MIS 381N
Stochastic Control and Optimization Homework 3

Jushira Thelakkat (jt39634)

Problem 1

It costs a company $12 to purchase an hour of labor and $15 to purchase an hour of capital. If L hours of labor and K units of capital are available, then 0.05(L)^2/3(K)^1/3 machines can be produced. Suppose the company has $100,000 to purchase labor and capital.
What is the maximum number of machines it can produce?

Here we have 12L+15K = 100000
We want to maximize the number of machines produced. This can be done using the optim function on R.

```
> library(quadprog)
> func <- function(L){ # function for num of machines produced given a level of L
+   K = (100000 - 12*L)/15
+   num_machines = 0.05*L^(2/3)*K^(1/3)
+   return(-num_machines)
+ }
> S = optim(2000, func, method="BFGS")
> S
$par
[1] 5649.843

$value
[1] -204.6088

$counts
function gradient
       5        4

$convergence
[1] 0

$message
NULL
```

The maximum number of machines that can be produced is 204.

Problem 2

The file homework4stocks.csv contains historical monthly returns for 27 companies. The first row contains stock names and the first column contains the dates. For each company, calculate the estimated mean return and the estimated variance of return. Then calculate the estimated correlations between the companies' returns.
Find a portfolio that achieves an expected monthly return of at least 1% and minimizes portfolio variance. What are the fractions invested in each stock? What are the portfolio's estimated mean, variance, and standard deviation?

```
> Stocks <- read.csv("homework4stocks.csv")
> MeanReturns = colMeans(Stocks[,2:ncol(Stocks)])
> VarReturns = apply(Stocks[,2:ncol(Stocks)], 2, var)
> StdDev = sqrt(VarReturns)
> Correlation = cor(Stocks[,2:ncol(Stocks)], use="pairwise.complete.obs")
> CovMatrix = diag(StdDev) %*% Correlation %*% diag(StdDev)
> Dmat = 2*CovMatrix
> dvec = rep(0,27)
> Amat = matrix(c(rep(1,27),rep(-1,27),MeanReturns),27)
> bvec = c(1,-1,0.01)
> Sol=solve.QP(Dmat,dvec,Amat,bvec)
> Sol
$solution
 [1] -0.11973810  0.03867549 -0.07492591  0.16636830 -0.01034152  0.02541211 -0.07436094 -0.09669737 -0.03890179  0.01443292
[11] -0.11294606 -0.05983834  0.09727562  0.06822219  0.16038834  0.11682045 -0.07211667  0.07338738  0.02102473  0.01460696
[21]  0.22763626 -0.07466514 -0.02873011  0.13824440  0.20171402  0.01403427  0.38501850

$value
[1] 0.0004698333

$unconstrained.solution
 [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

$iterations
[1] 3 0

$Lagrangian
[1] 0.0007860884 0.0000000000 0.0153578157

$iact
[1] 1 3

> sum(Sol$solution * MeanReturns)
[1] 0.01
> sum(Sol$solution * VarReturns)
[1] -0.0001122938
> sum(Sol$solution * sqrt(VarReturns))
[1] 0.03455156
```

The mean return of the new portfolio is 1%, the variance is -0.0001122 (the minimized value) and the standard deviation is 0.0345. The fractions invested in each stock are given below

```
$solution
 [1] -0.11973810  0.03867549 -0.07492591  0.16636830 -0.01034152  0.02541211 -0.07436094 -0.09669737 -0.03890179  0.01443292
[11] -0.11294606 -0.05983834  0.09727562  0.06822219  0.16038834  0.11682045 -0.07211667  0.07338738  0.02102473  0.01460696
[21]  0.22763626 -0.07466514 -0.02873011  0.13824440  0.20171402  0.01403427  0.38501850
```

Problem 3

The file 'variable_selection.csv' contains observations of variables y, x1, x2, and x3. Here, y is the dependent variable. We want to choose a linear model that uses at most 2 independent variables such that the sum of squared residuals is minimized. This can be formulated as a constrained quadratic programming problem.

This is called best subset problem which is usually very hard to solve. We will solve this problem by enumeration. Run six OLS regressions (3 with one independent variable and 3 more with two variables each) and choose the regression that best fits the data.

It's just 6 linear regression. This is the toy problem to let you warm up for your project. In your project, you will do it in the systematic NLP way.

```
> variables <- read.csv("variable_selection.csv")
> lm1 = lm(variables$y ~ variables$x1)
> lm2 = lm(variables$y ~ variables$x2)
> lm3 = lm(variables$y ~ variables$x3)
> lm4 = lm(variables$y ~ variables$x1 + variables$x2)
> lm5 = lm(variables$y ~ variables$x2 + variables$x3)
> lm6 = lm(variables$y ~ variables$x3 + variables$x1)
> sum(resid(lm1)^2)
[1] 7901.299
> sum(resid(lm2)^2)
[1] 878.8358
> sum(resid(lm3)^2)
[1] 8575.636
> sum(resid(lm4)^2)
[1] 26.19087
> sum(resid(lm5)^2)
[1] 878.1811
> sum(resid(lm6)^2)
[1] 7860.089
```

We want the least sum of squared residuals and from the results above, we find that the fourth regression which includes both x1 and x2 has the least SSR.

Problem 4

In an electrical network, the power loss incurred when a current of I amperes flows through a resistance of R ohms is I2R watts. In the figure below, 710 amperes of current must be sent from node 1 to node 4. The current flowing through each node must satisfy conservation of flow. For example, for node 1, 710 = flow through 1-ohm resistor + flow through 4-ohm resistor. Remarkably, nature determines the current flow through each resistor by minimizing the total power loss in the network.

1. Formulate a quadratic programming problem whose solution will yield the current flowing through each resistor.

Choose x1, x3, x4, x6 and x12
The objective here is to minimize sum(xi2 * i)
Constraints:
x1 + x4 = 710; x6 + x12 = x1 –> x1 + x6 + x12 = 0; x3 = x4 + x6 –> x3 - x4 - x6 = 0

2. Use R to determine the current flowing through each resistor.

```
> Dmat = diag(5)
> Dmat[2,2] = 3
> Dmat[3,3] = 4
> Dmat[4,4] = 6
> Dmat[5,5] = 12
> dvec = rep(0,5)
> Amat = matrix(c(1,0,1,0,0,-1,0,0,1,1,0,1,-1,-1,0),5,3)
> bvec = c(710,0,0)
> S=solve.QP(Dmat,dvec,Amat,bvec)
> S
$solution
[1] 371.3846 502.4615 338.6154 163.8462 207.5385

$value
[1] 1015955

$unconstrained.solution
[1] 0 0 0 0 0

$iterations
[1] 4 0

$Lagrangian
[1] 2861.846 2490.462 1507.385

$iact
[1] 1 2 3

> S$solution
[1] 371.3846 502.4615 338.6154 163.8462 207.5385
> S$value
[1] 1015955
> S$unconstrained.solution
[1] 0 0 0 0 0
> S$iterations
[1] 4 0
> S$Lagrangian
[1] 2861.846 2490.462 1507.385
> S$iact
[1] 1 2 3
```

Therefore, from the results, we see that the current flowing through each resistor is shown below (in ohms)

```
> S$solution
[1] 371.3846 502.4615 338.6154 163.8462 207.5385
```

Problem 5:

The file nflratings.csv contains the results of 256 regular-season NFL games from the 2009 season. The teams are indexed 1 to 32 as shown below:

The csv data file contains a matrix with the following columns:

- Week (1-17)
- Home Team Index (1-32 from the table above)
- Visiting Team Index (1-32 from the table above)
- Home Team Score
- Visiting Team Score

Index	Team Name	Index	Team Name	Index	Team Name	Index	Team Name
1	Arizona Cardinals	9	Dallas Cowboys	17	Miami Dolphins	25	Pittsburgh Steelers
2	Atlanta Falcons	10	Denver Broncos	18	Minnesota Vikings	26	St. Louis Rams
3	Baltimore Ravens	11	Detroit Lions	19	New England Patriots	27	San Diego Chargers
4	Buffalo Bills	12	Green Bay Packers	20	New Orleans	28	San Francisco 49ers
5	Carolina Panthers	13	Houston Texans	21	New York Giants	29	Seattle Seahawks
6	Chicago Bears	14	Indianapolis Colts	22	New York Jets	30	Tampa Bay Buccaneers
7	Cincinnati Bengals	15	Jacksonville Jaguars	23	Oakland Raiders	31	Tennessee Titans
8	Cleveland Browns	16	Kansas City Chiefs	24	Philadelphia Eagles	32	Washington Redskins

You will need to calculate the following:

- Actual Point Spread = Home Team Score – Visiting Team Score
- Predicted Spread = Home Team Rating – Visitor Team Rating + Home Team Advantage
- Prediction error = Actual Point Spread – Predicted Point Spread

```
> nfl <- read.csv('nflratings.csv')
> head(nfl)
  X1 X25 X31 X13 X10
1  1   2  17  19   7
2  1  29  26  28   0
3  1  21  32  23  17
4  1   3  16  38  24
5  1  13  22   7  24
6  1  20  11  45  27
> nfl$actual_point_spread = nfl$X13 - nfl$X10
> func <- function(R){
+   pred_error = 0
+   for (i in 1:nrow(nfl)) {
+     pred_error = pred_error + (nfl$actual_point_spread[i] - (R[nfl[i,2]] - R[nfl[i,3]] + R[33]))^2
+   }
+   return(pred_error)
+ }
> S = optim(rep(0, 33), func, method="BFGS")
> Sol = S$par[1:32] + (85 - mean(S$par[1:32]))
> Sol = c(Sol, S$par[33])
> mean(Sol[1:32])
[1] 85
```

Below are the ratings for the 32 NFL teams.

```
> Sol
 [1] 84.48792 89.83315 92.80631 83.04926 88.75252 79.83926 87.60801 76.94714 92.13457 85.67620 70.53597 92.28773 86.92186
[14] 90.79928 78.37611 76.92849 86.58862 92.09555 96.08356 95.62021 85.11249 93.10867 75.07121 90.97112 86.90721 67.68808
[27] 92.63373 85.20902 74.69670 79.16269 81.90324 80.16414  2.18786
```

The last number is the home team advantage.