FE 630 HW1

Utility function

2.1 Log Utility.

(a) $\ln(CE) = \frac{1}{2}\ln(1150) + \frac{1}{2}\ln(850)$

$= \ln(1150 \times 850)^{\frac{1}{2}} = \ln(988.69)$

$\Rightarrow CE = 988.69$

$E(W) = \frac{1}{2} \times 1150 + \frac{1}{2} \times 850 = 1000$

$\Rightarrow$ Risk Premium $= 1000 - 988.69 = 11.31$

Taylor serious expansion:

$y = \frac{1}{2} ARA(W_T^*) \cdot \sigma_x^2$, $ARA(x) = \frac{1}{x}$

$= \frac{1}{2} \times \frac{1}{1000} \times (150)^2 = 11.25$

The approximation is quite close to the real value

(b) initial wealth $= 2000$

$\ln(CE) = \frac{1}{2}\ln(2150) + \frac{1}{2}\ln(1850)$

$= \ln(1994.37)$

$\Rightarrow CE = 1994.37$

$\Rightarrow$ Risk Premium $= E(W) - CE = 5.63$

(c) $\ln(CE) = \frac{1}{2}\ln(1300) + \frac{1}{2}\ln(700)$

$= \ln(953.94)$

$\Rightarrow CE = 953.54.$

Risk Premium $= E(W) - CE = 1000 - 953.54$

$= 46.06$

2.2. CE and RP for a Power Utility

1) $ARA = -\dfrac{U''(W)}{U'(W)}$

$$= -\frac{k(k-1)W^{(k-2)}}{kW^{(k-1)}} = -\frac{k-1}{W} = \frac{1-k}{W}$$

① investor is risk-taker if $ARA < 0 \iff k > 1$

② investor is risk-neutral if $ARA = 0 \iff k = 1$

③ Investor is risk-averse if $ARA > 0 \iff k < 1$

2) $u(W) = W^{\frac{1}{2}}$

$$W = \begin{cases} 1250, & p = \frac{2}{3} \\ 600, & p = \frac{1}{3} \end{cases} \qquad W^* = E(W) = \frac{3100}{3} \approx 1033.33$$

$U(1250) = 25\sqrt{2}$ $\quad U(600) = 10\sqrt{6}$

$U(CE) = 25\sqrt{2} \times \frac{2}{3} + 10\sqrt{6} \times \frac{1}{3} \approx 31.74$

$CE \approx 1007.12$

$\Rightarrow$ Risk Premium $= 1033.33 - 1007.12 = 26.21$

$$y = \frac{1}{2} \cdot ARA(W^*) \cdot \sigma_x^2 = \frac{1}{2} \times \frac{1}{2 \times W^*} \cdot \left(\frac{2}{3} \cdot 250^2 + \frac{1}{3}(-400)^2\right)$$

$$= 22.98$$

3) $u(W) = W^2$ $\quad ARA = -\frac{1}{W}$ $\quad W^* = 1033.33$

$U(1250) = 1250^2$ $\quad U(600) = 600^2$

$U(CE) = 1250^2 \times \frac{2}{3} + 600^2 \times \frac{1}{3} = 1161666.67$

$CE = 1077.81$

Risk Premium $= 1033.33 - 1077.81 = -44.48$

$y = \frac{1}{2} \cdot ARA(W^*) \cdot \sigma_x^2 = -45.96$

2.3. Exponential Utility

(a) $ARA = -\frac{U''(r_p)}{U'(r_p)}$

$U(r_p) = -e^{-\lambda r_p}$ $\quad u'(r_p) = \lambda e^{-\lambda r_p}$

$u''(r_p) = -\lambda^2 e^{-\lambda r_p}$

$ARA = -\frac{u''(r_p)}{u'(r_p)} = \lambda > 0$

$\Rightarrow$ investor is Rise-averse

(b) $\mu_p(w) = \sum_{i=1}^{n} w_i \mu_i = w^T \mu.$

$\sigma_p^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} w_i \cdot \sigma_{ij} \cdot w_j = w^T \Sigma w$

$E[U(r_p(w))] = \lambda \cdot \mu_p - \frac{\lambda^2}{2} \cdot \sigma^2$

$= \lambda w \cdot \mu - \frac{\lambda^2}{2} w^T \Sigma w$

because λ is a constant, and r follows Normal Distribution

So maximize $E[U(r_p(w))]$ equals to

minimized $\frac{\lambda}{2} \cdot \sigma^2 = \frac{\lambda}{2} \cdot w^T \Sigma w$

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from numpy.linalg import inv
from cvxopt import matrix, solvers
from numpy import c_
import pandas_datareader as pdr
```

2.4 Numerical Application for Exponential Utility

In [2]:

```
#Download data
company = ['AAPL', 'GOOGL', 'FB', 'AMZN']
df = pd.DataFrame()

for i in company:
    df[i] = pdr.DataReader(i, data_source='yahoo', start='2020/09/01', end="2021/09/01")['Close'
]
```

In [3]:

```
df.head()
```

Out[3]:

	AAPL	GOOGL	FB	AMZN
Date				
2020-09-01	134.179993	1655.079956	295.440002	3499.120117
2020-09-02	131.399994	1717.390015	302.500000	3531.449951
2020-09-03	120.879997	1629.510010	291.119995	3368.000000
2020-09-04	120.959999	1581.209961	282.730011	3294.620117
2020-09-08	112.820000	1523.599976	271.160004	3149.840088

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 253 entries, 2020-09-01 to 2021-09-01
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   AAPL    253 non-null    float64
 1   GOOGL   253 non-null    float64
 2   FB      253 non-null    float64
 3   AMZN    253 non-null    float64
dtypes: float64(4)
memory usage: 9.9 KB
```

Compute Expect Return and Variance

In [5]:

```
returns = np.log(df / df.shift(1)) #compute log return
returns.fillna(value=0, inplace=True)
returns.head()
```

Out[5]:

Date	AAPL	GOOGL	FB	AMZN
2020-09-01	0.000000	0.000000	0.000000	0.000000
2020-09-02	-0.020936	0.036956	0.023615	0.009197
2020-09-03	-0.083448	-0.052526	-0.038346	-0.047389
2020-09-04	0.000662	-0.030089	-0.029243	-0.022028
2020-09-08	-0.069666	-0.037114	-0.041783	-0.044939

In [6]:

```
means = returns.mean() * 252
covariance = returns.cov() *252
```

In [7]:

```
print(means)
print(covariance)
```

```
AAPL      0.127542
GOOGL     0.560124
FB        0.256070
AMZN     -0.005744
dtype: float64
           AAPL     GOOGL        FB      AMZN
AAPL   0.097447  0.047372  0.061481  0.062521
GOOGL  0.047372  0.070965  0.058383  0.046372
FB     0.061481  0.058383  0.102010  0.061422
AMZN   0.062521  0.046372  0.061422  0.083245
```

In [8]:

```
# transfor to matrix
mean_list = np.array(means)
mean_matrix = matrix(means)
cov_list = np.array(covariance)
cov_matrix = matrix(cov_list)
```

In [9]:

```
la = np.linspace(0,0.5,501)
q = matrix(np.zeros((4,1)))
sol = []
A = matrix(np.c_[np.ones(4), mean_matrix]).T
risk = np.linspace(0,0,501)
exp_return = np.linspace(0,0,501)
```

In [10]:

```
for i in range(len(la)):
    b = matrix(np.c_[np.ones(1),la[i]]).T
    sol.append(solvers.qp(cov_matrix,q,A=A,b=b)['x'])
    risk[i] = np.sqrt(sol[i].T*cov_matrix*sol[i])
    exp_return[i]=np.matmul(sol[i].T,mean_matrix)
```

In [11]:

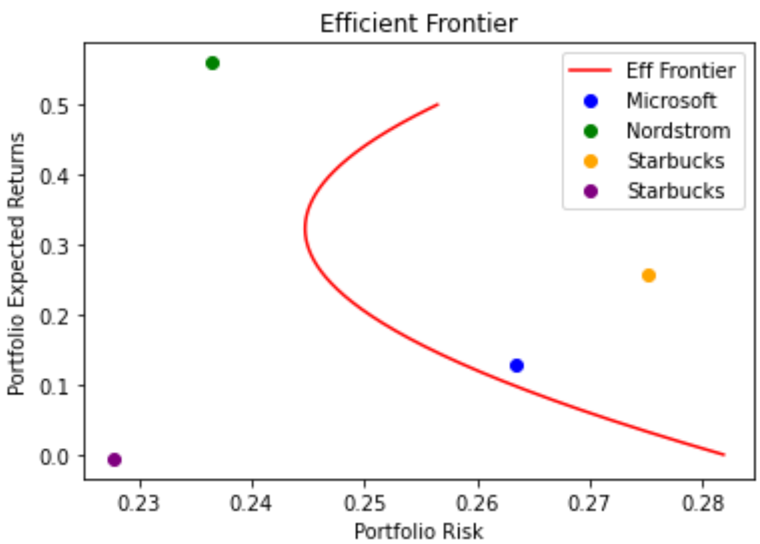
```
#print(exp_return)
```

In [12]:

```
#print(risk)
```

In [13]:

```
plt.xlabel("Portfolio Risk")
plt.ylabel("Portfolio Expected Returns")
plt.title("Efficient Frontier")
plt.plot(risk,exp_return,color='red',label="Eff Frontier")
plt.plot([np.sqrt(6.94e-02)], mean_matrix[0], 'ro', color='blue',label='Microsoft')
plt.plot([np.sqrt(5.59e-02)], mean_matrix[1], 'ro', color='green',label='Nordstrom')
plt.plot([np.sqrt(7.57e-02)], mean_matrix[2], 'ro', color='orange',label='Starbucks')
plt.plot([np.sqrt(5.19e-02)], mean_matrix[3], 'ro', color='purple',label='Starbucks')
plt.legend()
plt.show()
```



3. Diversification by Equally Weighted Protfolio.

1.

$$\sigma_p^2(n) = \sum_{i=1}^{n} \frac{1}{n^2}\sigma_i^2 + \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\sigma_{ij}$$

2

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from numpy.linalg import inv
from cvxopt import matrix, solvers
from numpy import c_
import pandas_datareader as pdr
```

In [2]:

```
class plotfunction:
    def __init__(self):
        self.mu1_ = 5
        self.sigma1_ = 3
        self.mu2_ = 10
        self.sigma2_ = 7

    def function(self, a, b, p):
        mu = np.linspace(a, b, 500)
        w = (mu - self.mu2_) / (self.mu1_ - self.mu2_)
        sigma =np.sqrt(w ** 2 * self.sigma1_ ** 2 + (1 - w) ** 2 * self.sigma2_ ** 2
                       + 2 * w * (1 - w) * self.sigma1_ * self.sigma2_ * p)
        plt.plot(sigma, mu)
        plt.xlabel("sigma")
        plt.ylabel("mu")
        plt.title('p = ' + str(p))

k = plotfunction()
for p in [1,0,-1]:
    k.function(0.0, 0.5, p)
    plt.show()
```