HW5_Saranpat

October 4, 2023

Saranpat Prasertthum: sp73@illinois.edu

[ ]:
```
import pandas as pd
import numpy as np
```

1) Use the data to estimate the mean return and covariance matrix.

[ ]:
```
data = pd.read_excel('DataforHomework5.xlsx',index_col='Year')
data.head()
```

[ ]:
```
      Stock  Treasury Bond  Money Market     NASDAQ
Year
1961  26.81           2.20          2.33  31.664780
1962  -8.78           5.72          2.93 -15.024354
1963  22.69           1.79          3.38  20.445586
1964  16.36           3.71          3.85  23.118500
1965  12.36           0.93          4.32  17.152602
```

[ ]:
```
expected_returns = data.mean()
expected_returns
```

[ ]:
```
Stock            12.044186
Treasury Bond     7.792326
Money Market      6.323023
NASDAQ           12.899098
dtype: float64
```

[ ]:
```
cov_matrix = data.cov()
cov_matrix
```

[ ]:
```
                    Stock  Treasury Bond  Money Market      NASDAQ
Stock          283.919768      38.850792      2.092916  357.149248
Treasury Bond   38.850792     114.793828     -2.448836   -6.498260
Money Market     2.092916      -2.448836     11.814812   -4.392481
NASDAQ         357.149248      -6.498260     -4.392481  649.448769
```

2) Let risk -free return be 3% solve a nonlinear optimization model to construct a portfoli of these four assets to maximize the Sharpe Ratio.

[ ]:
```python
from scipy.optimize import minimize
# using trick

# minimize y_t @ cov_matrix @ y
def objective(params):
    k = params[0]
    y = params[1:]
    return y @ cov_matrix @ y

# Constraint 1: Sum of y equals k
def constraint1(params):
    k = params[0]
    y = params[1:]
    return np.sum(y) - k

# Constraint 2: Weighted sum of (expected_returns - rf) @ y equals 1
def constraint2(params):
    k = params[0]
    y = params[1:]
    return np.sum((expected_returns - 0.03) * y) - 1

cons = (
    {'type': 'eq', 'fun': constraint1},
    {'type': 'eq', 'fun': constraint2}
)

# Initial guess
initial_y = np.array([0.25, 0.25, 0.25, 0.25])
initial_guess = [1] + list(initial_y)  # 1 for initial k and initial_y values␣
 ↪for y

# Bounds for k and y. k
k_bounds = (0, None)
y_bounds = [(0, None) for _ in initial_y]
all_bounds = [k_bounds] + y_bounds

result = minimize(objective, initial_guess, constraints=cons, bounds=all_bounds)

k_opt = result.x[0]
y_opt = result.x[1:]
opt_x = y_opt/sum(y_opt)

print("Optimal k:", k_opt)
print("Optimal y:", y_opt)
print("Optimal x:", opt_x)
print(f"Optimal Sharpe ratio is {1/np.sqrt(y_opt @ cov_matrix @ y_opt)}")
```

```
Optimal k: 0.14894533186969777
Optimal y: [5.11731844e-17 1.81559434e-02 1.25313935e-01 5.47545351e-03]
Optimal x: [3.43570247e-16 1.21896693e-01 8.41341809e-01 3.67614979e-02]
Optimal Sharpe ratio is 2.1110806469204175
```

[ ]:
```
print("the optimal portfolio of these four assets is")
for i in range(len(opt_x)):
    print(f"Optimal weight for {expected_returns.index[i]} is {opt_x[i]*100:.
↪2f} %")
```

```
the optimal portfolio of these four assets is
Optimal weight for Stock is 0.00 %
Optimal weight for Treasury Bond is 12.19 %
Optimal weight for Money Market is 84.13 %
Optimal weight for NASDAQ is 3.68 %
```

3) Construct the portfolio of these four assets to minimize the MAD unter the condition that mean reaturn of the porfolio is at least 9%

[ ]:
```
returns = data
y_minus_z = returns
for col in expected_returns.index:
    y_minus_z[col] = y_minus_z[col] - expected_returns[col]
y_minus_z.head()
```

[ ]:

Year	Stock	Treasury Bond	Money Market	NASDAQ
1961	14.765814	-5.592326	-3.993023	18.765682
1962	-20.824186	-2.072326	-3.393023	-27.923452
1963	10.645814	-6.002326	-2.943023	7.546487
1964	4.315814	-4.082326	-2.473023	10.219401
1965	0.315814	-6.862326	-2.003023	4.253503

0.0.1 Converting to LP using techinique on the slide

[ ]:
```
import cvxpy as cp
import numpy as np

n = len(returns)
y = cp.Variable(n, nonneg=True)  # y_t >= 0
z = cp.Variable(n, nonneg=True)  # z_t >= 0
x = cp.Variable(len(expected_returns))

# Define the constraints
constraints = [
    y - z == y_minus_z.values @ x,
    x @ expected_returns.values >= 0.09,
    cp.sum(x) == 1
```

```
]

objective = cp.Minimize(cp.sum(y + z))

problem = cp.Problem(objective, constraints)

problem.solve()

print("Optimal value:", problem.value)
print("Optimal y:", y.value)
print("Optimal z:", z.value)
print("Optimal x:", .xvalue)
```

```
Optimal value: 99.05460089338462
Optimal y: [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 1.13414150e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 1.73612254e+00 1.54443522e-01 0.00000000e+00 0.00000000e+00
 0.00000000e+00 2.88204870e+00 6.86767886e+00 1.18943332e+01
 4.72842090e+00 4.31019577e+00 2.94355487e+00 1.62609154e+00
 3.59436149e+00 1.20297053e+00 0.00000000e+00 2.33136350e+00
 2.93256441e+00 0.00000000e+00 3.53977629e-10 0.00000000e+00
 0.00000000e+00 0.00000000e+00 1.16742367e+00 0.00000000e+00
 2.15854966e-02 0.00000000e+00 8.72107081e-10 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00]
Optimal z: [3.24638173e+00 4.16076515e+00 2.60380988e+00 2.16019404e+00
 2.04744716e+00 1.74209767e+00 1.62625618e+00 2.87854037e-01
 0.00000000e+00 1.42356843e+00 1.52072039e+00 9.26698890e-01
 0.00000000e+00 0.00000000e+00 3.02669536e-01 6.99298451e-01
 5.94861214e-01 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 5.56523599e-01 0.00000000e+00
 0.00000000e+00 3.48940047e-09 0.00000000e+00 3.17610809e+00
 2.85397284e+00 1.99146385e+00 0.00000000e+00 8.75758417e-01
 0.00000000e+00 3.62432146e-01 0.00000000e+00 9.59489190e-01
 5.36747290e+00 5.88297192e+00 4.15848471e+00]
Optimal x: [0.01962577 0.04389065 0.91676895 0.01971462]
```

[ ]:
```
print("the portfolio of these four assets to minimize the MAD unter the␣
↪condition that mean reaturn of the porfolio is at least 9%. You most hold␣
↪the following poportion of this")
for i in range(len(x.value)):
    print(f"Optimal weight for {expected_returns.index[i]} is {x.value[i]*100:.
↪2f} %")
```

```
the portfolio of these four assets to minimize the MAD unter the condition that
```

```
mean reaturn of the porfolio is at least 9%. You most hold the following
poportion of this
Optimal weight for Stock is 1.96 %
Optimal weight for Treasury Bond is 4.39 %
Optimal weight for Money Market is 91.68 %
Optimal weight for NASDAQ is 1.97 %
```