Optimal_Portfolio

September 29, 2021

1 Constructing an Optimal Portfolio

[1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.optimize import fmin
import math

import warnings
warnings.filterwarnings("ignore")

# fix_yahoo_finance is used to fetch data
import fix_yahoo_finance as yf
yf.pdr_override()
```

[2]:

```
# input
symbols = ['BAC','AAPL', 'JNJ']
start = '2012-01-01'
end = '2019-01-01'

rf = 0.003
```

[3]:

```
def annual_returns(symbols, start, end):
    df = yf.download(symbols,start,end)['Adj Close']
    log_rets = np.log(df) - np.log(df.shift(1))
    date = []
    d0 = df.index
    for i in range(0, len(log_rets)):
        date.append(d0[i].strftime("%Y"))
    y = pd.DataFrame(log_rets, date, columns = [symbols])
    return np.exp(y.groupby(y.index).sum()) - 1
```

[4]:

```
def portfolio_var(M, W):
    cor = np.corrcoef(M.T)
    vol = np.std(M, axis=0)
    var = 0.0
    for i in range(n):
        for j in range(n):
            var += W[i] * W[j] * vol[i] * vol[j] * cor[i, j]
    return var
```

[5]:
```
def sharpe(M, W):
    var = portfolio_var(M, W)
    mean_return = np.mean(M, axis=0)
    ret = np.array(mean_return)
    return (np.dot(W, ret) - rf)/ np.sqrt(252)
```

[6]:
```
def negative_sharpe_n_minus_1_stock(W):
    w2 = np.append(W, 1-sum(W))
    return -sharpe(M, w2)
```

[7]:
```
n = len(symbols)
x2 = annual_returns(symbols[0], start, end)
for i in range(1,n):
    x_ = annual_returns(symbols[i], start, end)
    x2 = pd.merge(x2, x_, left_index=True, right_index=True)

M = np.array(x2)
```

```
[*********************100%***********************]  1 of 1 downloaded
[*********************100%***********************]  1 of 1 downloaded
[*********************100%***********************]  1 of 1 downloaded
```

[8]:
```
print('Efficient Portfolio (Mean-Variance)')
print('Symbols: ', symbols)
print('Sharpe ratio for an equal-weighted portfolio')
equal_weighted = np.ones(n, dtype=float) * 1.0/n
print(equal_weighted)
print(round(sharpe(M, equal_weighted), 4))
```

```
Efficient Portfolio (Mean-Variance)
Symbols:  ['BAC', 'AAPL', 'JNJ']
Sharpe ratio for an equal-weighted portfolio
[0.33333333 0.33333333 0.33333333]
-0.0002
```

[15]:
```
w0 = np.ones(n-1, dtype=float) * 1.0 / n
w1 = fmin(negative_sharpe_n_minus_1_stock, w0)

final_weight = np.append(w1, 1 - sum(w1))
final_sharpe = sharpe(M, final_weight)

print('Optimal weights:')
```

```python
print(final_weight)
print('Sharpe ratio:')
print(round(final_sharpe,4))
```

```
Optimization terminated successfully.
         Current function value: 0.000189
         Iterations: 9
         Function evaluations: 35
Optimal weights:
[0.33333333 0.33333333 0.33333333]
Sharpe ratio:
-0.0002
```