Stock Trading with Deep Reinforcement Learning

Shangyi Li 2819****

Southeast University - Monash University Joint Graduate School

29 May 2017

Abstract

Deep reinforcement learning (DRL) has a strong ability of decision making, learning in an end-to-end approach. In this paper, we present the first DRL architecture to tackle stock trading problems. This model can directly generate stock trading policy for making money, using deep neural network to directly approximate the trading policy. The policy gradient method is applied to optimize the trading policy. Our approach does not require any prior domain knowledge, thus can avoid the possible misjudgment caused by human experience in the trading process. Furthermore, recent DRL models deeply rely on the computing power of high-performance GPUs, which are generally very expensive. We use the multi-thread techniques to train multiple agents simultaneously, only requiring a multicore CPU, thus avoiding the high-expense of GPUs.

Keywords: deep reinforcement learning, reinforcement learning, stock trading

Introduction

Deep learning can discover the distributed feature representation of data, while reinforcement learning usually solves sequential decision-making problems, interacting with an environment over time. DRL integrates the advantages of the perception of deep learning and the decision making of reinforcement learning, and gains the output control directly based on raw inputs by the end-to-end learning process. DRL has recently achieved notable successes in a wide variety of domains, like Atari games (Mnih et al., 2013; 2015), the game of go (Silver et al., 2016), robot control (Lillicrap et al., 2016), and computer vision (Oh et al., 2015). In particular, Deep Q-Network (DQN) outperformed human players in playing various Atari 2600 games and what's more, AlphaGo defeated dozens of world-class Go players. DRL now is considered to be an important way to Artificial General Intelligence (AGI).

Advances in DRL have allowed autonomous agents to perform well in decision making. However, none of the above applications are relevant to the field of economics, which may benefit from the strong policy optimizing ability of DRL. Stock trading is one of the accepted ways of acquiring wealth in the field of economics. However, this is often perceived as a form a gambling for ordinary people since most of them lose money. The stock market accords with the Pareto Principle (also known as the 80/20 Rule) that 20% of the investors make money while the rest lose money. The reality is even worse. Even if people know the risk of the stock market, they still cannot get rid of the desire to become one of the 20%. There are many reasons for losing money, including psychological factors, cognitive ability and lack of expertise.

In this paper, we present the first DRL architecture to tackle stock trading problems, which is of great practical significance. Our approach is based on policy-based DRL methods. The idea behind is to develop a proper neural network to parameterize the policy and use policy gradient methods to optimize the policy. The network can directly learn from raw market data without hand-craft features and generate an optimal stock trading policy for making money. Compare to the present stock trading model, our approach does not require domain knowledge of the stock market, thus avoid the possible misjudgment caused by human experience in the trading process. What' more, previous DRL architectures deeply rely on the powerful computing ability of high-performance GPUs (Mnih et al., 2016), which are generally very expensive. We use the multi-thread technique to run multiple models simultaneously, only requiring a multicore CPU, thus avoid the high expense of high-performance GPUs.

Objectives

Our research aims to develop a deep neural network to directly approximate the trading policy. After training with the history market data, the model can be used to trade stock and make money. The model is also required to be trained every week or even every day, using the latest market data, to make sure that it can adapt constant changing of the stock market. Our multi-thread architecture makes it available that training our model on any personal computer with a multicore CPU, thus more people can use our proposed model to trade stock and make money.

Methodology

Policy gradient is a common policy optimization method, that updates policy parameters by continually calculating the gradient of the expectation of total reward, and eventually converges to an optimal policy. Therefore, when solving a DRL problem, we can use a deep neural network with parameter θ to approximate a policy, and optimize the policy by policy gradient method. Such an approach is known as the policy-based DRL method. It is worth noting that when solving a DRL problem, policy gradient algorithms are the first choice. The reason is that it can directly optimize the expected total reward, searching for an optimal policy in the policy space in an end-to-end manner. Compared with value-based DRL methods, policy-based DRL methods have a wider range of application and the better performance in policy optimization.

To optimize a policy π_θ, we first need to define a policy objective function as follow, which indicates the quality of π_θ. It is the discounted total future reward that the policy π_θ can get, starting from state s.

$$J(\theta) = E[V(s)] \tag{1}$$

The gradient of the objective function could be written as

$$\nabla_\theta J(\theta) = E[\nabla_\theta log\pi_\theta(s,a)r] \tag{2}$$

In which $\nabla_\theta log\pi_\theta(s,a)$ is the score function, which means that how much than usual of taking an action a, and *r* is the total future reward the policy can get when taking an action a. Combining the two components, policy gradient aims to improve the policy in the direction of taking more good actions and taking less bad actions. In practice, the total future reward *r* is usually replaced by advantage function $A(s,a)$, which indicates the advantage of taking an action a.

$$A(s,a) = Q(s,a) - V(s,a) = r + \gamma V(s') - V(s) \tag{3}$$

In the above equation, $V(s)$ is the value function that give out the expected total reward, which can also be approximated by a neural network. This is very easy to learn since the network only requires one state as the input and outputs the single value of that state. However, we can use a single network to estimate both policy $\pi_\theta(s)$ and value function $V(s)$. This approach can learn more faster and effectively.

The input of the model is the current state, which could be defined as (*B*, *T*). *B* is the available money in the account, and *T* is a list, the member of which is tuples that contains the stock symbol and its corresponding stock holding quantity. The action is defined as a tuple (*x*, *a*). *x* is the stock symbol which is an integer, and a is the action on that stock which is a real number. If *a* is a positive number, it means buying in *a* numbers of the stock. If *a* is a negative number, it means selling out *a* numbers of the stock. If the agent takes some invalid actions, like selling out the stock which it doesn't hold, or buying in a stock which beyond its balance, it will receive a great minus-value reward as punishment. If the prices of the stocks held rise, the agent will receive a plus-value reward based on the amount of earned money, or otherwise it will receive a minus-value reward based on the amount of money it loses. The optimization work of the network is based on policy gradient algorithms.

We train different models to and compare the performance. Different numbers of layers and neurons are tried, and we find that a 4-layer MLNN model with a LSTM layer (Hochreiter and Schmidhuber., 1997) is most appropriate to the stock trading problem. The memory ability of LSTM increases the performance of our model in long-term transactions.

We program to retrieve the history market data from 2000 to 2016, using Sina stock api. The model is trained with the dataset we collected. After training, the model has the ability to generate stock trading policy to make money. Furthermore, we use Sina stock API to collect the latest market data, to update the policy of the model every day to make sure our policy can adapt constant changing of the stock market.

In this project, we use a multi-thread mechanism to run multiple agents parallelly at the same time. These agents synchronize the parameters of the network each time step. This approach only requires a multicore CPU, instead of an expensive high-performance GPU, thus the propose model could be easily trained on personal computers and making it available for more people to use our model.

Novelty

In this paper, we present the first DRL architecture to tackle stock trading problems, which is of great practical significance. Previous DRL models deeply rely on the powerful computation ability of high-performance GPUs. Our architecture uses multiple threads to training multiple agents simultaneously, therefore speeding up the training process. The only requirement is a multicore CPU, avoiding the high expense of a high-performance GPU. Unlike present stock trading model, our proposed model learns in an end-to-end way, and directly generating stock trading policy for making money without any prior domain knowledge of the stock market, avoiding the possible misjudgment cause by human experience in the trading process. Our model can be easily trained on any personal computer with a multicore CPU, therefore, more people can use our model to make money.

Conclusion and Significance

Our research project aims to develop a deep neural network to directly approximate the trading policy. A 4-layer MLNN model with a LSTM layer is built to approximate the policy $\pi(s)$ and the value function $V(s)$. Then policy gradient method is applied to applied to optimize the policy π in direction of maximizing the value function. This approach is very fast and effective. What's more, we use the multi-thread technique to run multiple agents simultaneously, which could even speed up the training process and specifically, avoid the high expense of a high-performance GPU. The only requirement is a personal computer with a multicore CPU. This is the first DRL architecture to tackle stock trading problems. It can learn by the end-to-end training process and directly generate stock trading policy for making money without any prior domain knowledge about the stock market. After training with the raw market data collected by using Sina stock API, the proposed model is able to perform both short-term and long-term transactions to make money. Furthermore, the model is still required to trained with the latest market data which is also collected Sina stock API, to make sure the model can adapt constant changing of the stock market. We believe this high-performance and low-requirement model can help more ordinary people who are suffering in the stock market to make money.

References

Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. Neural computation, 9(8), 1735-1780.

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D. and Wierstra, D., 2015. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. and Riedmiller, M., 2013. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G. and Petersen, S., 2015. Human-level control through deep reinforcement learning. Nature, 518(7540), 529-533.

Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D. and Kavukcuoglu, K., 2016, June. Asynchronous methods for deep reinforcement learning. In International Conference on Machine Learning, 1928-1937.

Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M. and Dieleman, S., 2016. Mastering the game of Go with deep neural networks and tree search. Nature, 529(7587), 484-489.

Oh, J., Guo, X., Lee, H., Lewis, R.L. and Singh, S., 2015. Action-conditional video prediction using deep networks in atari games. In Advances in Neural Information Processing Systems, 2863-2871.