Nirbhay Sharma (B19CSE114)

Optimization for Machine Learning - Lab-3

Que-1

Code

```
import numpy as np
import pandas as pd
import cvxopt as cp
import json, sys

json_file = sys.argv[1]
with open(json_file, 'r') as jf:
    data = json.load(jf)

Q=np.array(data['Q'])
c=np.array(data['c'])
A=np.array(data['A'])
b = np.array(data['b'])
Aeq = np.array(data['Aeq'])
beq = np.array(data['beq'])

sol = cp.solvers.qp(
    cp.matrix(Q,tc="d"),
    cp.matrix(c,tc="d"),
    cp.matrix(A,tc="d"),
    cp.matrix(b,tc='d'),
    cp.matrix(Aeq,tc='d'),
    cp.matrix(beq,tc='d')
)

print(sol["x"])
print(sol['primal objective'] + data['const'])

"""
Optimal solution found.
[ 5.00e-01]
[ 5.67e-04]
[ 5.00e-01]

0.5000001605670136
"""
```

Json File

```
{
    "Q":[[2,1,0],[1,2,1],[0,1,2]],
    "c":[[0],[0],[0]],
    "A":[[-0.4,-0.4,-0.8],[-1,0,0],[0,-1,0],[0,0,-1]],
    "b":[[-0.14],[0],[0],[0]],
    "Aeq":[[1,1,1]],
    "beq":[[1]],
    "const":0
}
```

Que-2

code

```
import numpy as np
import pandas as pd
import cvxopt as cp
import json, sys

csv_file = "Two-Stocks.csv"
# csv_file = sys.argv[1]
data = pd.read_csv(csv_file)

np_data = np.array(data)[:50,1:]

def mean_return(np_data: np.array):
    # arr_1 = np.log(np_data[-1,:])
    # arr_0 = np.log(np_data[0,:])
    # return arr_1 - arr_0
    # return np.array(log_mean)

    log_mean = []
    ht, wd = np_data.shape
    for i in range(wd):
        stock_i = np_data[:,i]
        mean_i = 0
        for j in range(ht-1):
            ratio_return = stock_i[j+1]/stock_i[j]
            mean_i += np.log(ratio_return)
        log_mean.append(mean_i)
    return -np.array(log_mean)

def cov_matrix(np_data:np.array):
    np_data = np_data - np.mean(np_data,axis=0)
    return np.dot(np_data.T,np_data) / (np_data.shape[0] -1)

Q= cov_matrix(np_data)
cov_matrix_shape = Q.shape[0]
c= np.zeros((cov_matrix_shape, 1))
A = np.vstack([mean_return(np_data), -np.eye(cov_matrix_shape)])
b = np.array([[-0.014]]+[[0] for _ in range(cov_matrix_shape)])
```

```
Aeq = np.array([[1 for _ in range(cov_matrix_shape)]])
beq = np.array([[1]])

sol = cp.solvers.qp(
    cp.matrix(Q,tc="d"),
    cp.matrix(c,tc="d"),
    cp.matrix(A,tc="d"),
    cp.matrix(b,tc='d'),
    cp.matrix(Aeq,tc='d'),
    cp.matrix(beq,tc='d'),
)

print(sol["x"])
print(sol['primal objective'])

"""
Two stocks
Optimal solution found.
[ 4.62e-08]
[ 1.00e+00]

76.67343667829941
"""
```

Que-3

code

```
import numpy as np
import pandas as pd
import cvxopt as cp
import json, sys

csv_file = "Four-Stocks.csv"
# csv_file = sys.argv[1]
data = pd.read_csv(csv_file)

np_data = np.array(data)[:50,1:]

def mean_return(np_data: np.array):
    # arr_1 = np.log(np_data[-1,:])
    # arr_0 = np.log(np_data[0,:])
    # return arr_1 - arr_0
    # return np.array(log_mean)

    log_mean = []
    ht, wd = np_data.shape
    for i in range(wd):
        stock_i = np_data[:,i]
        mean_i = 0
        for j in range(ht-1):
```

```
            ratio_return = stock_i[j+1]/stock_i[j]
            mean_i += np.log(ratio_return)
        log_mean.append(mean_i)
    return -np.array(log_mean)

def cov_matrix(np_data:np.array):
    np_data = np_data - np.mean(np_data,axis=0)
    return np.dot(np_data.T,np_data) / (np_data.shape[0] -1)

Q= cov_matrix(np_data)
cov_matrix_shape = Q.shape[0]
c= np.zeros((cov_matrix_shape, 1))
A = np.vstack([mean_return(np_data), -np.eye(cov_matrix_shape)])
b = np.array([[-0.014]]+[[0] for _ in range(cov_matrix_shape)])
Aeq = np.array([[1 for _ in range(cov_matrix_shape)]])
beq = np.array([[1]])

sol = cp.solvers.qp(
    cp.matrix(Q,tc="d"),
    cp.matrix(c,tc="d"),
    cp.matrix(A,tc="d"),
    cp.matrix(b,tc='d'),
    cp.matrix(Aeq,tc='d'),
    cp.matrix(beq,tc='d'),
)

print(sol["x"])
print(sol['primal objective'])

"""
Four stocks
Optimal solution found.
[ 3.46e-01]
[ 5.91e-10]
[ 4.49e-10]
[ 6.54e-01]

59.33659417980698
"""
```

Homework

Dataset is also attached with the code

Code

```
import numpy as np
import pandas as pd
import cvxopt as cp
import json, sys
```

```
csv_file = "homework_data.csv"
# csv_file = sys.argv[1]
data = pd.read_csv(csv_file)

np_data = np.array(data)

def mean_return(np_data: np.array):
    # arr_1 = np.log(np_data[-1,:])
    # arr_0 = np.log(np_data[0,:])
    # return arr_1 - arr_0
    # return np.array(log_mean)

    log_mean = []
    ht, wd = np_data.shape
    for i in range(wd):
        stock_i = np_data[:,i]
        mean_i = 0
        for j in range(ht-1):
            ratio_return = stock_i[j+1]/stock_i[j]
            mean_i += np.log(ratio_return)
        log_mean.append(mean_i)
    return -np.array(log_mean)

def cov_matrix(np_data:np.array):
    np_data = np_data - np.mean(np_data,axis=0)
    return np.dot(np_data.T,np_data) / (np_data.shape[0] -1)

Q= cov_matrix(np_data)
cov_matrix_shape = Q.shape[0]
c= np.zeros((cov_matrix_shape, 1))
A = np.vstack([mean_return(np_data), -np.eye(cov_matrix_shape)])
b = np.array([[-0.014]]+[[0] for _ in range(cov_matrix_shape)])
Aeq = np.array([[1 for _ in range(cov_matrix_shape)]])
beq = np.array([[1]])

sol = cp.solvers.qp(
    cp.matrix(Q,tc="d"),
    cp.matrix(c,tc="d"),
    cp.matrix(A,tc="d"),
    cp.matrix(b,tc='d'),
    cp.matrix(Aeq,tc='d'),
    cp.matrix(beq,tc='d'),
)

print(sol["x"])
print(sol['primal objective'])

"""
homework
Optimal solution found.
[-1.07e-08]
[ 3.27e-09]
[-8.26e-09]
[-1.81e-09]
```

```
[-4.65e-09]
[ 1.66e-07]
[-1.82e-09]
[-1.43e-09]
[ 4.14e-01]
[ 5.86e-01]

459.5288989816814
"""
```