Practical Deep Reinforcement Learning Approach for Stock Trading

Zhuoran Xiong, Xiao-Yang Liu, Shan Zhong

Novosibirsk State University

April 27, 2021

Introduction

- Profitable stock trading strategy is vital to investment companies. It is applied to optimize allocation of capital and thus maximize performance, such as expected return.
- Return maximization is based on estimates of stocks' potential return and risk. However, it is challenging for analysts to take all relevant factors into consideration in complex stock market.
- The traditional approach is performed in two steps. First, the expected returns of the stocks and the co-variance matrix of the stock prices are computed. The best portfolio allocation is then found by either maximizing the return for a fixed risk of the portfolio or minimizing the risk for a range of returns.

Introduction

- Traditional can be very complicated to implement if the manager wants to revise the decisions made at each time step and take, for example, transaction cost into consideration.
- Another approach to solve the stock trading problem is to model it as a Markov Decision Process (MDP) and use dynamic programming to solve for the optimum strategy. However, the scalability of this model is limited due to the large state spaces when dealing with the stock market.
- In this paper, a deep reinforcement learning algorithm, namely Deep Deterministic Policy Gradient (DDPG), to find the best trading strategy in the complex and dynamic stock market. This algorithm consists of three key components: Actor-critic framework that models large state and action spaces; Target network that stabilizes the training process; Experience replay that removes the correlation between samples and increases the usage of data.

Problem Formulation for Stock Trading

- Considering the stochastic and interactive nature of the trading market, we model the stock trading process as a Markov Decision Process (MDP) as shown in fig. below.

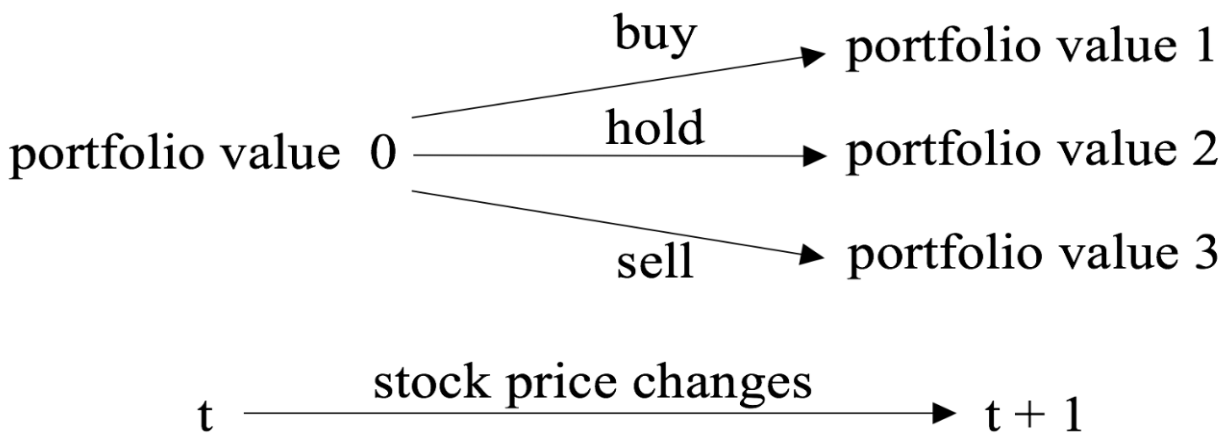


Figure 1: One starting portfolio value with three actions leading to three possible portfolio values where actions have probabilities that sum up to one. Note that "hold" can lead to different portfolio values if the stock prices change.

Problem Formulation for Stock Trading

- State s = [p, h, b]: a set that includes the information of the prices of stocks p $\in$ RD+ , the amount of holdings of stocks, h $\in$ ZD+, and the remaining balance b $\in$ R+, where D is the number of stocks that we consider in the market and Z+ denotes non-negative integer numbers.
- Action a: a set of actions on all D stocks. The available actions of each stock include selling, buying, and holding, which result in decreasing, increasing, and no change of the holdings h, respectively.
- Reward r(s, a, s′): the change of the portfolio value when action a is taken at state s and arriving at the new state s′. The portfolio value is the sum of the equities in all held stocks and balance b.
- Policy (s): the trading strategy of stocks at state s. It is essentially the probability distribution of a at state s.
- Action-value function Q (s, a): the expected reward achieved by action a at state s following policy .

Trading Goal as Return Maximization

- The goal is to design a trading strategy that maximizes the investment return at a target time. Due to Markov property of the model, the problem can be boiled down to optimizing the policy that maximizes the function Q (st , at). This problem is very hard because the action-value function is unknown to the policy maker and has to be learned via interacting with the environment. Hence in this paper, we employ the deep reinforcement learning approach to solve this problem.
- DDPG is an improved version of Deterministic Policy Gradient (DPG) algorithm. DPG combines the frameworks of both Q-learning [13] and policy gradient [14]. Compared with DPG, DDPG uses neural networks as function approximator. The DDPG algorithm in this section is specified for the MDP model of the stock trading market.

- The Q-learning is essentially a method to learn the environment. Instead of using the expectation of Q(st+1, at+1) to update Q(st, at), Q-learning uses greedy action at+1 that maximizes Q(st+1, at+1) for state ss+1.
- With Deep Q-network (DQN), which adopts neural networks to perform function approximation, the states are encoded in value function. The DQN approach, however, is intractable for this problem due to the large size of the action spaces. Since the feasible trading actions for each stock is in a discrete set, and considering the number of total stocks, the sizes of action spaces grow exponentially, leading to the "curse of dimensionality". Hence, the DDPG algorithm is proposed to deterministically map states to actions to address this issue

Learning Network Architecture

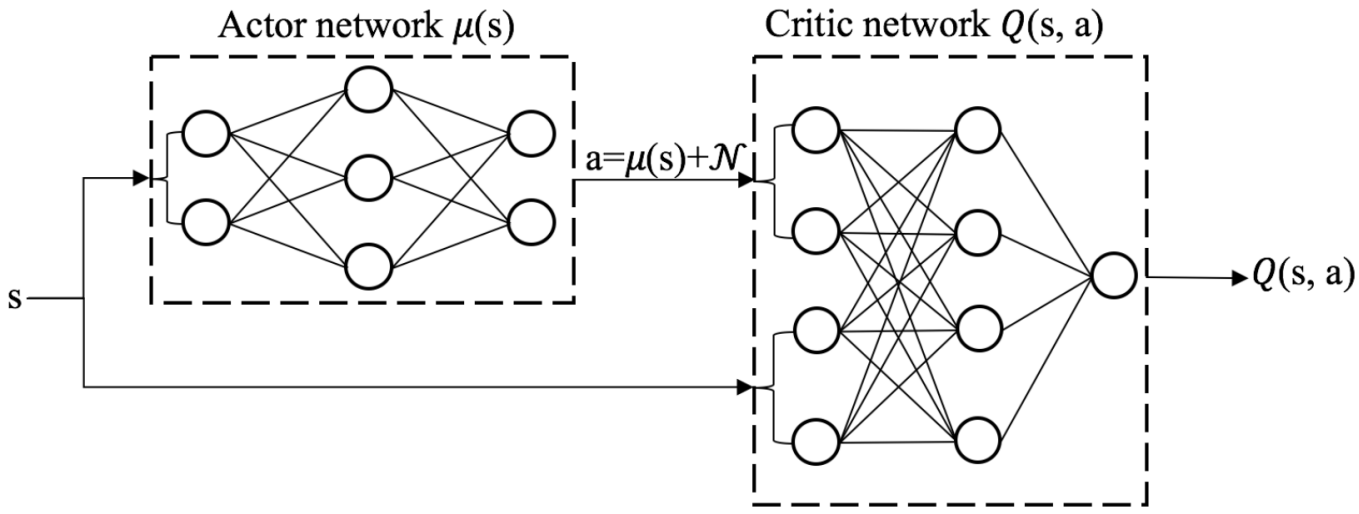
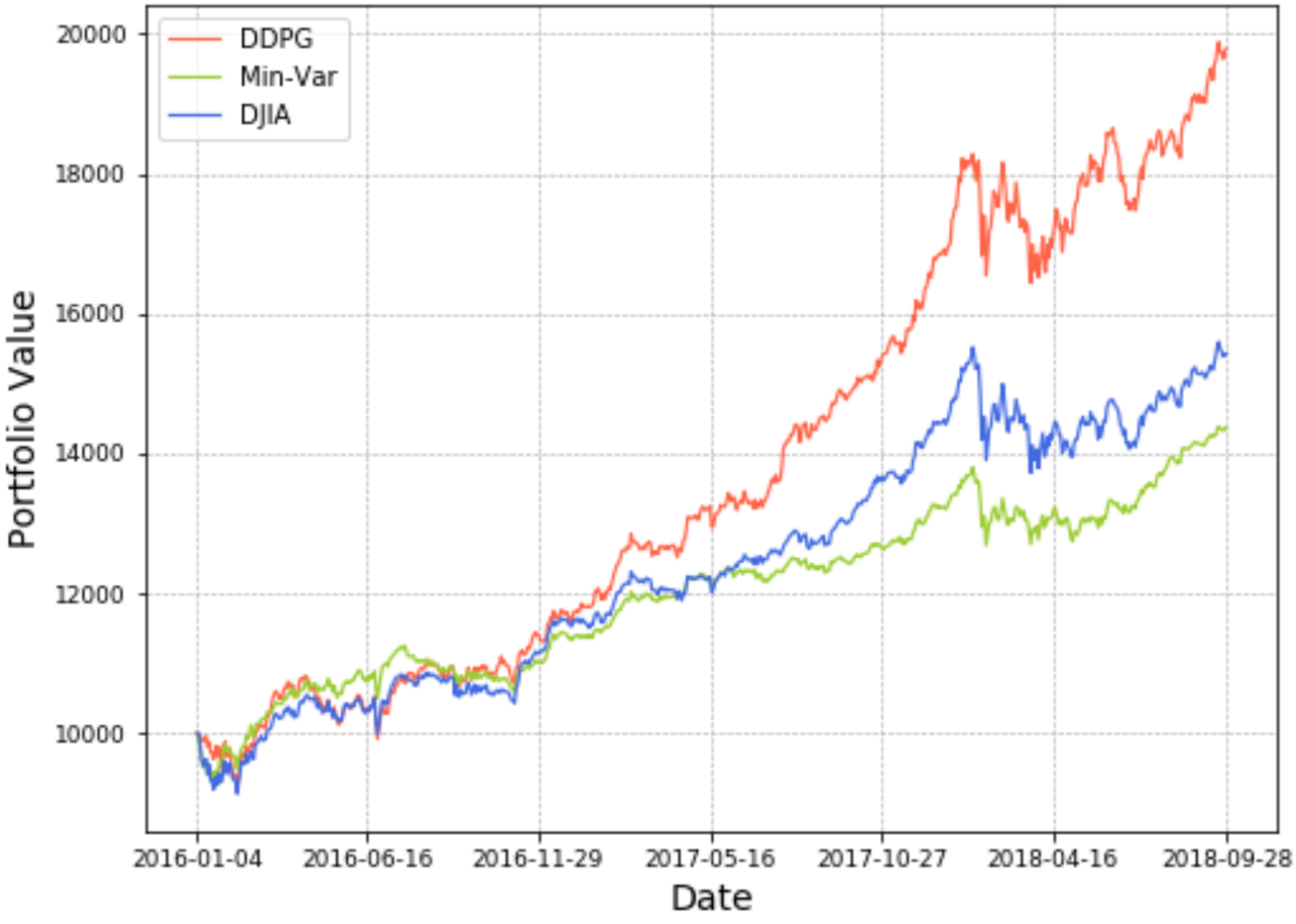


Figure 2: Learning network achitecture.

- As shown in Fig. 2, DDPG maintains an actor network and a critic network. The actor network μ(s—μ) maps states to actions where μ is the set of actor network parameters, and the critic network Q(s, a—Q) outputs the value of action under that state, where Q is the set of critic network parameters. To explore better actions, a noise is added to the output of the actor network, which is sampled from a random process N .

Performance Evaluation



Results

Table 1: Trading Performance.

	DDPG (ours)	Min-Variance	DJIA
Initial Portfolio Value	**10,000**	10,000	10,000
Final Portfolio Value	**19,791**	14,369	15,428
Annualized Return	**25.87%**	15.93%	16.40%
Annualized Std. Error	**13.62%**	9.97%	11.70%
Sharpe Ratio	**1.79**	1.45	1.27

Reproduced Results

```
# S&P 500: ^GSPC
# Dow Jones Index: ^DJI
# NASDAQ 100: ^NDX
backtest_plot(df_account_value,
              baseline_ticker = '^DJI',
              baseline_start = '2019-01-01',
              baseline_end = '2021-01-01')
```

Annual return	21.789%
Cumulative returns	48.441%
Annual volatility	19.916%

THANK YOU