공학석사학위논문

강화학습 기법을 사용한 효율적인 자산 배분 알고리즘

Efficient Portfolio Management using Deep Reinforcement Learning

2021년 2월

서울대학교 대학원

컴퓨터공학부

김정훈

강화학습 기법을 사용한 효율적인 자산 배분 알고리즘

Efficient Portfolio Management using Deep Reinforcement Learning

지도교수 강 유

이 논문을 공학석사 학위논문으로 제출함

2020년 10월

서울대학교 대학원

컴퓨터공학부

김정훈

김정훈의 석사 학위논문을 인준함

2020년 12월

위원장 김 선 (인)

부위원장 강 유 (인)

위원 김건희 (인)

Abstract

Efficient Portfolio Management using Deep Reinforcement Learning

Jung hoon Kim
Department of Computer Science & Engineering
The Graduate School
Seoul National University

Given historical stock prices in a portfolio, how can we efficiently allocate weights to maximize cumulative returns? Portfolio management is widely used in financial planning tasks that aim to maximize profits and minimize risks at the same time. Existing methods using deep learning and reinforcement learning algorithms have achieved significant improvement in efficient allocation problems. However, they perform poorly in downward trends of the financial markets because of their ability to deal with sudden downward trends.

In this paper, we propose **P**ortfolio **M**anagement with **S**hort **P**osition (PMSP) which employs a reinforcement learning algorithm to search the optimal allocations by adding a short position strategy to make profits even in downward trends. PMSP extracts and refines features from historical prices of stocks in order to reflect market dynamics. It then uses Deep Deterministic Policy Gradient (DDPG) algorithm for faster convergence of parameters by adding the concepts of memory buffers and target networks. Finally, instead of using the softmax function which transforms the sum of the input values 1 so that the function cannot apply a short position strategy, we apply the hyperbolic tangent at the end of the model to allow negative values, which allows the model to make short positions and earn profits even in downward trends. Experimental results show that PMSP achieves the highest portfolio value, which earns 102% profits in a year, giving state-of-the-art performance.

Keywords : Portfolio Management, Deep Deterministic Policy Gradient, Long Short-Term Memory, Short Position Strategy

Student Number : 2019-29394

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Given historical prices of stocks in the portfolio, how can we efficiently allocate weights to maximize cumulative returns? A portfolio refers to a collection of assets such as stocks, shares, and cash. Portfolio management is an asset allocation that selects the right investment policy for assets in the portfolio to minimize risks and maximize returns.

The rise of machine learning and deep learning has attracted the interest of its usage in financial fields, especially in portfolio management. Because portfolio management involves sequential decision making by continuously reallocating a number of funds into assets, many studies have attempted to employ deep reinforcement learning algorithms to find the optimal asset allocation [1, 2, 3, 4]. However, the existing methods are unable to solve two critical challenges; the models 1) cannot capture market dynamics, in which distributions of stock markets shift rapidly because of characteristics that reflect the fickleness of investors, and 2) are vulnerable to downward trends of the financial markets.

In this paper, we propose **P**ortfolio **M**anagement with **S**hort **P**osition (PMSP), a novel approach for portfolio management that considers both market dynamics and an ability to make short positions.

We build off the work of State Augmented Reinforcement Learning (SARL) proposed by [1], which increases the allocating performance by adding an additional information such as market trend predictions guiding whether prices of stocks will rise or fall into original states which only contain the daily price change ratio of the past days. Based on SARL, we use the relative price changes between prices (e.g. open / close) instead of the raw price data as features to capture market dynamics. Furthermore, PMSP comes up with Deep Deterministic Policy Gradient (DDPG), an off policy policy gradient actor-critic algorithm, and adding a short position strategy, which allows traders to sell stocks first and repurchase later at a lower price. In order to make a short position strategy, we apply the hyperbolic tangent at the end of the model to allow negative values.

We summarize our main contributions as follows:

- **Addition of short positions to boost earning profits.** We enable PMSP to take short positions on stocks in the portfolio so that it can make profits even in the downward trends of markets.
- **Capturing market dynamics and stabilizing learning processes.** We use the relative price changes between prices of open, high, low, and close to capture market dynamics. Furthermore, we employ the DDPG algorithm to acheive the faster convergence of parameters by adding the concepts of memory buffers and target networks.
- **Experiments.** Experimental results show that PMSP provides the best portfolio value, improving the portfolio value by 114% when compared to the SARL model. (see Figure 3)

Chapter 2

Related Works

We review previous researches on the portfolio management problems using deep reinforcement learning algorithms.

As the availability of large scale market data has been growing exponentially, it is natural to apply deep learning-based models that can capture hidden patterns of stock markets in portfolio management. Early studies have employed neural network models for market behavior predictions and proved their effectiveness in stock price prediction and asset allocation [5, 6, 7]. However, these deep learning-based methods cannot interact with markets which implies that they are unable to capture market dynamics

Deep reinforcement learning-based models, however, have proved their outstanding performance in decision-making problems by choosing the actions in every time step. The interaction between agents and environments can reflect market dynamics. [8] proposed a model that combines deep learning with reinforcement learning, and this integration became the basis in the financial field. [2] integrated a recurrent model with a reinforcement learning based method that suggests when to buy or sell stocks and how to allocate assets efficiently. [3] employed a model that uses a model-free Deep Deterministic Policy Gradient (DDPG) [9] to dynamically allocate assets composed of various cryptocurrencies. Furthermore, [4] used the DDPG and Proximal Policy Optimization (PPO) [10] to optimize asset portfolios. Deep reinforcement learning is also used to hedge the portfolio of derivatives under transaction costs [11]. [1] proposed a method called Stated Augmented Reinforcement Learning (SARL) that augments asset information with price signal predictions of assets, where they can be solely based on financial data such as asset's historical prices and optimizes the allocation of assets. Whereas these studies brought a significant improvement in the area of portfolio management, there has been no study that applies short positions in portfolio management.

Chapter 3

Preliminaries

Portfolio management is the art of selecting the optimal asset allocation in the portfolio. It is a fundamental financial planning task that targets to maximize returns and minimize calculated risks. A portfolio is composed of many assets such as stocks, cash, or mutual funds, and it is made up of their related information that affects the market. Our primary goal in portfolio management is to find the best allocation of assets that maximizes the total returns. We assume that every asset in the portfolio is liquid enough whenever we want to buy and sell. In this paper, we employ deep reinforcement learning algorithm to find the optimal weight of each asset in the portfolio.

Reinforcement learning is well known for solving decision-making problems. The agent and the environment are the main characters of reinforcement learning. Figure 1 expresses the interaction between the environment and the agent.

- **Environment and agent** The environment is where the agent lives in. For our case, it contains all the viable information of assets to the agent such as historical prices and price change ratio of stocks. The agent is a learner and decision-maker. At every time step t, the agent selects an action based on the information S_t that the environment provides.

- **Action space** Action space is the set of all actions available in a given environment. There are two kinds of action spaces. Discrete action spaces have a finite number of actions that the agent can choose. Otherwise, continuous

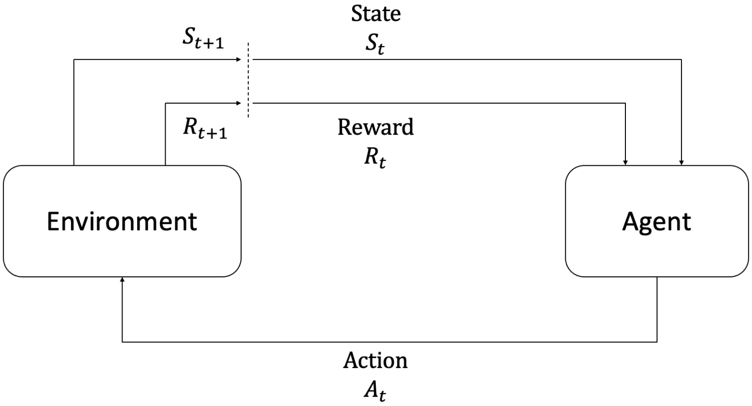


Figure 1: Reinforcement learning process. Based on a state S_t, the agent selects an action A_t, and the environment responds to the action and presents a new situation S_{t+1} with a reward R_{t+1}.

action spaces have an infinite number of possible actions. For the problem of portfolio management, the action space can be the possible amount of weight for the allocation in which the value is continuous.

- **Reward and return** Once the environment receives an action, it presents a new state, S_{t+1}, with a reward, R_{t+1} for the corresponding action. A reward is a scalar that tells how good or bad the action is in a given state. In general, Reinforcement Learning seeks to maximize the expected return.
- **Value functions** Value functions are the expected return of a state. They estimate how good it is for the agent to be in a given state (state-value function) or how good it is to perform a given action in a given state (action-value function). The value functions can be estimated from experience, and most reinforcement learning algorithms involve estimating value functions.
- **Policy** Policy is a mapping function that guides the agent to choose actions

in a given state. The agent changes its policy to maximize the total amount of reward it receives at the end.

Chapter 4

Proposed Method

We propose PMSP, an efficient asset allocation for portfolio management. The technical challenges are as follows:

- **How can we capture market dynamics?** Instead of using the raw price data, we use the relative price changes between prices (e.g. p_t^o/p_t^c, where p_t^o is an open price and p_t^c is a close price), and these features can capture the interaction of different prices.
- **How can we stabilize the learning process of the model?** SARL model faces a slow convergence problem because of the model it uses, Deterministic Policy Gradient (DPG) algorithm. To solve the corresponding problem, we employ the DDPG algorithm to enable faster convergence of parameters by adding memory buffers and target networks.
- **How can we make the model to earn consistent profits even in downward trends?** We apply a short position strategy into the model so that it can make profits even though the stock prices fall.

We first provide a brief overview of PMSP in Section 4.1. Then we explain the details of how we create an augmented state using the Long Short Term Memory (LSTM) model in Section 4.2, how we concatenate an original state and the augmented state in Section 4.3, and how we calculate the optimal weight of each asset by Deep Deterministic Policy Gradient (DDPG) algorithm in Section 4.4.

4.1 Overview

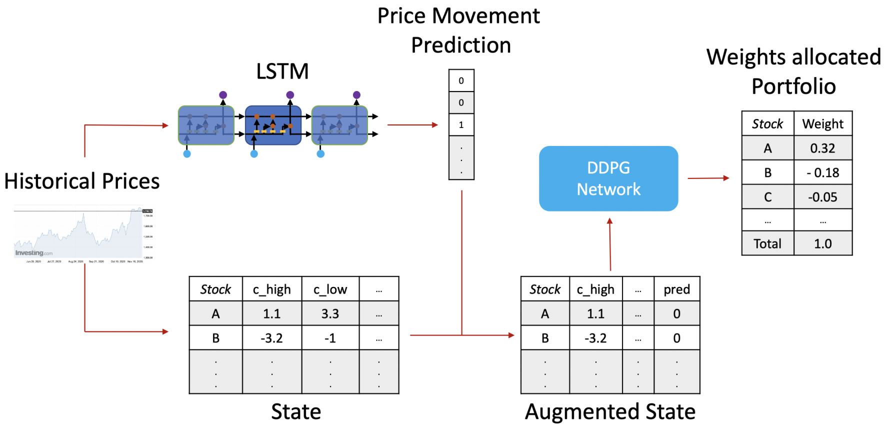
Given daily price data of stocks in the portfolio consisting of opening p_t^o, lowest p_t^l, highest p_t^h, and closing p_t^c prices where t refers to a day, our goal is to find the optimal allocation of stocks that maximizes the profit at the end of the day. To answer this problem, we design PMSP, a portfolio manager which optimizes the allocation of asset by using a deep reinforcement learning algorithm. PMSP is built off the work of [1] to come up with the DDPG algorithm and a short position strategy. Figure 2 shows the overview of PMSP.

We first create a state for each day, s_t, which contains the past asset prices data. [1] uses the normalized closing prices to create the state. However, the normalization process is a difficult problem for stock markets because of the dynamics that can change the whole distribution of stocks. Instead, we adopt [12], defining 4 temporal features ($\mathbf{x}_t^s$) to express the trend of each stock at a time step t. Table 1 shows the features that we use for the state. According to [12], these features can 1) normalize the prices of different stocks, 2) and explicitly capture the interaction of different prices (e.g. open and close).

Table 1: Features of daily trends of stocks.

Features	**Example**
$c_{open}, c_{high}, c_{low}$	e.g. $c_{open} = p_t^o/p_t^c - 1$
n_{adj_close}	e.g. $n_{adj\_close} = p_t^c/p_{t-1}^c - 1$

Next, we create an augmented state which contains the price movement prediction values. We build Long Short-Term Memory (LSTM) algorithm to extract asset movement information from the historical price data. After creating the augmented



Stock	c_high	c_low	...
A	1.1	3.3	...
B	-3.2	-1	...
.	.	.	.

Stock	c_high	...	pred
A	1.1	...	0
B	-3.2	...	0
.	.	.	.

Stock	Weight
A	0.32
B	- 0.18
C	-0.05
...	...
Total	1.0

Figure 2: The total structure of PMSP.

state, we concatenate it with the original state as seen in Figure 2. Finally, we adopt Deep Deterministic Policy Gradient (DDPG) algorithm based on the augmented state for learning the policy for portfolio management. As seen in the example in Figure 2, the DDPG network can allocate a negative weight which tells the agent to make a short position on a particular asset.

4.2 Augmenting State Information

According to [1], using only historical price data as states has several challenges. First, the collected information for each asset is very noisy and imbalanced. Second, as mentioned previously, financial markets are non-stationary and uncertain so that they often cause a distribution shift between training and testing data.

We, therefore add additional information that helps to overcome the aforementioned problems. [1] solves the problems by adding price movement predictions to a state. This method proves that the prediction values can capture the dynamics of stock markets and improve the performance of asset allocation.

We train a recurrent neural network with the LSTM model to get the prediction values of assets. The LSTM model was proposed by [13], and it is widely in time series prediction problems. It adds a cell and three gates which help to remember values over certain time intervals and regulate the flow of information. The equations of the LSTM model are:

$$
\begin{aligned}
&f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
&i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
&o_i = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
&\tilde{c}_i = \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
&c = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \\
&h_t = o_t \circ \sigma_h(c_t),
\end{aligned}
$$

where x_t is an input vector to the LSTM unit, f_t, i_t, and o_t are the gate's activation vectors of forget, input/update, and output, $\tilde{c}$ is a cell input activation vector, c_t is a cell state vector, W, U, b are the weight matrices and bias vector parameters to be learned, and finally, σ is an activation function such as sigmoid or hyperbolic tangent.

The output, $\mathbf{R}^{n\times 1}$, is shaped in a binary form of n stocks, in which 1 indicates 'rise' and 0 indicates 'fall'. We use the past 10 days of stock price data as the inputs of the model.

4.3 Creating Augmented State

We now integrate the prediction vector created from the LSTM model into the original state. The dimension of the original state is $\mathbf{R}^{n \times f}$, where n is the number of stocks, and f is the number of features, which is a combination of prices of past l days. After the concatenation, the final shape of the augmented state is $\mathbf{R}^{n \times (f+1)}$.

4.4 Allocating Weights of Assets

Deterministic Policy Gradient (DPG) was proposed by [14]. Given a deterministic policy $\mu_\theta : S \to A$ parameterized by θ, a reward function $r(s, a)$, and a discounted state distribution ρ^μ induced by the policy, an objective function can be defined:

$$J(\mu^\theta) = \mathbb{E}_{s\ \rho^\mu}[r(s, \mu_\theta(s)]$$

[14] proved that the gradient of the objective function is given by:

$$\nabla_\theta J(\mu_\theta) = \mathbb{E}_{s\ \rho^\mu}[\nabla_\theta Q^\mu(s, \mu_\theta(s)]$$

Deep Deterministic Policy Gradient (DDPG) is built off the work of [14] to come up with an actor-critic algorithm. Algorithm 1 shows the process of the DDPG algorithm. DDPG is an off-policy because it explores with a stochastic behavior policy but estimates a deterministic target policy.

DDPG suits our case because the action spaces are continuous, which are the weights of the stocks. We create both actor and critic networks with the two fully-connected layers, and the sizes of the hidden dimensions are 400 and 200. Therefore, we need to flatten the augmented state to a vector before we use it as inputs to the networks.

The outputs of the policy network in our case are the allocated weights of the assets, which will be used as the actions that the agent takes. Each weight has either a positive (long position) or negative (short position) value, and the sum of the absolute values of the weights should equal to 1 ($\sum_{i=1}^{n} |y_i|$, where y_i is a weight of i_{th} stock and n is the number of the stocks in the portfolio). Finally, in order to make the outputs to have negative values, we apply a hyperbolic tangent at the end of the layer of the policy network and then divide each value by the sum of the absolute values.

Algorithm 1: DDPG algorithm

```
Randomly initialize critic network $Q(s, a|\theta^{\mu})$ and actor $\mu(s|\theta^{\mu})$ with
  weights $\theta^Q$ and $\theta^{\mu}$
Initialize target network $Q^{'}$ and $\mu'$ with weights $\theta^{Q'} \leftarrow \theta^Q, \theta^{\mu'} \leftarrow \theta^{\mu}$
Initialize replay buffer $R$
for episode = 1, M do
    Initialize a random process $OU$ for action exploration
    Receive initial observation state $s_1$
    for t=1, T do
        Select action $a_t = \mu(s_t|\theta^{\mu}) + OU_t$ according to the current policy
          and exploration noise
        Execute action $a_t$ and observe reward $r_t$ and observe new state $s_{t+1}$
        Store transition $(s_i, a_i, r_, s_{i+1})$ from $R$
        Sample a random minibatch of $N$ transitions $(s_i, a_i, r_, s_{i+1})$ from $R$
        Set $y_i = r_i + \gamma Q'(s_{i+1}|\theta^{\mu})|\theta^{Q'})$
        Update critic by minimizing the loss:
            $L = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
        Update the actor policy using the sampled policy gradient:
            $\nabla_{\theta^{\mu}} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^{\mu} \mu(s|\theta^{\mu})}|_s$
        Update the target networks:
            $\theta^{Q'} \leftarrow \tau\theta^Q + (1 - \tau)\theta^{Q'}$
            $\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1 - \tau)\theta^{\mu'}$
    end
end
```

Chapter 5

Experiments

We run experiments to answer the following questions.

- **Q1. Cumulative portfolio values (Section 5.2).** How well does the proposed method perform in the portfolio management problem?
- **Q2. Effect of adding short positions (Section 5.3).** Does adding a short position strategy really work in the downward trends and increase the portfolio values consequently?

5.1 Experimental Settings

Dataset We evaluate PMSP on historical data of highly traded stocks in NASDAQ market. We scrape the datasets from investing.com.

- **NASDAQ.** The NASDAQ dataset contains the historical stock price information from 2006-10-20 to 2013-11-20. We use the data for the following companies: Google, Nvidia, Amazon, AMD, Qualcomm, Intel Corporation, Microsoft, Apple, and Baidu. The data consists of daily opening, highest, lowest, and closing prices for each stock. We split the dataset chronologically into 1529 business days (2006-10-20 to 2012-11-18) as the training set and 255 business days (2012-11-19 to 2013-11-20) as the testing set.

Competitor We compare the performance of PMSP to the following competitors.

- **CRP.** [15] Constant rebalanced portfolio (CRP) is widely used as a baseline. It keeps the uniform weights to each asset every day. For example, if there are 4

Table 2: Summary of stocks dataset.

Dataset	# Stocks	# Training days	# Testing days
NASDAQ	9	1,529	255

stocks in the portfolio, we allocate 25% of the asset to each stock. If there is any price change in the assets, we uniformly rebalance the weights.

- **SARL** [1] State augmented reinforcement learning (SARL) uses Deterministic Policy Gradient (DPG) algorithm to optimize the allocation of assets in the portfolio. SARL augments the asset information with price movement predictions as additional states, which are achieved by using the Long Short-Term Memory (LSTM) model.
- **PMSP_NOSP**. It is the same as our proposed method, but the only difference is that it is not able to make a short position for the stocks in the portfolio.

Evaluation metrics We evaluate the performance of a particular portfolio management strategy using Portfolio Value (PV). PV is widely used to analyze the accumulative change of assets' values over the testing time. According to [16], PV can be calculated by the final time horizon T.

$$p_T = p_0 \exp\left(\sum_{t=1}^{T} r_t\right) = p_0 \prod_{t=1}^{T} \left(\mathbf{a}_t \cdot \mathbf{y}_t - \beta \sum_{i=1}^{n} |a_{i,t} - w_{i,t}|\right),$$

where t refers to a t^{th} day, p_0 is a starting value, $\mathbf{a}_t$ is the allocated weights at time t, $\mathbf{y}_t$ is the price change ratio of each assets in the portfolio, n is the number of assets, and $w_{i,t}$ is a changed weight at the end of t. β is the transaction cost, and we apply it whenever we buy or sell assets. We set β to 0.25% for both buying and selling.

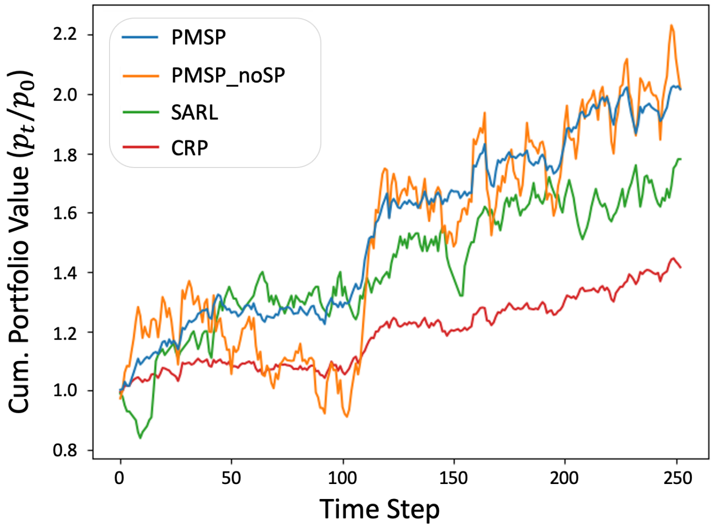


Figure 3: Portfolio value of difference methods on NASDAQ dataset.

Hyperparameters We build the three layers of the LSTM model with a hidden dimension of 100 for the price movement predictions. The model uses the prices of the past 10 days to train a classifier. and the binary cross entropy (BCE) is used as a loss function. We set the batch size to 1024 and apply early stop before 400 epochs of training. For the DDPG model, the state is composed of the previous prices of the past 30 days with a prediction output from the LSTM model. We set the replay size to 1,000,000, batch size to 256, discounting factor(γ) to 0.99, and τ to 0.001. The agent runs 100 episodes and saves experiences every time it moves to the next state. Both models use the Adam optimizer [17].

5.2 Cumulative portfolio values

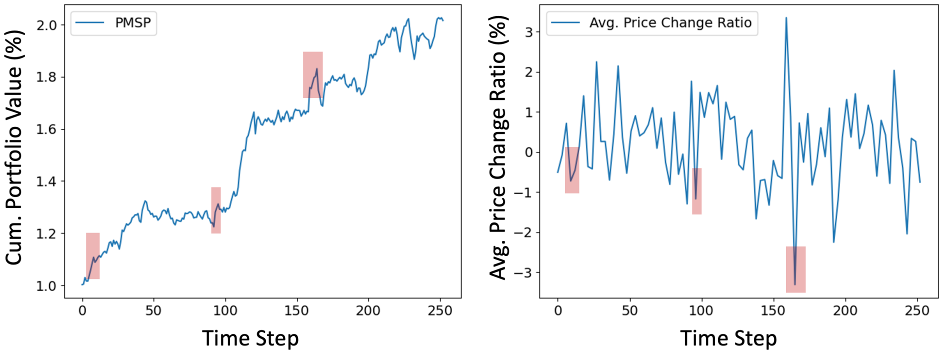
Figure 3 shows the cumulative portfolio values of PMSP and the competitors on the NASDAQ dataset. The figure clearly states that both PMSP and PMSP_NOSP have higher cumulative portfolio values than the competitors. At the end of the time step, PMSP earns 2.02 while SARL earns 1.76 showing that PMSP improves the portfolio value by 114% when compared to the previous methods.

One interesting observation in the figure is that although the portfolio value at time step T of PMSP and PMSP_NOSP are very similar to each other, PMSP has more stability in earning profits than PMSP_NOSP. From time step 50 to 100, it is clear that the portfolio value of PMSP_NOSP decreases over time, while PMSP sustains the steady earnings. This kind of aspect is easily found in the different time steps in the figure.

5.3 Effect of adding short position

In this section, we look into the effects of applying a short position strategy. As we discussed in 5.2, Figure 3 tells that PMSP has more stability in making profits, and Figure 4 explains how adding a short position strategy brings more profits. Figure 4 (a) is a cumulative portfolio value of PMSP, and (b) is the average price change ratio of the assets in the portfolio. In the figure, the red shaded areas are the times when the average price falls. As seen in Figure 4 (a), the portfolio value increases even in the red shaded areas, which implies that the model earns profits even in downward trends.

We also compute the Sharpe ratio of the profits for the different time periods. The Sharpe ratio is widely used in the financial fields to measure the performance



(a) Cum. Portfolio value of METHOD

(b) Avg. Price Change Ratio of assets

Figure 4: (a) Cumulative portfolio value of PMSP and (b) average price change ratio of assets over time. The results show that a short position strategy increases the portfolio value even the market has a downward trend (red shaded areas).

of an investment. It compares the portfolio's return to its risk, and the equation is defined as:

$$Sharpe\ Ratio = \frac{R_p - Rf}{\sigma_p},$$

where R_p is the portfolio's return, R_f is the risk-free rate, and σ_p is the standard deviation of the portfolio's return. In the experiment, we set the bank interest R_f to 2%. The higher the ratio represents the better performance of the model.

We test the Sharpe ratio on the different time periods, 1 week, 1 month, 3 months, and 6 months. Table 3 shows the results of the Sharpe ratio at different time periods. PMSP has the highest ratio in every time period, which implies that the additional amount of return that the model receives is higher than PMSP_NOSP.

Table 3: Sharpe Ratio of different time periods.

Method	Sharp Ratio			
	1w	**1m**	**3m**	**6m**
PMSP_noSP	0.236	0.385	0.645	0.584
PMSP	**0.474**	**0.698**	**1.31**	**0.905**

5.4 Flexibility in different markets

We measure the performance of PMSP in different markets to examine its flexibility. We experiment on stocks in Korea Composite Stock Price Index (KOSPI) and Korea Securities Dealers Automated Quotations (KOSDAQ). We use the data for the following companies: Samsung Electronics, Korean Air Lines, Shinsegae, Samsung Fire & Marine Insurance, Amorepacific, Posco, and GC. The dataset contains the historical stock price information from 2010-01-01 to 2019-12-31. We use the dataset from 2010-01-01 to 2018-12-31 as the training set and the data from 2019-01-01 to 2019-12-31 as the testing set. We use the cumulative portfolio value as an evaluation metric.

Figure 5 shows the cumulative portfolio value of the stocks in the Korean markets. We note several interesting observations. First, PMSP earns less profits in the Korean markets compared to the NASDAQ market. This is because the stocks in the Korean markets are selected from the different industries while the stocks in the NASDAQ dataset are all selected from the HighTech industry. The increase in industry variations makes the task more challenging. Despite the additional layer of challenge, PMSP still earns the highest portfolio values compared to its competitors. Moreover, the cumulative profit of PMSP grows steadily as the time step increases, while the cumulative portfolio value of PMSP_noSP fluctuates because of its inability to make

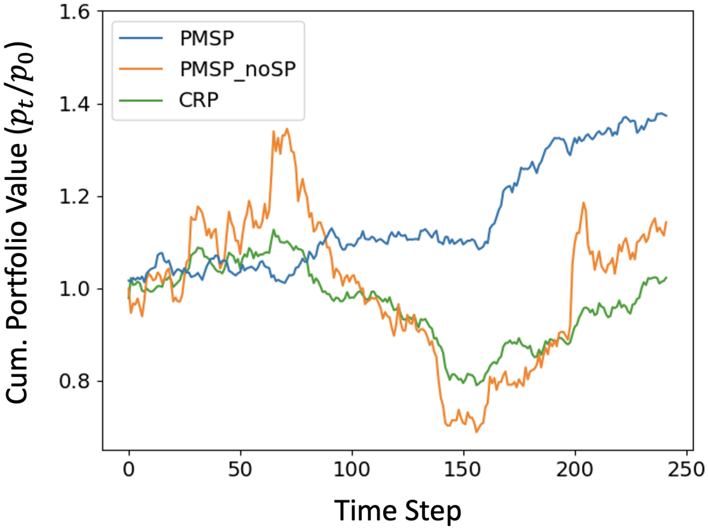


Figure 5: Portfolio value of difference methods on the stocks in the Korean markets.

profits in downward trends.

Chapter 6

Conclusion

We propose PMSP, an efficient portfolio management which considers both market dynamics and an ability to make a short position. PMSP uses the features that express the trend of stocks to capture the interaction of different prices. Furthermore, we add a short position strategy that PMSP can earn profits even in downward trends of stock markets. PMSP earns the highest portfolio value among the competitors and proves its stability by testing the Sharpe ratio.

References

[1] Y. Ye, H. Pei, B. Wang, P.-Y. Chen, Y. Zhu, J. Xiao, and B. Li, "Reinforcement-learning based portfolio management with augmented asset movement prediction states," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 1112–1119, Apr. 2020.

[2] S. Almahdi and S. Yang, "An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown," *Expert Systems with Applications*, vol. 87, 06 2017.

[3] Z. Jiang, D. Xu, and J. Liang, "A deep reinforcement learning framework for the financial portfolio management problem," 2017.

[4] Z. Liang, H. Chen, J. Zhu, K. Jiang, and Y. Li, "Adversarial deep reinforcement learning in portfolio management," 2018.

[5] J. B. Heaton, N. G. Polson, and J. H. Witte, "Deep learning in finance," *CoRR*, vol. abs/1602.06561, 2016.

[6] R. P. Schumaker, Y. Zhang, C.-N. Huang, and H. Chen, "Evaluating sentiment in financial news articles," *Decision Support Systems*, vol. 53, no. 3, pp. 458 – 464, 2012.

[7] T. H. Nguyen, K. Shirai, and J. Velcin, "Sentiment analysis on social media for stock movement prediction," *Expert Systems with Applications*, vol. 42, no. 24, pp. 9603 – 9611, 2015.

[8] L. Chen, H. Zhang, J. Xiao, X. He, S. Pu, and S. Chang, "Scene dynamics: Counterfactual critic multi-agent training for scene graph generation," *CoRR*, vol. abs/1812.02347, 2018.

[9] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2019.

[10] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *CoRR*, vol. abs/1707.06347, 2017.

[11] H. Buehler, L. Gonon, J. Teichmann, and B. Wood, “Deep hedging,” *Quantitative Finance*, vol. 19, no. 8, pp. 1271–1291, 2019.

[12] F. Feng, H. Chen, X. He, J. Ding, M. Sun, and T.-S. Chua, “Enhancing stock movement prediction with adversarial training,” 2019.

[13] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, p. 1735–1780, Nov. 1997.

[14] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, “Deterministic policy gradient algorithms,” ICML’14, p. I–387–I–395, JMLR.org, 2014.

[15] T. M. Cover, “Universal portfolios,” *Mathematical Finance*, vol. 1, no. 1, pp. 1–29, 1991.

[16] M. Ormos and A. Urbán, “Performance analysis of log-optimal portfolio strategies with transaction costs,” *Quantitative Finance*, vol. 13, no. 10, pp. 1587–1597, 2013.

[17] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2017.

요 약

여러 주식의 가격 정보를 담고 있는 포트폴리오가 주어졌을 때, 어떻게 각 주식에 자산을 효율적으로 배분하여 수익을 최대화할 수 있을까? 포트폴리오 매니지먼트 (Portfolio Management)는 포트폴리오 내의 각 주식에 자산을 배분하여 수익을 최대화하는 동시에 투자 위험을 최소화하는 것을 목표로 한다. 머신 러닝과 딥러닝 기술이 발전함에 따라 해당 기술을 사용하여 효율적으로 자산을 배분하는 선행 연구들이 많이 발표되었다. 그러나 선행 연구들은 주식 시장이 하락장일 때 좋지 못한 성능을 보였다.

따라서 해당 논문에서는 강화학습 기법과 인버스 투자 전략을 추가하여 주식의 하락장에서도 수익을 창출할 수 있는 **P**ortfolio **M**anagement with **S**hort **P**osition (PMSP) 알고리즘을 제안한다. PMSP 는 시시각각 변하는 주식의 변동성을 반영할 수 있도록 각 주가(시/고/저/종가)를 서로 비교하는 피처를 생성한다. 또한 replay buffers 및 타겟 네트워크를 사용하여 빠르고 안정적인 학습을 가능케하는 Deep Deterministic Policy Gradient (DDPG) 알고리즘을 사용한다. 마지막으로 각 입력값을 양수로 변환하여 인버스 투자(음수값)를 반영하지 못하는 소프트맥스 함수 (Softmax function)를 네트워크 끝 단에 사용하는 대신, 음수의 값을 취할수 있는 hyperbolic tangent 함수를 사용하여 인버스 투자를 가능케 하였다. 따라서 PMSP 는 인버스 투자를 통해 주식의 하락장에서도 수익을 얻을 수 있는 장점을 지니고 있다. PMSP 의 성능을 확인하기 위한 여러가지 실험을 진행하였으며 이를 통해 PMSP 가 선행 연구 기술보다 높은 수익률(연 102%)을 달성한 것을 확인할 수 있었다.

주요어 : 포트폴리오 매니지먼트 (자산 배분), Deep Deterministic Policy Gradient, 장단기 메모리 네트워크, 인버스 투자 전략

학번 : 2019-29394