# *Portfolio Optimization using Reinforcement Learning*

***(April 2018)***

Aditya Masurkar (*Author*)
Northeastern University
Boston, USA
masurkar.a@husky.neu.edu

Sachin Haldavanekar (*Author*)
Northeastern University
Boston, USA
haldavanekar.s@husky.neu.edu

*Abstract*— **In this project, we use Q-Learning to perform reinforcement learning to optimize a stock portfolio of five stocks. The system performs better than the do-nothing benchmark in most cases.**

***Keywords— stock portfolio optimization, q learning, reinforcement learning, markov decision process,***

## 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.

**Markov decision processes (MDPs)** provide a mathematical framework for modeling decision making in situations where outcomes are partly random and partly under the control of a decision maker. MDPs are useful for studying a wide range of optimization problems solved via dynamic programming and reinforcement learning.

**Reinforcement learning (RL)** is an area of machine learning inspired by behaviorist psychology, concerned with how software agents ought to take actions in an environment to maximize some notion of cumulative reward.

**Q-learning** is a reinforcement learning technique used in machine learning. The technique does not require a model of the environment. Q-learning can handle problems with stochastic transitions and rewards, without requiring adaptations.

For any finite Markov decision process (FMDP), Q-learning eventually finds an optimal policy, in the sense that the expected value of the total reward return over all successive steps, starting from the current state, is the maximum achievable. Q-learning can identify an optimal action-selection policy for any given FMDP.

## Literature Survey

The use of Q-Learning to optimize stock portfolio is a relatively new concept. We did not find any previous works where Q-Learning was used to train the AI agent.

We primarily referenced three papers: Portfolio Management using Reinforcement Learning, Reinforcement Learning for Trading Systems and Portfolios, Deep Reinforcement Learning in Large Discrete Action Spaces.

Although we referred a few techniques for discretizing states, formulating actions and designing constraints, each paper adopts a different approach when constructing the algorithm.

The approaches discussed were interesting, but we decided to focus on Q-Learning, to ensure that the problem was transformed into a Finite Markov Decision Process (FMDP). A few assumptions were made to limit the natively infinite state-action pairs.

## Experimental And Computational Details

The project design consisted of the following categories of tasks including formulation of the problem, understanding of the Q-Learning equation and all the factors which manipulate its outcome, also working with datasets from NASDAQ,

### *A. Problem Formulation*

We are trying to solve a simplified version of the classic Portfolio Optimization Problem, so that it can be within the scope of Reinforcement learning[Q-learning].

**Assumptions**:

We have made the following assumptions to restrict the state space that the program works with:

- A stock portfolio will consist of exactly 5 stocks [A, B, C, D, E] at any given time.
- Only one transaction of exchange (selling one stock and buying another) is allowed per day.
- Only 5% of one stock can be exchanged with another stock on any given day.

**State Description**:

To reduce the infinite state space concern, we have classified the state as follows:

We have represented the state of the stock in a string of 5 alphabets each alphabet is one from letters A to U, both inclusive.

Based on the percentage of stock in the portfolio in the current state, a letter is assigned to represent the state of that stock in the overall state of the portfolio.

The letter 'A' corresponds to a stock with 0% share in the portfolio, B corresponds to values in the set (0% - 5%], C corresponds to (5-10] and so on till U which corresponds to (95,100]. e.g. If a stock portfolio in share of percentages looks something like this: [9,24,44,20,3]; then it will be represented in the 5-character string representation as CFJEB. Similarly [19,24,44,7,6] is EFJCC and [95,1,1,2,1] is TBBBB.

Thus, the state transition is a simple and convenient change of alphabets in the string of 5 characters. The possible actions from the current state will be obtained by checking all possible 5% exchanges that can happen on the stock portfolio.

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.

In addition, a small transaction cost per transaction (0.001% of total value or 2% of each action) was used to encapsulate the various trading fees.

**Goal**:

Goal of the system is to maximize the value of the stock portfolio over a period of 5 years (which includes 4 years of exploration + 1 year of exploitation)

**Evaluation** (do-nothing benchmark):

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.

For all the stocks, the do-nothing benchmark is calculated by giving equal weightage to all stocks (i.e. 20% or “EEEEE”) and then allowing the stock value to increase over the period.

### *B. Q-Learning*

$$Q(s_t,a_t) \leftarrow \underbrace{Q(s_t,a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \left( \overbrace{\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1},a)}_{\text{estimate of optimal future value}}}^{\text{learned value}} - \underbrace{Q(s_t,a_t)}_{\text{old value}} \right)$$

Equation (1) – Q-Learning

The equation consists of the following components:

a. The old Q value of the current state – Q [s, a]

b. The Reward obtained for transition from current state to next state - r

c. The learning rate - α

d. The discount factor - γ

e. The optimal Q value of next state – Q [s’, a’]

**<u>Influence of Variables:</u>**

**Explore v/s Exploit or Learning Rate**

The learning rate or step size determines to what extent newly acquired information overrides old information. A factor of 0 makes the system learn nothing (exclusively exploiting prior knowledge), while a factor of 1 makes the agent consider only the most recent information (ignoring prior knowledge to explore possibilities). In fully deterministic environments, a learning rate of α=1 is optimal. When the problem is stochastic, the algorithm converges under some technical conditions on the learning rate that required it to decrease to zero. In practice, often a constant learning rate is used, such as α=1 for all t.

**Discount Factor**

The discount factor γ determines the importance of future rewards. A factor of 0 will make the agent "myopic" (or short-sighted) by only considering current rewards, while a factor approaching 1 will make it strive for a long-term high reward. If the discount factor meets or exceeds 1, the action values may diverge. For γ = 1, without a terminal state, or if the system never reaches one, all environment histories become infinitely long, and utilities with additive, undiscounted rewards generally become infinite. Even with a discount factor only slightly lower than 1, Q-function learning leads to propagation of errors and instabilities when the value function is approximated with an artificial neural network. In that case, it is known that starting with a lower discount factor and increasing it towards its final value accelerates learning.

Working with different values for constants like α and γ and comparing the performance of the system at various values yielded results that have been described using graphs in later sections.

### *C. Data Preprocessing*

Around 5 years of day-to-day historical stock prices were used as the dataset for five stocks from NASDAQ 100. The stocks that were used are Apple, Cisco, Fannie Mae, Marriott International, Sirius XM Holdings.

The python library Pandas.DataReader is used to read csv files and the data retrieved are fetched into a complex dictionary-based data structure to allow easy access to daily stock prices during execution.

The historical data was obtained from the Yahoo Finance website was used as is in the exploration and exploitation phases.

### *D. Exploration and Exploitation*

Keep in accordance with empirical research studies 75% of the data is used for training(exploration) the system and the remaining 25% is used for validation (exploitation).

After evaluating the dataset and we decided to select the first four years of data for exploration i.e. training the system, and the last year of data was selected for exploitation to validate the performance of the system.

### *E. Overfitting and Underfittng*

Experimentally we determine that if the system is trained more than 50000 iterations over a period of 4 years then we observe overfitting and the system performs below the benchmark.

Similarly, if we train the system for less than 1000 iterations it underfits and the system performed consistently below par.

## RESULTS AND DISCUSSION

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.

### *A. Influence of Alpha*

| **No.** | **Influence of Alpha** | | |
|---|---|---|---|
| | ***α*** | ***Benchmark*** | ***System*** |
| 1 | 0.95 | 36843 | 40900 |
| 2 | 0.8 | 36843 | 38300 |
| 3 | 0.6 | 36843 | 37500 |
| 4 | 0.4 | 36843 | 36870 |
| 5 | 0.2 | 36843 | 36900 |
| 6 | 0.1 | 36843 | 36860 |

(Note: Values in the table has been rounded off to the nearest multiple of 10.)

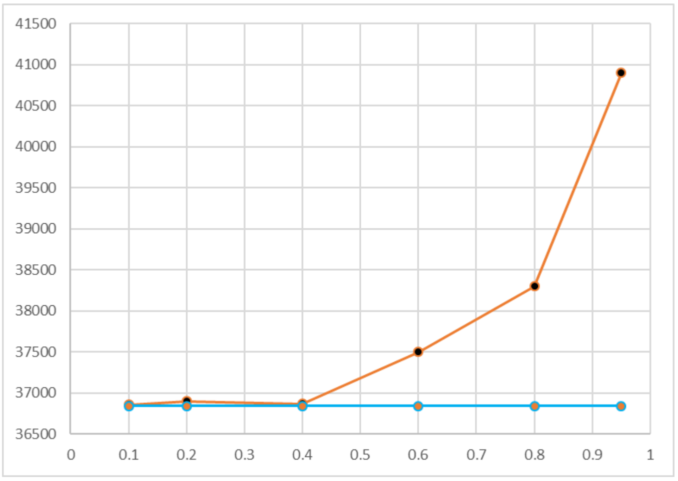

Fig 1. Alpha Influence – Benchmark (Blue Plotted Line) v/s System (Orange Plotted Line)

The system was tested on various inputs including of α which yielded the following observations:

- The value of α was varied between the range of 0.1 to 0.95 at different intervals.
- The system performed on par with the benchmark till the value α of the was below 0.4
- There was a steep increase in performance of the system for values of α which were beyond 0.4
- This is the expected behavior since the environment is deterministic and higher values of α tend to give better results in such cases.

### *B. Influence of Gamma*

Accordingly, a similar experiment was performed working with γ values. The outcome has been described below:

- The value of γ was varied between the range 0.1 to 0.8 at regular intervals.
- The system's performance was under par as compared to the benchmark for γ values under 0.3
- For γ = 0.3 the outcomes improved consistently and remained stable for all higher values.
- This shows that higher values of γ influenced better results as the stock portfolio has the goal of achieving long term high reward.

| **No.** | **Influence of Gamma** | | |
|---|---|---|---|
| | ***γ*** | ***Benchmark*** | ***System*** |
| 1 | 0.1 | 36843 | 35200 |
| 2 | 0.2 | 36843 | 35300 |
| 3 | 0.3 | 36843 | 39400 |
| 4 | 0.4 | 36843 | 38600 |
| 5 | 0.7 | 36843 | 38293 |
| 6 | 0.8 | 36843 | 38500 |

(Note: Values in the table has been rounded off to the nearest multiple of 10.)

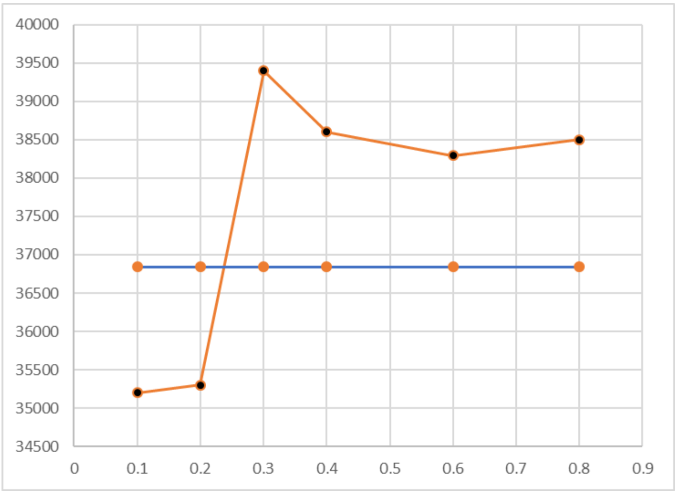

Fig 2. Gamma Influence – Benchmark (Blue Plotted Line) v/s System (Orange Plotted Line)

### C. *Comparison with Benchmark*

| **No.** | **Comparison with Benchmark** | | |
|---|---|---|---|
| | ***Benchmark*** | ***System*** | ***% Gain*** |
| 1 | 36843 | 40900 | 11 |
| 2 | 36843 | 38730 | 5 |
| 3 | 36843 | 37455 | 2 |
| 4 | 36843 | 37860 | 3 |
| 5 | 36843 | 37200 | 1 |
| 6 | 36843 | 39700 | 8 |
| 7 | 36843 | 38570 | 5 |

(Note: Values in the table has been rounded off to the nearest multiple of 10.)

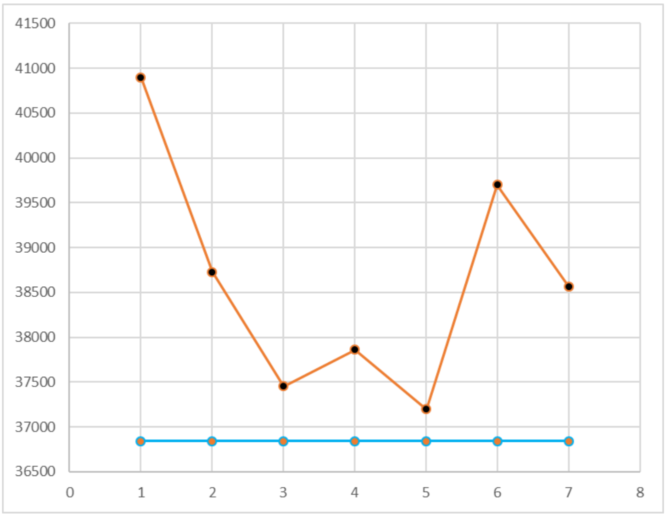

Fig 3. Performance w. r. t. Benchmark – Benchmark (Blue Plotted Line) v/s System (Orange Plotted Line)

The system's performance was compared with the benchmark. The system was tested by performing 7 runs at optimal values of $\gamma = 0.3$ and $\alpha = 0.95$.

It was observed that the system performs consistently better than the benchmark when the variables in the Q-learning equations are optimal.

## Future Work

After providing a proof of concept, with a simplified version of the problem, our future work on this can result in strong evidence that this technique is an effective way of approaching this problem. We limited the number of stocks to five and it can be made flexible for the system to learn independent of the number of stocks in the portfolio.

Another approach would be to train on weekly data or hourly data points which would allow the system to explore more before the exploitation phase. There is also a possibility of working with the current data along with historical data which can bring a different dimension to this problem. We can also work with more than 25 years of data to encapsulate the effects of bull market and bear market so that all kinds of market behavior is captured during the exploration period. This is will enable the system to recognize the type of market, amongst the two categories mentioned above, and would act optimally.

We can incorporate many other features like Sharpe ratio, price-to-earnings ratio and debt-to-equity ratio to ensure the system will have precise resemblance of the stock market.

## Conclusion

In summary, after applying the Reinforcement Learning technique to a simpler version of the Portfolio Optimization problem it can be concluded that this is one of the most unexplored approaches for solving this classic problem and can be improved upon by working on related factors like historical data, state variations and stocks from different industries.

## ACKNOWLEDGMENT

The idea was inspired from the work by Olivier Jin, Hamza El-Saawy in Portfolio Management using Reinforcement Learning.

The authors would like to thank Prof. David A. Smith, for providing valued feedback during various stages during the project. The Yahoo finance website for the datasets for five years of data.

Both the authors contributed to project proposal, problem formulation, code design, performance analysis and creation of the project report.

## REFERENCES

[1] Olivier Jin, Hamza El-Saawy "Portfolio Management using Reinforcement Learnin

[2] John Moody, Matthew Saffell "Reinforcement Learning for Trading Systems and Portfolios"

[3] Gabriel Dulac-Arnold, Richard Evans "Deep Reinforcement Learning in Large Discrete Action Spaces

[4] Online Portfolio Visualizer – https://www.portfoliovisualizer.com/optimize-portfolio

[5] CNBC Nasdaq 100 - https://www.cnbc.com/nasdaq-100/

[6] Yahoo Finance - https://finance.yahoo.com/

[7] Pandas documentation - https://pandas.pydata.org/pandas-docs/stable/api.html#excel

[8] Python documentation - https://docs.python.org/2/library/index.html

[9] Reinforcement Learning Wiki - https://en.wikipedia.org/wiki/Reinforcement_learning

[10] Portfolio Optmization Wiki - https://en.wikipedia.org/wiki/Portfolio_optimization

[11] Q-Learning Wiki - https://en.wikipedia.org/wiki/Q-learning

[12] Portfolio Optimization – https://link.springer.com/article/10.1007%2Fs11573-006-0006-z

[13] Portfolio Optimization – Stanford University https://stanford.edu/class/ee103/lectures/portfolio_slides.pdf

[14] Formulating MDPs from scratch - https://medium.com/@curiousily/solving-an-mdp-with-q-learning-from-scratch-deep-reinforcement-learning-for-hackers-part-1-45d1d360c120