April 25th, 2019

Portfolio Optimization with Q-Learning

Vikas Kumar
INFO 7978: Northeastern University

Abstract— **The objective of this research paper is to optimize a stock portfolio of five stocks using state-of-art model free Reinforcement Learning, Q Learning. The problem needs to be formulated like a Markov Decision Process to be able to apply Q Learning to it.**

INTRODUCTION

Portfolio optimization is the process of choosing the proportions of various assets to be held in a portfolio, in such a way as to make the portfolio better than any other according to some criterion. The criterion will combine the expected value of the portfolio's rate of return as well as some other measures of financial risk. The portfolio optimization problem is generally specified as a constrained utility-maximization problem. Although portfolio utility functions can take many forms, common formulations define it as the expected portfolio return. One approach to portfolio optimization is using model free Reinforcement Learning using Q Learning. The problem needs to be formulated like a Markov Decision Process to be able to apply Q Learning to it.

Reinforcement learning (RL) is an area of Artificial Intelligence. It is about taking suitable action to maximize reward in a particular situation. It is employed by various software and machines to find the best possible behavior or path it should take in a specific situation. RL involves an agent, a set of states, and a set of actions per state. By performing an action, the agent transitions from state to state. Executing an action in a specific state provides the agent with a reward (a numerical score). Markov decision processes are useful for studying a wide range of optimization problems solved via dynamic programming and reinforcement learning. MDP’s are meant to be a straightforward framing of the problem of learning from interaction to achieve a goal. The agent and the environment interact continually, the agent selecting actions and the environment responding to these actions and presenting new situations to the agent. Formally, an MDP is used to describe an environment for reinforcement learning, where the environment is fully observable. Almost all RL problems can be formalized as MDPs.

Q-learning is a model-free reinforcement learning algorithm. The goal of Q-learning is to learn a policy, which tells an agent what action to take under what circumstances. It does not require a model (hence the connotation "model-free") of the environment, and it can handle problems with stochastic transitions and rewards, without requiring adaptations.

For in the sense that it maximizes the expected any finite Markov decision process (FMDP), Q-learning finds a policy that is optimal value of the total reward over any and all successive steps, starting from the current state. Q-learning can identify an optimal action-selection policy for any given FMDP, given infinite exploration time and a partly-random policy. "Q" names the function that returns the reward used to provide the reinforcement and can be said to stand for the "quality" of an action taken in a given state.

APPROACH

1.Data Collection and Processing
2.Problem Formulation
3.Algorithm Implementation
4.Training and Backtesting

1.Dataset

The data for the project was gathered from Yahoo finance. The 5 stocks that were chosen are Apple, Google, Netflix, Tesla and Facebook due to their history of low volatility. The data consists day-to-day prices of each stock for a period of 5 years.

2.Problem Formulation

We have 5 stocks in our portfolio limited to only one transaction of exchange per day.

Instead of a continuous action space, where the system chooses what percentage of the portfolio each stock should constitute (e.g. stock A should constitute 35% of the portfolio's total value), the system was given one action 5% exchange. For every time it performs an action a, the portfolio sells at a × total of the low-valued stock and buys the corresponding amount of the high-valued stock (and vice versa for at a < 0). This discrete action-space, alongside the simplified state-space, helps make the problem tractable. AWe transaction cost of 1% of each action/transaction was used to include the trading cost.

Defining State-ID:
Based on the number of shares and the current price of each stocks I am calculating the weightage of each stock in the total portfolio value as percentages.
The range of these percentages will be denoted by an alphabet starting from A to U. Starting from alphabet 'A' which would represent 0% share in total portfolio value, B would represent 0-5 % share in total portfolio value, C would represent 5-10 % share in total portfolio value, D would represent 10-15 % so on till U which would represent 95-100% share in total portfolio value. So, consider we have a stock portfolio

with the weightage of 5 stocks in the portfolio as [12%, 22%, 48%, 10%, 8%], this can be represented by 5-character string as CFKCC. Similarly [20%, 20%, 20%, 20%, 20%] is EEEEE [21%, 22%, 32%, 10%, 15%] is FFHCD. These 5-character strings are used to represent the different states and are used as state- id's.

3.Algorithm Implementation

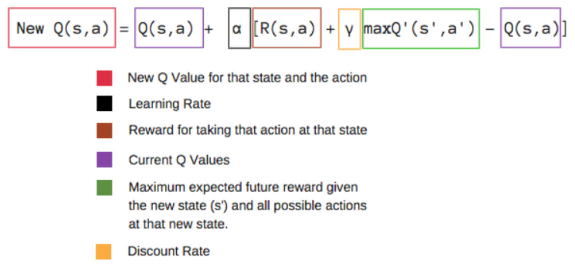
Q-Learning algorithm is implemented as follows:

Step 1: Initialize the Q-Table
We will first build a Q-table. We will initialize the values at 0.

Step 2: Choose an action
Step 3: Perform an action

Step2 and step3 is done for an undefined amount of time. This means that this step runs until the time we stop the training, or the training loop stops as defined in the code. We will choose an action (a) in the state (s) based on the Q-Table. But, as mentioned earlier, when the episode initially starts, every Q-value is 0.

Step 4 – Measure reward
Step 5 – Update Q Table



$$\text{New } Q(s,a) = Q(s,a) + \alpha\,[R(s,a) + \gamma\,\max Q'(s',a') - Q(s,a)]$$

🟥 New Q Value for that state and the action
⬛ Learning Rate
🟫 Reward for taking that action at that state
🟪 Current Q Values
🟩 Maximum expected future reward given the new state (s') and all possible actions at that new state.
🟧 Discount Rate

Now we have taken an action and observed an outcome and reward. We need to update the function Q(s,a). Here we perform iterations again and again until learning is stopped. In this way the Q-Table will be updated.

- Our goal is to maximize the value function Q.
- The Q table helps us to find the best action for each state.
- It helps to maximize the expected reward by selecting the best of all possible actions.
- Q(state, action) returns the expected future reward of that action at that state.
- This function can be estimated using Q-Learning, which iteratively updates Q(s,a) using the Bellman equation.
- Initially we explore the environment and update the Q-Table. When the Q-Table is ready, the agent will start to exploit the environment and start taking better actions.

4.Exploration and exploitation trade off

There is an important concept of the exploration and exploitation trade off in reinforcement learning. Exploration is all about finding more information about an environment, whereas exploitation is exploiting already known information to maximize the rewards.

In this project I am using 4 years of data for exploration/training (80%) and 1 year of data for exploitation/backtesting (20%).

Evaluation (do-nothing benchmark)

I am evaluating my system performance by comparing it with the benchmark values which is calculated by by giving equal weightage to all stocks (i.e. 20% or "EEEEE") and then allowing the stock value to increase over the period. If the stock portfolio is kept aside for the exploitation period, then the system should outperform the price rise of those stocks in that period.

Results

The system was only trained for 50 iterations. I experimentally found that training beyond 100 led to overfitting and below 20 led to underfitting and the system performed below benchmark

The system was tested by varying Q-Learning parameters such as Learning rate and discount factor and the optimum values were found out

The system was tested on various parameters and the pre-defined benchmark, although the most interesting outcomes of this test surfaced when the variables in the Q-learning equation were explored.

VARIATION IN LEARNING RATE

From the table below, we can observe that as we increase the values of Learning rate the system starts performing better than the benchmark. For the values beyond alpha=0.3 the there is steady increase in the system performance. In fully deterministic environments, higher learning rate tend to give better results and learning rate of 1 is optimal.

Variation in alpha

	ALP-HA	Bench-mark	System
1	0.9	2499934	2554361
2	0.8	2499934	2514662
3	0.7	2499934	2625355
4	0.6	2499934	2584354
5	0.5	2499934	2499934
6	0.4	2499934	2546146
7	0.3	2499934	2498868
8	0.2	2499934	2387937
9	0.1	2499934	2464770

Variation in Gamma

	GA-MMA	Bench-mark	System
1	0.9	2499934	2536933
2	0.8	2499934	2519398
3	0.7	2499934	2535833
4	0.6	2499934	2535543
5	0.5	2499934	2514032
6	0.4	2499934	2485100
7	0.3	2499934	2445100
8	0.2	2499934	2459368
9	0.1	2499934	2450415

VARIATION IN DISCOUNT FACTOR

From the table above, we can observe that as we increase the values of Discount factor the system starts performing better than the benchmark.

For the values beyond alpha=0.4 there is steady increase in the system performance. This improvement in performance is expected with increase in gamma values since a factor approaching 1 will make it strive for a long-term high reward and will lead to better system performance

FUTURE WORK

After approaching the simplified version of Portfolio Optimization problem with Reinforcement Learning we can improve further on this by incorporating more data of stocks and other factors which affect the market
Such as Sharpe ratio, ADS index and variables that capture overall market trend. The more data we train our algorithm it will be able to capture the long-term factors such as recessions, inflation, currency fluctuations etc.

Context:

I took a few worthy references for project initiation and applied the concepts to progress further by implementing Q-Learning algorithm on stocks data and finally building Portfolio optimization system.

CONCLUSION

In this project, I have formulated the portfolio optimization problem into a RL problem with Q Learning. It can be concluded that this is one of the most unexplored approaches for solving this classic problem. In my opinion to improve this project further we use neural networks to approximate Q values. Using this approach we can also integrate stock sentiments as well. We can also use a larger dataset to train and take into account the interactions between assets.

ACKNOWLEDGEMENTS
We are grateful to professor Nik Bear Brown for his true and dedicated guidance right from the project initiation to implementation.

REFERENCES

[1http://cs229.stanford.edu/proj2016/report/JinElSaawy-PortfolioManagementusingReinforcementLearning-report.pdf

[2] https://medium.freecodecamp.org/a-brief-introduction-to-reinforcement-learning-7799af5840db

[3]https://www.learndatasci.com/tutorials/reinforcement-q-learning-scratch-python-openai-gym/

[4https://github.com/simoninithomas/Deep_reinforcement_learning_Course/tree/master/Q%20learning/FrozenLake

[5] https://towardsdatascience.com/a-blundering-guide-to-making-a-deep-actor-critic-bot-for-stock-trading-c3591f7e29c2

[6] Formulating MDPs from scratch - https://medium.com/@curiousily/solving-an-mdpwith-q-learning-from-scratch-deep-reinforcementlearning-for-hackers-part-1-45d1d360c120

[7]https://finance.yahoo.com/quote/CSV/history/

[8]https://www.investopedia.com/university/concepts/concepts7.asp

[9]https://www.theschool.ai/courses/move-37-course/lessons/markovdecisionprocesses/

[10]https://en.wikipedia.org/wiki/Portfolio_optimization

[11]https://stanford.edu/class/ee103/lectures/portfolio_slides.pdf

[12]
https://finance.zacks.com/optimize-stock-portfolio-8617.html

[13]
https://github.com/dennybritz/reinforcement-learning/blob/master/TD/Q-Learning%20Solution.ipynb

[14]
https://github.com/sachinh19/portfolio-optimization/blob/documentation/ProjectReport.pdf

[15]
https://github.com/simoninithomas/Deep_reinforcement_learning_Course/blob/master/Q%20learning/FrozenLake/Q%20Learning%20with%20FrozenLake.ipynb

[16]https://medium.freecodecamp.org/diving-deeper-into-reinforcement-learning-with-q-learning-c18d0db58efe