**Mean Variance Optimization with Transaction Cost Project**

Yifei (Sarah) Wang, 2017-12-03

This report explains the implementation idea of mean variance optimization with transaction costs. From the ex-ante return and variance, we can use quadratic programming to find the optimal portfolio to minimize risk for given return, or maximize return for given risk. This project only implemented a baseline of portfolio optimization to minimize risk for a portfolio containing 4 assets(AAPL, F, SPT, and JBLU) without any short sale constraints. However, it could be very easily modified to accommodate more complicated strategies, such as market neutral strategy or strategy to maximize risk-adjusted return.

Theoretically the implementation should take expected return and risk as input. For simplicity here, historical monthly return is used as an approximation.

According to Quantitative Equity Portfolio Management by Ludwig B. Chicarini and Daehwan Kim, portfolio optimization with transaction cost is no longer a conventional quadratic programming problem. Instead, it is a highly nonlinear function with high dimensions. This book suggests an approximate solution to this optimal portfolio problem – solve the optimal problem ignoring the transaction cost first to find the optimal portfolio allocation and find which stocks to buy or sell, and then add transaction cost as part of quadratic function. The implementation follows this idea to find optimal portfolio after transaction costs.

1.1 Optimization without Transaction Cost Constraint

To find the portfolio with lowest level of risk, the utility function is as:

x T ∑x

where x is portfolio weights and ∑ is covariance matrix.

The objective then becomes

minimize x T ∑x

subject to µTx ≥µrequired ,

∑all i xi = 1

In the implementation, µrequired  is set as a parameter and the quadratic function is solved using cvxopt. If µrequired is not given, it will iterate through 200 different possible required return and return 200 optimal portfolios, which could construct the efficient frontier. If µrequired is given, it will return only one optimal portfolio.

To double check the accuracy of the efficient frontier, Monte Carlo simulation method is used to randomly generate 1000 portfolios with different weights. Their weights are normalized and sum to 1, and return and standard deviation are calculated for every portfolio. All portfolios perfectly sit within the efficient frontier.

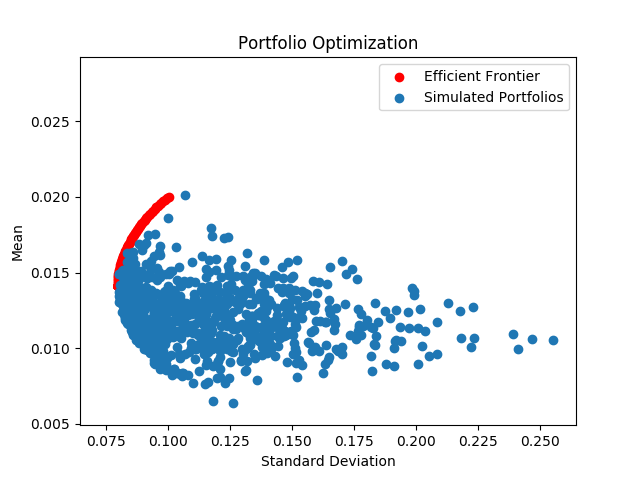


Chart1: Portfolio Efficient Frontier and Random Portfolios from Monte Carlo Simulation

1.2 Optimization with Transaction Cost Constraint

Transaction cost depends both on stock liquidity and trade size. Stocks with lower liquidity end up with larger bid ask spread, and large trade size increases market impact. Liquidity is modeled using average trade volume in the market. Assume transaction cost per share is an inverse function of each stock’s liquidity, then we have

ti = c/average trading volume of stock I

where c is a constant

And because transaction cost is a linear function on trade size, the total transaction cost to rebalance the portfolio will be

Total transaction cost = ∑all i ti (|ntargeti-ncurrenti|)

where ntarget is number of shares of stock i in target portfolio, and ncurrenti is the number of shares of stock i in current portfolio.

From 1.1 we get 200 optimal portfolios with different risk-return profile. Suppose our current portfolio are equally weighted among the 4 securities, then for each optimal portfolio, we can easily calculate how much the current portfolio is deviated from the optimal portfolio and therefore the total transaction cost for rebalancing.

With transactions, the objective function then is updated to be:

minimize x T ∑x + TT x

subject to µTx ≥µrequired ,

∑all i xi = 1

where T is transaction cost.

Compared with the efficient frontier in 1.1, the new efficient frontier is shifted to the right. This is expected because by adding an additional constraint to the optimization, the lowest risk portfolio we can create will have a higher risk than the lowest risk portfolio we were able to create without transaction cost constraint. One of the 200 optimal portfolio with transaction cost sits on the efficient frontier because for that given return, the optimal portfolio without transaction cost is very close to our current portfolio. Therefore transaction cost has very little impact.

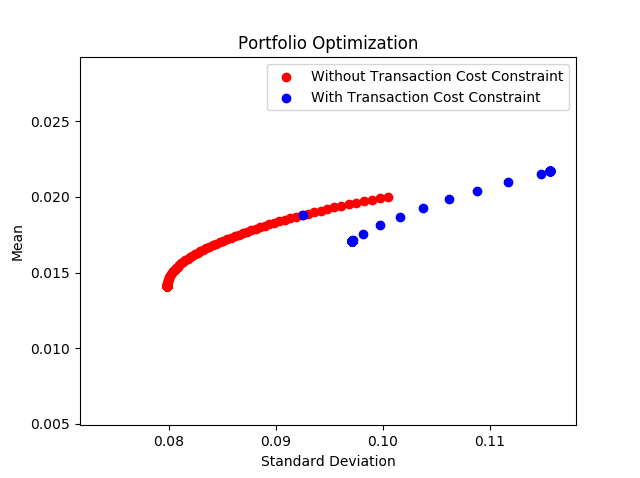


Chart 2: Portfolio Efficient Frontier with and without Transaction Costs

The efficient frontier contains all the optimal portfolios for different risk profile. Theoretically for investment, we can choose minimum variance portfolio, or the portfolio with highest Sharpe ratio, or any portfolio sitting on the efficient frontier. But I believe in practice, which portfolio to invest in highly depends on clients’ investment objective and risk tolerance. On the efficient frontier, portfolio allocation and risk is a one-to-one mapping relation, meaning that for given risk level, there is only one optimal portfolio allocation. After we have a thorough understanding of clients’ risk tolerance, we can easily determine the optimal portfolio allocation.

1.3 Files

There are two files in total. One is optimization.yml, which defines all the external dependencies, and the other one is optimization.py, which contains all the implementation.

Before running optimization.py, I suggest creating a virtual environment by typing “conda env update –f optimization.yml” and “activate optimization” in command line. And then go to python interpreter and select python.exe under optimization environment.

Please note that sometimes due to internet connection, you may get the error saying” No data fetched using 'YahooDailyReader'”. Please give it a shot for the second time, and usually the error message will be automatically fixed with good internet connection.

1.4 Possible Improvement in the Future

Since this project is a very simplified version of portfolio optimization, I believe in practice, there might be more constraints, such as diversity constraints, short sale constrains, etc. Currently the program is designed flexibly enough to accommodate more complex strategies and constraints. If the project gets large enough, we can adopt more object-oriented features such as design patterns, to improve readability and maintainability.