# Stock Portfolio Optimization with Markov Decision Processes and Q-Learning

**Joshua R. Chang**
Department of Computer Science
Stanford University
jrchang@stanford.edu

**Troy L. Shen**
Department of Computer Science
Stanford University
troyshen@stanford.edu

## Abstract

Portfolio optimization is a highly complex problem that is increasingly being approached with artificial intelligence techniques. Our project engages this problem using Markov Decision Processes with Q-Learning. We gathered data from the S&P 500, trained our state-based model, and tested the performance of the portfolio our model generated against both a baseline and an oracle. In our research, we found that Q-learning reinforced state-based models were effective at outperforming the market, but enormous state-action spaces and slow run-time forced us to contrain the problem.

## 1 Introduction

In recent years, investing firms and hedge funds have looked for strategies to streamline and improve their ability to create investments that maximize returns while minimizing risk. With its ability to provide inferences from millions of data points, artificial intelligence has increasingly been leveraged as a tool to make predictions about the market. Although reinforcement learning and deep learning strategies have been used by top investing firms like Blackrock and Two Sigma, as a report by Bloomberg in May 2019 notes, beating the market remains one of the most difficult challenges in artificial intelligence [3].

Our project attempts to solve the problem of stock portfolio optimization using a state-based approach, namely Markov Decision Processes with reinforcement learning. The reinforcement learning algorithm used is Q-Learning. We aim to develop a model that produces an optimal policy given the state of a portfolio and possible subsequent actions. Although it is unlikely our research will dramatically change the techniques used by modern investors, we hope it will shed light on the specific and intriguing ways artificial intelligence is being applied to the traditional problem of beating the market.

## 2 Task Definition

Given one year of input stock data from the S&P 500 index - namely daily opening, high, low, and closing prices - we want to predict the optimal policy for the state of a given portfolio and create a portfolio that performs well relative to the market. The starting portfolio begins with an initial amount of capital and invests in certain stocks in accordance with our model. At the end of a fiscal year (or any time interval specified), we calculate the return of the best portfolio our model was able to produce. As a point of comparison, we use several benchmarks to test against. Our baseline is a random selection of stocks from the S&P 500, and our oracle is the performance of top performing hedge funds in the year we tested on. To evaluate the effectiveness of our algorithm, we measure the

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

difference in percent gain over a given time period between our optimal portfolio and the S&P 500 as a whole.

## 3 Infrastructure

Our model utilizes a state-based approach with Markov Decision Processes and Q-Learning.

### 3.1 States

Each state contains four pieces of information: stock prices from time $t - N$ to time $t$ (where $N$ is the lookback window), the portfolio allocation from time $t - 1$, the time $t$, and the remaining capital that can be invested. Let $X$ be any natural number. Thus, the state effectively contains stock data within a certain time period as well as the portfolio choice we made previously.

$$State_t = [S^{t-X}, S^{t-X+1}, \ ... \ , S^t, P^{t-1}, t, C]$$

At each $S^t$, we know the price of each stock $A_k^t$ for our $k^{th}$ stock at time $t$.

$$S^i = [A_1^t, A_2^t, \ ... \ , A_{499}^t, A_{500}^t]$$

At each $P^t$, we know the proportion of each stock $w_k^t$ of our $k^{th}$ stock at time $t$.

$$P_t = [w_1^k, w_2^k, \ ... \ , w_{499}^k, w_{500}^k]$$

$C$ keeps track of how much capital has not been invested.

### 3.2 Actions

Because of the enormous state-action space that results from the countless combinations of stock selections and allocation to each, we opted to limit the actions our model could make to three actions: using 10% of our remaining capital to invest in $k$ random stocks selected from the S&P 500, selling 10% of our ownership in all stocks, or doing nothing. This enabled us to constrain the state-action space to a more reasonable size. Each action produces the new portfolio $P^t$ we select at time $t$. This makes our state-action pair the following:

$$[[S^{t-N}, S^{t-N+1}, \ ... \ , S^t, P^{t-1}, t+1, C'], P^t]$$

Note that C' denotes the remaining capital in our new state $s'$ after an action has been performed on state $s$.

### 3.3 Reward

Our reward for any state-action pair $[[S^{t-N}, S^{t-N+1}, \ ... \ , S^t, P^{t-1}, C'], P^t]$ is defined as the return on investment given that we choose portfolio $P^t$. This essentially means the difference in the value of our portfolio between the end of a period and the beginning of the period. Let the total value of our portfolio at time $t$ be defined as $V^t$. Thus, our reward function can be expressed as:

$$R_T = \sum_{i=1}^{N} \frac{V^t \cdot w_i^t}{A_i^t} \cdot (A_i^t - A_i^{t-X})$$

The left hand-side within the summation is the amount of shares we have of each stock (the valuation of our stake in that company divided by the cost of each share). The right hand-side is the difference of the stock prices within the time interval. Note that $X$ is a natural number. The summation thereby calculates how much we made or lost for each stock in our portfolio and sums these values together.

### 3.4 Policy

Our policy at each step is to select the portfolio $P^t$ that has the maximum reward function, accounting for the heuristic that is defined using feature extraction.

## 4 Methodology

The goal of our project was to reinforce a model to invest in stocks with good returns while using a heuristic to account for the viability of a portfolio.

We applied a Markov Decision Process using the following methodology. At each time interval $t$, the state $s$ was comprised of the economic state of all the stocks in our portfolio and the amount of each stock we had. Actions corresponded to changing the amount each stock occupied in our portfolio (i.e. buying more of a stock, selling some of a stock, doing nothing). Our portfolio began at time $t = 0$ with a fixed amount of capital. We trained our model using two key techniques: Q-Learning and Feature Extraction.

### 4.1 Q-Learning

We chose to use Q-Learning as a reinforcement-based approach to train our model. Q-Learning, a model free reinforcement-learning algorithm, was suitable for our task because it attempts to learn a policy that maximizes the total reward. In this case, the reward was the portfolio gain or loss as defined above in the Infrastructure section.

The parameters for our Q-Learning algorithm were as follows:

Learning Rate: Our learning rate decreases with each iteration $i$ according to the function:

$$\frac{1}{\sqrt{i}}$$

Gamma: We selected a discount factor of $\gamma = 1$, meaning present rewards were equally valued as future rewards.

Reward: See Section 3.3

### 4.2 Feature Extractor

Our feature extractor was a key component of our model as it enabled us to establish a heuristic for predicting the viability of a portfolio (in addition to the reward function). It is comprised of three major features: standard deviation of return on a portfolio, industry type of each stock in our portfolio, and stock momentum.

#### 4.2.1 Standard Deviation of Return

Standard deviation of return of an entire portfolio measures the variability of the expected rate of return of a portfolio and is comprised of three components: proportion of each asset in a portfolio, standard deviation of return of each asset in the portfolio, and the co-variance of returns between each pair of assets in the portfolio.

The intuition behind using standard deviation of return of a portfolio as opposed to aggregated standard deviations of individual stocks was twofold: the former enabled us to both account for risk and also optimize for diversification of portfolio, a good indicator of viable portfolios in investment theory. We were able to discern all these using our stock data. The standard deviation of return of a portfolio is given by the following equation:

$$\sigma_P = \sqrt{\sum_{i=1}^{N} w_i^2 \cdot \sigma^2(k_i) + \sum_{i=1}^{N} \sum_{i \neq j}^{N} w_i \cdot w_j \cdot \text{Cov}(k_i \cdot k_j)}$$

$N$ is the number of assets in our portfolio, $w_i$ is the weight or proportion of our $i^{th}$ asset relative to our total portfolio, $w_j$ is the weight or proportion of our $j^{th}$ asset relative to our total portfolio, $\sigma^2(k_i)$ is the variance of return of our $i^{th}$ asset, and Cov$(k_i \cdot k_j)$ is the covariance of return of our $i^{th}$ and $j^{th}$ asset.

### 4.2.2 Industry Type

The growth and profitability of specific industry types varies greatly from year to year, making industry type a logical feature to include in the feature extractor. Using data collected from the previous 12 years of the S&P 500 [2], we accounted for the growth rate of each industry per year to produce a score for each industry type. Scores for each stock in our portfolio were then aggregated and weighted to serve as a heuristic.

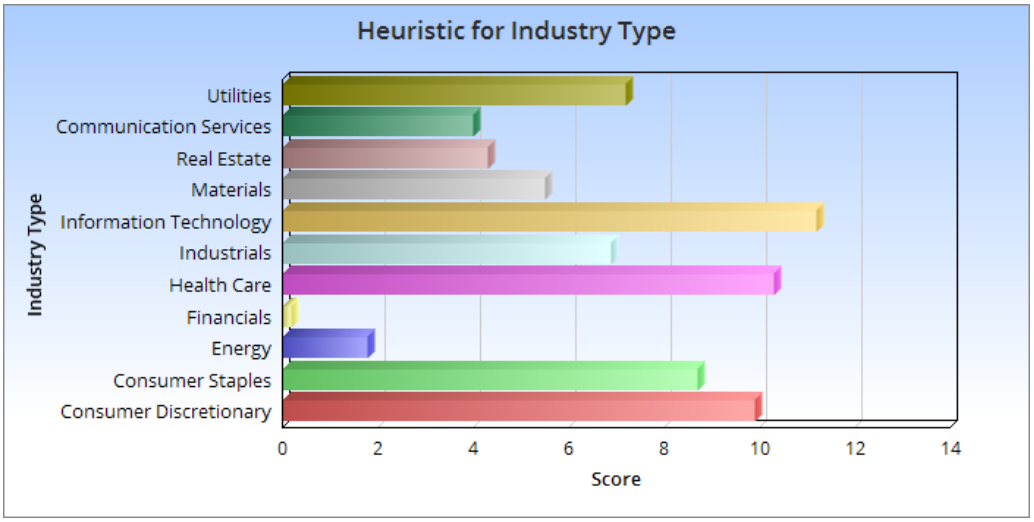


Figure 1: Scores by Industry Type in the S&P 500

### 4.2.3 Stock Momentum

Stock momentum is another useful indicator of the viability of a portfolio because it accounts for the change in the rate of rise or fall of each stock throughout a window of time. Without the inclusion of stock momentum, our model had a tendency to buy as frequently as possible, likely due to the immediate purchase of stocks as soon as an increase occurred. Stock momentum enabled our feature extractor to account for the movement of each stock within the period of time expressed in our state.

## 5 Results

Because of the large state-action space, we needed to implement constraints on our model in order to test. Figure 2 shows the optimal portfolio produced by the model when compared against the S&P 500, following a buy-only policy without the feature extractor. This means that we constrained the possible actions to only buying (not selling or doing nothing) and did not account for the features listed in Section 4. With this model, we were able to beat the S&P 500 in our best portfolio by approximately $3.88\%$ for a total return of $5.63\%$

Once we implemented the feature extractor, our results improved dramatically. However, because of the state-action space, we limited the scope of the time period to 30 days, beginning in January, 2015 (Figure 3). During these 30 days, our algorithm used the heuristics of the feature extractor (standard deviation of return on portfolio, industry type, and momentum of the portfolio) to select the best action each day out of buy, sell, or stay. For this period of time, our model created an optimal portfolio that beat the S&P 500 by approximately $30.07\%$ for a total return of $34.20\%$.

For comparison against our baseline and oracle:

**Baseline** (2015 fiscal year): 1.38%
**Our Model** (January 2015): 34.2%
**Oracle** (2015 fiscal year): 36% (Blackrock's European Hedge Fund)

It is important to note that despite the astounding results, our model simulation ran in exponential time. This is because our state-action space grew exponentially with each day that we evaluated, thus forcing us to constrain the evaluation period to only 30 days. Because of our limited time period, the accuracy of our model cannot be considered as valid as if we ran it for a longer period of time. Consequently, although our returns over a period of one month in 2015 were exceptional, we cannot extrapolate its performance to the time span of one fiscal year. We will discuss this further in our Error Analysis section.

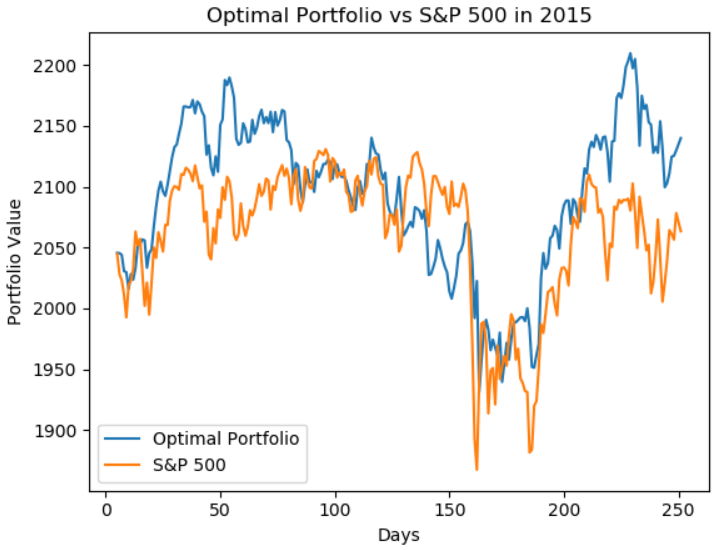


Figure 2: Scores by Industry Type in the S&P 500

## 6 Error Analysis and Discussion

One major challenge we struggled with was the enormous state-action space. Because there were 500 stocks in the S&P 500 and a continuous amount of money could technically be allocated to each stock in a real-world setting, we needed to constrain our model in certain ways. Constraints included limiting the amount we could invest or sell in each asset to a fixed percentage, as well as only exploring 5 new assets on each instance of a 'buy' action.

These constraints prevented us from finding the true optimal policy at each state - we were only able to find the relative best ones given the constraints. They also impacted the accuracy of our model. The random selection of assets made our model somewhat stochastic, at least for the 'buy' action. The feature extractor still had the ability to select selling assets or doing nothing if these actions were favorable to the random assets selected in the 'buy' action, but without exploring every single asset in the S&P 500 for each iteration it would be impossible to know if the assets selected were actually the best possible assets to select at a particular state. Given that our state-action space accumulates in exponential time and space, this would have been impossible.

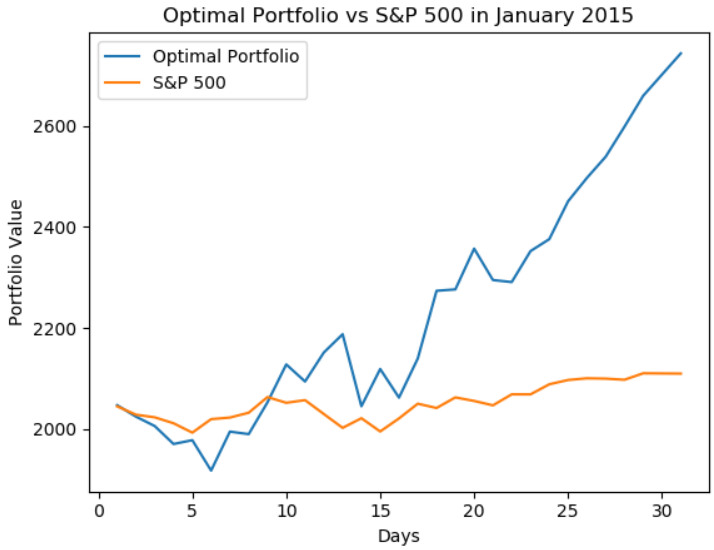


Figure 3: Optimal Portfolio vs. S&P 500 in January 2015

Future work could be done to reduce the state-action space, thus allowing for a longer training time and greater accuracy in the model. First, the state space could be simplified to reduce the amount of exploration and increase overlap between states. This would enable dynamic programming to be used to speed up computation. In addition, work could be done to implement Q-Learning with Function Approximation. This would make it possible to estimate Q-values rather than calculate them explicitly, thereby making run-time reasonable without the added constraints.

Other potentially useful extensions to our project are the inclusion of additional features. One feature in particular, beta ($\beta$), would be a useful indicator of the viability of a portfolio because it could further account for the systematic risk and volatility of a portfolio. Another feature, alpha ($\alpha$), could serve as a measure of the performance of a stock against the market as a whole.

Finally, it is important to note the limitations of predicting a stochastic stock market. In some cases, previous data can be used as an indication of future performance of the market, if the portfolio is diversified adequately. However, the stock market is subject to behavioral economics containing many irrational actors. At any point, a dramatic national or global event could send the stock market surging or plummeting, and no amount of algorithmic computational power or training data would be able to predict such trends.

## 7 Literature Review

Due to the widespread popularity of stock portfolio optimization, notably originating with Markowitz's mean-variance analysis strategy in 1952 [5], a large amount of literature in this field has been produced.

A popular approach to the traditional problem of portfolio optimization (PO) with artificial intelligence is to use Deep Reinforcement Learning. Liang et al. [4] discuss the implementation of various RL algorithms in their paper, including Deep Deterministic Policy Gradient (DDPG), Proximal Policy Optimization (PPO, and Policy Gradient (PG). DDPG is a strategy most similar to what we opted to use in our project; it employs a combination of Q-Learning, policy gradient, and a neural network for function approximation. Due to time constraints, we were unable to implement function approximation.

Our decision to use data from the S&P 500 was inspired in part by Agarwal et al. [1], who also used S&P 500 data with random selections of 50 stocks in each portfolio. By opting to only include 50

stocks in any given portfolio rather than all $500$, they were able to constrain and therefore, simplify the problem. As we struggled with the issue of large state-action spaces, limiting the number of stocks that could be selected in each iteration of our model also proved useful.

One critical component of our methodology was the use of a feature extractor to serve as a heuristic for the financial viability of a portfolio. In their paper, Qiu et al. [7] also incorporate co-variance and variance of portfolio return into their risk function. They do not however include industry type of various stocks. We believed the inclusion of industry type in our feature extractor was an appropriate indicator of a portfolio's risk.

## References

Acknowledgments also to Magdy Saleh for his role as an advisor to this project. Without him, this project would not have been possible.

[1] Agarwal, A., Hazan, E., Kale, S., Schapire, R. E. (n.d.). Algorithms for Portfolio Management based on the Newton Method. Retrieved from http://www.cs.princeton.edu/ ehazan/papers/icml.pdf

[2] Annual SP Sector Performance • Novel Investor. (n.d.). Retrieved from https://novelinvestor.com/sector-performance

[3] Dewey, R. (2019, May 21). Computer Models won't beat the Stock Market anytime soon. Retrieved from https://www.bloomberg.com/news/articles/2019-05-21/computer-models-won-t-beat-the-stock-market-anytime-soon

[4] Liang, Z., Chen, H., Zhu, J., Jiang, K., Li, Y. (2018). Adversarial Deep Reinforcement Learning in Portfolio Management. Retrieved June, 2019, from https://arxiv.org/pdf/1808.09940.pdf.

[5] Markowitz, H. (1952). Portfolio Selection. The Journal of Finance, 7(1), 77. doi:10.2307/2975974

[6] Perry, T., Perry, T. (2019, January 14). I Went To The German Alps and Applied Reinforcement Learning To Financial Portfolio Optimization. Retrieved from https://medium.com/@TalPerry/i-went-to-the-german-alps-and-applied-reinforcement-learning-to-financial-portfolio-optimization-b621c18a69d5

[7] Qiu, H., Liu, H., Ban, F., Caffo, B. (2016). Robust portfolio optimization. Financial Risk Modelling and Portfolio Optimization with R, 161-197. doi:10.1002/9781119119692.ch10

[8] Seo, J. D., Seo, J. D. (2018, December 04). [ Archived Post ] Tutorial: Introduction to Reinforcement Learning with Function Approximation 2. Retrieved from https://medium.com/@SeoJaeDuk/archived-post-tutorial-introduction-to-reinforcement-learning-with-function-approximation-2-72eaf9ab367d