Literature Survey

Youzhi Luo
Texas A&M University
College Station, TX 77843
yzluo@tamu.edu

Limei Wang
Texas A&M University
College Station, TX 77843
limei@tamu.edu

Abstract

Stocks trading can be easily transferred to a decision making problem, so it is natural to apply reinforcement learning to it. The objective of our course project is to build a stocks trading system with a deep reinforcement learning algorithm as the decision agent. We firstly describe our system in detail, including the state space, action space, reward, training data and implementation. Then we briefly summarize some related works to our project.

1 Introduction

Since the appearance of first financial market was set up, people have been trying to predict future trend of price movements for making as many profits as possible. One of the most widely happened financial activity in people's daily life is stocks trading. In this financial condition, what people is looking forward to achieving is to buy stocks at low prices and to sell them at high prices.

In fact, the process of stocks trading can be formalized as a decision problem easily — at each trading moment, people are access to some observation (e.g. historical prices), and make some decisions (e.g. buy, sell or wait) to maximize their expected future rewards (profits). Thus it is possible to apply reinforcement learning to stocks trading problem.

2 Project Plan

Generally speaking, our objective of this project is to develop a stocks trading program with Deep Q-learning [1] algorithm. We will try several different deep neural network structures (e.g. Multi-layer perceptron, 1D convolution neural network) as the function approximator, and use value-based or policy-based reinforcement learning architecture to train our neural network.

The stocks trading in the real world can be very complicated — one person can buy or sell multiple stocks in any time. To simplify the problem, we add some constraints to our agent. Since we only have the stock's pricing data of one company, in our project we only consider making agent learn to make trading about this stock. We divide the time into intervals, where each interval is one minute. The agent can observe one stock's pricing information during each interval, but it is allowed to make trading only at the end of the interval, and the agent is only allowed to buy a share or sell a share of stock at the trading moment. It is expected that our trained stocks trading program is able to make intelligent trading decision based on the observed historical price information of stocks so as to buy or sell a share of stock at the best time. Some detailed description about our project is below.

2.1 State Space

The state is represented as a four-item vector. Each state vector contains the price information of the stock in the interval of one minute, that is, the trading price at the beginning of the interval, the highest price during the interval, the lowest price during the interval and the price at the end of the interval.

2.2 Action Space

There are only three action at the trading moment — sell a share of the stock (only if the agent has a share of stock), buy a share of the stock (only if the agent does not have a share of stock) and do nothing. So the agent can only holds a share of the stock at any time.

2.3 Reward

The reward is the amount of profit that will get when the agent sell its share of stock, that is, the difference between the selling price and the buying price.

2.4 Data

We have collected the Russian stock market price records from 2015 to 2016 as our data in all experiments, which are saved as '.csv' files. Each record contains a four-number tuple, whose meaning is the same as the description of state space. We will choose part of these data for training and evaluate the trained stocks trading system on the remaining data.

2.5 Implementation

We plan to use python to implement our stocks trading system, and Pytorch package to implement the deep neural networks. In addition, we will make use of an implemented gym-like stocks trading environment [13] to simulate the stocks trading environment based on our collected stocks price data.

3 Related Works

Reinforcement learning is a main branch in the field of machine learning. It focuses on the problem of given an environment, how intelligent agents ought to take actions in this environment in order to maximize the cumulative rewards. Formally, a Markov Decision Process (MDP) is usually used to model this process. An MDP is typically a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \pi, p)$. $\mathcal{S}$ is the state space containing all possible states of the agent, $\mathcal{A}$ is the action space containing all possible actions that the agent can take, $\mathcal{R}$ is the reward space, $\pi(a|s)$ is a policy denoting the probability of taking action a under the state s, $p(s', r|s, a)$ denotes the environment dynamics defining the probability of changing to the state s' and receiving the reward r when taking action a under the state s in the environment. The objective of single-agent reinforcement learning is to find an optimal policy π^* so as to maximize the expected cumulative rewards along the time:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{s,r\sim p,a\sim\pi}(\sum_{t=0}^{\infty} \gamma^t r_t)$$

Deep reinforcement learning is the combination of reinforcement learning with deep learning. In the most recent years, people have made great progress in Atari game, Go chess [2,8] and many other complicated control problems, by combining the classical reinforcement algorithms with fancy deep neural network structure.

Basically, there are two class of methods in reinforcement learning. One is value-based methods, and the most widely used is Q-learning [3] method. This method constructs a greedy policy for the agent with a value function table or approximator, and use the following temporal-difference updates to modify value function estimation in each time step as a way of improving policy:

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$$

Deep Q-learning (DQN) [1] uses a deep neural network as the function approximator to compute value function. Instead of back propagating at each time step, the network is updated with a random selected batch of data from past time steps. The network is updated with the following loss function, where Q^{past} is computed by target network with old parameters.

$$L(\theta) = \mathbb{E}_{s,r\sim p,a\sim\pi}[r + \gamma \max_{a'} Q^{past}(s', a'|\theta^{past}) - Q(s, a|\theta)]^2$$

The other class of methods is policy-based methods. These methods use a policy estimator (e.g. deep neural network) to directly estimate the probability $\pi(a|s)$ of taking each action a under a certain state s. The policy estimator is updated using the policy gradient $\nabla_\theta \log \pi(a_t|s_t;\theta)R_t$ [4]. Deep Deterministic Policy Gradient (DDPG) [5] uses two deep neural networks in training — one is the actor neural network which determines the action, and a critic network which outputs the value function. The critic network is updated with the same method in Deep Q Learning, which the actor network is updated with the deterministic policy gradient (DPG) [6]. Proximal Policy Optimization (PPO) [7] is another kind of policy gradient method. It aims to take the biggest possible improvement step on a policy using the data we currently have, without stepping so far that performance collapse is caused.

Currently, deep learning and reinforcement learning have not been widely used in finance, but there are some exploration in applying them in business management, price prediction or trading. Heaton et al. [9] applied deep learning method in finance, and claimed that they are more powerful than some standard analysis methods in finance with a case study. Bao et al. [10] used a combination of wavelet transforms, stacked auto-encoders and long-short term memory (LSTM) [11] in the problem of stock price forecasting. Deng et al. [12] applied deep reinforcement learning techniques in trading problem.

References

[1] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Petersen, S. (2015). Human-level control through deep reinforcement learning. *Nature, 518(7540),* 529-533.

[2] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., ... & Dieleman, S. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature, 529(7587),* 484.

[3] Watkins, C. J. C. H. (1989). Learning from delayed rewards.

[4]Sutton, R. S., McAllester, D. A., Singh, S. P., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. *In Advances in neural information processing systems (pp. 1057-1063).*

[5] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., ... & Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint* arXiv:1509.02971.

[6] Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. (2014, June). Deterministic policy gradient algorithms.

[7] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint* arXiv:1707.06347.

[8] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., ... & Chen, Y. (2017). Mastering the game of go without human knowledge. *Nature, 550(7676),* 354-359.

[9] Heaton, J. B., Polson, N. G., & Witte, J. H. (2016). Deep learning for finance: deep portfolios. *Applied Stochastic Models in Business and Industry.*

[10] Bao, W., Yue, J., & Rao, Y. (2017). A deep learning framework for financial time series using stacked autoencoders and long- short term memory. *PLoS ONE,* 12(7).

[11] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation, 9*(8), 1735-1780.

[12] Deng, Y., Bao, F., Kong, Y., Ren, Z., & Dai, Q. (2016). Deep direct reinforcement learning for financial signal representation and trading. *IEEE Transactions on Neural Networks and Learning Systems.*

[13] Shmuma. (2019). https://github.com/PacktPublishing/Deep-Reinforcement-Learning-Hands-On-Second-Edition/blob/master/Chapter10/lib/environ.py