PORTFOLIO OPTIMIZATION BASED ON HARRY MARKOWITZ'S MODERN PORTFOLIO THEORY

by

PRIYANSHI GARG

PROJECT REPORT

CS 6301.012 - Special Topics in Computer Science

(Optimization in Machine Learning)

MASTERS IN

COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT DALLAS

May 2, 2020

PORTFOLIO OPTIMIZATION BASED ON HARRY MARKOWITZ'S MODERN PORTFOLIO THEORY

Priyanshi Garg, MS
The University of Texas at Dallas,

Course Instructor: Dr. Rishabh Iyer

A good investment requires maximizing the return and minimizing the risk and knowing the best risk-teturn trade off. A portfolio is a grouping of financial assets which includes stocks, bonds and any other tradable securities [1]. For low risk high return trading strategy investors need to find out what percentage of each asset is to be belonged for a particular period of time. This question led economist Harry Markowitz to propose Modern Portfolio Theory (MPT) which earn him the Economic Sciences Nobel Prize in 1990 [3]. MPT is all about assembling a portfolio that maximizes expected return for a set level of risk [7]. William Sharpe proposed a method based on his Sharpe ratio to select the optimum risk return trade off. His work on Capital asset pricing model earned him 1990 Nobel Prize in Economic Sciences [5]. Moreover, our proposed regularization method will be used for asset selection problem. Moreover, we also aim to build stacking based ensemble model to get an optimum time period of investment. The contribution to this work is to find out return-diversity trade off and best time period selection for portfolio optimization.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

BACKGROUND MATHEMATICS OF PORTFOLIO OPTIMIZATION

1.1 Portfolio Return and Risk

Suppose we are interested to invest in n assets $\alpha_1, \alpha_2, \ldots, \alpha_n$. Let, the total time is partitioned into $\mathbb{T}$ time periods. Let us consider a particular time period $t \in \{1, 2, \ldots, \mathbb{T}\}$. Let us assume, at the beginning of the time period t, we want to buy $(q_t)_i$ shares of α_i with unit price \$ $\left(p_t^-\right)_i$ where $i \in \{1, 2, \ldots, n\}$. Let us denote the dollar value holdings of the asset α_i at time period t by,

$$(h_t)_i := \left(p_t^-\right)_i (q_t)_i$$

At time t we we denote the dollar value holdings of all the assets by the following n-vector,

$$h_t = \left[(h_t)_1, (h_t)_2, \ldots, (h_t)_n\right]^T$$

Hence, at time t the total dollar value holdings of the assets or *portfolio value* is,

$$V_t = \sum_{i=1}^{n} (h_t)_i = \mathbf{1}^T h_t$$

Thus, *portfolio weights* or allocation (fraction of total portfolio value) at time period t is,

$$w_t = \left(\frac{1}{V_t}\right) \left[(h_t)_1, (h_t)_2, \ldots, (h_t)_n\right]^T = \left(\frac{1}{V_t}\right) h_t \Rightarrow \mathbf{1}^T w_t = 1$$

Now, at the end of the time period t the new share price of the asset α_i is \$ $\left(p_t^+\right)_i$. Hence, the *profit* on asset α_i for the time period t is,

$$\left[\left(p_t^+\right)_i - \left(p_t^-\right)_i\right] (q_t)_i = \left(\frac{\left(p_t^+\right)_i - \left(p_t^-\right)_i}{\left(p_t^-\right)_i}\right) (h_t)_i := (\widetilde{r}_t)_i (h_t)_i$$

where, we define $(\widetilde{r}_t)_i$ to be the *asset return* in period t for the asset α_i as,

$$(\widetilde{r}_t)_i = \left(\frac{\left(p_t^+\right)_i - \left(p_t^-\right)_i}{\left(p_t^-\right)_i}\right)$$

Hence, at time t we we denote the *asset returns* of all the assets by the following n-vector,

$$\widetilde{r}_t = \left[(\widetilde{r}_t)_1, (\widetilde{r}_t)_2, \ldots, (\widetilde{r}_t)_n\right]^T$$

Hence, dollar profit (increase in value) over period t is,

$$\sum_{i=1}^{n} (\widetilde{r}_t)_i (h_t)_i = \widetilde{r}_t^T h_t = \widetilde{r}_t^T (V_t w_t) = V_t \widetilde{r}_t^T w_t$$

$V_t \widetilde{r}_t^T w_t$ is the increase in portfolio value at time t with an initial portfolio value of $\$V_t$. Hence, dollar profit per unit of portfolio value is $r_t := \widetilde{r}_t^T w_t$ at time t. Thus we denote the portfolio returns over time period $t = 1, 2, \ldots, \mathbb{T}$ by a $\mathbb{T}$-vector r,

$$r = \left[r_1, r_2, \ldots, r_{\mathbb{T}}\right]^T$$

And finally, we define the following,

$$\begin{aligned} \text{portfolio return} &= \mathbb{T} avg(r), \text{ over time period } t = 1, 2, \ldots, \mathbb{T} \\ \text{portfolio risk} &= \sqrt{\mathbb{T}} std(r), \text{ over time period } t = 1, 2, \ldots, \mathbb{T} \end{aligned}$$

We have a detailed analysis of data collection and exploratory analysis in Chapter 3 and 4. After we have the data we store it in a dataframe called `table`. We will now create the `python` variables explained in this section for coding,

1. `table:` This is the table of historical stock prize data of size $\mathbb{T} \times n = 1841 \times 65$ with index as date. $\left(p_t^-\right)_i$ indicates the closing price of the stock on t-th day of stock i.

$$\texttt{table}_{\mathbb{T}\times n} = \begin{pmatrix} \left(p_1^-\right)_1 & \left(p_1^-\right)_2 & \cdots & \left(p_1^-\right)_n \\ \left(p_2^-\right)_1 & \left(p_2^-\right)_2 & \cdots & \left(p_2^-\right)_n \\ \vdots & \vdots & \ddots & \vdots \\ \left(p_{\mathbb{T}}^-\right)_1 & \left(p_{\mathbb{T}}^-\right)_2 & \cdots & \left(p_{\mathbb{T}}^-\right)_n \end{pmatrix}_{\mathbb{T}\times n = 1841\times 65}$$

   And we assume,

$$\left(p_{t-1}^+\right)_i = \left(p_t^-\right)_i$$

2. `weights:`

$$\texttt{weights}_t = [(w_t)_1\,, (w_t)_2\,, \ldots, (w_t)_n] = w_t = \left(\frac{1}{V_t}\right) [(h_t)_1\,, (h_t)_2\,, \ldots, (h_t)_n]^T$$

This is a $n \times 1 = 65 \times 1$ matrix. We generate random numbers $(w_t)_1\,, (w_t)_2\,, \ldots, (w_t)_n$ for our $n = 65$ assets (stocks) such that

$$\mathbf{1}^T w_t = 1$$

3. `returns:` This is of the size `table` (one row less 1840×65) containing the daily percentage change data. We need this as different stock vary a lot in their prices. The following is the matrix representing the matrix,

$$\texttt{returns}_{\mathbb{T}\times n} = \begin{pmatrix} (\widetilde{r}_1)_1 & (\widetilde{r}_1)_2 & \cdots & (\widetilde{r}_1)_n \\ (\widetilde{r}_2)_1 & (\widetilde{r}_2)_2 & \cdots & (\widetilde{r}_2)_n \\ \vdots & \vdots & \ddots & \vdots \\ (\widetilde{r}_{\mathbb{T}})_1 & (\widetilde{r}_{\mathbb{T}})_2 & \cdots & (\widetilde{r}_{\mathbb{T}})_n \end{pmatrix}$$

Observe,

$$(\widetilde{r}_{\mathbb{T}})_i = \left(\frac{(p^+_{\mathbb{T}})_i - (p^-_{\mathbb{T}})_i}{(p^-_{\mathbb{T}})_i}\right) = \left(\frac{(p^-_{\mathbb{T}+1})_i - (p^-_{\mathbb{T}})_i}{(p^-_1)_i}\right) = \texttt{NA}$$

4. `mean_returns:` We calculate the mean of `returns` over 1840 data for each of the 65 companies generating a 65×1 matrix. But for ease of symbols we will consider the number of time periods is still $\mathbb{T}$. This will introduce a `NA` row in `returns` matrix. But that will not create any calculation problem.

$$\texttt{mean\_returns} = \left[\frac{1}{\mathbb{T}}\sum_{j=1}^{\mathbb{T}} (\widetilde{r}_j)_1\,,\ \frac{1}{\mathbb{T}}\sum_{j=1}^{\mathbb{T}} (\widetilde{r}_j)_2\,,\ \ldots,\ \frac{1}{\mathbb{T}}\sum_{j=1}^{\mathbb{T}} (\widetilde{r}_j)_n\right]$$

5. `cov_matrix:` This is 65×65 covariance matrix of `returns` table calculated as `cov_matrix = returns.cov()`. Let σ_i^2 is the variance of i-th asset is the variance of the vector

$$X_i := \left[(p^-_1)_i\,,\ (p^-_2)_i\,,\ \ldots,\ (p^-_{\mathbb{T}})_i\right]^T$$

The covariance of i-th and k-th stock is,

$$\sigma_{i,k}^2 = \operatorname{cov}[X_i, X_k] = \mathbb{E}[\,(X_i - \mathbb{E}[X_i])(X_k - \mathbb{E}[X_k])\,]$$

Hence, the covariance matrix is,

$$\texttt{cov\_matrix}_{n\times n} = \begin{pmatrix} \sigma_1^2 & \sigma_{1,2}^2 & \cdots & \sigma_{1,n}^2 \\ \sigma_{2,1}^2 & \sigma_2^2 & \cdots & \sigma_{2,n}^2 \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n,1}^2 & \sigma_{n,2}^2 & \cdots & \sigma_n \end{pmatrix}$$

6. portfolio_std_dev: As we have obtained the formula,

$$\texttt{portfolio\_return} = 252avg(r), \text{ over time period } t = 1, 2, \ldots, \mathbb{T}$$

$$\texttt{portfolio\_std\_dev} = \sqrt{252} \times std(r), \text{ over time period } t = 1, 2, \ldots, \mathbb{T}$$

It can be proven that

$$var(r) = \texttt{weights}^T \times \texttt{cov\_matrix} \times \texttt{weights}$$

Thus we have the following formula,

$$\begin{aligned} \texttt{portfolio\_std\_dev} &= \sqrt{252} \times std(r) \\ &= \sqrt{252} \times \sqrt{var(r)} \\ &= \sqrt{252} \times \sqrt{\texttt{weights}^T \times \texttt{cov\_matrix} \times \texttt{weights}} \end{aligned}$$

1.1.1 Risk Minimization

Let us define $\mathbb{T} \times n$ asset returns matrix R with rows $\widetilde{r}_t^T$,

$$R_{\mathbb{T}\times n} = \texttt{returns}_{\mathbb{T}\times n} = \begin{pmatrix} (\widetilde{r}_1)_1 & (\widetilde{r}_1)_2 & \cdots & (\widetilde{r}_1)_n \\ (\widetilde{r}_2)_1 & (\widetilde{r}_2)_2 & \cdots & (\widetilde{r}_2)_n \\ \vdots & \vdots & \ddots & \vdots \\ (\widetilde{r}_{\mathbb{T}})_1 & (\widetilde{r}_{\mathbb{T}})_2 & \cdots & (\widetilde{r}_{\mathbb{T}})_n \end{pmatrix}$$

Suppose the portfolio weight doesn't change over the period of time $t = 1, 2, \dots, \mathbb{T}$, i.e.

$$w_1 = w_2 = \cdots = w_T := w$$

Then, we get,

$$r = Rw$$

Now, let us consider the following,

1. We want to know for what portfolio weight vector w, past risk would be minimum for a given past return. And we will use the same weight vector for future investment. The big assumption here is,

   ```
   future returns will look something like past ones.
   ```

2. We know the past realized asset returns but not future ones. That means we know the asset returns matrix for past returns, denoted by R or in our code `returns`. And so, Rw is the (past) portfolio return time series. We also know, the mean past return $\rho := avg(r) = avg(Rw)$. Observe that,

   $$\texttt{portfolio\_return} = 252 \times avg(r) = 252 \times avg(Rw) = 252 \times \rho,\ \mathbf{1}^T w = 1$$

3. Thus to get the same annualized return ρ with minimum portfolio risk, we need to solve the following optimization problem,

   $$\min_{w} (\texttt{portfolio\_std\_dev})^2 = \left(\sqrt{252} \times std(r)\right)^2 = \frac{252}{\mathbb{T}} ||Rw - \rho\mathbf{1}||^2$$
   $$\text{subject to, } avg(Rw) = \rho = \frac{\texttt{portfolio\_return}}{252},\ \mathbf{1}^T w = 1$$

   which means, we minimize risk for specified value of return. This is equivalent to the following optimization problem,

   $$\min_{weights} \left\| \texttt{returns} \times weights - \frac{\texttt{portfolio\_return}}{252} \right\|^2$$
   $$\text{subject to, } \mathbf{1}^T weights = 1$$

and the optimum portfolio in that case is,

$$w_{optimum} = arg\left(\min_{\texttt{weights}}\left\|\texttt{returns}\times\texttt{weights} - \frac{\texttt{portfolio\_return}}{252}\right\|^2\right)$$

subject to, $\mathbf{1}^T\texttt{weights} = 1$, $\texttt{weights}$ vector has all positive component $\in [0,1]$

We will first create a lot of weights vector and calculate portfolio return and volatility pair and plot them. This has been demonstrated in Chapter 5.1.

1.1.2 Efficient Frontier

The efficient frontier is the set of optimal portfolios that offer the highest expected return for a defined level of risk or the lowest risk for a given level of expected return. This has been demonstrated in Chapter 5.3. Portfolios that lie below the efficient frontier are sub-optimal because they do not provide enough return for the level of risk. Portfolios that cluster to the right of the efficient frontier are sub-optimal because they have a higher level of risk for the defined rate of return [2].

1.1.3 Sharpe Ratio maximization

We have seen in previous section that the optimization problem gives us the solution as Efficient Frontier curve. The next question is of course how much increase of the return will be if someone wants to take a bit more risk. This was explained by William F. Sharpe, in 1966.

Sharpe Ratio describes how much excess return you receive for the extra risk. If the return is ρ and the best available rate of return is x. Then we can calculate the Sharpe ratio from efficient frontier line. Suppose for the portfolio return $r_\rho = avg(Rw_\rho)$ the minimum risk is $\sigma_\rho = std(r_\rho)$. Then for weight vector w_p,

$$\text{Sharpe ratio } S_\rho = \frac{r_\rho - x}{\sigma_\rho}$$

To get the optimum portfolio we want to now solve the following optimization problem,

$$\max_\rho S_\rho$$

In other words solving the following minimization problem,

$$\min_{\rho} \left[- \left(\frac{r_\rho - x}{\sigma_\rho} \right) \right]$$

For diffrent random weights Sharpe ratio was calculated and plotted in Chapter 5.1.

1.1.4 Summary

We have two optimization problem. First we want to minimize the risk for a particular desired return ρ. That gives us the portfolio w_p

$$w_\rho = arg \left(\min_{\texttt{weights}} \left\| \texttt{returns} \times \texttt{weights} - \frac{\texttt{portfolio\_return}}{252} \right\|^2 \right)$$

subject to, $\mathbf{1}^T \texttt{weights} = 1$, $\texttt{weights}$ vector has all positive component $\in [0, 1]$

Next we want to change the $\texttt{portfolio\_return} = \rho$ and get different w_ρ values and risk $std(Rw_\rho)$ versus return $avg(Rw_\rho)$ plot (called efficient frontier). Finally we want to calculate

$$\arg\min_{\rho} \left[- \left(\frac{r_\rho - x}{\sigma_\rho} \right) \right] = \arg\min_{\rho} \left[- \left(\frac{avg(Rw_\rho) - x}{std(Rw_\rho)} \right) \right]$$

1.2 Project Plan

Finding out the optimum portfolio weights based of multiple stocks using Modern Portfolio Theory of Markowits.

Data Source: The source of our training, testing and validation data will be obtained from Yahoo Finance.

CHAPTER 2

OPTIMIZATION TECHNIQUE

2.1 Portfolio Optimization Problem

Let us recall the optimization problem we have is,

$$w_\rho = arg\left(\min_{\texttt{weights}}\left\|\texttt{returns}\times\texttt{weights}-\frac{\texttt{portfolio\_return}}{252}\right\|^2\right)$$

subject to, $\mathbf{1}^T\texttt{weights}=1,\ \texttt{weights}$ vector has all positive component $\in[0,1]$

Or equivalently,

$$\min_{\texttt{weights}}(\texttt{portfolio\_std\_dev})^2=\texttt{weights}^T\times\texttt{cov\_matrix}\times\texttt{weights}$$

subject to, $avg(R\times\texttt{weights})=\rho=\dfrac{\texttt{portfolio\_return}}{252},\ \mathbf{1}^T\texttt{weights}=1$

Or

$$\min_x\frac{1}{2}x^TGx+x^Tc$$

subject to, $a_1^Tx=b_1,\ a_2^Tx=b_2,\ a_i^Tx\ge b_i,\ i\in\mathcal{I}=\{3,4,\dots,n+2\}$

where, $G=2\times\texttt{cov\_matrix}$, $a_1=\frac{1}{\mathbb{T}}R^T\mathbf{1}_{\mathbb{T}\times1}$, $b_1=\rho$, $a_2=\mathbf{1}_{n\times1}$ and $b_2=1$. The variable $x:=\texttt{weights}$. a_i is a vector with all elements zero except $i-2$-th element for all $i\in\mathcal{I}$ and $b_i=0$ for all $i\in\mathcal{I}$. Moreover, $c=0_{n\times1}$ and $\mathcal{E}=\{1,2\}$. Then the problem becomes,

$$\min_x f(x)=\frac{1}{2}x^TGx+x^Tc$$

$$\text{subject to } a_i^Tx=b_i\quad i\in\mathcal{E}\tag{16.1b}$$

$$a_i^Tx\ge b_i\quad i\in\mathcal{I}$$

Statement 1. *G is positive semi-definite matrix because covariance matrix is always symmetric positive semi-definite.*

Statement 2. . *The objective function f, defined as $f(x) = \frac{1}{2}x^T Gx + x^T c$ is a convex function for positive semi-definite $G \succeq 0$.*

Lemma 1. *In Portfolio Optimization problem a local minimizer of the objective function f is also a global minimizer.*

Proof. Using Statements 1 and 2 □

2.2 Karush–Kuhn–Tucker Conditions

Let us consider the following problem,

$$\min_{x \in \mathbb{R}^n} f(x) \quad \text{subject to} \quad \begin{cases} c_i(x) = 0, & i \in \mathcal{E} \\ c_i(x) \geq 0, & i \in \mathcal{I} \end{cases} \tag{12.1}$$

where f and the functions c_i are all smooth, real-valued functions on a subset of $\mathbb{R}^n$, and $\mathcal{I}$ and $\mathcal{E}$ are two finite sets of indices.

Theorem 1 (Linear Independence Constraint Qualification (LICQ)). *Given the point x and the active set $\mathcal{A}(x)$ defined as the set of constraints with x at their boundary,*

$$\mathcal{A}(x) = \mathcal{E} \cup \{i \in \mathcal{I} : c_i(x) = 0\}$$

We say that the linear independence constraint qualification (LICQ) holds at x if the set of active constraint gradients $\{\nabla c_i(x), i \in \mathcal{A}(x)\}$ is linearly independent.

Theorem 2 (First-Order Necessary Conditions (Karush–Kuhn–Tucker conditions)). *Suppose that x^* is a local solution of (12.1) that the functions f and c_i in (12.1) are continuously differentiable, and that the LICQ holds at x^*. Let us consider the Lagrangian function,*

$$\mathcal{L}(x, \lambda) = f(x) - \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i c_i(x) \tag{12.33}$$

Then there is a Lagrange multiplier vector λ^*, with components λ_i^*, where $i \in \mathcal{E} \cup \mathcal{I}$, such that the following conditions are satisfied at (x^*, λ^*),

$$\nabla_x \mathcal{L}(x^*, \lambda^*) = 0 \tag{12.34a}$$

$$c_i(x^*) = 0 \quad \forall i \in \mathcal{E} \tag{12.34b}$$

$$c_i(x^*) \geq 0 \quad \forall i \in \mathcal{I} \tag{12.34c}$$

$$\lambda_i^* \geq 0 \quad \forall i \in \mathcal{I} \tag{12.34d}$$

$$\lambda_i^* c_i(x^*) = 0 \quad \forall i \in \mathcal{E} \cup \mathcal{I} \tag{12.34e}$$

The conditions (12.34) are often known as the *Karush-Kuhn-Tucker* conditions, or KKT conditions for short. The conditions (12.34e) are *complementarity conditions* ; they imply that either constraint i is active or $\lambda_i^* = 0$, or possibly both. In particular, the Lagrange multipliers corresponding to inactive inequality constraints are zero, we can omit the terms $i \notin \mathcal{A}(x^*)$ for indices from (12.34a) and rewrite this condition as

$$0 = \nabla_x \mathcal{L}(x^*, \lambda^*) = \nabla_x f(x)\big|_{x^*} - \sum_{i \in \mathcal{A}(x^*)} \lambda_i \nabla_x c_i(x)\big|_{(x^*, \lambda^*)}$$

Let us define the the set of linearized feasible directions $\mathcal{F}(x)$,

$$\mathcal{F}(x) = \{d \mid d^T \nabla c_i(x) = 0,\ \forall i \in \mathcal{E} \text{ and } d^T \nabla c_i(x) \geq 0,\ \forall i \in \mathcal{A} \cap \mathcal{I}\}$$

$$\mathcal{C}(x^*, \lambda^*) = \{y \in \mathcal{F}(x^*) \mid \nabla c_i(x^*)^T y = 0,\ \forall i \in \mathcal{A}(x^*) \cap \mathcal{I}\ with\ \lambda_i^* > 0\}$$

Theorem 3 (Second-Order Sufficient Conditions)**.** *Suppose that for some feasible point $x^* \in \mathbb{R}^n$ there is a Lagrange multiplier vector λ^* such that the KKT conditions (12.34) are satisfied. Suppose also that*

$$y^T \nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*) y > 0 \qquad \forall y \in \mathcal{C}(x^*, \lambda^*), y \neq 0$$

Then x^* is a strict local solution for (12.1).

Corollary 1. *Portfolio Optimization Problem satisfying Theorem 3 is a strict global solution.*

Proof. Using Lemma 1 and Theorem 3. □

2.3 General Quadratic Program

The general quadratic program (QP) can be stated as

$$\min_x q(x) = \frac{1}{2}x^T Gx + x^T c \tag{16.1a}$$

$$\text{subject to } a_i^T x = b_i \quad i \in \mathcal{E} \tag{16.1b}$$

$$a_i^T x \geq b_i \quad i \in \mathcal{I} \tag{16.1c}$$

By specializing the KKT conditions to this problem, we find that any solution x^* of (16.1) satisfies the following first-order conditions, for some Lagrange multipliers $\lambda_i^*, i \in \mathcal{A}(x^*)$:

$$Gx^* + c - \sum_{i \in \mathcal{A}(x^*)} \lambda_i^* a_i = 0 \tag{13.37a}$$

$$a_i^T x^* = b_i, \forall i \in \mathcal{A}(x^*) \tag{13.37b}$$

$$a_i^T x^* \geq b_i, \forall i \in \mathcal{I} \backslash \mathcal{A}(x^*) \tag{13.37c}$$

$$\lambda_i^* \geq 0, \forall i \in \mathcal{I} \cap \mathcal{A}(x^*) \tag{13.37d}$$

If we are lucky, and we have only equality conditions then the solution is much easier as it boils down to solving a system of linear equations. But as our constraints are not equality we need the following sufficient condition for the solution,

Theorem 4. *If x^* satisfies the conditions (13.37) for some $\lambda^* i$, $i \in \mathcal{A}(x^*)$, and G is positive semidefinite, then x^* is a global solution of (16.1).*

In our portfolio optimization problem as G is a positive semidefinite matrix we can conclude that any point satisfying the KKT condition will be the global solution.[1]

[1] Section 2.2 and section 2.3are explained in detail over here [6].

2.4 Sequential Quadratic Programming

`Sequential Quadratic Programming` is an optimizer developed by Dieter Kraft in 1988 for the solution of constrained non linear optimization problem [4]. This is a sequential least squares programming algorithm which uses the Han–Powell quasi–Newton method with a BFGS update of the B–matrix and an L1–test function in the step–length algorithm.

2.4.1 Basics Of Algorithm

Consider a nonlinear programming problem of the form:

$$\begin{aligned} \min_x f(x) \\ \text{subject to } b(x) \geq 0 \\ c(x) = 0 \end{aligned}$$

The Lagrangian for this problem is,

$$\mathcal{L}(x, \lambda, \sigma) = f(x) - \lambda b(x) - \sigma c(x),$$

where λ and σ are Lagrange multipliers. At an iterate x_k, a basic sequential quadratic programming algorithm defines an appropriate search direction d_k as a solution to the quadratic programming subproblem.

$$\begin{aligned} \min_d f(x_k) + \nabla f(x_k)^T d + \tfrac{1}{2} d^T \nabla^2_{xx} \mathcal{L}(x_k, \lambda_k, \sigma_k) d \\ \text{subject to } b(x_k) + \nabla b(x_k)^T d \geq 0 \\ c(x_k) + \nabla c(x_k)^T d = 0. \end{aligned}$$

Note that the term $f(x_k)$ in the expression above may be left out for the minimization problem, since it is constant under the $\underset{d}{min}$ operator.

2.4.2 A Brief study of Algorithm

Sequential Quadratic Algorithm solves the non linear problem(NLP) iteratively.

$$x^{k+1} := x^k + \alpha^k d^k$$

To get the search direction d^k it changes a non linear problem(NLP) to a quadratic problem by doing quadratic approximation of dual of NLP(using Taylor Series Expansion) and linear approximation of the constraints. Doing this the problem which is already quadratic is not affected proof of which is given here [4]. The step size is chosen like in Newton's method of solving system of nonlinear equations with $\alpha = 1$ if nonlinear functions are near the local optimum, if not it uses the method proposed by Han with a penalty term proposed by Powell which guarantees global convergence of the method. An additional variable δ is introduced into the quadratic problem to keep the constraints consistent through the course of iterations.

$$\min_{d} f(x_k) + \nabla f(x_k)^T d + \tfrac{1}{2} d^T \nabla^2_{xx} \mathcal{L}(x_k, \lambda_k, \sigma_k) d + \tfrac{1}{2}\rho^k (\delta^k)^2$$

subject to $\delta^k b(x_k) + \nabla b(x_k)^T d \geq 0$

$$\delta^k c(x_k) + \nabla c(x_k)^T d = 0$$

$$0 \leq \delta^k \leq 1,$$

where δ^k has the value

$$\delta^k = \begin{cases} 1, & \text{if } c(x_k) > 0 \\ \sigma^k, & \text{otherwise} \end{cases}$$

To update the Hessian matrix of Lagrange function in each iteration an approximation method is used as used by quasi-Newton Methods and BFGS-formula for unconstrained optimization. If only equality constraints are there then it is equivalent to applying Newton's method to KKT conditions of the problem, described in section 2.3. To use first order information to approximate Hessian matrix for constrained optimization a method proposed by `Powell` is used as Hessian matrix need not be anymore positive definite. Kraft has proposed an algorithm to implement `composite-t methods` of Fletcher and Powell which guarantees the updated Hessian-matrix of Lagrange to be positive definite. As described in theorem 4 the solution x^* will be unique(global) solution of Quadratic programming. This algorithm instead of finding the local minimum from the interior of the feasible domain rather find it from its exterior. The solutions x lying on linearly independent set of active constraints are called S-pair, doing this it satisfies theorem 2.2.

The algorithm starts by initializing x to be the unconstrained minimum of quadratic problem. If x satisfies the condition mentioned in Theorem 4 then it stops, since the solution is feasible and optimal, else determines the step direction and computes the step length making the constraints feasible. Determines the new S-pair and takes the step, if conditions are satisfies it stops else it repeats. Algorithm finishes here.

CHAPTER 3

DATA COLLECTION

3.1 Dataset 1: Stock List Collection

`Finviz` is one of our most favorite site to get the free stock data. The link for `Finviz` that we used in this project is, `https://finviz.com/screener.ashx?v=152&o=ticker&c=`.
We are interested to create a dataset using web scrapping on `Finviz` for which we used `BeautifulSoup` package in `python`. `BeautifulSoup` is a quite a strong package that can fetch website data from server end, clean up, parse the web documents and give a tagged HTML file from where we can run easy queries using built in functionalities `BeautifulSoup` [8].
This dataset contains *71* attributes(features) and 7673 records (companies). These are a lot of companies, we want to short list the companies, our strategy is to work with the top most richest companies, based on their `Market Capital` as `'Market Cap'` in the dataset.

3.2 Dataset 2: Short List **`finviz`** Dataset based on Market Capital

We created a new column in our dataset called `MarketCap`, where we did the conversion of Billions to Millions for the data present in `'Market Cap'`, then removed the companies whose `Market capital` was not retrieved. Since, We want to work only with the topmost portion of the companies, we set this threshold to 20%, which leaves us with 1059 companies out of 7673 companies.

3.3 Dataset 3: Get consistently profitable companies those just got a downfall

Out of this 1059 companies, we shortlist those companies who are consistently profitable (in number of business days) but had a recent downfall, assuming they will turn back (*It is our opinion*). Now to get the stock history of these companies we used `Yahoo Finance` (One can use Google finance as well). The link for it is, `https://finance.yahoo.com/lookup/`. To scrape data from here we used Python 3 package called `yahoo_fin` [10].

We are calling those companies profitable, firstly, whose *correlation coefficient* $\% > CFCutoff$ on the *daily closing data* of *number Of business days*. *CFCutoff* can be $60, 70, \ldots etc$, it is just a variable. We take $CFCutoff = 70,$ number of business days $= 300$, *correlation coefficient %* is calculated using `Linear Regression` and secondly, who had a recent downfall, steepness of which is calculated using `BullishProbability` as:

$$\frac{\text{Maximum stock price of the company} - \text{current stock price of the company}}{\text{Maximum stock price of the company} - \text{Minimum stock price of the company}}$$

If current stock price equals to the Maximum price then BullishProbability will be 0 and If current stock price equals to the Minimum price then BullishProbability will be 1. [1]

The following factors after `CFCutoff` filtering removes continuously down going companies and tells us that if the price is still lower. This leaves us with 289 companies.

In conclusion, From 1059 richest companies, we take 289 stocks whose history data from June 29, 2019 to April 24, 2020 (300 days) shows a positive correlation factor of minimum 70% , but recently the stock price had a small downfall. We collect all the history data for 289 companies from the date 01/01/2013 to 04/24/2020 which gives us a data frame of size 1841×289 and `date` as the index. But still 289 companies are way too many for our portfolio optimization. We want to play with a shorter list, to achieve that we do exploratory analysis of data.

[1]There can be many other factors like `50 days max`, `Price Earning Ratio`, it differs from investor to investor. This is our own logic of how we like investing.

CHAPTER 4

EXPLORATORY ANALYSIS

4.1 Try to understand the data

To understand the data, We plotted all the closing price data of all the 289 companies. We see from

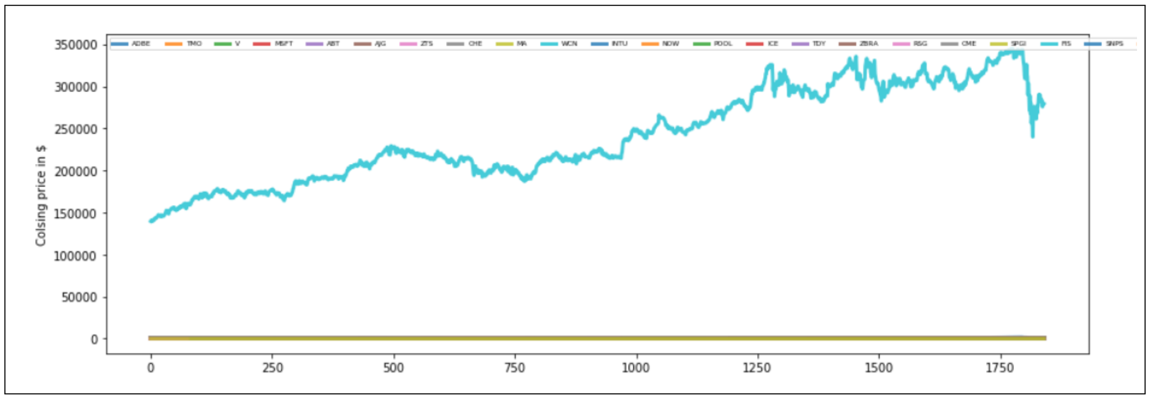


Figure 4.1: Closing Price of 289 Companies for 1841 day

the figure above that there is a big price variation of stocks. Specially `BRK.A` price is much higher than the other stocks. This makes every other stock insignificant in the graph. So, we cannot work directly with price. We decide to work with daily percentage change data which will make every stock significantly comparable to each other. Let us plot percentage change versus stocks.

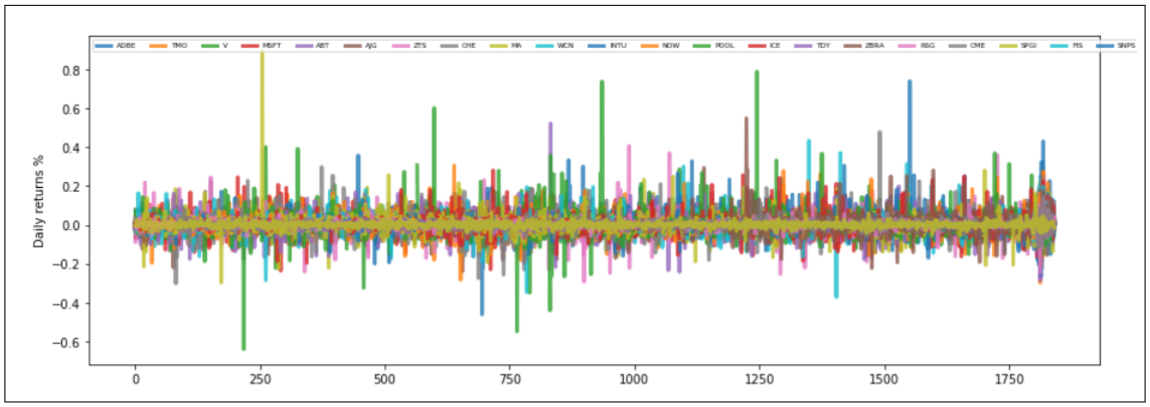
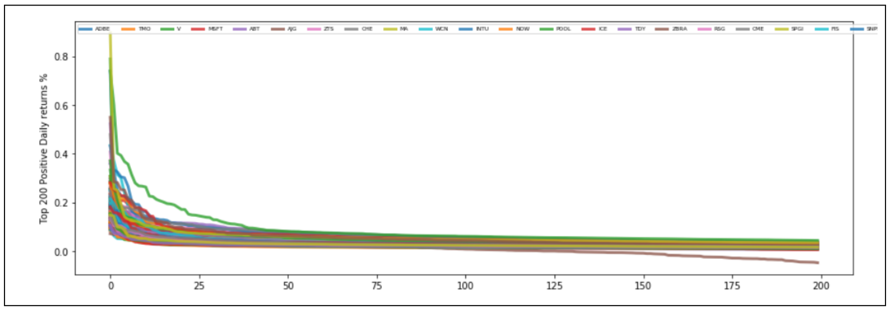


Figure 4.2: Daily returns% Change of 289 Companies for 1841 day

We see that many stocks spiked high and low in many situations. Some stocks may not be very volatile. Let us investigate them all. To do that we create two data frames where the top 200 positive and negative percentage change of companies will be listed.


(a) Positive Daily returns% Change of Companies

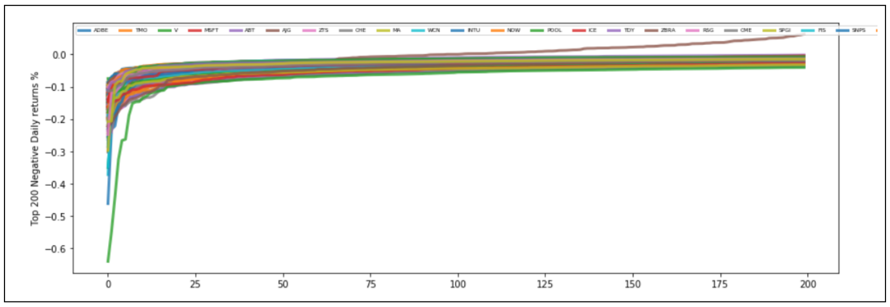
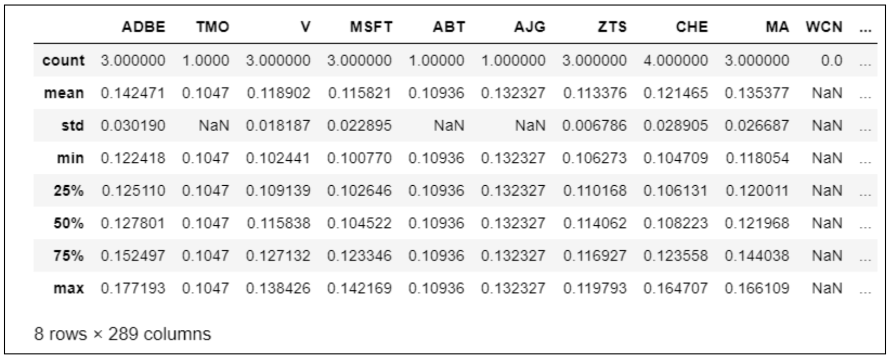

(b) Negative Daily returns% Change of Companies

Figure 4.3: Top 200 Positive and Negative Daily returns% Change of Companies

From the graph above we can conclude that around 50 maximum days having daily percentage change more than 15% comparing to maximum number of total days which is 1841 . We created another data frame with the statistical descriptions of each stock with count of number of times companies had more than 10% change in one day. We are interested to know about the statistical data of companies where the daily percentage change was more than 15% in one day.

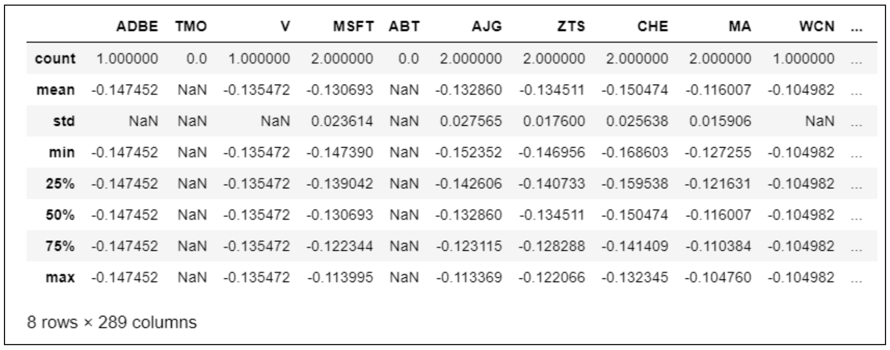
We set the threshold for each statistics to get the volatile companies, such as $count = 5, mean = 0.17, std = 0.01, min = 0.13, per25 = 0.15, per50 = 0.17, per75 = 0.19, max = 0.21$. The companies whose statistics are greater then the set threshold (for positive daily return% change) and smaller then the negative of set threshold (for negative daily return% change), we term them



	ADBE	TMO	V	MSFT	ABT	AJG	ZTS	CHE	MA	WCN	...
count	3.000000	1.0000	3.000000	3.000000	1.00000	1.000000	3.000000	4.000000	3.000000	0.0	...
mean	0.142471	0.1047	0.118902	0.115821	0.10936	0.132327	0.113376	0.121465	0.135377	NaN	...
std	0.030190	NaN	0.018187	0.022895	NaN	NaN	0.006786	0.028905	0.026687	NaN	...
min	0.122418	0.1047	0.102441	0.100770	0.10936	0.132327	0.106273	0.104709	0.118054	NaN	...
25%	0.125110	0.1047	0.109139	0.102646	0.10936	0.132327	0.110168	0.106131	0.120011	NaN	...
50%	0.127801	0.1047	0.115838	0.104522	0.10936	0.132327	0.114062	0.108223	0.121968	NaN	...
75%	0.152497	0.1047	0.127132	0.123346	0.10936	0.132327	0.116927	0.123558	0.144038	NaN	...
max	0.177193	0.1047	0.138426	0.142169	0.10936	0.132327	0.119793	0.164707	0.166109	NaN	...

8 rows × 289 columns

(a) Positive Daily returns% Change



	ADBE	TMO	V	MSFT	ABT	AJG	ZTS	CHE	MA	WCN	...
count	1.000000	0.0	1.000000	2.000000	0.0	2.000000	2.000000	2.000000	2.000000	1.000000	...
mean	-0.147452	NaN	-0.135472	-0.130693	NaN	-0.132860	-0.134511	-0.150474	-0.116007	-0.104982	...
std	NaN	NaN	NaN	0.023614	NaN	0.027565	0.017600	0.025638	0.015906	NaN	...
min	-0.147452	NaN	-0.135472	-0.147390	NaN	-0.152352	-0.146956	-0.168603	-0.127255	-0.104982	...
25%	-0.147452	NaN	-0.135472	-0.139042	NaN	-0.142606	-0.140733	-0.159538	-0.121631	-0.104982	...
50%	-0.147452	NaN	-0.135472	-0.130693	NaN	-0.132860	-0.134511	-0.150474	-0.116007	-0.104982	...
75%	-0.147452	NaN	-0.135472	-0.122344	NaN	-0.123115	-0.128288	-0.141409	-0.110384	-0.104982	...
max	-0.147452	NaN	-0.135472	-0.113995	NaN	-0.113369	-0.122066	-0.132345	-0.104760	-0.104982	...

8 rows × 289 columns

(b) Negative Daily returns% Change

Figure 4.4: Snapshot of Statistical data of companies where percentage change was more than 15% in one day

as volatile companies. Removing all the companies falling on either positive side, negative side or both positive and negative side, we are left with 45 comparatively less volatile companies. But we also want to forcefully keep all the top 30 richest company in our list. As 10 of these top 30 companies are already in our list we end up with a total of 65 companies and a data frame of size 1841×65, for which we will perform *Portfolio Optimization*.[1]

[1]These type of shortlisting of companies absolutely depends on the investors. I am noway cleverer than many clever investors in terms of this shortlisting business. So, we can only focus on these companies and don't bother about how I shortlisted. Again, this shortlisting is no way better than many other better algorithms to short list the companies.

CHAPTER 5

EXPERIMENTS AND RESULTS

For our experimental purpose, we have divided the shortlisted 65 companies based on the sectors which are *Basic Materials*, *Consumer Goods*, *Financial*, *Healthcare*, *Industrial Goods*, *Services*, *Technology* and *Utilities*.

5.1 Randomized Weights and Risk Return Relation

From the discussion of Section 1.1 we first create random weights and calculate their risk and return for the optimal solution of both the optimization problems as mentioned in Section 1.1.4. We want to see how the (risk, return) points look like. We create 20000 random weight vectors, keeping the constraint in mind i.e. , $\mathbf{1}^T\texttt{weights} = 1$ and calculate return and risk of 4 random companies of each sector.

Figure 5.1 are the tables of weights for *minimum risk* and *maximum Sharpe ratio* along with annualized return and volatility. It is observed that the sector *Basic Materials* have only 1 stock, so we did not study that.

In figure 5.2, we try to understand annualized risk, annualized volatility for *minimum risk* and *maximum Sharpe ratio* visually.

5.2 Quadratic Programming

From the discussion of 2.3, where we showed how SQP - Sequential Quadratic Programming algorithm works, is used here to find the optimal solution of both the optimization problem as mentioned in 1.1.4. Due to time constraint, instead of building the algorithm from scratch as of now I have used Python in built function `scipy.optimize.minimize` with `method='SLSQP'` [9], to solve both of our optimization problem, i.e; maximizing Sharpe ratio and minimizing the risk for a particular return.

The weights obtained for randomly selected companies of each sector are shown in Figure 5.3

5.3 Efficient Frontier

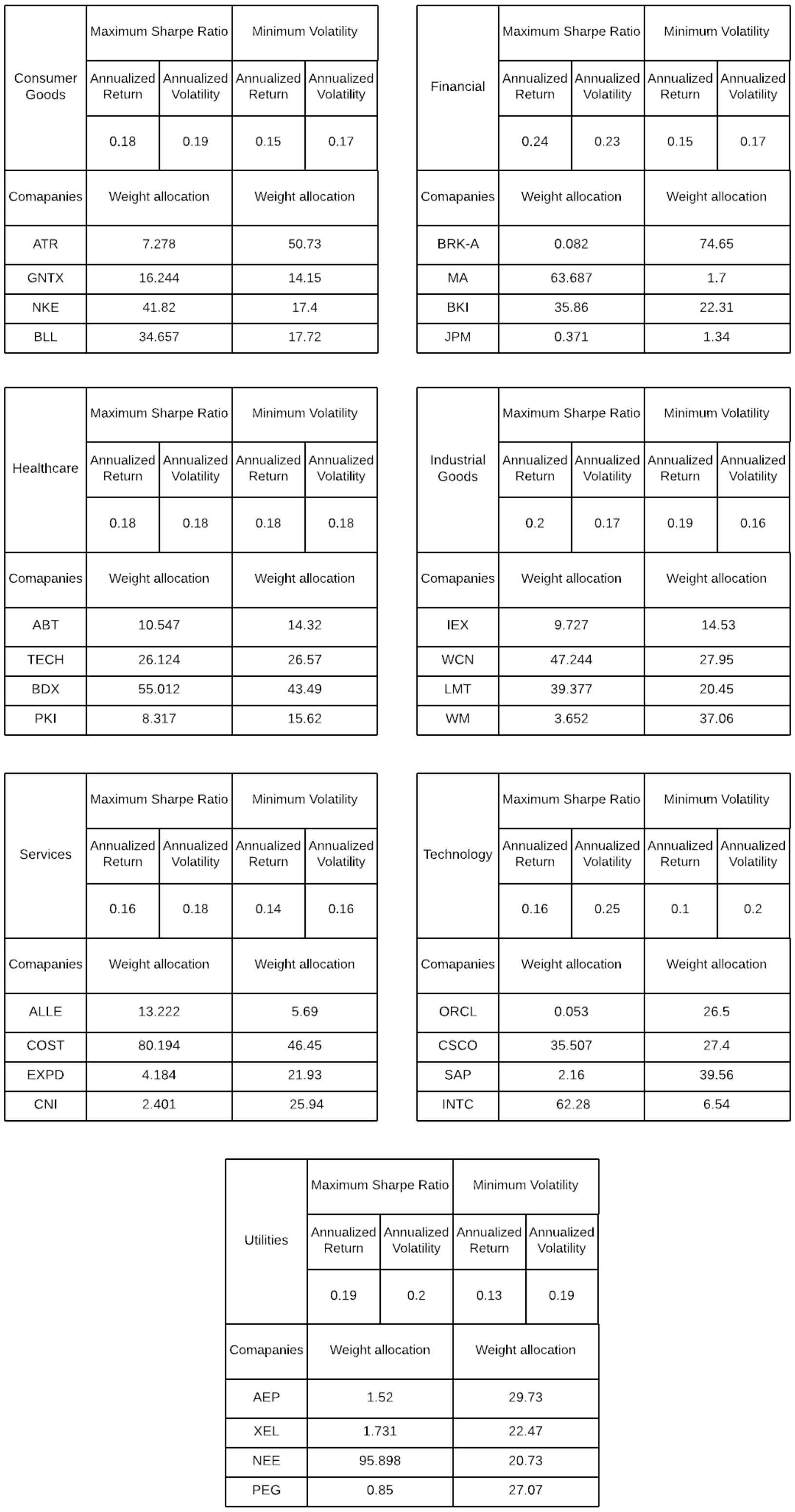
To find efficient frontier we optimize over a lot of randomly generated rate of return from the range 0 to 100.

The Efficient frontier along with maximum Sharpe ratio and minimum risk for each displayed in Figure 5.4.

5.4 Discussion

We see from the graphs that the risk return points follows a nearly parabolic shape that suggests us the following,

1. There is a lower bound of risk.
2. For a particular fixed risk there is an upper bound of return.
3. For a particular fixed return there is an lower bound of return.
4. Increment of risk increases the probability of return.
5. Increment of desired return increases the risk.
6. There is a point of minimum risk where the return is also low.
7. There is an optimum point called maximum Sharpe ratio according to the previous discussion which gives us an optimum point for risk-return trade off.
8. Portfolios lying below the efficient frontier are sub optimal as they don't provide enough return for the level of risk.
9. Portfolios that cluster on the right side of the efficient frontier are sub-optimal as they have high risk for defined rate of return.



Consumer Goods	Maximum Sharpe Ratio		Minimum Volatility	
	Annualized Return	Annualized Volatility	Annualized Return	Annualized Volatility
	0.18	0.19	0.15	0.17
Comapanies	Weight allocation		Weight allocation	
ATR	7.278		50.73	
GNTX	16.244		14.15	
NKE	41.82		17.4	
BLL	34.657		17.72	

Financial	Maximum Sharpe Ratio		Minimum Volatility	
	Annualized Return	Annualized Volatility	Annualized Return	Annualized Volatility
	0.24	0.23	0.15	0.17
Comapanies	Weight allocation		Weight allocation	
BRK-A	0.082		74.65	
MA	63.687		1.7	
BKI	35.86		22.31	
JPM	0.371		1.34	

Healthcare	Maximum Sharpe Ratio		Minimum Volatility	
	Annualized Return	Annualized Volatility	Annualized Return	Annualized Volatility
	0.18	0.18	0.18	0.18
Comapanies	Weight allocation		Weight allocation	
ABT	10.547		14.32	
TECH	26.124		26.57	
BDX	55.012		43.49	
PKI	8.317		15.62	

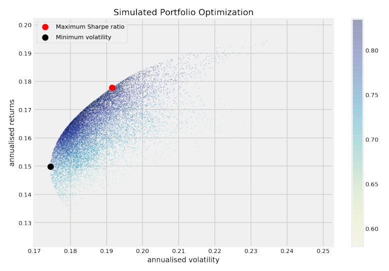
Industrial Goods	Maximum Sharpe Ratio		Minimum Volatility	
	Annualized Return	Annualized Volatility	Annualized Return	Annualized Volatility
	0.2	0.17	0.19	0.16
Comapanies	Weight allocation		Weight allocation	
IEX	9.727		14.53	
WCN	47.244		27.95	
LMT	39.377		20.45	
WM	3.652		37.06	

Services	Maximum Sharpe Ratio		Minimum Volatility	
	Annualized Return	Annualized Volatility	Annualized Return	Annualized Volatility
	0.16	0.18	0.14	0.16
Comapanies	Weight allocation		Weight allocation	
ALLE	13.222		5.69	
COST	80.194		46.45	
EXPD	4.184		21.93	
CNI	2.401		25.94	

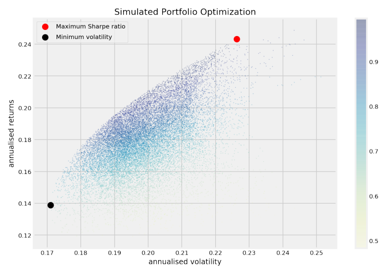
Technology	Maximum Sharpe Ratio		Minimum Volatility	
	Annualized Return	Annualized Volatility	Annualized Return	Annualized Volatility
	0.16	0.25	0.1	0.2
Comapanies	Weight allocation		Weight allocation	
ORCL	0.053		26.5	
CSCO	35.507		27.4	
SAP	2.16		39.56	
INTC	62.28		6.54	

Utilities	Maximum Sharpe Ratio		Minimum Volatility	
	Annualized Return	Annualized Volatility	Annualized Return	Annualized Volatility
	0.19	0.2	0.13	0.19
Comapanies	Weight allocation		Weight allocation	
AEP	1.52		29.73	
XEL	1.731		22.47	
NEE	95.898		20.73	
PEG	0.85		27.07	

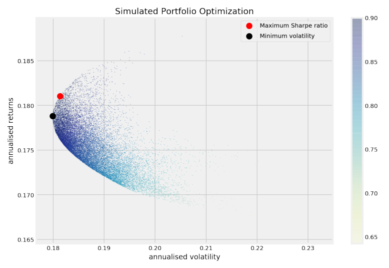
Figure 5.1: Sector wise final weights, annualized return and volatility.



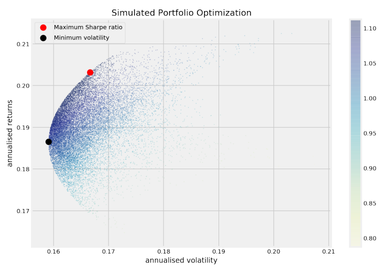
(a) Consumer Goods



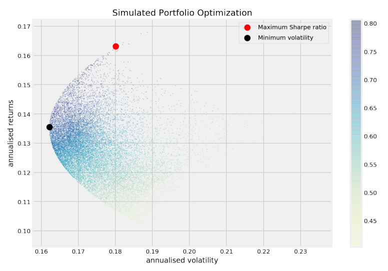
(b) Financial



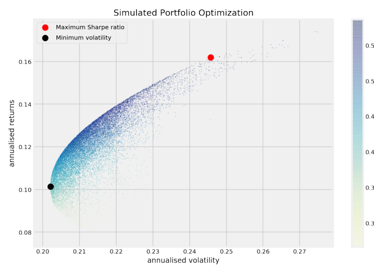
(c) Healthcare



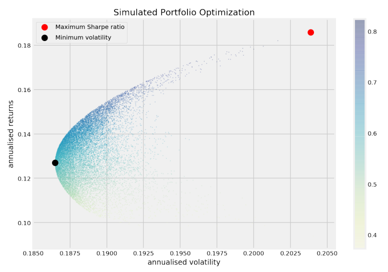
(d) Industrial Goods



(e) Services



(f) Technology



(g) Utilities

Figure 5.2: Sector wise Risk Return Plot using Randomized Weights

Consumer Goods	Maximum Sharpe Ratio		Minimum Volatility	
	Annualized Return	Annualized Volatility	Annualized Return	Annualized Volatility
	0.18	0.19	0.15	0.17
Comapanies	Weight allocation		Weight allocation	
ATR	7.278		50.73	
GNTX	16.244		14.15	
NKE	41.82		17.4	
BLL	34.657		17.72	

Financial	Maximum Sharpe Ratio		Minimum Volatility	
	Annualized Return	Annualized Volatility	Annualized Return	Annualized Volatility
	0.24	0.23	0.15	0.17
Comapanies	Weight allocation		Weight allocation	
BRK-A	0.082		74.65	
MA	63.687		1.7	
BKI	35.86		22.31	
JPM	0.371		1.34	

Healthcare	Maximum Sharpe Ratio		Minimum Volatility	
	Annualized Return	Annualized Volatility	Annualized Return	Annualized Volatility
	0.18	0.18	0.18	0.18
Comapanies	Weight allocation		Weight allocation	
ABT	10.547		14.32	
TECH	26.124		26.57	
BDX	55.012		43.49	
PKI	8.317		15.62	

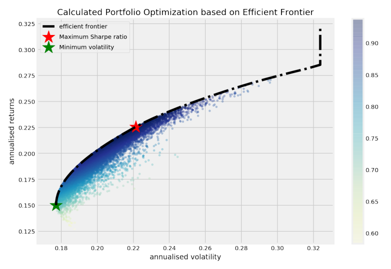
Industrial Goods	Maximum Sharpe Ratio		Minimum Volatility	
	Annualized Return	Annualized Volatility	Annualized Return	Annualized Volatility
	0.2	0.17	0.19	0.16
Comapanies	Weight allocation		Weight allocation	
IEX	9.727		14.53	
WCN	47.244		27.95	
LMT	39.377		20.45	
WM	3.652		37.06	

Services	Maximum Sharpe Ratio		Minimum Volatility	
	Annualized Return	Annualized Volatility	Annualized Return	Annualized Volatility
	0.16	0.18	0.14	0.16
Comapanies	Weight allocation		Weight allocation	
ALLE	13.222		5.69	
COST	80.194		46.45	
EXPD	4.184		21.93	
CNI	2.401		25.94	

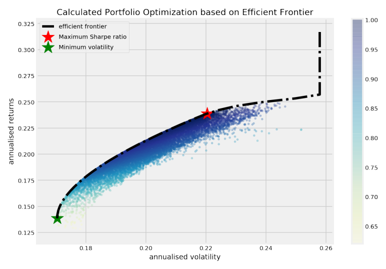
Technology	Maximum Sharpe Ratio		Minimum Volatility	
	Annualized Return	Annualized Volatility	Annualized Return	Annualized Volatility
	0.16	0.25	0.1	0.2
Comapanies	Weight allocation		Weight allocation	
ORCL	0.053		26.5	
CSCO	35.507		27.4	
SAP	2.16		39.56	
INTC	62.28		6.54	

Utilities	Maximum Sharpe Ratio		Minimum Volatility	
	Annualized Return	Annualized Volatility	Annualized Return	Annualized Volatility
	0.19	0.2	0.13	0.19
Comapanies	Weight allocation		Weight allocation	
AEP	1.52		29.73	
XEL	1.731		22.47	
NEE	95.898		20.73	
PEG	0.85		27.07	

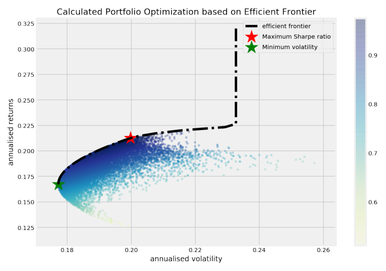
Figure 5.3: Sector wise final weights, annualized return and volatility.



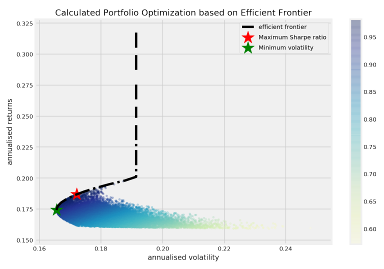
(a) Consumer Goods



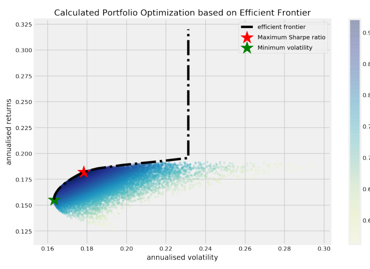
(b) Financial



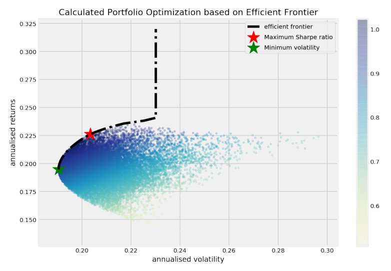
(c) Healthcare



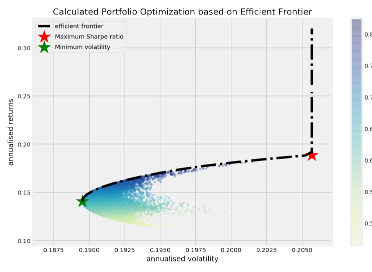
(d) Industrial Goods



(e) Services



(f) Technology



(g) Utilities

Figure 5.4: Sector wise Risk Return Plot and Efficient Frontier using Optimization Algorithm

CHAPTER 6

LEARNING OUTCOME

I first started with constrained optimization with quadratic objective function and linear constraints. This required me to understand linear programming as well as quadratic programming techniques. For the existence and uniqueness of the solution, I went through several theorems and their proofs, e.g. LICQ conditions, first odrer necessary conditions, second order sufficient conditions. Besides, understood why a quadratic problem reduces to a very simple optimization problem when it has only equality constraints. Moreover, I did hand calculation as well as programming using python on the simplex method and KKT conditions while learning constrained optimization techniques. This was a very interesting journey to see how the quadratic and linear functions are optimized over convex sets. This was enough to solve minimum risk maximum return problem of portfolio optimization. Then I took it one step further to maximize Sharpe ratio. This required optimizing a non linear objective function over linear constraints. But according to the knowledge I obtained, for a non convex general non-linear function the uniqueness of minimum is not guaranteed. Moreover, the existence of solution for smooth non-linear function over a bounded domain is guaranteed if the domain has a boundary. But we can always find the infimum even though minimum is not guaranteed. Hence, the minimization of non-linear functions are quite hard. I tried my best to see what is there in literature and found few methods where I have chosen Sequential Quadratic Programming, which in a sense converts non-linear optimization problem with arbitrary constraints to a quadratic optimization problem with linear constraints, which I explained in Section 2.3. Doing this project along with the course gave me a good amount of understanding of how optimization works. I am still continuing and trying to build my own non-linear optimization library.

REFERENCES

[1] Chen, J. (2019). What is a portfolio? `https://www.investopedia.com/terms/p/portfolio.asp`.

[2] Ganti, A. (2019). Efficient frontier. `https://www.investopedia.com/terms/e/efficientfrontier.asp`.

[3] GuidedChoice (2010). Harry markowitz's modern portfolio theory: The efficient frontier. `https://www.guidedchoice.com/video/dr-harry-markowitz-father-of-modern-portfolio-theory/`.

[4] Kraft, D. (1988). A software package for sequential quadratic programming. `http://degenerateconic.com/wp-content/uploads/2018/03/DFVLR_FB_88_28.pdf`.

[5] Lioudis, N. (2019). Understanding the sharpe ratio. `https://www.investopedia.com/articles/07/sharpe_ratio.asp`.

[6] Nocedal, J. and S. J. Wright (2000). Numerical optimization. `https://www.csie.ntu.edu.tw/~r97002/temp/num_optimization.pdf`.

[7] Posey, L. (2019). Optimizing portfolios with modern portfolio theory using python. `https://towardsdatascience.com/optimizing-portfolios-with-modern-portfolio-theory-using-python-60ce9a597808`.

[8] Richardson, L. (2020). Beautiful soup documentation. `https://www.crummy.com/software/BeautifulSoup/bs4/doc/`.

[9] SciPy.org (2019). minimize(method='slsqp'). `https://docs.scipy.org/doc/scipy/reference/optimize.minimize-slsqp.html`.

[10] Treadway, A. (2020). Yahoo_fin documentation. `http://theautomatic.net/yahoo_fin-documentation/`.