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

Cynthia Chen (cchen24@stanford.edu), Isaac Aguilar (isaac007@stanford.edu), Nolawi Ayelework (nolaye94@stanford.edu)

Stanford
Computer Science

Project Overview

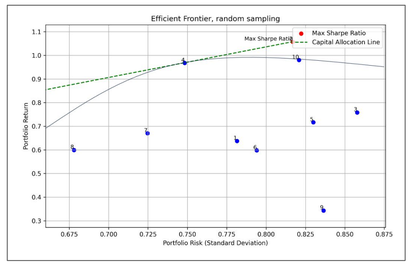
In the world of finance, investors and hedge fund managers are interested in determining how to optimize their capital allocation across sets of securities to maximize long-term profitability. It becomes natural to ask whether a formulation for the perfect investment strategy exists. The Markowitz model is a common formula used to do so. This model is 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 (1), but this model depends on accurately computing the risks and expected returns, both of which are inputs to the model. [1] Our project seeks to counter the Markowitz model in two ways: 1) optimizing both inputs to the model using machine learning (specifically ridge regression), and 2) creating a model independent from the Markowitz model by employing reinforcement learning, to see whether we can generate an investing strategy that outperforms the original model.

Datasets & Metrics

Both of our models are trained on data provided by Yahoo Finance from the set of stocks that were on the S&P 500 from December 2013 to December 2018. We then calculated a set of features from those stocks. 6 of these were given in our data (Open Price, Close Price, Adjusted Close Price, High Price, Low Price, Volume) and the last 4 were calculated based on those previous 6 (Relative Strength Index, Simple Moving Average, Exponential Moving Average and Volatility).

For our RL model, we took the same set of stocks and features 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 every month.



Results

Markowitz Model

From the Markowitz model, we evaluated the performance of our weights 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 in Table 1.

We saw that when we modeled using the first 50 stocks, the SMA performs better on the loss as well as generates better returns with the "Simple" Markowitz model. However, when we selected 100 stocks randomly from the entire S&P 500 dataset, we see that the ridge performs better in the cases where it previously didn't, as seen in Table 2:

We have also visualized the efficient frontier graph, with the Capital Allocation Line and portfolio with the Maximum Sharpe Ratio, labeled, using 10 portfolios: the first via sampling 50 from a set of 100, ten times (left), and the second by splitting all of the S&P 500 stocks into 10 unique portfolios (right).

Reinforcement Learning:

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

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

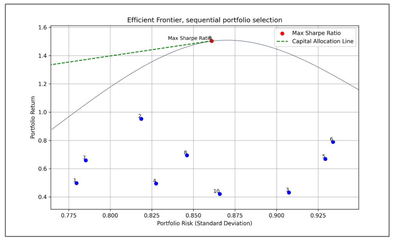
Complex Model	Historical Covariance	Predicted Covariance
SMA Returns	49.80871109742626	44.741334854290535
ML Returns	49.80890414356125	44.74011310674608

Table 1

Simple Model	Historical Covariance	Predicted Covariance
SMA Returns	36.1103830107817 25	36.1103830107817 25
ML Returns	94.11977736818342	94.11977736818342

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

Table 2



Methods & Experiments

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:

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

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 that 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 is computed in the following way:

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

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:

$$\begin{aligned} &\textbf{maximize: } \mu^\top \mathbf{x} \\ &\textbf{subject to: } \mathbf{x}^\top \Sigma \mathbf{x} \leq \gamma^2, \\ &\qquad 1^\top \mathbf{x} = 1 \end{aligned}$$

Ridge Regression

We trained the model to find a list of weights that would predict 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. We then use the weights from training to predict the future prices of the stocks on those features 5 years in the future, (December 28th, 2023.)

We also calculated both a "historical" and a "predicted" covariance 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 first 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 the vector of these results to calculate the predicted covariance and the predicted covariance matrix.

For comparison, we decided to use SMA returns, which is typically used by investors. We calculated SMA by taking 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 the returns over that time period.

Reinforcement Learning

Here, 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 Keras as our neural network API.

For 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 trained 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 (gamma_squared, again), 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 return *Xθ*, a set and a covariance matrix *B*:

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

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

Discussions & Future Research

Discussion:

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, from calculating Markowitz portfolio returns for different combinations of inputs, 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 selected securities, 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.

Future Research:

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. Possible future work for RL would be to see if increasing the amount of layers or inputs would lead to a better predicting model, without overfitting to our data.