Stevens Institute of Technology

School of Business

Combining Reinforcement Learning and Goals-Based Investing

Author:

Fangchi Wu

Rui Wang

Huijun Wen

Supervisor:

Cristian Homescu

An Assignment submitted for the Stevens:

FE 800 Project in Financial Analytics

December 20, 2020

Substract

This project aims to explore a reinforcement learning algorithm combining with goal-based investing and implement it in python. The algorithm will use the Deep-Q network to identify the signals given by the market and tell the agent what action to take under different circumstances. Furthermore, for the convenience of users, the whole project will be automated, and the entire training simulation and data extraction can be achieved only by entering the stock symbol.

There are several parts in our project.

First, a function will be defined to derive the data from the yfinance and calculate the technical index data and then compose the data that meets the training format, and save it as '.csv' format automatically for multiple repetitions.

Second, two classes will be defined to describe the environment and implement deep Q learning algorithm. In addition, the main training function is defined with several important training parameters as inputs and output pictures recording the yield curve and trading information in the test set.

Besides, the implement of goal-based investing can be satisfied by changing the target function of profit to the target function of utility function, so that different users can define different risk preferences.

After that, several stocks will be used to test the validity of our algorithm and perfect our parameters.

Finally, We will optimize the whole model and test the influence of various important parameters on the simulation effect. And the effectiveness of risk control is verified by taking utility function as objective function.

Key Words: Deep Q learning, Goal-Based Investing, Utility Function

1 Introduction

Trading strategy plays an important role in investment. A good automated trading strategy is vital to investment companies and hedge funds. However, it is challenging to design a profitable strategy in a complex and dynamic stock market. As the rapid development of machine learning, more and more algorithms are being discovered. Reinforcement learning is one of the training of machine learning models to make a sequence of decision. It becomes more and more popular in many areas. Of course, reinforcement learning can be used in stock trading. People are willing to explore a deep reinforcement learning scheme that automatically learns a stock trading strategy by maximizing investment return.

In this project, we are going to explore a reinforcement learning algorithm combining with goal-based investing using for stock trading. We will use the Deep-Q network to identify the signals given by the market and tell the agent what action to take under each circumstance. We are going to use several stocks to test our algorithm and perfect our parameters.

Next, different people have different opinions on investing. Some people do not consider the risk and want to make as much money as possible. However, some people prefer stable returns and reduce risks as much as possible. So, we are going to use the concept of goal-based learning to make our algorithm popularization. We will make specific plan for different people by changing the parameters and make as much profit as possible.

2 Literature Review

Improving the stock returns is a long-term issue. Many researchers have tried to learn from the past marketing experiences. A long-standing ambition of artificial intelligence has been to create programs that can instead learn for themselves from first principles. Therefore, it is easy to think about apply a general reinforcement learning algorithm to the stock market, just like apply it to the chess.[1] However, there are already researchers using reinforcement learning for financial signal representation and trading.[2][3]

We use the RL module interacts with deep representations and makes trading decisions to accumulate the ultimate rewards in an unknown environment. [2]Hence, it leads to an interesting question in the context of trading: Can we train an RL model to beat experienced human traders on the financial markets? What kind of the RL model is the best to choose?

After reading the first part of the Alex's paper about Q-learning[4] , we learned that a Q-learning approach is used to maximize the expected utility of consumption. It fits our needs perfectly, thus, we decide to use this kind of method in our paper. In terms of the comparison part, we could also consider if the RL model is overact in stocks market or not.[5]

3 Methodology

3.1 Theoretical Knowledge

Deep Q learning is a model-free reinforcement learning technique proposed by Watkins in 1989. It compares the expected utility of available operations (for a given state) without the need for an environment model.It can also handle random transitions and rewards without adjustment. It has been shown that for any finite MDP, Q learning will eventually find an optimal strategy, that is, starting from the current state, the expected value of the total return for all successive steps is the maximum that can be achieved. Before learning begins, Q is initialized to any possible fixed value (chosen by the programmer).Then at each time t, the Agent selects an action a_t, gets a reward R_t, enters a new state [formula] and updates the Q value.Its core is the iterative process of value function, that is

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha[r_t + \gamma * maxQ(s_{t+1},a_t) - Q(s_t,a_t)]$$

where α is learning rate and γ is discount factor.

First, initialize the value function matrix, start episode, and then select a state. At the same time, the agent selects an action according to its own greedy strategy. After the agent uses the action, it gets a reward R and S, calculates the value function, and continues to iterate the next process.

Goal-based investing is a multiplicity of requirements, determined by investor preferences. The most common requirement is a combination of benefits and risks.

3.2 Implement Method

(1) Create the Deep Q network (the brain of the agent)

In the class DeepQNetwork, we define some basic variables such as n_action(buy sell and hold), n_features, batch_size of neural network training etc, initialize the variable ‘memory’(use to record the previous operation), and import the build net function. Then build the evaluate net and the target net and use the q_target and q_eval value to train the eval network.

(2) Create the environment.

In the class stock, we set the initial money = 10000, window_size=6 (6 days of the closing price), reward, state and other necessary variables. Then get current state for each single moment T, and we use the array as the status code. What’s more the buy and sell functions will calculate the quantity we can buy or sell, and update the stock market value and remaining cash. We need to calculate the total market value once a day, update the profit obtained compared to the previous day (used as a Q-learning reward), store daily operations, and update the network once a day. Finally, we put our goal-based target function $U(\lambda) = r - \lambda\sigma^2$ in the code to calculate reward. Different λ means different goals. Larger the λ is, the more we care about risk.

(3) Write main function code

Combine the previous functions we defined. Draw the graph and return the total profit.

4 Results

4.1 Simulation Results

We set the original stock curve as the benchmark – assuming that we have a full position in the stock over that period of time without doing anything, which is showed by the blue line. Besides, the my account line shows represents the revenue performance on the test set if the strategy is followed after the training. Our initial asset is 1,000,000 dollars. The following two charts respectively show the account return and trading history of Amazon stock as an example. And subsection 6.3 shows the total simulation results in other stocks depends on different λ.

4.1.1 Account Perfomance in AMZN Trading ($\lambda = 0$)

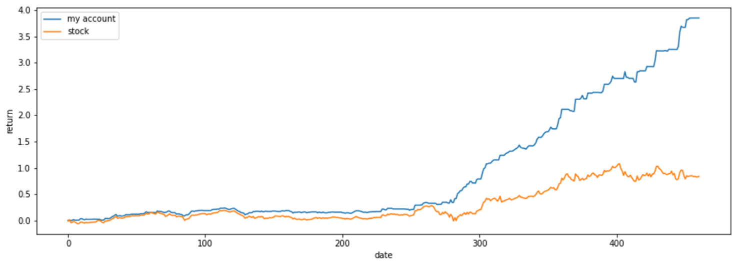

Figure 1: AMZN Profit Performance

4.1.2 Account Trading History ($\lambda = 0$)

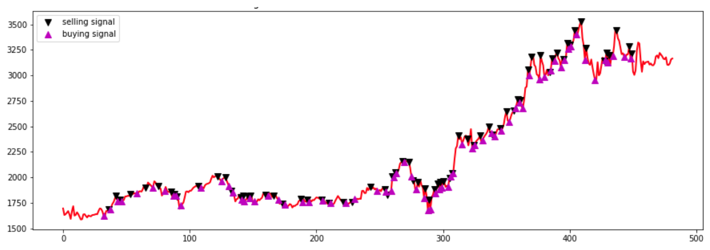

Figure 2: AMZN Trading History

4.1.3 Total Simulation Profit Results

Stock / Lambda Value	$\lambda = 0$	$\lambda = 10$	$\lambda = 1000$	benchmark
AAPL	889.0%	921.0%	99.0%	91.2%
AMZN	748.0%	632.0%	27.0%	83.2%
BA	1845%	1434%	214.0%	102.2%

4.1.4 Total Simulation Volatility Results (Annualized)

Stock / Lambda Value	$\lambda = 0$	$\lambda = 10$	$\lambda = 1000$	benchmark
AAPL	25%	21%	22%	38%
AMZN	20%	21%	13%	32%
BA	38%	49%	34%	68%

4.1.5 Analysis

1. The fully trained model well amplifies the returns from a single stock.

2. Compared with Benchmark, the model after training seldom keeps full position, so the corresponding volatility should be smaller than the volatility of the stock with full position.

3. Lambda's effect on risk control still needs to be tested, and more simulations are needed to test the effectiveness of the utility function.

4. Presumably, the rise of lambda means that users are more risk-conscious, thus sacrificing some potential gains.

4.2 Optimization and Parameters Effects

After completing the series of simulations in the previous section, we wanted to know how each of the important parameters in the model affected the performance in the test set, so as to better optimize our model.

4.2.1 Influence by Training Epochs

We choose the AMZN stock as the underlying stock and leave the other variables unchanged and set the feature numbers to 24 and want to test the effect by the number of training epochs, which represents the training intensity of the machine in the training set. The benchmark line is assuming that we have a full position in the stock over that period of time without doing anything. My account line represents the revenue performance on the test set if the strategy is followed after the training. The result is showed below:

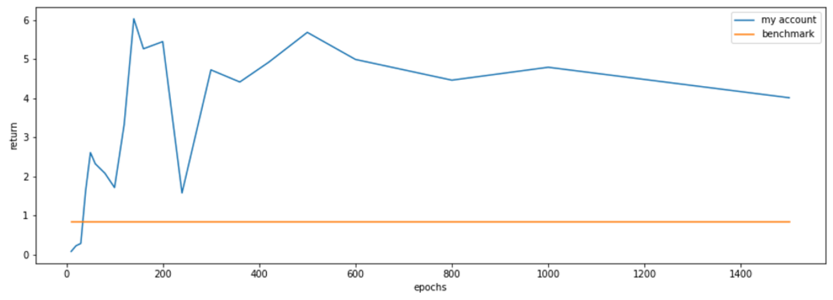

Figure 3: Training Epoch Effect

From the plot:

1. As the number of training epochs increases at the beginning, the performance quickly improves, and then exceeds the level of benchmark.

2. When the number of training continues to rise to more than 300 times, it will take more than 5 minutes, and the training result will tend to stably gradually decline.

3. There is some randomness in the training results.

4.2.2 Influence by Feature Numbers

The number of features is not only the number of columns of input data. The machine will dig up the data of the previous days as features according to the requirements of the number of features, so the number of features used in training can be much larger than the original number of input features. This time we set the training epochs equal to 200 constantly and the result is showed below:

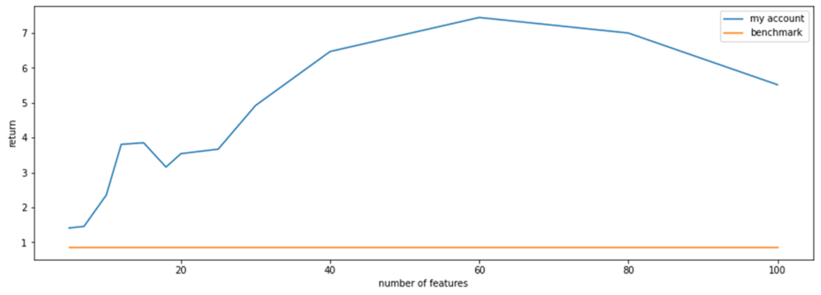

Figure 4: Feature Numbers Effect

From the plot:

1. with the rise of the number of features, the profit of the account continues to rise.

2. When the number of features reaches about 60, the profit of the account reaches its peak. After that, the profit of the account gradually declines.

4.2.3 Influence by λ

As we mentioned before, the utility function is defined by:

$$U(\lambda) = r - \lambda\sigma^2$$

where r is the return in each step, the σ is calculated and updated by EWMA model.

and we want to test whether our utility function can really do a good job of realizing goal-based risk-related problems. The result is showed below (The red line represents the total return, the blue line represents the annualized volatility):

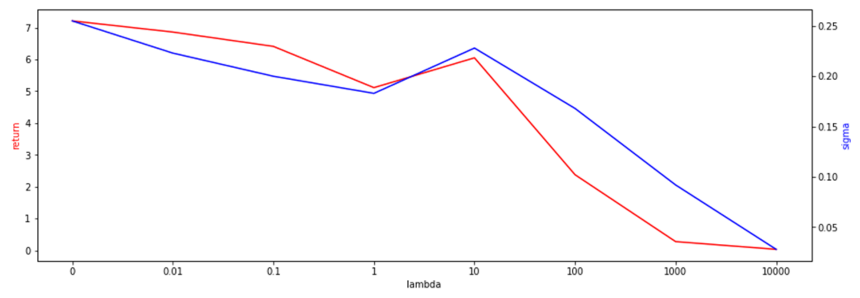

Figure 5: Feature Numbers Effect

From the plot:

1. As the lambda increases, machine will pay more attention to the risk, and even sacrifice some potential gains.

2. With the rise of lambda, the volatility and profit of the account gradually declined, and the rise and fall kept pace.

5 Conclusion and Future Work

5.1 Conclusion

1. Like many good models in the market, our model in the test set also greatly magnified the earnings of the U.S. stock market.

2. With adequate training, the trading signals are also very clear, without excessive frequency or sparsity.

3. From the point of view of parameter influence and optimization, the results are also very consistent with the basic behavior pattern of machine learning parameter influence, which changes continuously and generally has extreme value points.

4. After converting a goal-based problem into a utility function, the risk-related problem can be directly determined by a parameter(λ).

5.2 Future Work

1. In this project, historical data are used for backtracking, and we will conduct mock transactions and actual transactions in the future.

2. According to the requirements given by customers, we can add more nonlinear constraints to the problem, such as the limitation of the max drawdown.

3. There are many more detailed parameters in the model that can be optimized.

4. Finally convert the single stock problem to portfolio problem, which can be achieved by setting the reward from profit and volatility of one stock to multiple stocks in training process.

Reference

[1]. Silver, David, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot et al. "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play." Science 362, no. 6419 (2018): 1140-1144.

[2]. Deng, Yue, Feng Bao, Youyong Kong, Zhiquan Ren, and Qionghai Dai. "Deep direct reinforcement learning for financial signal representation and trading." IEEE transactions on neural networks and learning systems 28, no. 3 (2016): 653-664.

[3]. Horel, Enguerrand, Rahul Sarkar, and Victor Storchan. "Dynamic Asset Allocation Using Reinforcement Learning." (2016).

[4]. Weissensteiner, Alex. "A Q-Learning Approach to Derive Optimal Consumption and Investment Strategies." IEEE transactions on neural networks 20, no. 8 (2009): 1234-1243.

[5]. Chopra, Navin, Josef Lakonishok, and Jay R. Ritter. "Measuring abnormal performance: do stocks overreact?." Journal of financial Economics 31, no. 2 (1992): 235-268.

[6]. Bøe, Peter Røtvold, and Emil Sverre Heiervang Ruud. "Goal-Based Portfolios-A mean-variance optimization approach with subportfolios." Master's thesis, Handelshøyskolen BI, 2018.

[7]. Das, Sanjiv R., and Subir Varma. "Dynamic Goals-Based Wealth Management using Reinforcement Learning." (2019).

[8]. Dixon, Matthew, and Igor Halperin. "G-Learner and GIRL: Goal Based Wealth Management with Reinforcement Learning." arXiv preprint arXiv:2002.10990 (2020).

[9]. Friedman, Eli, and Fred Fontaine. "Generalizing across multi-objective reward functions in deep reinforcement learning." arXiv preprint arXiv:1809.06364 (2018).

[10]. Gu, Shixiang, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. "Continuous deep q-learning with model-based acceleration." In International Conference on Machine Learning, pp. 2829-2838. 2016.

[11]. Kim, Woo Chang, Do-Gyun Kwon, Yongjae Lee, Jang Ho Kim, and Changle Lin. "Personalized goal-based investing via multi-stage stochastic goal programming." Quantitative Finance 20, no. 3 (2020): 515-526.

[12]. Mossalam, Hossam, Yannis M. Assael, Diederik M. Roijers, and Shimon Whiteson. "Multi-objective deep reinforcement learning." arXiv preprint arXiv:1610.02707 (2016).

[13]. Parker, Franklin J. "Goal-based portfolio optimization." The Journal of Wealth Management 19, no. 3 (2016): 22-30.

[14]. Siddique, Umer, Paul Weng, and Matthieu Zimmer. "Learning Fair Policies in Multi-Objective (Deep) Reinforcement Learning with Average and Discounted Rewards." In International Conference on Machine Learning, pp. 8905-8915. PMLR, 2020.