Implementing Linear Programming in Python

Vitthal Srinivasan
CO-FOUNDER, LOONYCORN
www.loonycorn.com

Overview

Use historical data of stocks to estimate risk and return of a stock portfolio

Initially define risk as a linear function

Formulate an LPP in the standard form

Re-define risk as portfolio variance, creating a quadratic optimization

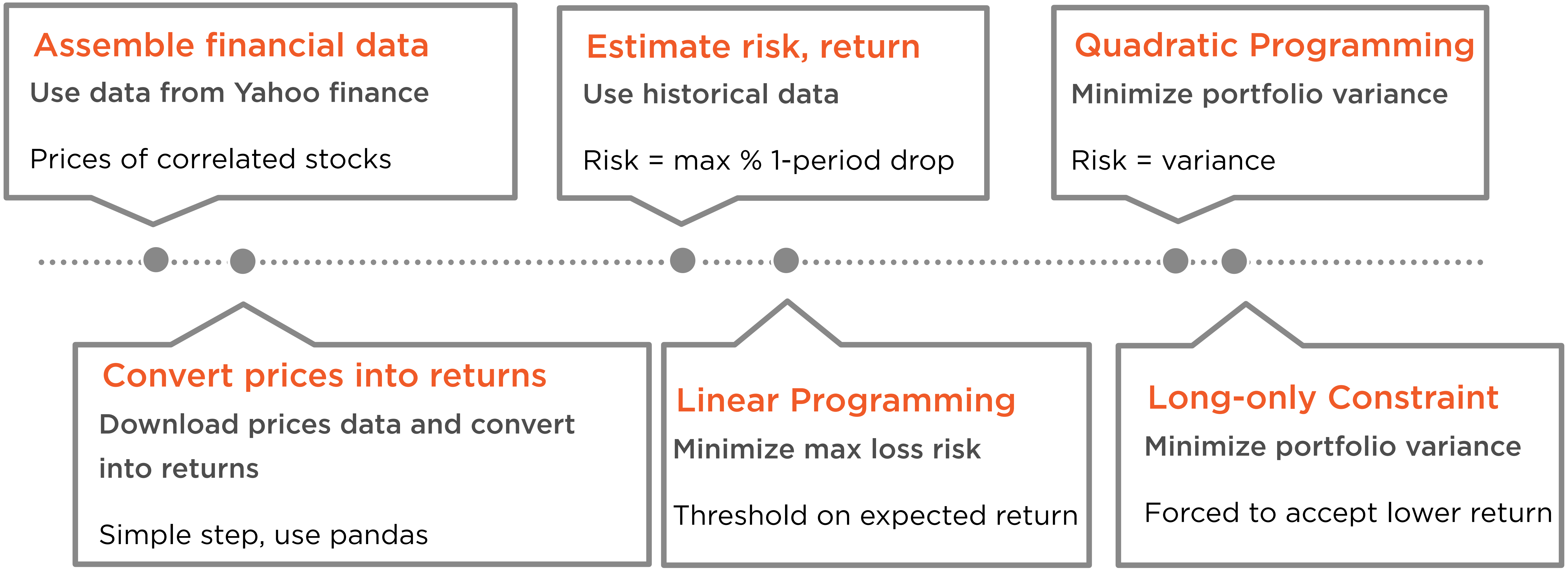
Turn on and off non-negativity constraints

Use Python to solve all of these optimization problems

Demo

Implement portfolio optimization using Python

Portfolio Optimization in R



Portfolio Optimization in R

Assemble financial data

Use data from Yahoo finance

Prices of correlated stocks

Portfolio as Sum of Random Variables

$$P = w_1 Y_1 + w_2 Y_2 + w_3 Y_3 \ldots + w_k Y_k$$

Modelling a portfolio as the sum of random variables is an extremely common use-case

Portfolio as Sum of Random Variables

$$P = w_1Y_1 + w_2Y_2 + w_3Y_3 \ldots + w_kY_k$$

P_i = % return of stock portfolio on day i

Portfolio P consists of $\$w_1$ stocks of Y_1, w_2 of Y_2, w_3 of Y_3 and w_k of Y_k

Set up the Problem

DATE	EXXON	GOOGLE		APPLE
2017-01-01	Y^1_E	Y^1_G		Y^1_A
2016-12-01	Y^2_E	Y^2_G		Y^2_A
2007-01-01	Y^n_E	Y^n_G		Y^n_A

Download prices from Yahoo finance (refer Adjusted close)

Data Frame: Data in Rows and Columns

Each column represents 1 variable (a list or vector)

Each row represents 1 observation

DATE	OPEN	...	ADJUSTED CLOSE
2016-12-01	772	...	779
2016-11-01	758	...	747
2006-01-01	302	...	309

From File to Data Frame

DATE	OPEN	...	ADJUSTED CLOSE
2016-12-01	772	...	779
2016-11-01	758	...	747
2006-01-01	302	...	309

File

read.table

DATE	OPEN	...	ADJUSTED CLOSE
2016-12-01	772	...	779
2016-11-01	758	...	747
2006-01-01	302	...	309

Data Frame

Portfolio Optimization in R

Assemble financial data

Use data from Yahoo finance

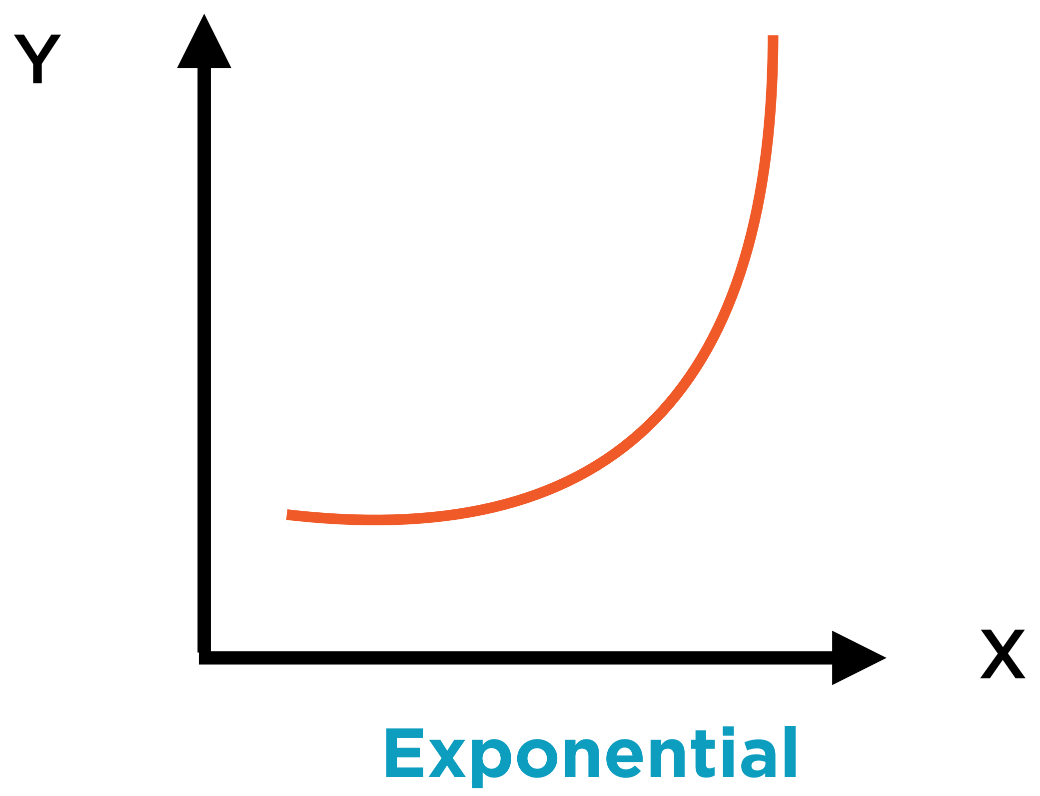
Prices of correlated stocks

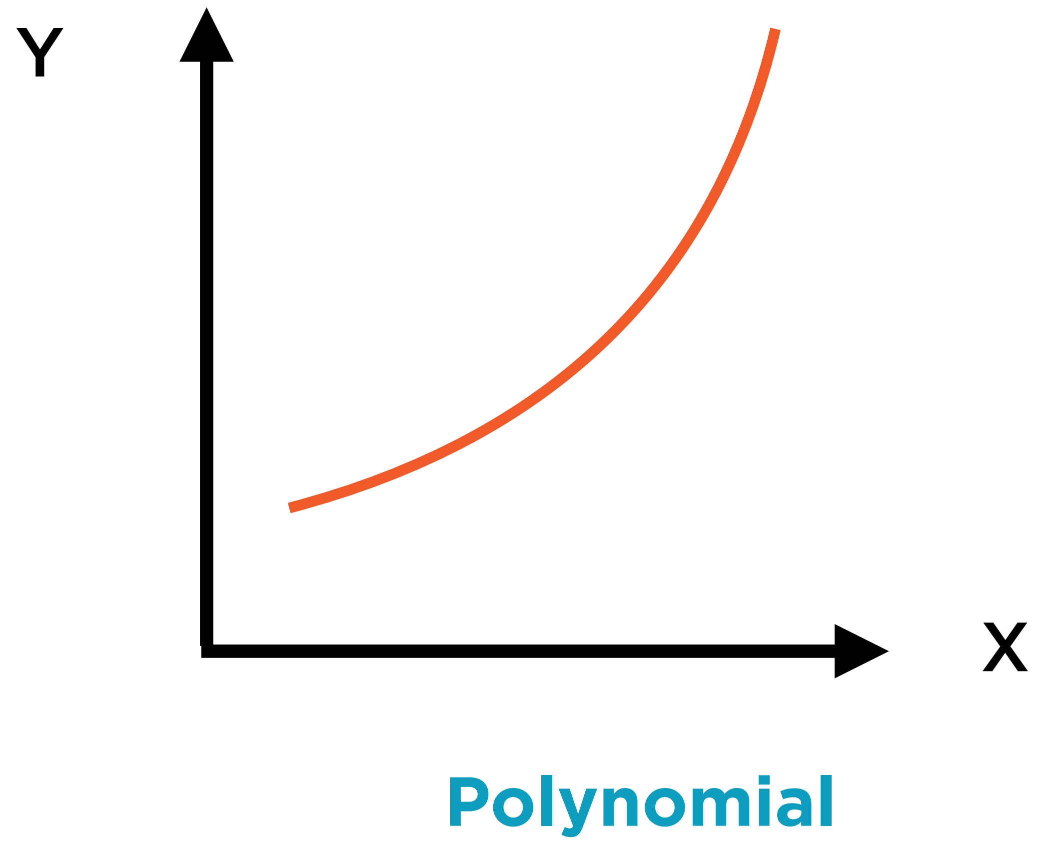
Convert prices into returns

Download prices data and convert into returns

Simple step, use Python pandas

Convert Prices to Returns





Smoothly trending data will lead to poor quality regression and covariance models

Convert Prices to Returns

$$y'_{12} = \log y_2 - \log y_1$$

$$x'_{12} = \log x_2 - \log x_1$$

Regress y' and x'

Log Differences

$$y'_{12} = (y_2 - y_1)/y_1$$

$$x'_{12} = (x_2 - x_1)/x_1$$

Regress y' and x'

Returns

Take first differences of smooth data converting either to log differences or returns

Set up the Problem

DATE	EXXON	GOOGLE		APPLE
2007-01-01	Y^n_E	Y^n_G		Y^n_A
2016-12-01	Y^2_E	Y^2_G		Y^2_A
2007-01-01	Y^n_E	Y^n_G		Y^n_A

Sort date from oldest to newest to calculate returns

Negative Indices => Exclude Data

goog

DATE	GOOG. PRICE	NASDAQ. PRICE	
2016-12-01	779	5550	Row 1
2016-11-01	747	5324	
2006-01-01	309	1900	Row nrow(goog)
Column 1			

goog[-nrow(goog),-1]

Negative Indices => Exclude Data

goog

DATE	GOOG. PRICE	NASDAQ. PRICE	
2016-12-01	779	5550	Row 1
2016-11-01	747	5324	
2006-01-01	309	1900	Row nrow(goog)

Exclude

Column 1

goog[-nrow(goog),-1]

Negative Indices => Exclude Data

goog

DATE	GOOG. PRICE	NASDAQ. PRICE	
2016-12-01	779	5550	Row 1
2016-11-01	747	5324	
2006-01-01	309	1900	Row nrow(goog)

Exclude

Column 1

goog[-nrow(goog),-1]

Negative Indices => Exclude Data

goog

DATE	GOOG. PRICE	NASDAQ. PRICE	
2016-12-01	779	5550	Row 1
2016-11-01	747	5324	
2006-01-01	309	1900	Row nrow(goog)

Exclude

Column 1

goog[-nrow(goog),-1]

Element-wise Operations

779	5550
...	

/

747	5324

=

779/747	5550/5324
...	...
...	...

goog[-nrow(goog),-1]/
goog[-1,-1]

Prices to Returns

779/747	5550/5324
...	...
...	...

-

1	1
1	1
1	1
1	1
1	1

=

779/747 - 1	5550/5324 -1
...	...
...	...

goog[-nrow(goog),-1]/
goog[-1,-1] - 1

This converts prices to returns

Portfolio Optimization in R

Assemble financial data

Use data from Yahoo finance

Prices of correlated stocks

Convert prices into returns

Download prices data and convert into returns

Simple step, use Python pandas

Estimate risk, return

Use historical data

Risk = max % 1-period drop

Estimate Portfolio Return and Risk

$$P = w_1Y_1 + w_2Y_2 + w_3Y_3 \ldots + w_kY_k$$

Expected Return

Simple - use average of historical returns

Forecast Risk

Conservative - define as sum of max loss in each stock

Max Loss refers to largest % fall experienced by a stock in any period in our data

Estimate Portfolio Return and Risk

$$P = w_1Y_1 + w_2Y_2 + w_3Y_3 \ldots + w_kY_k$$

Expected Return = Mean(y)

Simple - mean of sum is sum of means

Forecast Risk = MaxLoss(y)

Conservative - define as sum of max loss in each stock

Max Loss refers to largest % fall experienced by a stock in any period in our data

Estimating Return

$$P = w_1Y_1 + w_2Y_2 + w_3Y_3 \dots + w_kY_k$$

$$\begin{aligned}\text{Mean}(P) = \; & w_1 \times \text{Mean}(Y_1) + \\ & w_2 \times \text{Mean}(Y_2) + \\ & w_3 \times \text{Mean}(Y_3) + \\ & \dots \\ & w_k \times \text{Mean}(Y_k)\end{aligned}$$

k terms, all linear

Mean of sum = sum of means

Estimating Return

$$P = w_1Y_1 + w_2Y_2 + w_3Y_3 \ldots + w_kY_k$$

$$\begin{aligned} \text{Mean}(P) = \; & w_1\bar{Y}_1 + \\ & w_2\bar{Y}_2 + \\ & w_3\bar{Y}_3 + \\ & \ldots \\ & w_k\bar{Y}_k \end{aligned}$$

k terms, all linear

Mean of sum = sum of means

Estimate Portfolio Return and Risk

$$P = w_1Y_1 + w_2Y_2 + w_3Y_3 \ldots + w_kY_k$$

Expected Return = Mean(y)

Simple - mean of sum is sum of means

Forecast Risk = MaxLoss(y)

Conservative - define as sum of max loss in each stock

Estimating Risk

$$P = w_1Y_1 + w_2Y_2 + w_3Y_3 \ldots + w_kY_k$$

$$\left.\begin{aligned} Risk(P) = \; & w_1 \times MaxLoss(Y_1) + \\ & w_2 \times MaxLoss(Y_2) + \\ & w_3 \times MaxLoss(Y_3) + \\ & \ldots \\ & w_k \times MaxLoss(Y_k) \end{aligned}\right\} \text{k terms, all linear}$$

Portfolio Risk = Sum of individual asset risks

Portfolio Variance in R

Assemble financial data

Use data from Yahoo finance

Prices of correlated stocks

Convert prices into returns

Download prices data and convert into returns

Simple step, use Python pandas

Estimate risk, return

Use historical data

Risk = max % 1-period drop

Linear Programming

Minimize max loss risk

Threshold on expected return

Portfolio Allocation as an Optimization Problem

Objective Function

Minimize Risk(P)

Risk(P) = MaxLoss(P)

Constraints

$\bar{P} >= R_{threshold}$

$\bar{P} = w_1\bar{Y}_1 + w_2\bar{Y}_2 + \dots\ w_k\bar{Y}_k$

Decision Variables

W

$W = [\ w_1 \quad w_2 \quad w_3 \ \dots \ w_k\ ]$

Portfolio Optimization in R

Assemble financial data

Use data from Yahoo finance

Prices of correlated stocks

Convert prices into returns

Download prices data and convert into returns

Simple step, use Python pandas

Estimate risk, return

Use historical data

Risk = max % 1-period drop

Linear Programming

Minimize max loss risk

Threshold on expected return

Quadratic Programming

Minimize portfolio variance

Risk = variance

Estimate Portfolio Return and Risk

$$P = w_1Y_1 + w_2Y_2 + w_3Y_3 \ldots + w_kY_k$$

Expected Return

Simple - use average of historical returns

Forecast Risk

Change definition of risk to refer to variance of portfolio

Change definition of risk to use portfolio variance (a more common, but less conservative approach)

Estimate Portfolio Return and Risk

$$P = w_1Y_1 + w_2Y_2 + w_3Y_3 \ldots + w_kY_k$$

Expected Return = Mean(y)

Simple - mean of sum is sum of means

Forecast Risk = Variance(y)

Tricky - requires use of covariance matrix

Change definition of risk to use portfolio variance (a more common, but less conservative approach)

Covariance Matrix

$$Y = Y_1 + Y_2 + Y_3 \ldots + Y_k$$

$$\begin{array}{cc} \begin{bmatrix} Var(Y_1) & Cov(Y_1, Y_2) & \ldots & Cov(Y_1, Y_k) \\ Cov(Y_2, Y_1) & Var(Y_2) & \ldots & Cov(Y_2, Y_k) \\ Cov(Y_k, Y_1) & Cov(Y_k, Y_2) & \ldots & Var(Y_k) \end{bmatrix} & \text{k rows} \\ \text{k columns} & \end{array}$$

A kxk matrix - diagonal elements are variances, off-diagonal elements are covariances

Adding Random Variables

$$P = w_1Y_1 + w_2Y_2 + w_3Y_3 \ldots + w_kY_k$$

$$\text{Variance}(P) = \sum_{i=1}^{k} \sum_{j=1}^{k} w_i w_j \,\text{Covariance}(Y_i, Y_j)$$

k^2 terms,

quadratic

Variance of the portfolio can be found by multiplying the weight vector with the covariance matrix

Portfolio Variance

$$P = w_1Y_1 + w_2Y_2 + w_3Y_3 \ldots + w_kY_k$$

$$\underset{1 \times 1}{Var(P)} = \underset{1 \times k}{W} * \underset{k \times k}{Cov(Y)} * \underset{k \times 1}{W^T}$$

Variance of the portfolio can be found by multiplying the weight vector with the covariance matrix

Portfolio Variance

$$P = w_1Y_1 + w_2Y_2 + w_3Y_3 \ldots + w_kY_k$$

$$W = \begin{bmatrix} w_1 & w_2 & w_3 & \ldots & w_k \end{bmatrix}$$

1 row

1 x k

k columns

The weight vector simply contains the weights of different stocks in the portfolio

Portfolio Variance

$$P = w_1Y_1 + w_2Y_2 + w_3Y_3 \ldots + w_kY_k$$

$$\underset{k \times 1}{W^T} = \begin{array}{c} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \ldots \\ w_k \end{bmatrix} \\ \text{1 column} \end{array} \begin{array}{c} \text{k rows} \end{array}$$

Transposing a vector reverses its rows and columns

Portfolio Variance

$$P = w_1Y_1 + w_2Y_2 + w_3Y_3 \ldots + w_kY_k$$

$$\underset{1 \times 1}{Var(P)} = \underset{1 \times k}{W} * \underset{k \times k}{Cov(Y)} * \underset{k \times 1}{W^T}$$

Variance of the portfolio can be found by multiplying the weight vector with the covariance matrix

Standard Form of Linear Programming Problems

Maximize

$$Z = c_1x_1 + c_2x_2 + \dots + c_nx_n$$

Subject to constraints:

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n <= b_1$$

$$a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n <= b_2$$

$$\vdots$$

$$a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n <= b_m$$

$$x_1 , x_2 \dots x_n >= 0$$ **(Non-negativity constraints)**

Quadratic Programming Problems

Maximize

$$Z = c_1x_1 + c_2x_2 + \dots + c_nx_n$$

$$+ q_{11}x^2{}_1 + q_{12}x_1x_2 + \dots + q_{nn}x^2{}_n$$

Subject to constraints:

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n <= b_1$$

$$a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n <= b_2$$

$$\vdots$$

$$a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n <= b_m$$

$$x_1 , x_2 \dots x_n >= 0$$

(Non-negativity constraints)

Quadratic Programming Problems

Maximize

$$Z = cx - \frac{1}{2}x^TQx$$

Subject to constraints:

$$Ax <= B$$

$$x >= 0$$

Matrix form of quadratic programming problems

Quadratic Programming Problems

Maximize

$$Z = cx - \frac{1}{2} x^T Q x$$

Subject to constraints:

$$Ax <= B$$

$$x >= 0$$

Can be solved using the Modified Simplex Method

Quadratic Programming Problems

Maximize

$$Z = cx - \frac{1}{2} x^T Q x$$

Subject to constraints:

$$Ax <= B$$

$$x >= 0$$

Here $c = 0$, $Q = Cov(Y)$, $x = W^T$

Quadratic Programming Problems

Maximize

$$Z = cx - \frac{1}{2} x^T Q x$$

Subject to constraints:

$$Ax <= B$$

$$\cancel{x >= 0}$$

Also, relax the non-negativity constraint to allow short selling

Portfolio Allocation as an Optimization Problem

Objective Function

Minimize Risk(P)

Risk(P) = Variance(P)

Constraints

$\bar{P} >= R_{threshold}$

$\bar{P} = w_1\bar{Y}_1 + w_2\bar{Y}_2 + \ldots\ w_k\bar{Y}_k$

Decision Variables

W

$W = [\, w_1 \quad w_2 \quad w_3 \quad \ldots \quad w_k \,]$

Portfolio Optimization in R

Assemble financial data
Use data from Yahoo finance

Prices of correlated stocks

Convert prices into returns
Download prices data and convert into returns

Simple step, use Python pandas

Estimate risk, return
Use historical data

Risk = max % 1-period drop

Linear Programming
Minimize max loss risk

Threshold on expected return

Quadratic Programming
Minimize portfolio variance

Risk = variance

Long-only Constraint
Minimize portfolio variance

Forced to accept lower return

Quadratic Programming Problems

Maximize

$$Z = cx - \frac{1}{2} x^T Q x$$

Subject to constraints:

$$Ax <= B$$

$$\cancel{x >= 0}$$

Here $c = 0$, $Q = Cov(Y)$, $x = W^T$

Also, relax the non-negativity constraint to allow short selling

Quadratic Programming Problems

Maximize

$$Z = cx - \frac{1}{2} x^T Q x$$

Subject to constraints:

$$Ax <= B$$

$$x >= 0$$

Re-impose the constraint on short-selling - the optimal return will reduce

Summary

Use historical data of stocks to estimate risk and return of a stock portfolio

Initially define risk as a linear function

Formulate an LPP in the standard form

Re-define risk as portfolio variance, creating a quadratic optimization

Turn on and off non-negativity constraints

Use Python to solve all of these optimization problems