Deep Reinforcement Learning For Stock Trading Execution

Shuo Yang
Department of Computer Science
Shanghai Jiao Tong University
No. 800 Dongchuan Road, Shanghai
andy_yang@sjtu.edu.cn

Abstract

Deep reinforcement learning has performed well in many fields. Many people in the investment industry hope to apply it in the field of stock market trading, and many researchers have made a lot of efforts. Nonetheless, past research has not met the needs of the investment industry. This paper establishes a completely different deep reinforcement learning model for the stock market and overcomes the problems of the past trading model. This model saves a lot of action space and can train an excellent agent in a short time. On the test set, the model's return is over 1.8 times the original funding.

1 Introduction

Deep reinforcement learning (DRL) is the process of learning through the interaction between the agent and the environment to find a strategy that maximizes the cumulative return for a specific task. Therefore, deep reinforcement learning is very naturally introduced into the field of stock market trading, and lots of work has been done.

However, the previous work did not meet the needs of investors. There are many reasons why DRL is rarely used in stock market trading. Firstly, in terms of modeling, the stock market has a huge and complex observation space, and there are many variables that cannot be quantified. Secondly, the stock market also has a huge action space, making it extremely difficult to train models. Last but not least, the lack of interpretability of DRL means that the investment industry refuse to use it on a large scale.

In this paper, I propose a completely different action space for the model. This method saves the cost of sampling and is more in line with our understanding of stock market trading. This model can avoid risks while giving more benefits. The most unexpected thing is that under different algorithms, the policy of the model converges together, so it has stronger robustness and interpretability.

2 Background and related works

Due to the complexity of the stock market model, the action space is often simplified in previous works. Therefore, many models have good results in the literature, but do not work well in actual transactions. For larger action spaces, the model will be more difficult to train and therefore will not perform better.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

2.1 Strategies based on stock price prediction

Currently, the vast majority of DRL stock market trading strategies are based on stock price prediction. [1] Such models have a clear loss function. As long as you predict the rise and fall of the stock correctly, you can choose the stock with the largest increase.
However, stock price predictions are unrealistic. Taking the LSTM algorithm as an example, price prediction on time series relies on the inertia of price and time. Previous works[2] have come to the conclusion that the correlation coefficient of LSTM's prediction of ups and downs is close to 0.

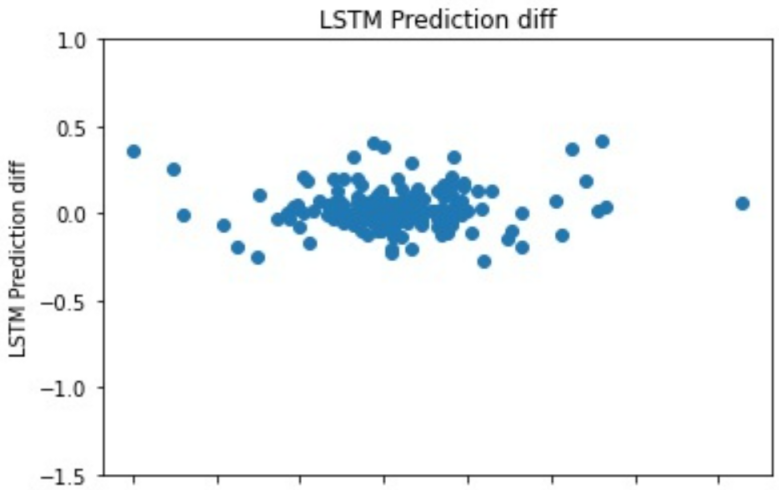

Figure 1: LSTM performance

2.2 Strategies based on discrete action space

There are two main DRL stock market models based on discrete action spaces. The first model has only two actions for a stock: buy as many stocks as possible or sell all stocks. Rather than learning stock prices explicitly, this model learns strategies for buying and selling stocks directly. Since there are only three possibilities in the action space, the sampling for training is very convenient, and the model can generally exceed the rise and fall of the stock itself[3]. However, the narrowing of the action space greatly reduces the ability of risk control, and it is impossible to consider the buying and selling of multiple stocks at the same time, so its profits cannot be guaranteed.

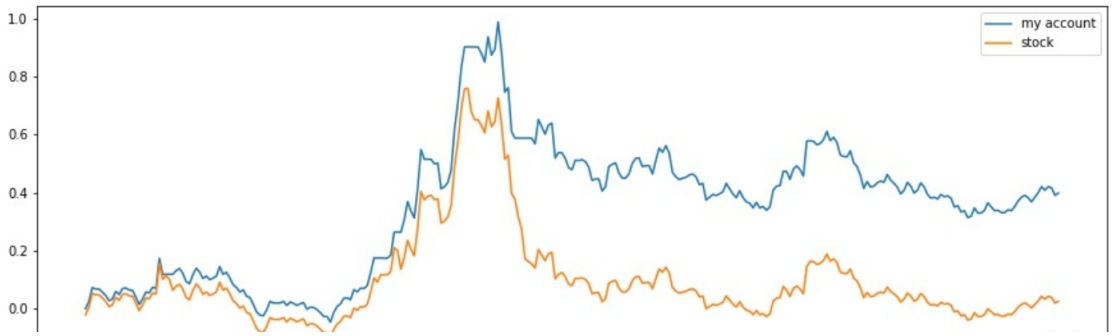

Figure 2: { -1 , 0 , 1} action space performance

[1] https://github.com/wbbhcb/stock_market
[2] https://zhuanlan.zhihu.com/p/460042522
[3] https://github.com/kanghua309/strategy

Another strategy is to use a huge, high-dimensional action space with positive values for buying and negative values for selling. This model is very close to the real stock trading, because it supports the trading of any amount of multiple stocks[4].However, its huge action space makes the training of the model extremely difficult. First, the vast majority of training samples are wasted. When cash is at a small value, all buying sampling are invalid. Similarly, all sell sampling is ineffective when there are very few stocks held.Therefore, the benefits of this model is even lower than the benefits of the former model.

3 Problem description

The principle of designing the model is to treat stock market trading as a Markov decision process and maximize the final total assets.

3.1 Observation space

In order to provide the most important information in the stock market as possible, the observation space provides the current cash holdings and the price, holdings, average similarities and differences indicators, relative strength indicators, trend indicators and average movement indicators of 30 stocks. Therefore, the observation space is a 182-dimensional vector. Except for the price and holding information, the other elements in the vector are all real numbers. In summary, we choose a 182-dimensional real vector as the observation space.

$$s = [proportion, price, feature]$$

3.2 Action space

The choice of action space is the core of this article. Considering that the action space of previous articles is either clumsy or wasteful, we use a 31-dimensional real vector as the action space. This vector represents the reallocation of current assets: the first dimension represents the proportion of cash in total assets, and the remaining thirty dimensions represent the proportion of this stock in assets. Therefore, each dimension of the action space is a real number from 0 to 1, and sums up to 1 over the thirty-one dimensions.

$$a = [new\ proportion]$$

In this action space, any action is a valid operation, so there is no waste of action space. In addition, this action space contains any valid pattern of asset allocation, so optimal decisions can be learned. The most important point is that this strategy is in line with our understanding of the stock market, which is to hold as many potential high-quality stocks as possible, and select multiple high-quality stocks to share the risk.

3.3 Reward

Any action is an allocation of assets, so it can be understood as selling all stocks and then buying stocks according to the allocation. Therefore, we can calculate the number of new holdings of each stock according to the asset allocation, and the total amount of assets on the next day will also change due to the rise and fall of the stocks held. The reward for this action is the total assets of the next day minus the total assets of the day.

$$h = asset \times proportion/price$$
$$asset' = h \cdot price'$$
$$r = asset' - asset$$

[4]https://github.com/AI4Finance-Foundation/Deep-Reinforcement-Learning-for-Automated-Stock-Trading-Ensemble-Strategy-ICAIF-2020

3.4 Algorithm

In the selection of training algorithms, we used three algorithms, including Advantage Actor Critic(A2C), Proximal Policy Optimization(PPO) and Deep Deterministic Policy Gradient(DDPG). The study tests the performance of the three algorithms at different timesteps and batchsizes and compares them with random strategies.

4 Experiment

4.1 Data set

The experiment selected thirty stocks of US stocks from 2009 to 2019 as the data set. Among them, the data from 2009 to 2015 is used as the training set, and the stocks from 2015 to 2019 are used as the test set.The reason why US stocks are selected as samples is that A-shares have great instability, and most of the data are not open source.

Figure 3: Dow Jones Industrial Average

4.2 Implementation details

4.2.1 environment configuration

This experiment is done in the ubuntu environment. In order to use the "stable baselines" deep neural network algorithm library and give it a chance to run on windows, the following configuration[5] needs to be done on the environment.

4.2.2 action space's design

Most reinforcement learning algorithms work on a continuous space of actions based on a Gaussian distribution, so setting the action space to be a 31-dimensional vector of 0 to 1 makes the training process very difficult.[6]
The action space for the gym library used in the experiment is -1 to 1. In order to make the model discriminating against stocks, each dimension of the vector is multiplied by a larger coefficient here. Finally, we normalize this action space with the softmax function, and we get the required action space.

4.3 Experiment procedure

step 1 Set up a training environment and split the data set into training set and test set.

step 2 Train multiple agents for each of the three algorithms according to timestep and batchsize. Validate the agent on the test set, record its final earnings and record its stock holding history.

step 3 Compare trained agents to each other and against the random strategy. Adjust the hyperparameters for a new round of training.

[5]https://stable-baselines.readthedocs.io/en/master/guide/install.html
[6]https://github.com/hill-a/stable-baselines/issues/473

Table 1: Hyperparameters

Item		
Name	Description	Size
initial_asset	original assets on the account	100000
cash_bias	agent's propensity for cash	10000
softmax_regularization	original action space's differentiation coefficient	10000

5 Performance and results

5.1 Compare with random strategy

After training, almost all agents can outperform random strategy, and almost all agents are guaranteed to have positive returns.

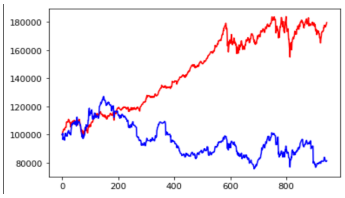

Figure 4: A2C vs random

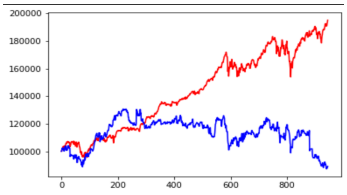

Figure 5: DDPG vs random

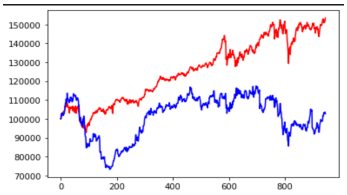

Figure 6: PPO vs random

5.2 Comparison between agents

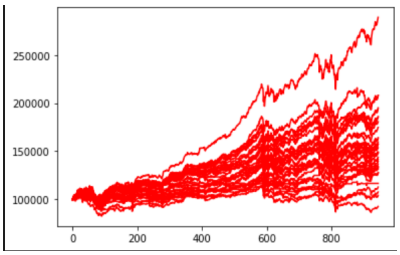

Figure 7: All trained agents

By comparing various agents, it is not difficult to find that DDPG performs the best and PPO performs the worst. Therefore, we analyze why DDPG can provide a better result. DDPG is a deterministic policy gradient algorithm that can solve continuous control problems.[7]

The behavior strategy of DDPG is different from the evaluation strategy. Its behavior strategy is a random strategy, but the evaluation strategy is a deterministic strategy. Random strategies can explore and generate diverse data, and deterministic strategies use these data to improve strategies. For continuous control tasks, the DDPG algorithm has great advantages. When the Q-function in DDPG is learned sufficiently reliable, the algorithm works very well in practice.[8]

5.3 Model convergence

In general model training, the longer the training time, the better the general model fitting effect will be. When the model training time is too long, the model may overfit the training set.From the experimental results, the DDPG algorithm does not appear to be overfitting. As the training time increases, the agent performs better and better on the test set.

[7] https://arxiv.org/pdf/1509.02971.pdf

[8] https://spinningup.openai.com/en/latest/algorithms/td3.html

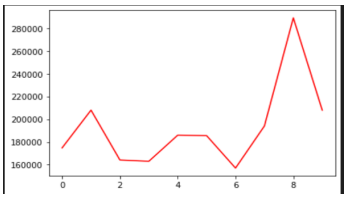

Figure 8: A2C time step

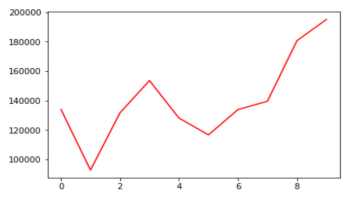

Figure 9: DDPG time step

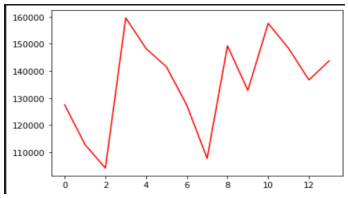

Figure 10: PPO time step

The most surprising thing for us is that when the training time is long enough, the policies of the different algorithms converge together.After sufficient learning, the two models get almost identical strategies. In other words, the two algorithms found the same strategy in different ways. This means that the trained strategy is likely to be an excellent strategy for stock market trading at this stage.

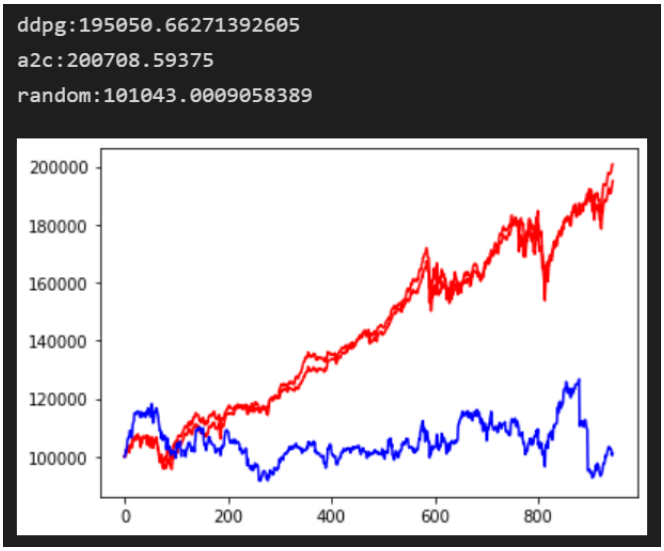

Figure 11: A2C and DDPG converge together

6 Conclusion

In this paper, we use deep reinforcement learning to specify strategies for stock market trading. During the research process, we redesigned the observation space and action space for stock market trading. During the experiments, we found that this model performed quite well. The vast majority of models can beat random strategy, and all models are guaranteed to have positive returns.

For future work, we will continue to develop and refine this project.[9] I believe that one day, deep reinforcement learning will become a powerful tool of traders, not a toy of computer scientists.

References

[1] Zihao Zhang & Stefan Zohren Deep Reinforcement Learning for Trading.

[2] Uta Pigorsch & Sebastian Schäfer *HIGH-DIMENSIONAL STOCK PORTFOLIO TRADING WITH DEEP REINFORCEMENT LEARNING.*

[3] Hasselmo, M.E., Schnell, E. & Barkai, E. (1995) Deep Reinforcement Learning for Automated Stock Trading: An Ensemble Strategy.

[9] https://github.com/andy-yang-1/Milagro-Man