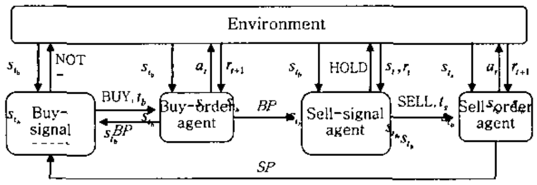

Fig. 1. Overall framework of Q-Trader

The buy-signaling agent and sell-signaling agent perform prediction[1] to generate buy / sell trading signals. Buy-signaling agent estimates stock prices and decides if they are likely to increase in near future. In parallel to purchasing stocks, sell-signaling agent is responsible for determining the specific day for selling stocks. The rationale underlying the separation of these two agents can be explained as follows.

From the viewpoint of an investor, one has different decision mechanisms depending on whether one is buying or selling stocks. When buying stocks, one considers the overall possibility of price increase of stocks. On the contrary, when selling stocks, it is a common practice to consider not only the tendency of stock prices but also the current profits that can be realized to the investor.

Specifically, since the selling agent maintains the profit as a part of its states, it would be desirable to have two different agents for prediction task. The buy-ordering agent and sell-ordering agent determine the prices of *bids* and *offers*[2]. We remark that ordering agents do not perform predictions as their common goal is to provide intra-day policies for optimal ordering execution in order to increase overall success rates of trades.

An episode starts by selecting a stock at a certain day (t_b)[3] randomly chosen by environment. If the buy-signaling agent makes NOT-BUY decision for the selected stock, the episode ends and a new episode starts. On the other hand, in case that the buy-signaling agent decides to BUY, it invokes the buy-ordering agent who purchases the stock on the next day of t_b after determining the buying price (*BP*).

Subsequently, the sell-signaling agent is activated and it carries out HOLD or SELL action during the successive trading days following the buying day. When the sell-signaling agent decides to HOLD, the environment provides the agent with reward and state information. In contrast, when the sell-signaling agent decides to SELL, the stock is sold on the next day by sell-ordering agent after the sell price (*SP*) is determined. This ends an episode. Finally, the profit rate is provided to the buy-signaling agent as a reward. Fig. 2 depicts an example of episode.

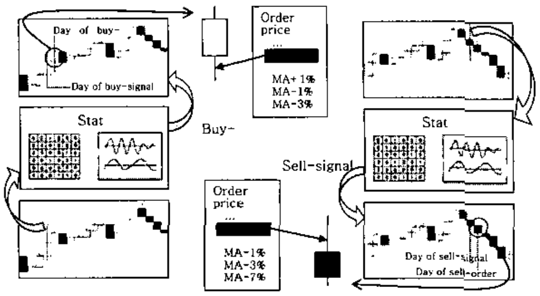

Fig. 2. An example of an episode

III. LEARNING ALGORITHMS

The Q learning algorithms defined for the agents in Q-trader are presented in Fig. 3 ~ Fig. 6. Although each agent has its own equations for computing the reward, delta and Q-value, they are represented in the same notation for brevity. The subscripts denoting time steps are also abbreviated. Update rules and function approximations are based on the ordinary Q-learning. Finally, we remark that early-stopping [3] is employed for training, which stops training when the validation error rate starts to grow.

```
Repeat
    Environment produces s of $t_b$
    $a \leftarrow Greedy(s)$
    If ($a$ = BUY)
        Invoke buy-order agent and wait for the invocation
        from sell-order agent
        $r \leftarrow 100 \times ((1-TC) * SP - BP) / BP$
    Else
        $r \leftarrow 0$
    $\delta \leftarrow r - Q(s,a)$ ; $\theta \leftarrow \theta + \eta * \delta * \nabla_\theta Q(s,a)$
Until (Early stopping criterion is met)
```

Fig. 3. Q-learning algorithm for buy-signaling agent. *TC* is the transaction cost, *SP* is the sell price, and *BP* is the buy price.

Buy-signaling agent refers to the state of the t_b-th trading day of a stock which is randomly chosen by the environment and takes an action by use of its policy function *Greedy*. We use the well-known ε-greedy for *Greedy*:

$$a = \begin{cases} \arg\max_a Q(s,a) & \text{with prob. } 1-\varepsilon, \\ \text{random selection} & \text{with prob. } \varepsilon. \end{cases}$$

If the action taken is BUY, the agent invokes buy-ordering agent and waits until the sell-ordering agent in-

[1] The sell-signaling agent also affects trading strategy since it considers current profit rates of stocks.
[2] A bid is an order to buy at a certain price and an offer is an order to sell at a certain price.
[3] The time index defined for the day of buy signal. Actual purchase is made on the next day.

vokes it. The reward, the profit of the trade, is calculated based on the buy and sell prices that are determined by buy-ordering and sell-ordering agents respectively. The transaction cost is also considered in this calculation.

TABLE I
Possible actions of ordering agents.

Action	Coding	Action	Coding
-12	00000001	+12	10000000
-7	00000010	+7	01000000
-3	00000100	+3	00100000
-1	00001000	+1	00010000

```
Environment produces $s$ of $t_b$
Repeat
    $a \leftarrow Greedy(s)$
    $d \leftarrow MA^5 * (1 + a/100) - Low$
    If ($d < 0$)
        $r \leftarrow 0$
        $\delta \leftarrow r - Q(s,a)$ ; $\theta \leftarrow \theta + \eta * \delta * \nabla_\theta Q(s,a)$
Until ($d \geq 0$)
$r \leftarrow e^{-100*d/Low}$
$\delta \leftarrow r - Q(s,a)$ ; $\theta \leftarrow \theta + \eta * \delta * \nabla_\theta Q(s,a)$
$BP \leftarrow MA^5 * (1 + a/100)$
Invoke sell-signal agent
```

Fig. 4. Q-learning algorithm for buy-ordering agent.

Buy-ordering agent also uses its *Greedy* function to choose an action. Table I shows the possible actions of ordering agents. An action a represents that the agent chooses $MA_i^5(1+a/100)$ as a bid or offer price. For a given state, the buy-ordering agent attempts to make an optimal bid, which is the bid at the expected lowest possible price on the next day. The reward of buy-ordering agent is defined as:

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

where d = bid price - Low[4]. If d is 0, the agent receives the maximal reward.

The sell-signaling agent first considers the state of (t_b+1)-th trading day of the stock purchased in order to take an action, HOLD or SELL. The price change rate of the stock is given as a reward to the agent when it decides to HOLD. On the other hand, when it makes a decision to SELL, no reward is given since its decision implies that it exits the market. The agent refers to the buy price determined by the algorithm of buy-ordering agent to compute the current profit rate.

[4] The lowest price of the next day.

```
Environment produces $s$ of $t_b$+1
$a \leftarrow Greedy(s)$
$t_s \leftarrow t_b + 1$
If $a$ = SELL
    $r \leftarrow 0$
    $\delta \leftarrow r - Q(s,a)$ ; $\theta \leftarrow \theta + \eta * \delta * \nabla_\theta Q(s,a)$
    Invoke sell-order agent
Else
    Repeat
        $s' \leftarrow$ state of the next trading day; $r \leftarrow RC$
        $\delta \leftarrow r + \gamma * \max_{a'} Q(s',a') - Q(s,a)$ ;
        $\theta \leftarrow \theta + \eta * \delta * \nabla_\theta Q(s,a)$
        $s \leftarrow s'$
        $t_s \leftarrow t_s + 1$
        $a \leftarrow Greedy(s)$
    Until ($a$ = SELL)
$r \leftarrow 0$
$\delta \leftarrow r - Q(s,a)$ ; $\theta \leftarrow \theta + \eta * \delta * \nabla_\theta Q(s,a)$
Invoke sell-order agent
```

Fig. 5. Q-learning algorithm for sell-signaling agent. *RC* is the rate of change of the closing price.

Finally, sell-ordering agent determines the offer price. If the price is higher than the highest price of the next day, the agent sells the stock at the closing price of the day. The reward of sell-ordering agent is then defined as:

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

where d = $High$[5] - offer price. If d is 0, the agent receives the maximal reward.

```
Environment produces $s$ of $t_s$
$a \leftarrow Greedy(s)$
$d \leftarrow High - MA^5 * (1 + a/100)$
If ($d \geq 0$)
    $r \leftarrow e^{-100*d/High}$
    $SP \leftarrow MA^5 * (1 + a/100)$
Else
    $r \leftarrow 0$
    $SP \leftarrow Close$
$\delta \leftarrow r - Q(s,a)$ ; $\theta \leftarrow \theta + \eta * \delta * \nabla_\theta Q(s,a)$
Invoke buy-signal agent
```

Fig. 6. Q-learning algorithm for sell-ordering agent. *High* is the highest price and *Close* is the closing price of the next day.

[5] The highest price on the next day.

IV. Experimental Results

In this section, we test the predictability of Q-Trader. We constructed data set of the stock candidates by using 200 representative stocks, KOSPI 200. The entire historical data of KOSPI 200 stocks are classified into three mutually exclusive groups: a training set, a validation set, and a test set as shown in Table II.

TABLE II
Partitions and specifications 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

We used several architectures of feedforward neural networks for the value function approximation. Each of them is trained 10 times with different initial weights. We also limited the network structure to two hidden layers at maximum. Table III shows the best prediction performance of each architecture considered. In Table III, the correlations between the predicted values $Q(s, \text{BUY})$s and the actual values computed by using the discount factor $\gamma=0.9$ are presented. The network with 80 neurons in the first hidden layer and 20 neurons in the second hidden layer turns out to be the best in terms of the predictability[6].

TABLE III
Prediction performance of neural networks

One hidden layer		Two hidden layers	
Structure	Correlation	Structure	Correlation
80	0.274	40 x 20	0.291
100	0.283	60 x 20	0.304
120	**0.291**	80 x 20	**0.336**
160	0.285	100 x 20	0.315
200	0.278	120 x 40	0.296

Fig. 7 shows the tendency of training performance of Q-Trader with the best network architecture described above. The strategy of shrinkage of learning rate was not adopted, and the learning rate η was fixed to 0.005. The exploration factor ε was set to 0.1. For the purpose of analyzing the tendency of training, the statistics were logged at the intervals of 20,000 episodes.

The lower part of Fig. 7 shows the average number of trades made during each interval. The upper part shows the average profit from trades during the same period. Before 3,200,000 episodes were experienced, the trades had been made about 50 times in the training set and no trades in the validation set. This implies that the system is trying to buy and sell only through the random exploration of ε-policy. In this first period, trades lead to losses, -0.5% ~ -0.1%. After 3,200,000 episodes, the number of trades and the profits begin to increase in both data sets. This means that the system begins to buy and sell stocks by its greedy policy and make profits from those trades. But after 5,000,000 episodes, we note that there is, though less significant, degradation of average profits in the validation set. Therefore the system stops training at this point.

[6] Because this architecture also achieves the best trading performance in the simulation test, the architecture is used for the performance comparison of Q-Trader and SNN.

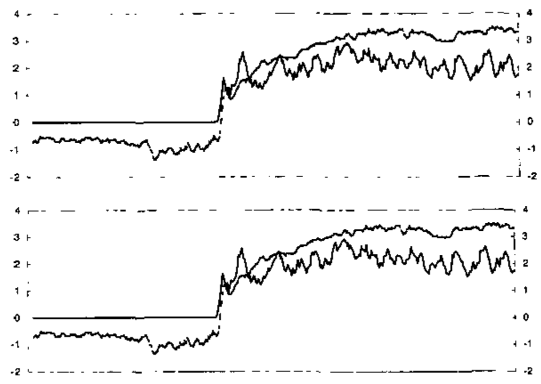

Fig. 7. Average profit induced during every 20,000 episodes (above) and average number of trades occurred in that period (below). The solid line represents the case of the validation set and the dotted line represents the case of the training set. The left vertical axis represents the average percentage of profit and the average number of trades in the case of the validation set. The right vertical axis represents the value in the case of the training set.

V. Conclusion

In this paper, we proposed a stock trading system, Q-trader, which consists of four cooperative reinforcement learning agents. A novel data structure for summarizing historical information of price changes for a long period has been proposed to address the complexity problem. Experimental results on Korean stock market shows that the proposed system is able to achieve a high profit and reasonable risk management. Nevertheless, we believe that there are several additional considerations that can be taken when designing future stock trading systems. These include the incorporation of multiple portfolios, the distribution of the assets to each portfolio, and the adaptation to the trend of the stock market. While the reinforcement learning is promising, introduction of these considerations will make the problem more complex. Therefore, one of future research issues will be to make the reinforcement learning formulation with these considerations tractable.

References

[1] Malkiel, *A Random Walk Down Wall Street*. Norton. New York, 1996.

[2] Y. Bengio, P. Simard and P. Frasconi, "Learning Long-Term Dependencies with Gradient is Difficult," *IEEE Transactions on Neural Networks*, 5(2), 1994, pp. 157-166.

[3] R. Caruana, S. Lawrence and L. Giles, "Overfitting in Neural Nets: Backpropagation Conjugate Gradient, and Early Stopping," *Neural Information Processing Systems*, Denver, Colorado, November 28-30, 2000.