Lecture 14

So far we have only dealt with constrained optimization problems where the objective and the constraints are linear. We now turn attention to general problems of the form

$$\begin{array}{ll} \text{minimize} & f(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{0} \\ & \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0}, \end{array} \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{g} = (g_1, \ldots, g_m)^\top$, $\boldsymbol{g} = (g_1, \ldots, g_\ell)$, and the inequalities are componentwise. The problem (1) is *convex*, if f and the g_i are convex, and the h_j are linear. We also denote by $\mathrm{dom}(f)$ the *domain* of f, which is the set of points $\boldsymbol{x}$ where f takes a finite value. The feasible set

$$\mathcal{F} = \{\boldsymbol{x} : g_i(\boldsymbol{x}) \leq 0,\ h_j(\boldsymbol{x}) = 0,\ 1 \leq i \leq m,\ 1 \leq j \leq \ell\}$$

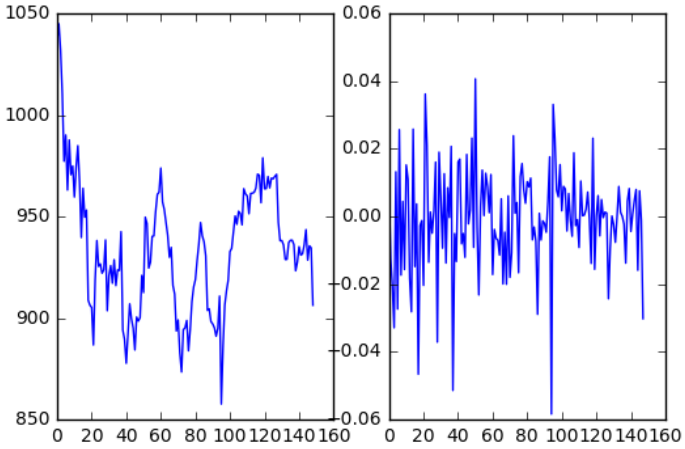
for a convex constrained problem is a convex set.

14.1 Quadratic Programming and Portfolio Optimization

The simplest case of non-linear, constrained convex optimization is **quadratic programming**. Quadratic programming problems are of the form

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{Q}\boldsymbol{x} + \boldsymbol{c}^\top \boldsymbol{x} \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \end{array}$$

with $\boldsymbol{Q}$ symmetric and positive semidefinite. Note that this problem combines two problems we studied in some detail before: minimizing a quadratic function, and linear constraints. Just as we did for unconstrained minimization and for linear programming, we will derive optimality conditions for such problems. Before studying the theory, we first present an important application: portfolio optimization.



Mean-variance portfolio theory

If we invest an amount x^0 into a product at period 0, and at period 1 (for example, one day) the value is x^1, then the **relative return** is defined as

$$r = \frac{x^1 - x^0}{x^0}.$$

The following figure shows the price movement and the returns of a stock from January 2016 until now.

As we can't predict the future, we usually work with the **expected return** $\boldsymbol{E}[r] = \mu$, which is a statistical estimate of the future return. One naive method of estimating the future return is by taking the average of past returns, but other more sophisticated methods are possible.

In **portfolio optimization**, we have a proportion x_i of our available funds that we want to invest in a stock i (in particular, as x_i measures a proportion, we have $\sum_{i=1}^n x_i = 1$). We may or may not allow $x_i < 0$, which would correspond to short-selling or borrowing. Given this allocation, the overall return is $\boldsymbol{x}^\top \boldsymbol{r}$, where $\boldsymbol{x} = (x_1, \ldots, x_n)^\top$ and $\boldsymbol{r} = (r_1, \ldots, r_n)^\top$ is the vector of (relative) returns. If $\boldsymbol{\mu} = \boldsymbol{E}[\boldsymbol{r}]$ denotes the vector of expected returns, then the total expected return is

$$\mu = \boldsymbol{x}^\top \boldsymbol{\mu} = \sum_{i=1}^n x_i \mu_i.$$

The **risk** is an investment is measured in terms of the **variance** of the returns $r = \boldsymbol{x}^\top \boldsymbol{r}$. Let $\boldsymbol{\Sigma}$ denote the **covariance matrix**, where the (i,j)-th entry is $\mathrm{Cov}(r_i, r_j) = \boldsymbol{E}[(r_i - \mu_i)(r_j - \mu_j)]$. The (i,j)-th entry measures how much products i and j are correlated. The **variance** of the returns r is then the quadratic function

$$\boldsymbol{x}^\top \Sigma \boldsymbol{x}.$$

A portfolio optimization problem either seeks to maximize the return while bounding the risk,

$$\begin{aligned} \text{maximize} \quad & \boldsymbol{x}^\top \boldsymbol{\mu} \\ \text{subject to} \quad & \boldsymbol{x}^\top \boldsymbol{\Sigma} \boldsymbol{x} \leq \sigma, \\ & \sum_{i=1}^{n} x_i = 1 \\ & x_i \geq 0, \end{aligned}$$

or minimize the risk given a certain target return μ,

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{x}^\top \boldsymbol{\Sigma} \boldsymbol{x} \\ \text{subject to} \quad & \boldsymbol{x}^\top \boldsymbol{\mu} = \mu \\ & \sum_{i=1}^{n} x_i = 1 \\ & x_i \geq 0. \end{aligned}$$

Both of these problems are convex optimization problems. The constraints $x_i \geq 0$ mean that we are not allowed to short-sell; when dealing with futures or options, or if we are a large institutional investor, we may drop these constraints. As we will see later, dropping the inequality constraints from the second formulation allows the problem to be solved in closed form.

A third form of the portfolio optimization problem is to combine the expected return and the risk into a single objective function, as follows:

$$\begin{aligned} \text{maximize} \quad & \boldsymbol{\mu}^\top \boldsymbol{x} - \gamma \boldsymbol{x}^\top \boldsymbol{\Sigma} \boldsymbol{x} \\ \text{subject to} \quad & \sum_{i=1}^{n} x_i = 1 \\ & x_i \geq 0. \end{aligned}$$

Note that this is a convex quadratic problem: we can transform it into a minimization problem with the matrix $\boldsymbol{\Sigma}$ by changing the sign. The parameter γ is called a **risk aversion parameter**; it adjusts the level of risk we are willing to take. If $\gamma = 0$, then the quadratic term does not feature in the objective and we just aim to maximize the expected return. If γ is big, then the risk term weighs heavily on the objective function, and the optimal value will like be one where $\boldsymbol{x}^\top \boldsymbol{\Sigma} \boldsymbol{x}$ is small. By varying the value of γ, we get different expected return / risk (variance) trade-offs. The following code computes this trade-off curve for a portfolio of 10 stock from the FT100 index. The mean and covariance where computing by averaging over the past 60 trading days (3 months). The x-axis is the **standard deviation**, which is given as the square root of the variance (risk).

In [1]:
```
from datetime import datetime, date
import pandas as pd
from pandas_datareader import data, wb
import numpy as np
import matplotlib.pyplot as plt
from cvxpy import *
```

We first use the functionality of the Pandas module to load the price data from the Yahoo Finance web site.

In [2]:
```
START = datetime(2016,1,1)
END = date.today()
TICKER = ['ADN', 'AZN', 'EZJ', 'GSK', 'ITV', 'LSE', 'TSCO', 'PSON', 'PRU', 'DGE']
mydata = data.DataReader('TSCO', "yahoo", START, END)
dates = mydata.index
df = pd.DataFrame(index=dates, columns=TICKER)
for x in TICKER:
    mydata = data.DataReader(x, "yahoo", START, END)
    df.loc[:,x] = mydata['Adj Close']
df = df.dropna()
df.head(2)
```

The following shows a small sample of the data loaded.

	ADN	AZN	EZJ	GSK	ITV	LSE	TSCO
Date							
2016-01-04	2.41197	31.926706	84.900002	37.813187	0.4	0.002	82.800593
2016-01-05	2.41197	32.404650	86.620003	38.028194	0.4	0.001	82.741275

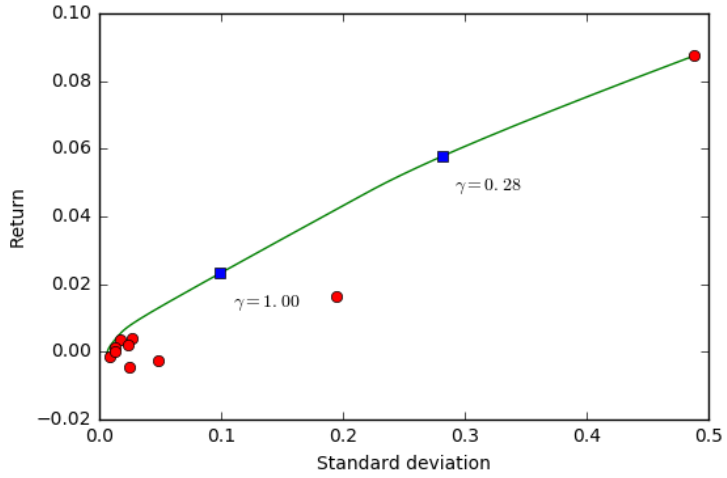
We next compute the returns and the mean and covariance.

In [3]:
```
price_matrix = df.values
returns = (price_matrix[1:]-price_matrix[0:-1])/price_matrix[0:-1]
mu = np.mean(returns[-60:,:], axis=0)
Sigma = np.cov(returns[-60:,:].T)
n = 10
```

In the next step, we start the CVXPY engine and compute the mean-variance trade-off curve. We include the constraint $\boldsymbol{x} \geq 0$ to disallow for short-selling.

In [4]:
```
x = Variable(n)
gamma = Parameter(sign='positive')
ret = mu.T*x
risk = quad_form(x, Sigma)
prob = Problem(Maximize(ret - gamma*risk),
               [sum_entries(x) == 1,
                x >= 0])
```

In [5]:
```
SAMPLES = 1000
risk_data = np.zeros(SAMPLES)
ret_data = np.zeros(SAMPLES)
gamma_vals = np.logspace(-2, 3, num=SAMPLES)
for i in range(SAMPLES):
    gamma.value = gamma_vals[i]
    prob.solve()
    risk_data[i] = sqrt(risk).value
    ret_data[i] = ret.value
```



In [6]:
```
markers_on = [290, 400]
fig = plt.figure()
ax = fig.add_subplot(111)
plt.plot(risk_data, ret_data, 'g-')
for marker in markers_on:
    plt.plot(risk_data[marker], ret_data[marker], 'bs')
    ax.annotate(r"$\gamma = %.2f$"
    % gamma_vals[marker], xy=(risk_data[marker]+0.01, ret_data[marker]-0.01))
for i in range(n):
    plt.plot(sqrt(Sigma[i,i]).value, mu[i], 'ro')
plt.xlabel('Standard deviation')
plt.ylabel('Return')
plt.show()
```

The red dots represent the standard deviation and expected return of the individual stocks. The two blue dots indicate the standard deviation and expected return of the portfolio for two values of γ. Which value of γ we go for depends on how much risk we are willing to take: if we are risk-averse, we may prefer the $\gamma = 1$ value to the $\gamma = 0.028$ value, at the expense of smaller expected returns.

In practical applications there are a lot of other factors to be considered, such as whether the estimation procedure for the mean and covariance makes sense. In addition, it is often common to add terms that account for **transaction costs** into the objective funtion.