International Journal of Grid and Distributed Computing
Vol. 11, No. 2 (2018), pp.11-20
http://dx.doi.org/10.14257/ijgdc.2018.11.2.02

Deep Reinforcement Learning based Multi-Agent Collaborated Network for Distributed Stock Trading

Jung-Jae Kim[1], Si-Ho Cha[2], Kuk-Hyun Cho[1] and Minwoo Ryu[3*]

[1]*Department of Computer Science, Kwangwoon University, Seoul, South Korea*
[2]*Department of Multimedia Science, Chungwoon University, Incheon, South Korea*
[3]*Telecom R&D Center, Korea Telecom (KT), Seoul, 06763, South Korea*
[1]*{kjj6929, chokh}@kw.ac.kr,* [2]*shcha@chungwoon.ac.kr,* [3]*mw.ryu@kt.com*

Abstract

Recently, interest in financial transactions is increasing, and the number of investors in the stock market is increasing. These investors are applying financial analysis methods to stock trading in order to gain more profits, and combining with artificial intelligence techniques has made it possible to achieve returns in excess of the market average. As a result, the stock trading system based on reinforcement learning has attracted attention, and in recent years, studies are being conducted to optimize financial time series data by Multi-Agent Reinforcement Learning (MARL). However, MARL, which is used in existing stock trading, cannot be fully collaborated because of lack of generalization of experience. Therefore, in this paper, we propose Multi-agent Collaborated Network (MCN) that can share and generalize the experience by agent, and experiment on collaboration in distributed stock trading.

Keywords: *We would like to encourage you to list your keywords in this section*

1. Introduction

Cryptocurrency nowadays is emerging rapidly, and interest in financial transactions is increasing. As a result, the number of investors in the stock market is increasing, and these investors want to predict the peak or the bottom of the financial time series data in their own way and to earn more money with the own capital. However, financial time series data is usually expressed as a random walk, and it follows the Brownian movement, it is very difficult to predict price fluctuations or expected returns [1]. To reduce this complexity, individual investors are increasingly applying financial analysis methods to stock trading. In financial engineering, fundamental analysis or technical analysis techniques are used to predict stock prices, but due to noise and non-stationarity of financial time series data, financial analysis based market analysis methodology does not have such a great effect. For this reason, for more accurate prediction and high profit, researchers in artificial intelligence field, especially machine learning, have tried to study the combination of artificial intelligence technique and technical analysis technique. Several studies have suggested that it is possible to achieve returns in excess of the market average by combining machine learning and technology analysis techniques [2].

A machine learning based stock trading system deals with the optimization of forecasting based on supervised learning based on financial time series data in one integrated structure. There is a limitation in that is no big difference in the system trading and profitability of the system trading by repeating buying and selling in a specific pattern through prediction of the rise and fall of the stock price. Recently, research has been proposed to use

Received (October 15, 2017), Review Result (December 19, 2017), Accepted (January 20, 2018)
* Corresponding Author

ISSN: 2005-4262 IJGDC

reinforcement learning as an alternative to optimize the stock trading system [3]. Reinforcement learning aims not to optimize the sum of squared error as supervised learning but to acquire the optimal policy for the learning agent to obtain the maximum average rewards from the environment.

Existing reinforcement learning based trading systems were focused on a single time series such as the market composite index, and which is able to earn higher profit than market average. However, there is a problem that a single agent has a limited policy. Therefore, Multi-Agent Reinforcement Learning(MARL) is introduced to divide the transaction system such as buying agent, selling agent, and the like into one agent for a single purpose. In other words, by configuring transaction system as a multi agent and establishing the optimal policy for each job, and it allows various and detailed policies can be learned. As a result, it was able to achieve an excess profit that exceeded the market average. However, this existing MARL based trading system is not a true collaboration because it utilizes multiple agents in a situation where there is a sequence such as learning the buy signal policy and optimizing the decision for the next agent [4]. Also, by learning the optimal policy for a single type of stocks, moving the agent to a different type of stocks requires learning the optimal policy again. And finally, when it is building a portfolio for another type of stocks, it takes a lot of time for new agents to be added and the agent to learn optimal policy.

In order to solve these problems, it is necessary to have a reinforcement learning structure that can share the experiences about the optimal policy learned by the agents for each type of stocks. In this paper, we propose a Multi-agent Collaborated Network (MCN) that can share learning experiences between different agents based on Long Short-Term Memory (LSTM) structure of deep learning [5]. This makes it possible to generalize patterns that occur between different time series data. In addition, learning speed is improved by sharing learning experiences for collaboration and newly added agents can quickly adapt to the shared learning experience.

The composition of this paper is as follows. Section 2 describes the reinforcement learning and LSTM used in this paper. Section 3 gives an overview of the MCN. Section 4 describes in detail the method. Section 5 describes the experimental method and results. Finally, section 6 describes the conclusion of the proposed method.

2. Background

This chapter discusses the basics of reinforcement learning and LSTM.

2.1. Reinforcement Learning

Reinforcement learning is an approach to automating decision making to achieve an objective and give a goal to an agent. In this paper, reinforcement learning is described by the notation of Sutton [6]. In the reinforcement learning framework called the Markov Decision Process (MDP), there is generally one agent and an environment (E) that can interact during the discrete time process (step; $t = 0, 1, 2, \cdots, T$). Then the agent observes the state of the environment ($s_t \in S$), and then selects an action ($a_t \in A$) that the agent can perform based on the state. In this way, the state of the environment and the behavior of the agent are called a policy ($\pi = (s_t, a_t)$). If the agent performs an action after observing the state, the environment responds to the agent's action a_t, and transforms it into other state s_{t+1}, and returns the reward ($r_{t+1} \in R$) to the agent. Figure 1 shows the framework of such reinforcement learning.

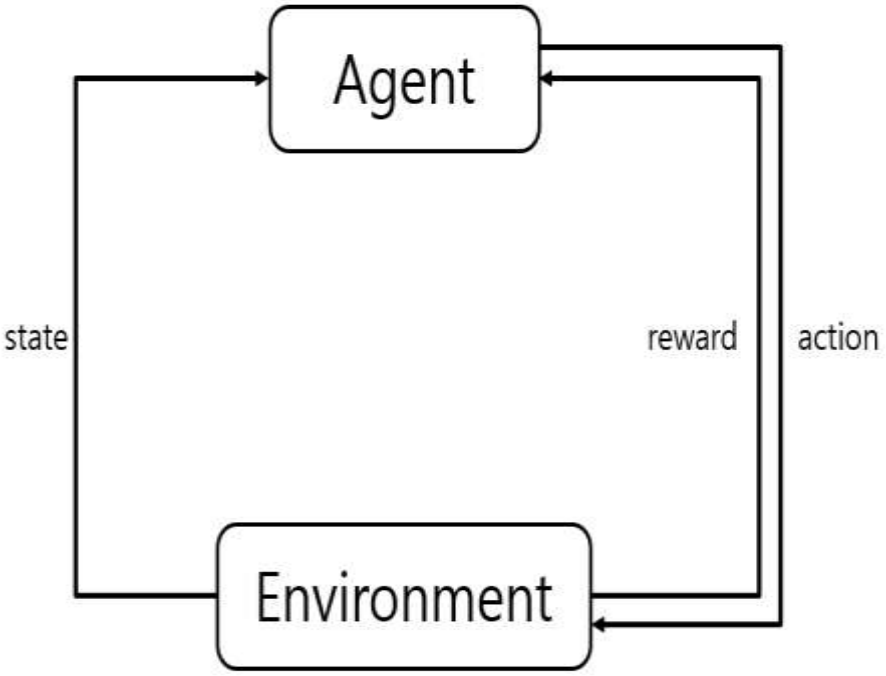

Figure 1. Framework of Reinforcement Learning

In this framework, the purpose of the agent is to learn the optimal policy to maximize rewards. The policy means to map the optimal action that the agent can take in each state. Thus, it means maximizing the value function $Q^{\pi}(s,a)$ for action according to the state-action pair (s,a). The value function $Q^{\pi}(s,a)$ obtained by interacting with the environment from the initial state (s_0) to the final state (s_T) is define as Equation (1).

$$Q^{\pi}(s,a) = E_{\pi}\{r_1 + \gamma r_{1+1} + \cdots + \gamma^{T-t-1} r_T \mid s_t = s\,, a_t = a\} \quad (1)$$

γ is defined $0 \leq \gamma \leq 1$, which means a discounting factor. At this time, the optimal policy Q^* is defined as Equation (2) for all s and a.

$$Q^*(s,a) = max_{\pi}\, Q^{\pi}(s,a) \quad (2)$$

Therefore, the fact that the agent always makes the best decision means that it can act to maximize the policy value in the state.

2.2. LSTM (Long Short-Term Memory)

In this paper, only the idea of LSTM is borrowed. Figure 2 shows the memory block of LSTM. LSTM is a model developed from Recurrent Neural Network (RNN), and is a structure for time series data processing. It has improved performance over long term memory compared to existing RNN by controlling input, storage, and output of incoming data in time unit from previous node. And LSTM stores important data among input data in C_t called cell state and moves it to the next node to share information that can be forgotten in time series data. This enables smooth collaboration among multiple agents.

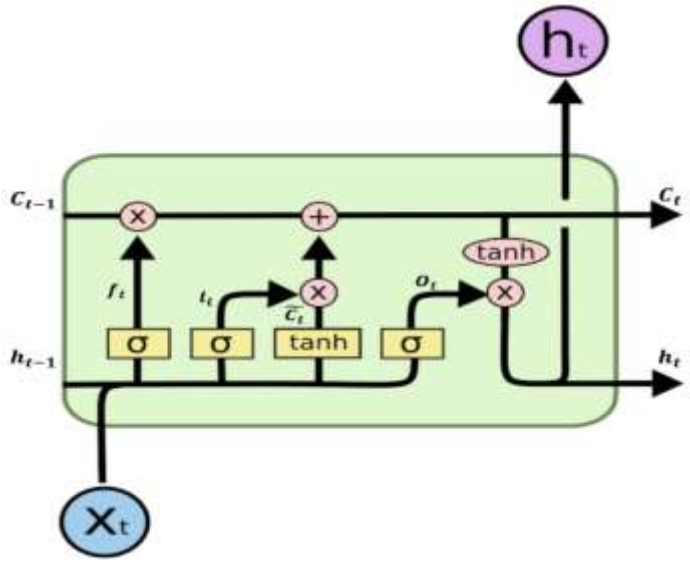

Figure 2. Memory Block of LSTM

3. Overview of Multi-agent Collaborated Network (MCN)

This chapter overview the overall framework and single agent of proposed architecture.

3.1. Basic Architecture

Stock trading is a non-zero-sum game with a typical goal oriented that of making more profit with limited capital. And therefore can be defined more simply than collaboration among different multi-agents. Figure 3 shows the basic architecture of a distributed stock trading system using MCN. In an MCN, an agent selects one of the actions of selling, buying, or staying at every time step for one stock item. By learning their optimal action in time series data, they learn the optimal policy at that time. In addition, because of the long term memory allows long-term memorization and co-operation of the main learning patterns among agents, if the stock predicts less profit than the other stocks, the stock is sold to other agents cooperate to get bigger profit.

3.2. Structure of Agent in MCN

MCN focuses on micro-management tasks for agents in the stock trading system. Therefore, there are combinations of stocks (heterogeneous) and there are infinite states due to the nature of stocks. In order to enable each agent to operate in this situation, the Actor-Critic Framework [7] is followed. Figure 4 shows the basic structure of an actor-critic framework agent with long-term memory in MCN.

The agent of the MCN has a separate structure that deals with the value function and the policy function and the policy like the existing actor-critic model. At this time, the policy structure is called actor because it is used to select an action, and the value estimator for evaluating the value function is called critic because it is responsible for evaluating the selected action. Critic must criticize the policy the actor follows, so the learning process is always on-policy. In MCN, both actor and critic have their own long-term memory to share memories.

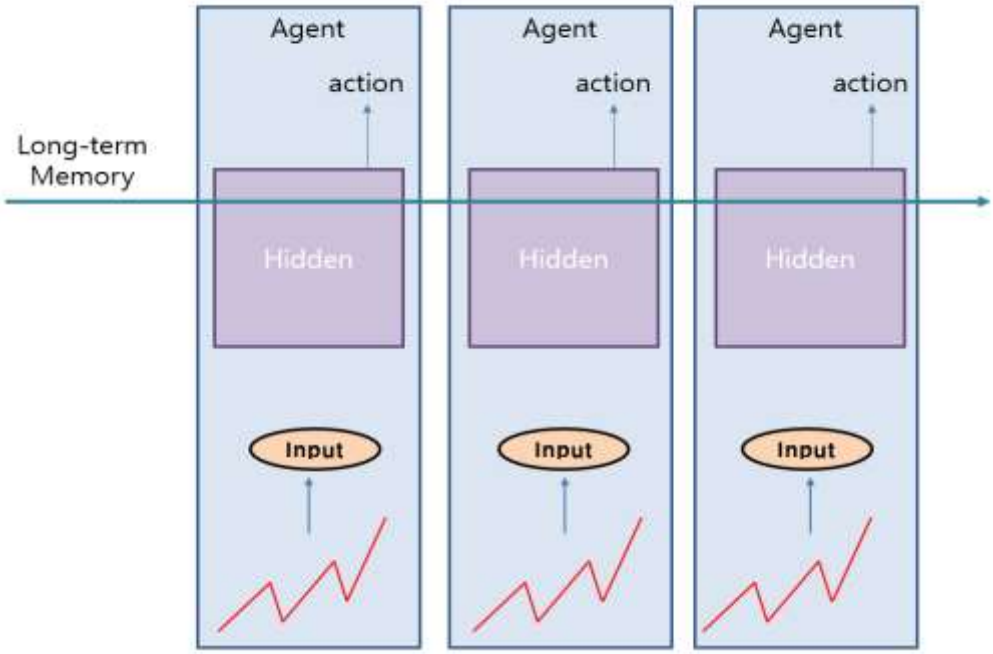

Figure 3. Basic Architecture of MCN

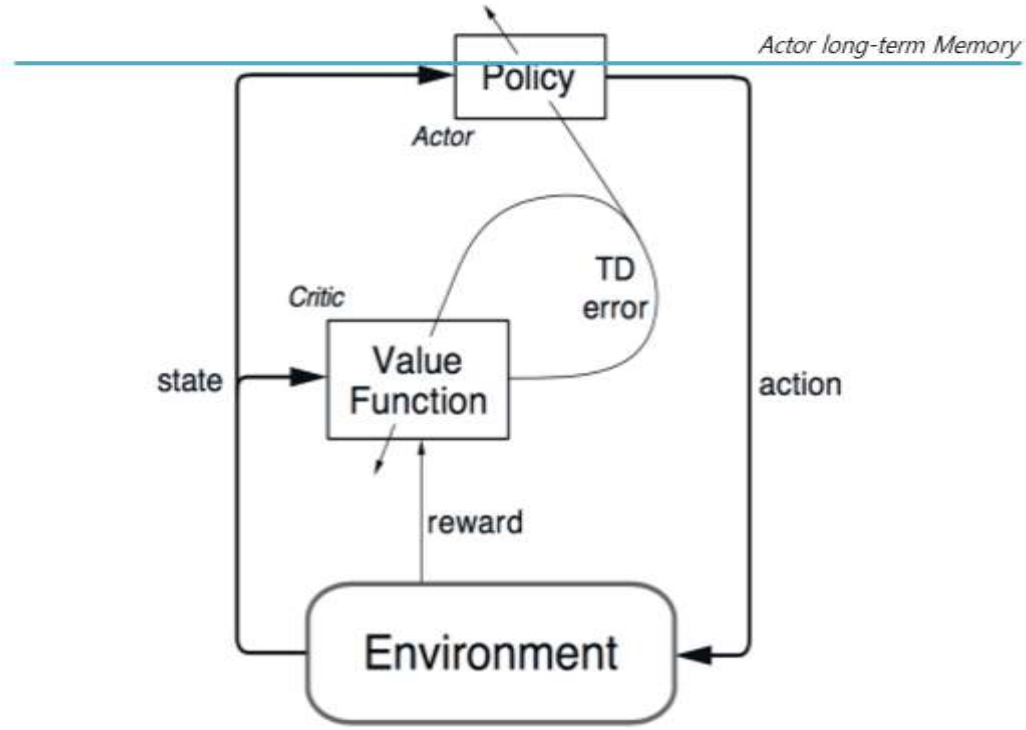

Figure 4. Basic Architecture of MCN

4. Learning Process of Multi-agent Collaborated Network (MCN)

One of the ways to achieve good performance in machine learning is to effectively select or extract input features. Reinforcement learning is especially important because the outcome depends on how the state, action and reward are designed. Therefore, this chapter describes how State, Action, Reward and MCN are learned.

4.1. State, Action, Reward

MCN basically uses deep reinforcement learning. Therefore, the issue of feature selection is relatively small due to the high abstraction of deep learning compared to the existing reinforcement learning method. And MCN uses stock price information in minutes. In addition, Granville's Law [8] is used in the technical analysis method for high accuracy due to the characteristics of stock prices that are not easy to predict. Granville's law describes the future trend of stock prices in terms of the moving average, which is the average of stock prices over a certain period. Therefore, we assume that the moving average has a pattern that can predict the stock price. The moving average can be defined as Equation (3).

$$MA_t^n = \frac{(\sum_{i=t-n+1}^{t} close_t)}{n} \tag{3}$$

Where $close_t$ is the closing price at the relevant minute. In MCN, we use 9 kinds of data for stock price prediction. In addition, we use the sliding window method for time series prediction to group the previous 10 minutes into one window. Therefore, the input feature to predict the stock after one minute is 90 pieces. Table 1 shows the input characteristics used in MCN.

Table 1. Symbol Table of State Learned in MCN

Symbols	Definition
OP_t	Starting transaction amount in the last 24 hours
CP_t	Last transaction amount in last 24 hours
$MinP_t$	Minimum transaction amount in last 24 hours
$MaxP_t$	Maximum transaction amount in last 24 hours
$AvrP_t$	Average transaction amount in last 24 hours
$VolToday_t$	Stock trading volume in the last 24 hours
$VolOne_t$	Stock trading volume in yesterday
$VolWeek_t$	Stock trading volume in week
MA_t	Moving average of last 10 minutes

Therefore, the input set $InputSet_t$ of the MCN can be defined as the following Equation (4).

$$\begin{aligned} MinuteSet_t &= \{OP_t, CP_t, MinP_t, MaxP_t, AvrP_t, VolToday_t, VolOne_t, VolWeek_t, MA_t\} \\ InputSet_t &= \{MinuteSet_{t-9}, MinuteSet_{t-8}, \cdots, MinuteSet_t\} \end{aligned} \tag{4}$$

The agents of an MCN take only three kinds of actions, BUY, STAY, SSELL. Finally, the reward given to the agent is given as the sum of all the stock values at the time the SELL action is taken. Through this, all actions of the agents are affected by global rewards.

4.2. Learning Algorithm

MCN is based on an actor-critic algorithm. The actor-critic framework uses approximations of infinite states through deep learning. Equation (5) represents the expression of the basic actor. θ at this time can be expressed as Equation (6).

$$\begin{aligned} \mathrm{J}(\theta) &= \mathrm{E}_{\pi_\theta}[r] \\ &= \sum_{s\in S} d(s) \sum_{a\in A} \pi_\theta(s,a) R_{s,a} \end{aligned} \tag{5}$$

$$\Delta\theta \approx \alpha \frac{\partial p}{\partial \theta} \tag{6}$$

In the MCN, the policy gradient learned by the agent is assumed to be an experience and stored in long-term memory. The procedure for storing long-term memory is the same as for LSTM. The first step is to decide whether new information is stored in cell state C_t. Equation (7) serves to determine which value to update through the policy gradient for the current input and input.

$$i_t = \sigma(W_i \cdot [\theta_t, x_t] + b_i) \tag{7}$$

The next step is to create a new candidate value that can be added to the cell state and is defined as Equation (8).

$$\tilde{C}_t = \tanh(W_c \cdot [\theta_t, x_t] + b_c) \tag{8}$$

The last step is the step of updating the long-term memory C_{t-1} to C_t, which is defined as Equation (9).

$$C_t = C_{t-1} + i_t * \tilde{C}_t \tag{9}$$

Through this process, the learning experience of each agent can be stored in the long term memory, and the learning speed is improved because the long-term memory can experience more various situations by the multi-agent.

5. Experiment and Results

In this chapter, the stock trading system constructed based on the proposed structure is experimented and analyzed. The actor and critic of the MCN have three hidden layers, each consisting of 50 neurons. All agents fixed the learning rate to 0.01. also, with the discounting factor of 0.95, the most recent results have had a large effect. The exploration factor was 0.1. And MCN learned 5000 episodes and then verified using verification data. The structure of the data is shown in Table 3. The data were randomly selected from five KOSPI in Korea.

Table 3. Table Label

Training Data		Validation Data	
Periods	Number of Data	Periods	Number of Data
2016/01 ~ 2016/12	366 day / 1830 * 5	2017/03 ~ 2017/06	122 day / 610 * 5

Figure 5 shows the trends in the returns obtained during the episode of the MCN. This is the average rate of return obtained from training data and learning data. A total of 5,000 runs were performed. The graph is that graph showing average data for 100 times. Table 4 shows the final rate of return for each episode. Analysis of the graphs and tables reveals negative returns in the early stages of learning, but it is getting better and more profitable, with a return of around 45 percent from the moment it finally reaches 5,000 episodes. In particular, we can see that the rate of return has risen considerably since about 1000 episodes. After 5,000 episodes, one of the five stocks stopped trading and traded only four stocks. This can be interpreted as stopping the transaction by collaborating with agents of the remaining stock because the return on the other stock cannot be expected. Therefore, the learning speed is slightly higher than that of a single agent, which is repeated until it is initially learned.

Table 4. Symbol Table of State Learned in MCN

Episodes	Yield (%)
1000	21.16
2000	31.47
3000	31.47
4000	38.87
5000	46.68

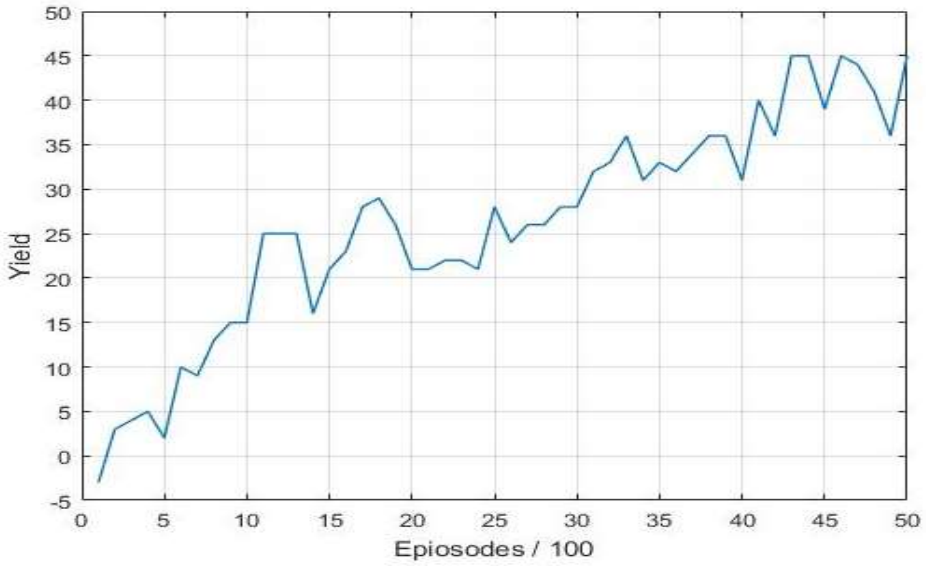

Figure 4. Experimental Yield Rate of MCN

6. Conclusion

In this paper, we design the MCN to perform distributed stock trading through the agents of reinforcement learning through collaboration. Also, through experiment, it was confirmed that the agents corresponding to each item recorded high rate of return at high speed.

However, in the case of MCN, there is no analytic portfolio selection method that selects the optimal stocks. Therefore, future researches should be conducted to select the optimal stock among the various stocks.

Acknowledgment

This paper is a revised and expanded version of a paper entitled 'Multi-Agent Reinforcement Learning based Collaboration for Distributed Stock Trading Method' presented at GST2017, Jeju National University, Jeju, Korea, 1st December, 2017.

References

[1] B. G. Malkiel, "A Random Walk Down Wall Street", Norton, New York, **(1996)**.
[2] T. M. Sands, D. Tayal, M. E. Morris and S. T. Monteiro, "Robust stock prediction using support vector machines with particle swarm optimization", Evolutionary Computation (CEC), 2015 IEEE Congress on, Sendai, Japan, **(2015)** May 25-28.
[3] M. Tirea, I. Tandau and V. Negru, "Multi-agent Stock Trading Algorithm Model", Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2011 13th International Symposium, **(2013)**.
[4] Y. Luo, K. Liu and D. N. Davis, "A multi-agent decision support system for stock trading", IEEE Network, vol. 16, no. 1, **(2002)**.

[5] F. A. Gers, J. Schmidhuber and F. Cummins, "Learning to forget: continual prediction with LSTM", ICANN '99, **(1999)**.
[6] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction", The MIT Press, **(1998)**.
[7] V. R. Konda, John N. Tsitsiklis, "Actor-Critic Algorithms", In Advances in Neural Information Processing Systems 12: proceedings of the 1999 Conference, Denver, Colorado, **(2000)**, pp. 1008-1014.
[8] R. D. Edwards and J. Magee, "Technical Analysis of Stock Trends", John Magee. Inc, **(1974)**.

Authors

Jung-Jae Kim, He received his B.S. and M.S. degrees in Computer Science from Kwangwoon University, Seoul, Korea in 2013 and 2015, respectively. He is currently working on Ph.D. course at the Department of Computer Science, Kwangwoon University, Seoul, Korea. His research interests include Intelligent Network System, Machine Learning, Deep Learning and Deep Reinforcement Learning.

Si-Ho Cha, He is a professor in the Department of Multimedia Science, Chungwoon University, Incheon, South Korea. He received his Ph.D. degree in Computer Science from Kwangwoon University, Seoul, South Korea in 2004. From 1997 to 2000, he worked as a senior researcher at Daewoo Telecom R&D Center, Korea. His research interests include network management, wireless sensor networks, vehicular ad hoc networks, semantic web, and web of things.

Kuk-Hyun Cho, he received his B.S. degree in Electronic Engineering from Hanyang University, Seoul, Korea in 1977 and his M.S. and Ph.D. degrees in Electronic Engineering from Tohoku University, Sendai, Japan in 1981 and 1984, respectively. Since 1984, he has been a Professor in the Department of Computer Science and Engineering, Kwangwoon University, Seoul, Korea. From 1998 to 2000, he was President of Open Standards and Internet Association (OSIA). His research interests include network and service management, wireless sensor networks, and vehicular ad hoc networks

Minwoo Ryu, He received his B.S. degree in Internet Information Processing from Yeoju Institute of Technology, Korea in 2007, and his M.S. and Ph.D. degrees in Computer Science from Kwangwoon University, Seoul, Korea in 2009 and 2012, respectively. From 2011 to 2016, he worked as a research scientist at Korea Electronics Technology Institute (KETI), Korea. He is now a research scientist in the KT R&D center, Korea Telecom (KT), Korea. His research interests include Internet of Things, Semantics, Cognitive Computing, Intelligence Service and Vehicular Ad Hoc Networks.

