STOCK PRICE PREDICTION USING REINFORCEMENT LEARNING

Jae Won Lee

School of Computer Science and Engineering
Sungshin Women's University
Seoul, 136-742, South Korea

ABSTRACT

Recently, numerous investigations for stock price prediction and portfolio management using machine learning have been trying to develop efficient mechanical trading systems. But these systems have a limitation in that they are mainly based on the supervised learning which is not so adequate for learning problems with long-term goals and delayed rewards. This paper proposes a method of applying reinforcement learning, suitable for modeling and learning various kinds of interactions in real situations, to the problem of stock price prediction. The stock price prediction problem is considered as Markov process which can be optimized by reinforcement learning based algorithm. TD(0), a reinforcement learning algorithm which learns only from experiences, is adopted and function approximation by artificial neural network is performed to learn the values of states each of which corresponds to a stock price trend at a given time. An experimental result based on the Korean stock market is presented to evaluate the performance of the proposed method.

1. INTRODUCTION

There has been so much work done on ways to predict the stock prices for the sake of providing investors with the basis for an optimal decision-making in trading. Various technical indicators such as moving averages [1] have been developed by researches in economic area and nowadays numerous investigations in computer science have intended to develop different decision support systems for stock price prediction. Especially, some recent systems using machine learning methods such as artificial neural network achieved better performance than those using only conventional indicators [2][3][4]. But these systems have a limitation in that they are mainly based on the supervised learning. This is an important kind of learning, but alone it is not adequate for learning from interactions with long-term goals.

In the learning task of stock prediction, it is more natural and effective to represent target values by the successive relative changes in price since the previous time point than the absolute prices after a fixed time horizon which are generally adopted in conventional time series approaches based on supervised learning methods. In this sense, reinforcement learning can be a promising alternative approach for stock price prediction, representing and learning the delayed rewards, as well as the immediate rewards, from interactive processes more effectively [5].

This paper adopts reinforcement learning to the problem of stock price prediction regarding the process of stock price changes as a Markov process. For this, the process of stock price changes is modeled by the elements of reinforcement learning such as state, action, reward, policy, etc. And TD(0) algorithm [6], a kind of reinforcement learning methods is used. TD (temporal-difference) algorithms can learn directly from raw experience without a complete model of the environment's dynamics. Like DP (dynamic programming), TD algorithms update estimates based in part on other learned estimates, without waiting for a final outcome (they bootstrap) [7].

Though the ultimate purpose of reinforcement learning is to get an optimal policy, that is to resolve control problem, the use of TD algorithm in this paper is confined to the prediction problem because the policy in stock market is assumed to be determined by each investor and to be beyond the scope of learning here. In this paper, each stock price trend, the trace of successive price changes, at a given time is mapped to a state of reinforcement learning that is represented by the combination of some numerical features. The total number of possible states defined in this way is not finite and thus a generalization method is needed to approximate the value of expected future cumulative reward for a state. Here a multi-layer neural network is used for this purpose.

2. REINFORCEMENT LEARNING

Reinforcement learning is a computational approach to understanding and automating goal-directed learning and decision-making [8]. It is distinguished from other computational approaches by its emphasis on learning by

0-7803-7090-2/01/$10.00 © 2001 IEEE.

the individual from direct interaction with its environment as shown in Figure 1.

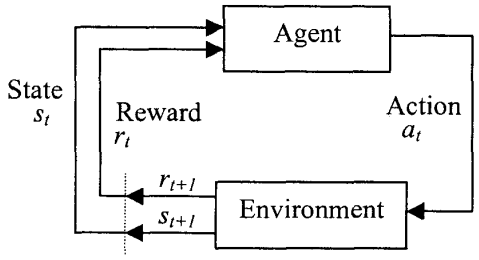

Figure 1: The agent-environment interaction in reinforcement learning.

At each discrete time step t, the agent senses the current state s_t, chooses a current action a_t, and performs it. The environment responds by giving the agent a reward $r_{t+1} = r(s_t, a_t)$ and by producing the succeeding state $s_{t+1} = \delta(s_t, a_t)$. Here the functions δ and r are the part of the environment and are not necessarily known to the agent. In MDP (Markov decision process), the functions $\delta(s_t, a_t)$ and $r(s_t, a_t)$ depend only on the current state and action not on earlier states or actions. The task of the agent is to learn a policy, $\pi : S \rightarrow A$, where S is the set of states and A is the set of actions, for selecting its next action based on the current observed state s; that is, $\pi(s_t)=a_t$. An optimal policy is a policy that can maximize the possible reward from a state, called value, $V^{\pi(s)}$, for all states. (1) is a typical definition of this:

$$V^{\pi}(s_t) = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,. \quad (1)$$

Here γ ($0 < \gamma < 1$) is a constant that determines the relative value of delayed versus immediate rewards. $V^{\pi(s)}$ means the possible cumulative reward achieved by following an arbitrary policy π from an arbitrary initial state. Then an optimal policy, π^*, is defined as follows:

$$\pi^* = \arg\max_{\pi} V^{\pi}(s), (\forall s) \,. \quad (2)$$

An iterative process in Figure 2, called Generalized Policy Iteration (GPI), is necessary to get π^*.

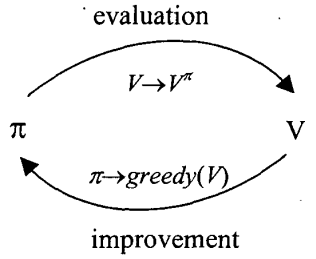

Figure 2: Generalized policy iteration

This paper applies the learning only to the prediction problem by utilizing the result of evaluation step in Figure 2.

3. TD ALGORITHM

Various reinforcement learning algorithms (eg. dynamic programming, temporal-difference, Monte Carlo, etc.) can be classified in view of GPI and the differences in the algorithms lies primarily in their approaches to the prediction problem. Like Monte Carlo algorithms, TD algorithms can learn directly from raw experience without a model of the environment's dynamics. Like DP, TD algorithms are bootstrapping algorithms, that is, they base their update in part on an existing estimate. After all, TD algorithms combine the sampling of Monte Carlo algorithms with the bootstrapping of DP algorithms.

TD algorithms have an advantage over DP algorithms in that they do not require a model of the environment, of its reward and next-state probability distributions [9]. TD algorithms are different from Monte Carlo algorithms in that they are suitable for continuous (not episodic) task such as stock trading or task with very long episode[1] [10]. At time step $t+1$ they immediately form a target and make a useful update using the observed reward r_{t+1} and the estimate $V(s_{t+1})$.

The simplest TD algorithm, known as TD(0), is

$$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \,. \quad (3)$$

In effect, the target for the TD update is $r_{t+1} + \gamma V(s_{t+1})$. TD(0) in complete procedural form is as follows:

Initialize $V(s)$ arbitrarily, π to the policy to be evaluated
Repeat (for each episode):
    Initialize s

[1]This is a unit for separating interactions between agent and environment into subsequences.

Repeat (for each step of episode)
  $a \leftarrow$ action given by π for s
  Take action a
  observe reward, r, and next state, s'
  $V(s) \leftarrow V(s) + \alpha[r + \gamma V(s') - V(s)]$
  $s \leftarrow s'$
Until s is terminal

4. STOCK PRICE CHANGES IN TD VIEW

It is a highly difficult problem to get a model that can clearly illustrate the stock price changes in terms of actions or policies of investors because they take many different kinds of policies and actions. For example, an investor may make a decision to sell or buy a stock based only on the current price trend of a stock, while another one deeply concerns in fundamental analysis. But it is sufficient for achieving the purpose of this paper to define only the state and reward in the process of stock price change for TD learning, assuming that each agent's policy and actions can be implicitly reflected into the stock data such as a chart showing series of changes of stock prices.

4.1. State

The next to be considered for modeling stock price changes in TD View is the state of a stock at a given time. By the definition of the policy in section 2, a state is the unique input parameter of a policy function. That is, an action is selected depending only on a given state if we assume overall policies of investors rarely change. Thus any kind of data that is referred every day by investors can be considered in defining the state of a stock.

The general daily available data for each stock is represented by the following time series:

- y_O : The price of the first trade of the day
- y_H : Highest traded price during the day.
- y_L : Lowest traded price during the day.
- y_C : Last price that the security traded during the day.
- v : The number of shares (or contracts) that were traded during the day.

Most market experts refer the conventional indicators in technical analysis area. Therefore various kinds of technical indicators as well as the simple kinds of derived data such as return, volume increase, etc. should also be considered. Finally the state of a stock in is paper is defined as a vector with a fixed number of real valued components each of which corresponds to one of the raw daily data or the derived data enumerated above:

$$\vec{\phi}_s = (\phi_s(1), \phi_s(2), ..., \phi_s(n)) \qquad (4)$$

In the rest of this paper, this vector is called *state vector* and is used as the input part of the prediction task.

4.2. Reward

The output part z of most inductive learning algorithms for general prediction task based on the time series approach have the following form:

$$z(t+h) = y(t+h). \qquad (5)$$

That is, they aim to predict the time series value after the h, a fixed prediction time horizon, time steps. But in the task of stock prediction, representing the output (or target) as (5) has a drawback. The price y normally vary greatly for a longer period of time and y for different stocks may differ over several decades, making it difficult to create a valid model. Therefore the *returns* defined as the relative change in price since the previous time step are often used instead [11]:

$$R_h(t) = 100 \cdot \frac{y(t) - y(t-h)}{y(t-h)}. \qquad (6)$$

From this, using one day for the time horizon, h, the immediate reward at time step t, r_t, is defined as follows:

$$r_t = 100 \cdot \frac{y_c(t) - y_c(t-1)}{y_c(t-1)}, \qquad (7)$$

where $y_c(t)$ is the close price of a stock at t. With t substituted by $t+h$, equation (6) corresponds to the *h-step truncated return* with $\gamma = 1$ in the reinforcement learning framework which is used for computing the values of states defined as (1) in section 2.

By the definition of the reward and the target term in equation (3), the current estimated value of a state, that is, the current price trend of a stock, can be computed at every time step. The estimates may result in different values according to the discounting factor γ. If $\gamma = 0$, only the immediate reward, the price rate of change of the next day, is reflected to the values. So γ close to 0 is expected to be suitable for short-term prediction. On the other hand, if $\gamma = 1$, then all the rewards after the state are treated equally regardless of the temporal distance from that state and thus γ close to 1 seems to be suitable for long-term prediction.

t	y_{Ac}	y_{Bc}	r_{At}	r_{Bt}
0	1000	1000	-	-
1	1100	900	10.0	-10.0
2	1150	950	4.55	5.56
3	1300	1100	13.04	15.79
4	1200	1200	-7.69	9.09

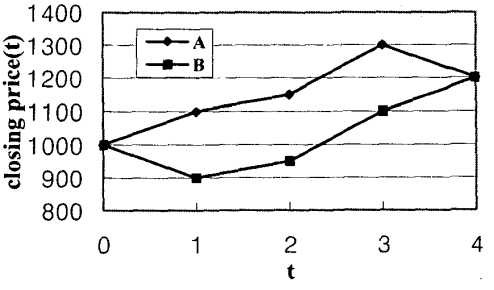

Figure 3: An example of stock price changes.

In stock price prediction, the intermediate prices between the price at current time step and the one after the given time horizon of interest are also meaningful. In this sense, the discounting factor make it possible to represent the target terms more effectively. For example, consider the price changes of stock *A* and *B* in Figure 3. According to the value function (1), the target value of *A* at time step 0 is greater than that of *B* when assumed $\gamma < 1$ and $r_{At} = r_{Bt} = 0$ for $t > 4$, which reflects the price differences at intermediate time steps. But the returns of both stocks calculated by (6), setting the time horizon h to 4, have the same value, *20.0*.

5. FUNCTION APPROXIMATION BY NEURAL NETWORK

According to the definition of the previous section, the state space of a stock is continuous. This means that most states encountered in training examples will never have been experienced exactly before. So it is needed to generalize from previously experienced states to ones that have never been seen. In gradient-descent methods, the approximate value function at time t, V_t, is represented as a parameterized functional form with the parameter vector which is a column vector with a fixed number of real valued components, $\vec{\theta}_t = (\theta_t(1), \theta_t(2), ..., \theta_t(n))^T$ (the T here denotes transpose). In TD case, this vector is updated using each example by:

$$\vec{\theta}_{t+1} = \vec{\theta}_t + \Delta\vec{\theta}_t, \tag{8}$$

where

$$\Delta\vec{\theta}_t = \alpha[r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t)]\nabla_{\vec{\theta}_t} V_t(s_t) \cdot$$

If the value function, V_t, is a linear function of the parameter vector as:

$$V_t(s_t) = \vec{\theta}_t^{\,T}\,\vec{\phi}_{s_t} = \sum_{i=1}^{n} \theta_t(i)\phi_{s_t}(i), \tag{9}$$

the gradient term in equation (8) is reduced to a simple form, just the state vector itself at time t as:

$$\nabla_{\vec{\theta}_t} V_t(s_t) = \vec{\phi}_{s_t} \cdot \tag{10}$$

But the linear method is not suitable for stock price prediction problem, because any strong relationships between inputs and outputs in stock market are likely to be highly nonlinear. A widely used nonlinear method for gradient-based function approximation in reinforcement learning is the multilayer artificial neural network using the error backpropagation algorithm [12][13]. This maps immediately onto the equation (8), where the backpropagation process is the way of computing the gradients.

6. EXPERIMENT

In order to experiment the method presented, data collected from the Korean stock market shown in Table 1 is used.

Table 1: The experiment data.

Training Data		Test Data	
Stocks	100	Stocks	50
Period	2 years	Period	1 year

Table 2 is the list of the indicators consisting the state vector in this experiment. Some of them are themselves conventional indicators in technical analysis and the others are primitive indicators referred commonly in computing several complex indicators. All these input indicators are normalized between 0 and 1 based on their values between a predetermined minimum and maximum.

Training proceeds by on-line updating, in which updates are done during an episode as soon as a new target value is computed. To compute the target value for TD update, $r_{t+1} + \gamma V(s_{t+1})$, of an input example, the weigts of the net learned from the previous examples are used to approximate the term $V(s_{t+1})$.

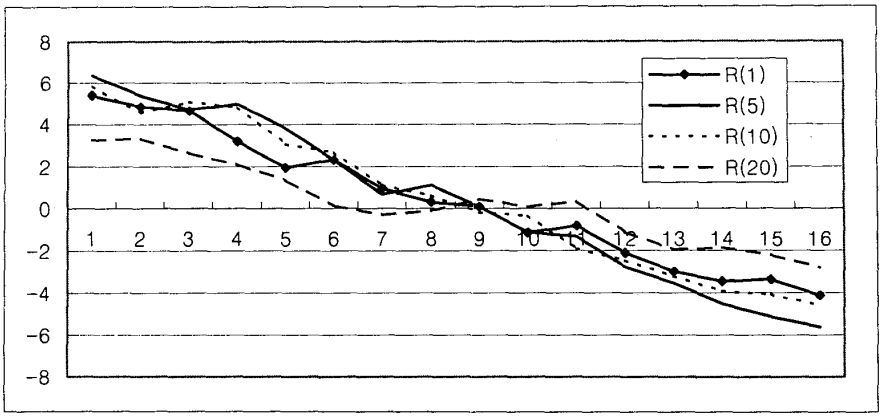

Figure 4: The experiment result.

Table 2: Components of state vector.

Open price	MACD
High price	EOM
Low price	Momentum
Closing price	Williams' % R
Volume	RSI
Price rate of change	Stochastic oscillator
Volume rate of change	Volume oscillator
5-day moving average	5-day disparity
20-day moving average	20-day disparity
60-day moving average	60-day disparity

Though the stock price change is a continuous task, in this experiment, the total period, 2 years, of subsequent price changes of each stock is regarded as an episode. Therefore the training is performed with 100 episodes and each episode corresponds to a stock included in Table 1.

Figure 4 shows the result of the test experiment after the training of 3000 epoch with 60 hidden nodes using γ = 0.8. The predicted values for the test input patterns are classified into 16 distinct grades, each of which corresponds to a predefined range of the value. For this, a heuristic function which determines the upper and lower limits of each range is used. This function considers the maximum and minimum of actual returns over a period, found in the training examples. The average actual returns of all the examples that are predicted to belong to each grade are plotted. $R(1)$, $R(5)$, $R(10)$, and $R(20)$ are the average returns truncated after 1, 5, 10 and 20 days respectively and are computed from the n-step truncted return $r(n)$ of each example:

$$r(n) = r_{t+1} + \gamma r_{t+2} + \ldots + \gamma^{n-1} r_{t+n}, \qquad (11)$$

with γ = 0.8. According to this result, as expected, higher grades result in higher returns in overall and vice versa. But as shown in Figure 5, the majority of the test samples cannot be predicted to have extremely positive or negative values.

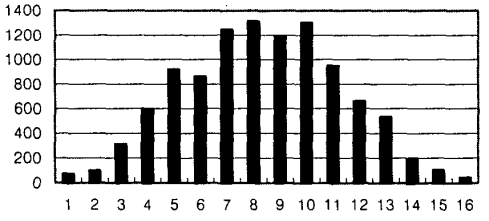

Figure 5: The grade distribution of the test examples.

Table 3 shows the average root mean-squared (RMS) errors between the predicted grades and the grades of actual returns. Here the grade of each return is also computed by the same fuction mentioned above. The performance in terms of $R(1)$ and $R(20)$ was not as satisfactory as that of the others, showing that making the period of prediction extremely shorter or longer results in decay of performance.

Table 3: RMS errors between value grades and return grades.

Grade Range	RMS			
	$R(1)$	$R(5)$	$R(10)$	$R(20)$
1-4	3.84	3.07	3.80	4.65
5-12	3.14	2.62	2.81	3.46
13-16	4.11	3.38	3.49	4.99

7. CONCLUSION

This paper presents an approach for applying reinforcement learning to the problem of stock market prediction. An alternative way of representing target values for the patterns of stock price changes is also presented. The experimental result shows that the proposed method might be utilized as a more useful indicator for stock trading. The state-values calculated by the method are different from the conventional indicators in that they intend to represent directly the expected cumulative reward, the price rate of change, not the auxiliary information for making buying or selling signals.

The proposed approach provides a basis for applying reinforcement learning to stock market prediction and it might be possible to extend this to applying another reinforcement algorithms such as TD(λ), instead of TD(0), where the definition of the reward is slightly different from that of this paper.

REFERENCES

[1] S.M. Kendall and K. Ord, *Time Series*, Oxford University Press, New York, 1997.

[2] R.J. Kuo, "A decision support system for the stock market through integration of fuzzy neural networks and fuzzy Delphi," *Applied Artificial Intelligence*, 6:501-520, 1998.

[3] N. BaBa and M. Kozaki, "An intelligent forecasting system of stock price using neural networks," *Proc. IJCNN*, Baltimore, Maryland, 652-657, 1992.

[4] S. Cheng, *A Neural Network Approach for Forecasting and Analyzing the Price-Volume Relationship in the Taiwan Stock Market*, Master's thesis, National Jow-Tung University, 1994.

[5] R.S. Sutton and A.G. Barto, *Reinforcement Learning*, The MIT Press, 1998.

[6] R.S. Sutton, "Learning to predict by the method of temporal differences," *Machine Learning*, 3:9-44, 1988.

[7] P.J. Werbos, "Building and understanding adaptive systems: A statistical/numerical approach to factory automation and brain research," *IEEE Transactions on Systems, Man, and Cybernetics*, 17:7-20, 1987.

[8] T.M. Mitchell, *Machine Learning*, The McGraw-Hill Companies, Inc, 1997.

[9] C.J. Watkins, *Learning from Delayed Rewards*, Ph.D. theis, Cambridge University, 1989.

[10] M.H. Kalos and P.A. Whitlock, *Monte Carlo Methods*, Wiley, New York, 1998.

[11] T. Hellstroem, *A Random Walk through the Stock Market*, Ph.D. theis, University of Umea, Sweden, 1998.

[12] C.W. Anderson, *Learning and Problem Solving with Multilayer Connectionist Systems*, Ph.D. theis, University of Massachusetts, Amherst, 1986.

[13] S.E. Hampson, *Connectionit Problem Solving: Computational Aspects of Biological Learning*, Birkhauser, Boston, 1989.