Name: Yuvaraj Vivekanadan

Student ID - 200252347

Project 506 – Portfolio Selection Problem

1. Consider the following portfolio selection problem. The objective is to find an optimal portfolio of securities (stocks, bonds etc.). An investor wishes to invest a certain amount of money in securities S1, S2, · · · , Sn. Each security Si has an uncertain return, i.e., the return is a random variable. Suppose the expected value of Si is μi and its standard deviation is σi. the correlation of the returns of the securities Si and Sj is ρij. Denote the covariance matrix of the returns of the securities Si and Sj is given by Σij , Σij = ρijσiσj, i/= j and σii = σ2 i. Suppose xi is the proportion of the investor's total money that is invested in security Si. Setting x = (x1, · · ·, xn)T, E(x) =%n i=1 uixi var(x) = %n j=1 %n i=1 ρijσiσjxixj x1 + · · · + xn = 1 xi ≥ 0, i= 1, · · · , n. The aim is to find a portfolio x where the return is maximized, and the variance is minimized. Of course, these are competing goals.

Stock A B C D
ri1 r11 = 0.0063 r21 = 0.0066 r31 = 0.0107 r41 = 0.0234
ri2 r12 = 0.0015 r22 = 0.00762 r32 = −0.0684 r42 = −0.0753
ri3 0.01861 -0.0248 0.02876 0.08761
ri4 0.0356 -0.0551 0.0320 0.0315
ri5 0.1011 0.0112 0.1181 0.1039
ri6 0.0911 0.0019 - 0.01123 - 0.1009
ri7 0.0981 0.7891 -0.00121 - 0.0121
ri8 0.1009 0.0912 0.01231 0.0978
ri9 0.0670 0.0781 0.0791 0.0782
ri10 0.1819 0.0911 0.0812 0.1012

Solution:

The given multi-objective constrained problem is solved using **Exterior Penalty Method** using unfactored **BFGS nonlinear optimization** Method.

Output:

For the multiplying constant 10^ (4), we get the below set of observation:

Sr.No	Alpha1	Alpha2	X	Y	W	Z	Fun_value
1	0.1	0.9	0. 30581	0. 39909	-0.017504	0. 30457	-0. 00151
2	0.2	0.8	0. 23966	0. 53872	-0.007363	0. 21263	-0. 00604
3	0.3	0.7	0. 09326	0. 86995	-0.00569	0. 00464	-0. 00677
4	0.4	0.6	0. 0559	0. 61349	0.32791	-0. 00679	-0. 01906
5	0.5	0.5	0. 04932	0. 51691	0. 43363	-0. 00199	-0. 02553
6	0.6	0.4	0. 04324	0. 48136	0. 48187	-0. 00454	-0. 03137
7	0.7	0.3	0. 04281	0. 45813	0. 50743	-0. 00234	-0. 03709
8	0.8	0.2	0. 04275	0. 44338	0. 52425	-0. 00073	-0. 04275

9	0.9	0.1	0. 02564	0. 44615	0. 55701	-0. 01999	-0. 04842

Problem Analysis:

For the given set of data, we have 4 companies to choose the investment. The initial step in any portfolio selection problem is to find the rate of return of the stock over a period. In our example, we consider the returns each year from the stocks A, B, C and D. Using the return from each year, we find the rate of return for the corresponding year. A negative value in the rate of return implies that the return has reduced from the previous year.

Having calculated the rate of return table, we use this data for further analysis. At first, we find the geometric mean of the data. This geometric mean is used to find the expected return of the stock at the end of 10 years. The expected return of the stock is given by the product of the geometric mean of the stock and the percentile of the amount invested in the stock.

Secondly, we find the standard deviation of the population. The standard deviation is used to calculate the variance covariance matrix. One of our objective deals with minimizing the variance of the return given by the stock. In order to achieve that objective, we find the covariance and find the product of the proportions and the covariance matrix.

Once we have the data set, we dive deep into the problem. We have 2 primary objectives to accomplish.

1. Maximize the Expected return
2. Minimize the Variance of the Stock

Maximize the Expected return:

In order to achieve this primary objective, we multiply the geometric mean of each stock with their percentage of investment. We then sum all the individual data to get the cumulative expected return. In order to achieve the optimality, we can minimize the negative of cumulative return along with the second objective.

Note: For the given data, the geometric mean for each stock is given by

A	B	C	D
0.068908	0.080776	0.026878	0. 030986

Inference: Maximum Individual Return = Stock B

Minimize the Variance of the Stock:

The variance covariance matrix is calculated by the inbuilt MATLAB function. Using the inbuild function of the population covariance, we find the covariance of the data.

A	B	C	D
0.002826	0.003736	0.0014	0.001288
0.003736	0.054989	-0.00159	-0.00221
0.0014	-0.00159	0.002581	0.002852
0.001288	-0.00221	0.002852	0.005069

From this data, we formulate the second minimization function.

F2 = ½*X*Covariance*X';

Note: Stock C has the least variance.

Conversion of Multi Objective to Single Objective:

Using the weights alpha_1, alpha_2 we make the multi objective problem to single objective problem.

Objective function: alpha_1*(-Cumulative Expected Return) + alpha_2*(F2);

Where 0<= alpha_1, alpha_2 <= 1

Constraints:

1. Sum of all the proportions of the investment should be equal to 1.
   X + Y + Z + W = 1
2. Each individual proportion must be strictly greater than or equal to 1.

   X >= 0;

   Y >= 0;

   Z >= 0;

   W >= 0;

Using the Exterior penalty method, we change the value of multiplying constant in every iteration with the increasing factor of 10. Once it reaches 10^(6), we stop the iteration. The same is repeated for various alpha_1 and alpha_2 values and the result is tabulated.

Interpretation of the result:

Case 1:

When alpha1 = alpha2 = 0.5, meaning we have equal weightage to both the objectives.

Sr.No	Alpha1	Alpha2	X	Y	W	Z	Fun_value
5	0.5	0.5	0. 04932	0. 51691	0. 43363	-0. 00199	-0. 02553

In this scenario of optimization, we have the above solution. Here the solution is to invest in the stocks in the following proportions:

A – 4.9%

B – 51.69 %

C – 43.36 %

D – 0%

Since the stock B gives maximum return and Stock C gives minimum variance, we have a converged solution when we give equal importance to maximizing the return as well as minimizing the variance. This corresponds to the weights we have obtained as solution.

Case 2:

When we must maximize the return (alpha1 = 1, alpha2 = 0):

In this case, the user is more concerned about maximizing the expected return and least bothered about the variance in return of each year. Hence, we assign the maximum weight 1 to the objective 1 of our case.

Solution: By theory, we must invest in the stock that has maximum geometric mean over the period to have maximum expected return. Hence, we must invest only in stock B.

Case 3:

When we need to minimize the variance of return (alpha1 = 0, alpha2 = 1):

In this case, the user is more concerned about minimizing the variance of return of the stock than the expected return of the portfolio. Hence, we assign maximum weight of 1 to objective 2.

Source Code:

```
function [ output ] = Final_Project()
format short;
%We import the input data from excel
input_data = readtable('Final_Project.xlsx','Sheet','Sheet1','Range','B1:E11');
%In order to calculate the grometric mean of negative numbers, we need to add 1
and then calculate the GM
for i = 1:10
    a(i) = input_data.A(i) + 1;
end
```

```
for i = 1:10
    b(i) = input_data.B(i) + 1;
end
for i = 1:10
    c(i) = input_data.C(i) + 1;
end
for i = 1:10
    d(i) = input_data.D(i) + 1;
end
gm = [geomean(a)-1
    geomean(b)-1
    geomean(c)-1
    geomean(d)-1];
%Formation of the variable to hold the solution of weights
syms x y z w;
x_new = [x y z w];
% Negative of Maximum of Expected Return
F1 = -x_new*gm;
%Covariance matrix of the population
cov_matrix = [ input_data.A input_data.B input_data.C input_data.D];
cm = cov(cov_matrix,1);
%Second objective
F2 = 1/2*x_new*cm*x_new';
%Constraints
c1 = x + y + z + w -1;
output = Exterior_Penalty_Method(c1,F1,F2);
end
function output = Exterior_Penalty_Method(f1,fun1,fun2)
outputTable = table([],[],[],[],[],[],[],[],[],'VariableNames',
{'Iteration_Count','x','y','w','z','alpha1','alpha2','e','fun_value'});
i =0;
e = 10^i;
for i = 0:0.1:1
alpha1 = i;
fun = alpha1*fun1 + (1-alpha1)*fun2 + e*(f1)^2;
while (e<10^6)
    %BFGS to solve NLP
    [a b c d] = BFGS_2(fun,f1,e);
    x = a;
    y = b;
    z = c;
    w = d;
    fun_value = eval(fun);
    Iteration_Count = i+1;
    alpha2 = 1- alpha1;
    outputTable = [outputTable; table(
Iteration_Count,x,y,z,w,alpha1,alpha2,e,fun_value)];
    i = i+1;
    e = 10^i;
end
```

```
output = outputTable;
end
end
function [x_value y_value z_value w_value] =BFGS_2(fun,f1,c)
epsilon = 10^(-8);
% Starting point outside the feasible region
x = 1;
y = 1;
z = 1;
w = 1;
f1value = eval(f1);
if(f1value < 0)
    f1 = 0;
end
x_old = [x y z w]';
%Formation of function value based on
fun = fun + c*(f1^2 + x^2 + y^2 + x^2 + w^2);
grad_fun = gradient(fun);
H_old_value = [1 0 0 0
               0 1 0 0
               0 0 1 0
               0 0 0 1];
x = x_old(1);
y = x_old(2);
z = x_old(3);
w = x_old(4);
x_new = x_old;
grad_fun_new_value = eval(grad_fun);
fun_value = eval(fun);
for i = 1:1000
    criteria = norm(grad_fun_new_value)/(1+abs(fun_value));
    if(criteria < epsilon)
        break;
    end

    d_new = -H_old_value*grad_fun_new_value;
    x_old = x_new;
    dk = -grad_fun_new_value;
    alpha = LineSearch(x_old,fun,dk);
    x_new = double(x_old + (alpha*d_new));
    s_k = alpha*d_new;

    x = x_old(1);
    y = x_old(2);
    z = x_old(3);
    w = x_old(4);

    grad_fun_old_value = eval(grad_fun);

    x = x_new(1);
```

```
    y = x_new(2);
    z = x_new(3);
    w = x_new(4);

    grad_fun_new_value = eval(grad_fun);
    y_k = grad_fun_new_value - grad_fun_old_value;

    H_old_value = double(H_old_value + ((s_k*s_k') /(s_k'*y_k)) -
(H_old_value*y_k*y_k'*H_old_value)/(y_k'*H_old_value*y_k)) ;

    fun_value = eval(fun);
end
x_value = x_new(1);
y_value = x_new(2);
w_value = x_new(3);
z_value = x_new(4);
end
function output1 = LineSearch(x_old,fun,d)
x = x_old(1);
y = x_old(2);
z = x_old(3);
w = x_old(4);
f_old = eval(fun);
toggle = false;
for i = .01:.01:1
    x_new = x_old + i*d;
    x = x_new(1);
    y = x_new(2);
    z = x_new(3);
    w = x_new(4);
    f = eval(fun);

    if(length(x_new(x_new<0))>0)

        if(f>f_old && toggle)
            i = i-0.01;
            break;
        else
            i = 1;
            break;
        end

    end


    if(f<f_old)
        toggle = true;
    elseif(f>f_old && toggle)
        i = i-.01;
        break;
```

```
    end
    f_old=f;
end
if(i==1)
x = x_old(1);
y = x_old(2);
z = x_old(3);
w = x_old(4);
f_old = eval(fun);
toggle = false;
for i = .001:.001:.01
    x_new = x_old + i*d;
    x = x_new(1);
    y = x_new(2);
    z = x_new(3);
    w = x_new(4);
     f = eval(fun);

    if(length(x_new(x_new<0))>0)

        if(f>f_old && toggle)
            i = i-0.001;
            break;
        else
            i = 0;
            break;
        end

    end


    if(f<f_old)
        toggle = true;
    elseif(f>f_old && toggle)
        i = i-.001;
        break;
    end
    f_old=f;
end
if(i ~= 0.01 && i ~= 0)
    output1 = i;
else
    output1 = 0;
end
else
    output1 = i;
end
end
```