Portfolio Optimization using ML-Generated Inputs for the Markowitz Model and a Reinforcement Learning Approach

Stanford CS229 Final Project

Isaac Aguilar
Department of Mathematics
Stanford University
isaac007@stanford.edu

Nolawi Ayelework
Department of Symbolic Systems
Stanford University
nolaye94@stanford.edu

Cynthia Chen
Department of Computer Science
Stanford University
cchen24@stanford.edu

Abstract

Portfolio optimization is a crucial aspect of financial research which impacts investment strategies and risk management. This study compares Reinforcement Learning and the Markowitz Model for portfolio re-weighting, utilizing predictive analytics obtained through a Ridge Regression model. Analyzing a random sample of S&P 500 stocks reveals that Reinforcement Learning outperforms the Markowitz Model in risk-adjusted returns. These findings underscore the potential of machine learning in enhancing portfolio management and adapting to dynamic market conditions, offering valuable insights for investors and academics alike.

1 Introduction

In the world of finance, investors and hedge fund managers are interested in determining how to optimize their capital allocation across sets of securities to maximize long-term profitability. It becomes natural to ask whether a formulation for the perfect investment strategy exists. In an attempt to answer this question, in the 1950s, economist Harry Markowitz invented modern portfolio theory (MPT), a method that argued it was possible to create the ideal portfolio that gave maximum returns via taking on an optimal risk (1). As part of MPT, he introduced the Markowitz model, a specific formulation of a mean-variance model which seeks to find the optimal set of portfolio weights for a set of assets to achieve a desired risk-return tradeoff. However, shortcomings of the model include its inability to accurately compute the expected returns input and the associated risk.

As a result, we seek to find a way to mitigate these limitations. To be more specific, in this paper, we explore two ways to counter the traditional Markowitz model: the first of which involves optimizing both inputs to the model using machine learning, and one that is independent of the model entirely which employs reinforcement learning, to see whether we can generate an investing strategy that outperforms the original model. We determined a common 5-year time frame to evaluate the final returns of our investment starting on December 28th, 2018 and ending on December 28th, 2023.

The inputs to the Markowitz model are the vector of expected returns of a set of stocks over the 5-year period (calculated historically and the covariance matrix of the daily returns of those stocks over a 50-day period, the output is the weights to invest in over a 5-year period. The input to the reinforcement learning model is the features we have chosen to highlight and train our data on (detailed in Section 4), the output is the monthly weights to invest in over the 5-year period. We will run a simulation to invest monthly according to the weights over the period (5x12 = 60 months total).

2 Related Work

Markowitz Model

Jang and Seong (2023) look to "propose a novel deep reinforcement learning approach that combines the modern portfolio theory and a deep learning approach, (4)" using Tucker decomposition to solve the multimodal problem. We instead use reinforcement learning as a replacement/comparison to the Markowitz model.
O.A. Awoye uses a machine learning method (Graphical Lasso) to generate improved covariance matrices for the Markowitz model compared to the sample covariances commonly used. We also use a regression method, but instead to calculate the expected returns of a model.
Samo et al. (2016) explore finding an optimal set of weights for an investment strategy looking at future returns given the Stochastic Portfolio Theory(SPT). We also look to find an optimal set of weights, but instead we look to use the Markowitz model.

Reinforcement Learning

Filos (2019) tests both model-based and model-free approaches to come to the conclusion that model-free approaches offer the most promise at using reinforcement learning to portfolio management. We used this paper to guide our thinking that a model-free approach would be the best way to create a reinforcement learning algorithm.
Benhamou et al. (2020) found that using Deep Reinforcement Learning for portfolio allocation better predicts portfolio performance than the Markowitz model and other metrics for portfolio allocation. They use a policy gradient approach with a ReLU activation and inputting recent portfolio returns observed over the last two weeks, one month and one year, common asset features observed over the last week, as well as the previous portfolio allocation. We thought this approach was very intuitive specifically and it guided our reinforcement learning model; we also used a policy gradient approach based off of this paper, but we decided to use just chosen asset features and expected returns as our inputs.

3 Dataset and Features

Ridge Regression Model

Our ridge regression model is currently trained on data from the S&P 500, called ALL_STOCKS from December 2013 to December 2018. We cleaned and processed this dataset to only include the stocks that lasted over the entire time frame. Then, we calculated a set of features from those stocks, and trained the model to find a list of weights that would predict the y-values: the adjusted closing price of each stock 5 years in the future. For the training set, this would give predicted adjusted closing prices on December 2018. The test dataset is analogous, with features calculated leading up to and on December 28th, 2018. We then use the weights from training to predict the future prices of the stocks on those features 5 years in the future, which is on December 28th, 2023.

So far, our model takes on 10 features. These 10 features are:

• **Open Price**: Price at the beginning of the market day.

• **Close Price**: Price at the end of the market day.

• **Adjusted Close Price**: The close price is adjusted by different weights to take into consideration any corporate actions that occurred after the close time that could affect the value of the stock, such as stock splits, dividend investments, spin-offs, and mergers.

• **High Price**: The high point of the stock price over the given day.

• **Low Price**: The low point of the stock price over the given day.

• **Volume**: Number of shares of a product sold on a given day.

• **RSI (Relative Strength Index)**: a technical indicator used in financial markets, particularly in stock trading, to assess the magnitude of recent price changes and identify overbought or oversold conditions in an asset.

• **SMA (Simple Moving Average)**: A technical indicator using the average taken from a given set of previous days' adjusted closing prices.

• **EMA (Exponential Moving Average)**: A technical indicator also using moving averages, but compared to a SMA it more highly bases off of more recent data using exponential weighting.

• **Volatility**: The degree of variation or dispersion in the returns of a stock over a certain period of time. It measures the rate at which the price of a stock changes.

We chose the last four features because they are common features that we thought represented information about the stock's recent trends and characteristics, not just the results of that day's trading.

Reinforcement Learning

For our reinforcement learning model, we took the entire set of stocks that we used for our ridge regression model. We then used the same dataset of 10 features listed above for each of those stocks and created a matrix matching those feature values to every stock for a given month. This matrix multiplied by our expected returns, became the input into our reinforcement learning model for every month. The reinforcement model would then tell us how to weight our portfolio to maximize our profits while maintained a preassigned risk threshold of .15.

4 Methods

Our way of obtaining the training and testing datasets is described above in the previous section. We chose regression as our first model as a proof of concept in order to compare the results of the expected returns to the one based on an SMA, and we used ridge to introduce a form of regularization to prevent overfitting.

Input 1: Generating Expected Returns Vectors

After using the ridge regression to predict adjusted closing price for 2023, we created an $expected\_returns$ vector of the 5-year expected returns of our testing stocks which we named $expected\_returns\_ridge$, by taking the percent increase of the predicted 2023 price from the original 2018 price.

We also wanted to compare this ML-generated expected returns vector with a simple moving average (SMA)-generated returns vector. We decided to approach calculating the 5-year return in two ways, which we named Method 1 and Method 2. In the first method, we took the 252-trading-day time period before the 2018 start date and split it into 12 equal-sized periods, where we calculated the SMA return over each period from the first adjusted closing price of that period, and added up all of these returns over that time period.

The second method, Method 2, involved us calculating the SMA return over the 252-day period as a whole but over the average adjusted closing price of the first 5 values of the period.

Both methods were drawn up to reduce variance. Furthermore, both of these returns were then projected to create a 5-year return. In order to calculate which method was more accurate, we calculated the percent error from these projected returns from the actual returns, and annualized them. Method 1 produced the lower annualized error, so we proceeded to compute our $expected\_returns\_sma$ vector with Method 1.

Input 2: Generating Covariance Matrices

We calculated both a "historical" and a "predicted" covariance, called $covariance\_matrix\_historical$ and $covariance\_matrix\_pred$, as inputs for the model. To compute the historical covariance matrix, we took 50 days worth of historical close price data up to the 2018 date and calculated the covariance between each set of stocks based on their daily return rates. To compute the predicted covariance matrix, we had to first obtain a vector of predicted 50-day prices for each stock we were interested in. To do this, we modified our model 50 times to predict the adjusted closing price for each of the 50 days leading up to December 28th, 2023. We then used this vector to calculate the predicted covariances and the predicted covariance matrix.

Markowitz Model

Below is the formulation of the version of the Markowitz model that we used. We are optimizing the following convex function, and we want to maximize the expected portfolio return:

$$\mu_x = \mathbb{E}(R_x) = \mathbf{x}^\mathsf{T}\mathbb{E}(R) = \mathbf{x}^\mathsf{T}\mu$$

We decided to implement two versions of the Markowitz model for comparison: the "Simple" model and the "Complex" model. The "Simple" model uses only one constraint, that the final optimal weights sum to 1. In the "Complex" model, we have both the weights constraint and a risk constraint that the portfolio variance must be less than or equal to a constant we call $gamma\_squared$. The portfolio variance:

$$\sigma_{\mathbf{x}}^2 = Var(R_x) = \sum_i Cov(R_i, R_j)x_i, x_j = \mathbf{x}^\mathsf{T}\Sigma\mathbf{x}$$

For the sake of our project, we chose 0.15 to be the upper bound for our portfolio risk, which would be a realistic level of risk for an investor to set. Below is the formulation:

$$\begin{gathered}\textbf{maximize: } \mu^\mathsf{T}\mathbf{x} \\ \textbf{subject to: } \mathbf{x}^\mathsf{T}\Sigma\mathbf{x} \leq \gamma^2, \\ 1^\mathsf{T}\mathbf{x} = 1\end{gathered}$$

Reinforcement Learning

For the reinforcement learning model, we first created a training matrix with the stocks in a given portfolio and the features that we planned to use to train our model. For our method, we decided that we would use a model-free method, which better matches the environment of the stock market, where the inputs are continuous values. More specifically, we chose to use the policy gradient method, which, as a method that maps states to probability distributions for actions, fit with our desired inputs and outputs. For our model, we used Tensorflow's API for our neural network. Specifically, we used Keras as our neural network API and inputted our gradients into the Adam optimizer.

For every month (or, more specifically, for the last day of every month), we used the data from the previous 50 days as the environment for our agent. Over 200 episodes for a given month, we train the agent to find the best set of weights for that month below a certain level of risk using calculated expected returns, and then we save the set of weights that gives the best rewards for that month's predictions. We did this for every month of the time period, giving us a set of weighting of the stocks that we invest in.

For our reinforcement learning model, we had to create a reward function that punished weights that would go over a given level of risk γ^2 while giving an accurate reflection of what a good set of weights are. Therefore, we decided to set the punishment for going over the risk to be an arbitrarily large negative number regardless of how much the weights went over, to encourage staying as far away as possible, and we set the general reward to be the expected returns for the entire portfolio multiplied by our normalized weights. Our reward equation looked like this, for a given set of weights w, a set of expected returns $X\theta$, a set and a covariance matrix B:

$$\text{If } \frac{w^\mathsf{T}Bw}{||w||_2^2} > \gamma^2, \text{ set reward equal to } -1000.$$

$$\text{Otherwise, set reward equal to } \frac{w^\mathsf{T}X\theta}{||w||_2}$$

5 Experiments / Results / Discussion

Ridge Loss vs SMA Loss
Upon using ALL_STOCKS as the first 50 stocks of the S&P 500 dataframe, we got that the loss of the ridge regression model with lambda = 0 and lambda = 2 respectively was 11162.039302055455 and 10257.803461137175. Alternatively, the loss of the SMA predicted returns using Method 1 was 8848.011336285965, indicating that in this particular case the SMA had done better in predicting the closing prices.

Markowitz Model
From the Markowitz model, we wrote a function to evaluate the actual performance/percent return of the model given that we had invested on the December 2018 date and sold our portfolio on the December 2023 date according to the weights recommended by the model. We did this simply by computing the dot product of the optimal weights obtained from the Markowitz model with the actual

returns of the stocks over the 2018-2023 time period. We were able to get returns for each of the four combinations of inputs, which we have shown below.

Simple Model	Historical Covariance	Predicted Covariance
SMA Returns	727.2166452666834	727.2166452666834
ML Returns	30.089232846481444	30.089232846481444

Complex Model	Historical Covariance	Predicted Covariance
SMA Returns	49.80871109742626	44.741334854290535
ML Returns	49.80890414356125	44.74011310674608

Continuing from the Ridge Loss vs SMA Loss section, we can see that when we choose the new ALL_STOCKS to be ALL_STOCKS[:50], the SMA performs better on the loss as well as generates better returns with the "Simple" Markowitz model. However, when selecting ALL_STOCKS to be a random sample of 100 stocks from the entire S&P 500 dataset, we see that the ridge performs better in the cases where it previously didn't:

Simple Model	Historical Covariance	Predicted Covariance
SMA Returns	36.110383010781725	36.110383010781725
ML Returns	94.11977736818342	94.11977736818342

Complex Model	Historical Covariance	Predicted Covariance
SMA Returns	75.72643924859644	90.32552339737155
ML Returns	75.72674654725056	90.32206702959117

For the Simple Models, we expect the results to be the same when the return method (SMA or ML) is the same, because the simple method does not care about the covariance matrix being used. For the Complex Model, the covariance matters because it determines our risk. The returns being the same are a caveat of the projected returns being similar.

Reinforcement Learning

When using reinforcement learning to find monthly re-weightings of the portfolio, we find that this method predicted a 71.39 percent increase over the entire 5 year span. We do this by summing each re-weighting dotted with its associated monthly returns.

6 Conclusion / Future Work

After running our model on different groupings of stocks, we found that our ridge regression model generally had a lower test loss for final predicted closing price over the SMA method. We also found that the first method for calculating the SMA, of adding up returns over 12 periods, was more accurate than the second. Finally, from calculating Markowitz portfolio returns for different combinations of inputs, we found that for randomly generated stocks the model performs better using the ML model than the SMA projections, but for the two-constraint complex model, the two methods tend to perform equally well. However, our work is highly dependent on the set of securities that are selected for the portfolio, as the return accuracy can perform better on some than for others, due to traits from their historical time-series data. In terms of our reinforcement learning model, we found that our method predicted a 71.39 percent increase, a significant improvement on the 49 percent and 44 percent increase that we got using our ridge regression into our Markowitz model. For future work, we plan on doing more exploration on our ridge regression model by running k-fold cross validation to achieve the best lambda. Next, we could explore more ways to optimize the risk constraint such as through GARCH volatility clustering. We can also consider what it would look like if the Markowitz model outputted monthly weights rather than the weights after a 5-year period. We hypothesize that our returns would not differ by much, from our results from the 5-year period, but we'd be interested to find out either way. In terms of our reinforcement learning algorithm, one possible future work would be to see if increasing the amount of layers or inputs would lead to a better predicting model, without overfitting to our data.

7 Contributions

Each member of the team contributed uniquely to our project. Cynthia wrote the Markowitz optimization functions, and expected returns and covariance matrix calculations. Nolawi wrote the reinforcement learning model. Isaac processed the data from Yahoo Finance's database, wrote the regression features, and helped debug the Markowitz and RL models. Everyone contributed to writing the final report.

References

[1] Investopedia. (n.d.). Modern Portfolio Theory. Retrieved from `https://www.investopedia.com/terms/m/modernportfoliotheory.asp`

[2] Awoye, Oluwatoyin Abimbola. Markowitz minimum variance portfolio optimization using new machine learning methods. Diss. (UCL) University College London, 2016.

[3] Benhamou, Eric, et al. "Deep reinforcement learning (drl) for portfolio allocation." Machine Learning and Knowledge Discovery in Databases. Applied Data Science and Demo Track: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part V. Springer International Publishing, 2021.

[4] Jang, Junkyu, and NohYoon Seong. "Deep reinforcement learning for stock portfolio optimization by connecting with modern portfolio theory." Expert Systems with Applications 218 (2023): 119556.

[5] Samo, Yves-Laurent Kom, and Alexander Vervuurt. "Stochastic portfolio theory: A machine learning perspective." arXiv preprint arXiv:1605.02654