Once you face your fear, nothing is ever as hard as you think. - Olivia Newton-John

Nonlinear Optimization

Types of Optimization

There are 4 main types of optimization problems:

1. Linear Programming – objective function and constraints are linear.
2. Integer Linear Programming – objective function and constraints are linear but decision variables must be integers.
3. Mixed Integer Linear Programming – same as ILP with only some decision variables restricted to integers.
4. Non-linear Programming – objective function and constraints continuous but not all linear.

Types of Optimization

There are 4 main types of optimization problems:

~~**1. Linear Programming** – objective function and constraints are linear.~~

~~**2. Integer Linear Programming** – objective function and constraints are linear but decision variables must be integers.~~

~~**3. Mixed Integer Linear Programming** – same as ILP with only some decision variables restricted to integers.~~

4. Non-linear Programming – objective function and constraints continuous but not all linear.

Linear vs. Nonlinear

Examples of linear relationships:

$y = ax + b$ $\quad\quad$ $z = ax + by$

Examples of nonlinear relationships:

$y = ax^b$ $\quad\quad$ $z = axy$

Algorithms

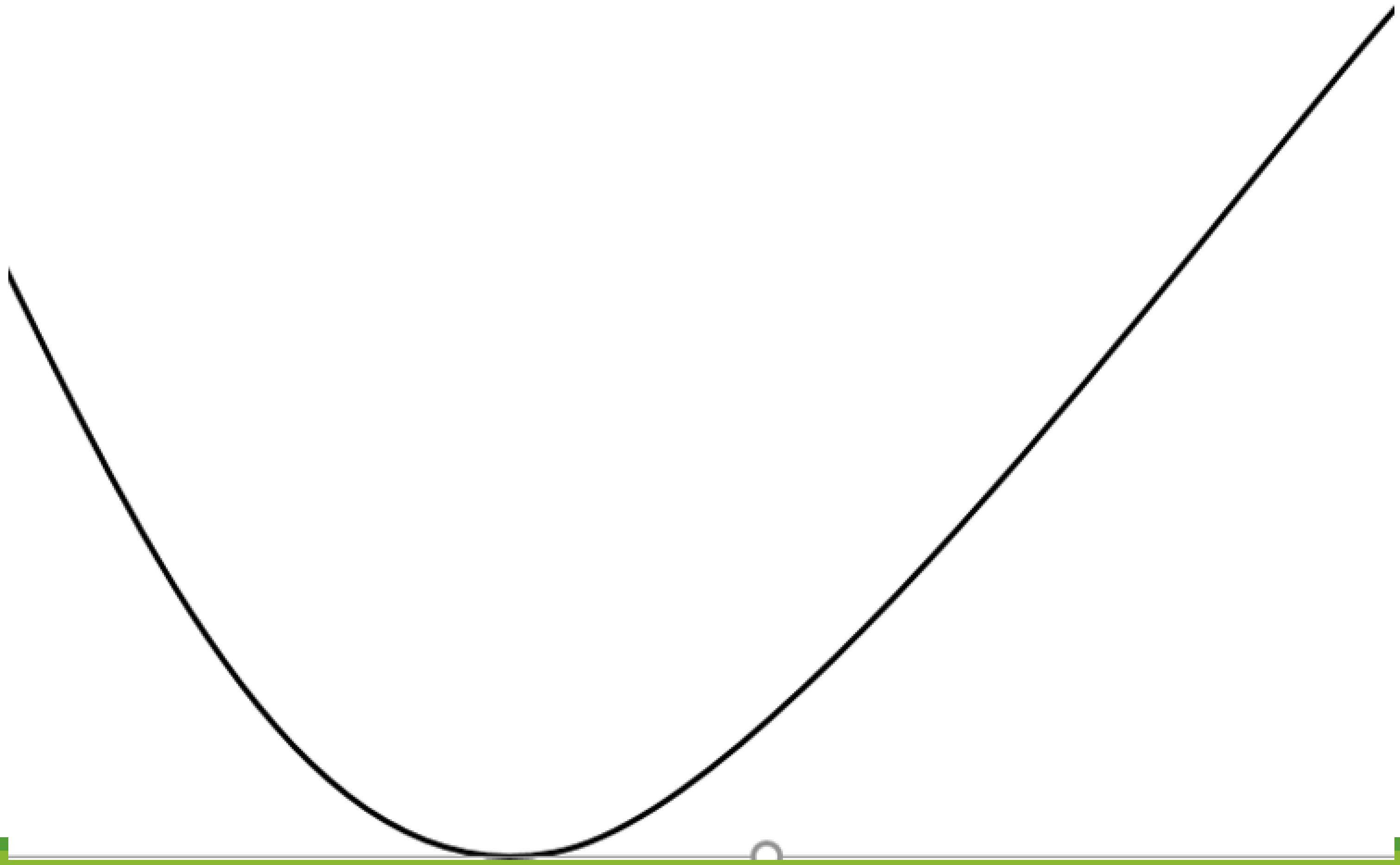
Nonlinear optimization is a lot harder of a process than linear optimization (careful of local optimum)

Many algorithms use gradients to solve the optimization.

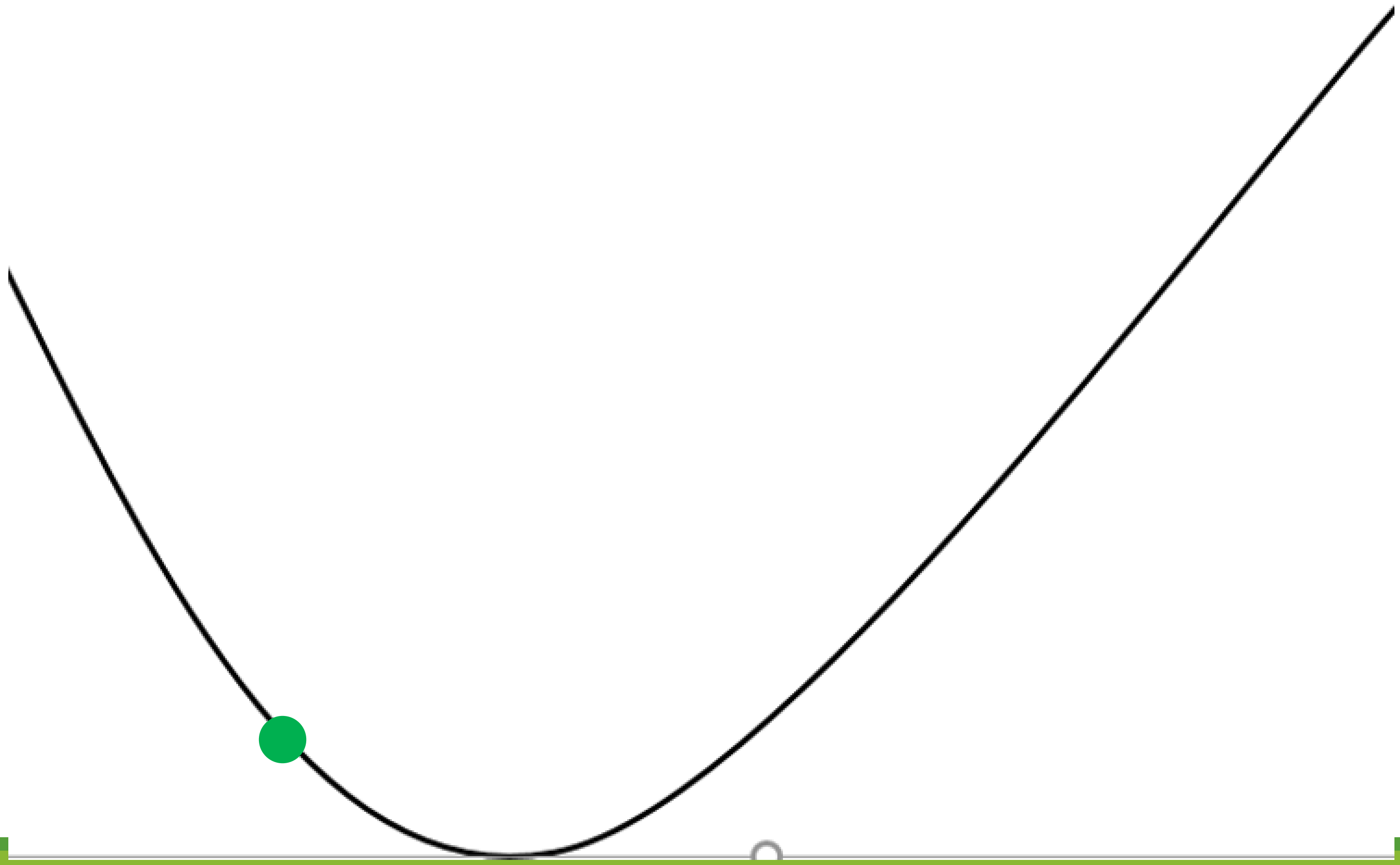
- Conjugate gradient method
- Newton method with line search
- Trust region

Genetic Algorithms

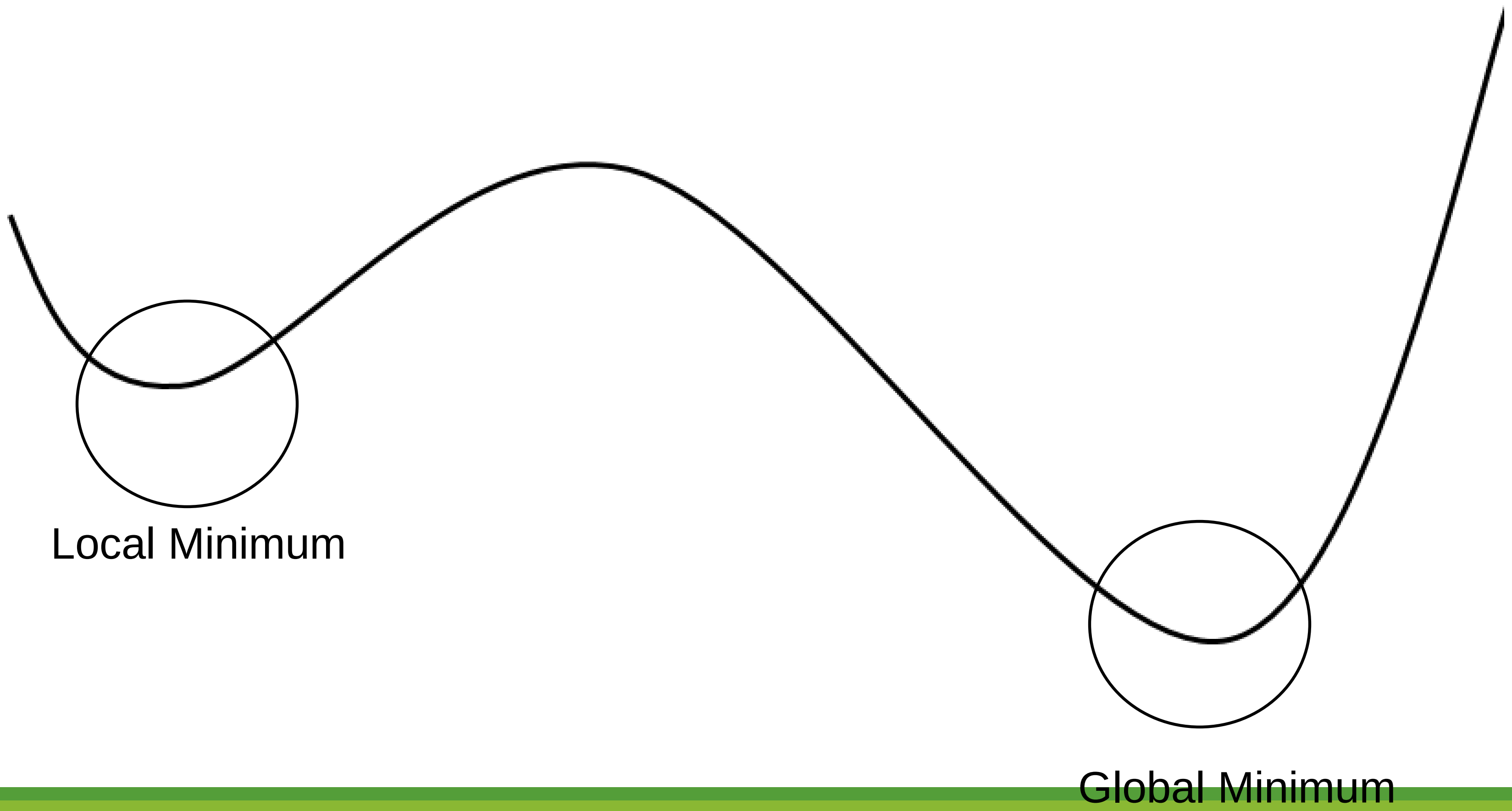
An example of Gradient Descent (minimizing a function)



An example of Gradient Descent (minimizing a function)



Potential issues



Multiple Answers

Depending on where we start determines our answer!

In cases with both local and global optima, there is no guarantee that a single run will produce the correct answer.

The best method is to try many different starting points to get an idea of how good our answer actually is.

Portfolio Optimization

Financial Portfolio

A ***portfolio*** is a collection of assets where the investor chooses the investment amount of each investment in the portfolio.

Portfolio performance is typically measured by total value of the portfolio at the end of a period of time.

To determine how much to allocate in each part of a portfolio, two things must be considered – risk and return.

Risk versus Return

Return –growth in the value of an asset (for example percentage growth)

Risk – variability / volatility associated with the returns on the stock (can use standard deviation or variance)

We can look at historical data to estimate both risk and return.

Example: overall means and variance over a certain period of time (use historical data).

Optimizing a portfolio

When optimizing a portfolio, we focus on risk and return, therefore, we could either:

1. Minimize risk for a given return (typical)

2. Maximize return for a given risk

Risk and Return are generally related (Higher return = higher risk)

Return of a portfolio

One way of estimating return is calculating the percentage growth

$$\frac{Recent - Previous}{Previous} x100$$

In a portfolio, we have numerous stocks, each with their own return, r_1, r_2, r_3,…r_k

The return of the whole portfolio will depend on the amount in each stock, p_1, p_2, p_3…p_k (these are the decision variables)

Return for portfolio: $p_1r_1 + p_2r_2 + p_3r_3 + \ldots + p_kr_k$

Risk of a portfolio

Risk of a portfolio is the variation (volatility of the portfolio)

Individual assets have their own variability (denote variance of asset 1 as σ_{11})

- Need to go back to basic statistics, where we define Cov(Y_1 +Y_2)

$$Cov(aY_1 + bY_2) = a^2V(Y_1) + b^2V(Y_2) + 2abCov(Y_1, Y_2)$$

- We want to find the Cov(p_1r_1 +p_2r_2+p_3r_3 + p_4r_4 + p_5r_5)

  where p_1,..p_5 are the proportion in each stock and r_1…r_5 are the returns for each stock

$$Cov(p_1r_1 + p_2r_2 + p_3r_3 + p_4r_4 + p_5r_5) = p_1^2V(r_1) + p_2^2V(r_2) + p_3^2V(r_3) + p_4^2V(r_4) + p_5^2V(r_5) + 2p_1p_2Cov(r_1, r_2) + 2p_1p_3Cov(r_1, r_3)+\ldots++2p_4p_5Cov(r_4, r_5)$$

In Matrix form… (bet you thought you wouldn't see this again….)

$$
\begin{bmatrix} p_1 & p_2 & p_3 & p_4 & p_5 \end{bmatrix}
\begin{bmatrix}
\sigma_{11} & \sigma_{12} & \sigma_{13} & \sigma_{14} & \sigma_{15} \\
\sigma_{21} & \sigma_{22} & \sigma_{23} & \sigma_{24} & \sigma_{25} \\
\sigma_{31} & \sigma_{32} & \sigma_{33} & \sigma_{34} & \sigma_{35} \\
\sigma_{41} & \sigma_{42} & \sigma_{43} & \sigma_{44} & \sigma_{45} \\
\sigma_{51} & \sigma_{52} & \sigma_{53} & \sigma_{54} & \sigma_{55}
\end{bmatrix}
\begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{bmatrix}
$$

Portfolio Optimization Example

- Advising Ms Womack
  - Ms Womack has some savings to invest and has 5 preferable stocks to invest in.
  - These five stocks are in 5 different industries over the past two years.
  - High return → High volatility
  - Return:

$$Return = p_1 r_1 + p_2 r_2 + p_3 r_3 + p_4 r_4 + p_5 r_5$$

Average return of each stock

Portfolio Optimization Example

- Advising Ms Womack
  - Ms Womack has some savings to invest and has 5 preferable stocks to invest in.
  - These five stocks are in 5 different industries over the past two years.
  - High return → High volatility
  - Return:

$$Return = p_1 r_1 + p_2 r_2 + p_3 r_3 + p_4 r_4 + p_5 r_5$$

sum p > 1 must borrow money
sum p < 1 not investing all money
(invest in risk free rate)

Proportion of wealth in each stock

$$p_1 + p_2 + p_3 + p_4 + p_5 = 1$$

Portfolio Optimization Example

- Advising Ms Womack
  - Ms Womack has some savings to invest and has 5 preferable stocks to invest in.
  - These five stocks are in 5 different industries over the past two years.
  - High return → High volatility
  - Return: At least 0.015
  - Risk:

$$\sum_j \sum_k p_j * \sigma_{j,k} * p_k = p_1 \sigma_{1,1} p_1 + p_1 \sigma_{1,2} p_2 + \cdots + p_5 \sigma_{5,5} p_5$$

Covariance between each stock combination

Gurobi Code

Need to get libraries:

```
from gurobipy import *
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from math import sqrt
```

Optimization set up

Decision variables: 5 (what proportion should go into each investment)

Objective function: Minimize risk: $\sum_i \sum_j p_i \sigma_{i,j} p_j$

Constraints:

Sum of proportions equals 1: $\sum_i p_i = 1$

Return of at least 0.015: $\sum_i p_i r_i \geq 0.015$

NOTE: This is using the Portfolio data set on website.

```
data_stock=pd.read_csv('Q:\My Drive\Spring 1  2017 - Optimization\portfolio_r.csv')
stocks = data_stock.columns
num_stocks=len(stocks)
stock_return = data_stock.mean()
print(stock_return)
cov_mat=data_stock.cov()
# Create an empty model
m = Model('portfolio')

# Add a variable for each stock
vars = pd.Series(m.addVars(stocks,lb=0), index=stocks)
portfolio_risk = cov_mat.dot(vars).dot(vars)
m.setObjective(portfolio_risk, GRB.MINIMIZE)
## constraints
m.addConstr(vars.sum() == 1, 'budget')
m.addConstr(stock_return.dot(vars) >=0.015,'return')
m.optimize()
print('Minimum Risk Portfolio:\n')
for v in vars:
    if v.x > 0:
        print('\t%s\t: %g' % (v.varname, v.x))
```

Output

Optimal objective 1.27497282e-03

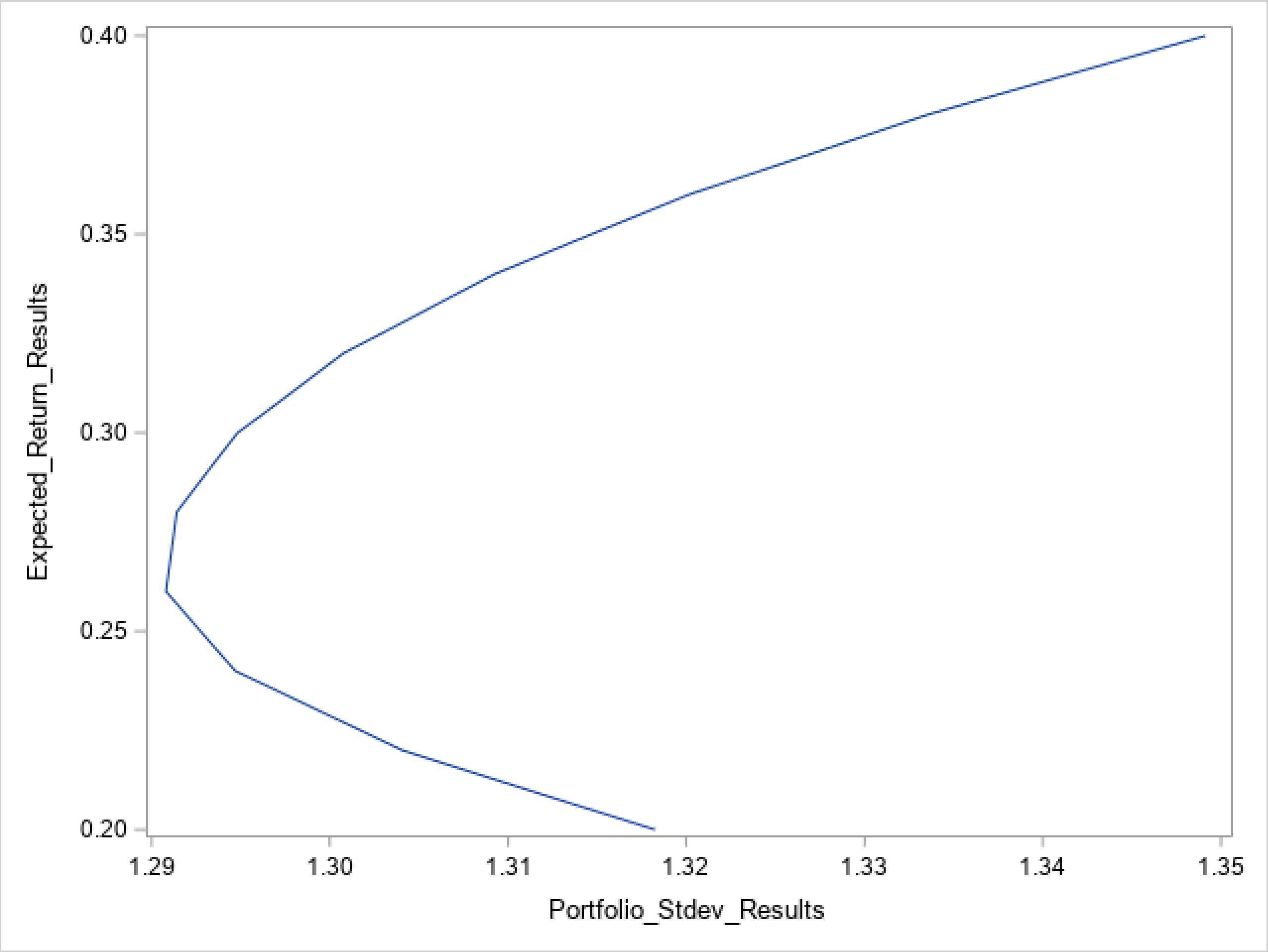
Minimum Risk Portfolio:

C0 : 0.132348
C1 : 0.304061
C2 : 0.158047
C3 : 0.224714
C4 : 0.18083

Efficient Frontier

Higher risk yields higher return. What is the best return we can achieve for a given level of risk (or what is the lowest risk for a given level of return)

The efficient frontier is the set of optimal portfolios that offers the highest expected return for a defined level of risk or the lowest risk for a given level of expected return. Portfolios that lie below the efficient frontier are sub-optimal, because they do not provide enough return for the level of risk. (Investopedia)



Note: this is using the stocks data set on website.

NOTE: THIS IS ROLLING UP TO MONTHLY!!

```
data_stock=pd.read_csv('Q:\My Drive\Spring 1  2017 - Optimization\stocks_r.csv')
data_stock.Date=pd.to_datetime(data_stock.Date)
data_stock['year'] = data_stock['Date'].dt.year
data_stock['month'] = data_stock['Date'].dt.month
data2=data_stock.drop(['Date'],axis=1)
data3 = data2.groupby(['year','month'],as_index=False).sum()
data3=data3.drop(['year','month'],axis=1)
stocks = data3.columns
num_stocks=len(stocks)

# Calculate basic summary statistics for individual stocks
stock_volatility = data3.std()
stock_return = data3.mean()
cov_mat=data3.cov()

returns = np.linspace( 0.2, 0.4, 100 )
ret_list = []
risks = []
props = []
```

```
for ret in returns:
    m.reset(0)
    m = Model("Portfolio_Optimization")
    m.setParam('OutputFlag', 0)
    vars=pd.Series(m.addVars(stocks,lb=0), index=stocks)
    portfolio_risk = cov_mat.dot(vars).dot(vars)
    m.setObjective(portfolio_risk, GRB.MINIMIZE)
    m.addConstr(vars.sum() == 1, name = 'budget' )
    m.addConstr(stock_return.dot(vars) == ret , name = 'return_sim' )
    m.update()
    m.optimize()
    risks.append(np.sqrt(m.objval ) )
    ret_list.append(stock_return.dot(m.x) )
    props.append(m.x)


plt.rcParams.update({'font.size': 22})
plt.plot( risks, returns )
plt.xlabel( 'Risk' )
plt.ylabel( 'Return' )
plt.title( 'Efficient Frontier' )
plt.plot()
```