Electrical and Computer Engineering
Linear Programming (ECE 236A)
Homework 1

Noyan Evirgen
205220656

October 4, 2018

Problem 1

First we need to define parameters for the problem. Let us assume that days of a week starting from Monday are numerated as $i = 1, 2, ..., 5$ respectively.

We have 2 constant vectors whose values can be defined as:

- b_i = Buying price defined for the ith day.
- s_i = Selling price defined for the ith day.

We have 4 variables whose values can be defined as:

- m_i = Number of shares bought on ith day.
- n_i = Number of shares sold on ith day.
- x_i = Number of shares owned at the end of ith day.
- y_i = Amount of money owned at the end of ith day.

The objective is,

$$max\{y_5\} \tag{1.1}$$

such that,

$$\begin{aligned} y_0 &= 100 \\ x_0 &= 0 \\ y_i &= y_{i-1} + s_i * n_i - b_i * m_i \\ x_i &= x_{i-1} + m_i - n_i \\ n_i &\leq x_{i-1} \\ b_i * m_i &\leq y_{i-1} \\ m_i, n_i, x_i, y_i &\geq 0 \end{aligned} \tag{1.2}$$

Using the variables, the objective and the constraints, objective is optimized using cvxpy library (for the code see Appendix). The maximum amount of money one can have at the end of Friday is \$189.723. Optimal strategy is to buy shares on Tuesday with all the available money then sell all these shares

on Wednesday. Next, shares are bought on Thursday with the entire available money and these shares are sold on Friday (for the results see Appendix).

Problem 2

This optimization problem can not be expressed as an LP. Because, the optimization problem has strict inequality,

$$p_1 > p_i \quad \forall i \neq 1 \tag{2.1}$$

Strict inequalities can not be expressed in an LP.

Problem 3

Let us define x_1 as any point in the first hyperplane. We also define a line L which passes through the point x_1 in the direction of a which is the normal of the first hyperplane. Next, we define x_2 as the point where the line L intersects with the second hyperplane. The distance between these two hyperplanes are equal to the distance between these two points. The equation of L can be given by $x_1 + ak, \quad \forall k \in \mathbb{R}$. Since x_2 satisfies both the second hyperplane and L equations, it can be found as follows,

$$\begin{aligned} a^T(x_1 + ak) &= b_2 \\ k &= \frac{b_2 - a^T x_1}{a^T a} \\ k &= \frac{b_2 - b_1}{a^T a} \end{aligned} \tag{3.1}$$

The intersection point x_2 can be found as,

$$x_2 = x_1 + a\frac{b_2 - b_1}{a^T a} \tag{3.2}$$

Therefore the distance between these hyperplanes are(which is equal to the distance between x_1 and x_2),

$$\|x_1 - x_2\| = \frac{|b_1 - b_2|\|a\|}{a^T a} = \frac{|b_1 - b_2|}{\|a\|} \tag{3.3}$$

Problem 4

(a)

The optimal solution is,

$$x_i^* = \begin{cases} 1, c_i & \leq 0 \\ 0, c_i & > 0 \end{cases}$$

and the optimal value is,

$$p^* = \sum_{i=1}^{n} min(c_i, 0)$$

(b)

The optimal solution is,

$$x_i^* = \begin{cases} 1, c_i & \leq 0 \\ -1, c_i & > 0 \end{cases}$$

and the optimal value is,

$$p^* = -\sum_{i=1}^{n} |c_i|$$

(c)

The optimal value is the summation of largest k values. Let us define s vector as the sorted vector whose values come from c ($s_1 \geq s_2 \geq ... \geq s_n$). Then the optimal value is,

$$p^* = s_1 + s_2 + ... + s_k$$

The optimal solution has all the components zero except the ones that are in the first k components of s. Their values are 1. In other words,

$$x_i^* = \begin{cases} 1, i \in A \\ 0, i \notin A \end{cases}$$

where A is the set of the indices of c with the largest k values.

(d)

The optimal value is the sum of the largest k positive components of c. If c has less number of positive components than k, then the optimal solution is the summation of all of the positive components of c.

Let us define s vector as the sorted vector whose values come from c ($s_1 \geq s_2 \geq ... \geq s_n$). Then,

$$t = max(i) \quad s.t. \quad (i \in 1, 2, ..., n \quad and \quad s_i \geq 0) \tag{4.1}$$

Then the optimal value can be defined as,

$$p^* = s_1 + s_2 + ... + s_{min(t,k)}$$

Optimal solution,

$$x_i^* = \begin{cases} 1, i \in A \\ 0, i \notin A \end{cases}$$

where A is the set of the indices of c with the largest positive $min(t, k)$ values.

(e)

In order to maximize $c^T x$, we can also maximize $\alpha c^T x$. Therefore, we maximize $(d^T x)c^T x$. The term in the parenthesis can be rewritten to make the entire term, $(x^T d)c^T x$. This can be rewritten as $x^T A x$ where $A = dc^T$. The Lagrangian for this problem can be written as,

$$\mathcal{L}(x) = x^T A x + \mu_1(x) + \mu_2(1 - x) \tag{4.2}$$

Taking the derivative and setting it to zero,

$$x^T(A^T + A) = \mu_2 - \mu_1 \tag{4.3}$$

such that

$$\begin{aligned} \mu_1(x) &= 0 \\ \mu_2(1-x) &= 0 \\ 0 \leq x &\leq 1 \\ \mu_1, \mu_2 &\geq 0 \end{aligned} \tag{4.4}$$

The equation can easily be solved to find the optimal solution and value.

Appendix- Code for Problem 1

Using the output of the code, the optimal strategy and the optimal value is given in Problem 1.

In [1]:
```
import cvxpy as cp
import numpy as np

b = np.array([2.7, 2.2, 1.6, 2.3, 1.1])
s = np.array([2, 2.4, 3.2, 2.3, 3])
```

In [2]:
```
m = cp.Variable(5)
n = cp.Variable(5)
x = cp.Variable(5)
y = cp.Variable(5)
```

In [3]:
```
constraints = []
constraints.append(y[0] == 100)
constraints.append(x[0] == 0)
for i in range(4):
    constraints.append(y[i+1] == y[i] + s[i+1]*n[i+1] - b[i+1]*m[i+1])
    constraints.append(x[i+1] == x[i] + m[i+1] - n[i+1])
```

In [4]:
```

constraints.append(m[0] == 0)
constraints.append(n[0] == 0)
for i in range(4):
    constraints.append(n[i+1] <= x[i])
    constraints.append(b[i+1]*m[i+1] <= y[i])
    constraints.append(m[i+1] >= 0)
    constraints.append(n[i+1] >= 0)
    constraints.append(x[i+1] >= 0)
    constraints.append(y[i+1] >= 0)
```

In [5]:
```
objective = cp.Maximize(y[4])
```

In [6]:
```
prob = cp.Problem(objective, constraints)
prob.solve()
print(y[4].value)
```

```
189.72331922639782
```

In [11]:
```
print('Number of Shares Bought per Day')
print(m.value)
print('--------------------------------')
print('Number of Shares Sold per Day')
print(n.value)
print('--------------------------------')
print('Total Money at the End of the Day')
print(y.value)
```

```
Number of Shares Bought per Day
[[0.00000000e+00]
 [4.54545385e+01]
 [9.46382052e-06]
 [6.32410969e+01]
 [7.21582025e-08]]
--------------------------------
Number of Shares Sold per Day
[[0.00000000e+00]
 [2.41931630e-09]
 [4.54545385e+01]
 [3.32196626e-07]
 [6.32411060e+01]]
--------------------------------
Total Money at the End of the Day
[[1.00000000e+02]
 [1.52206589e-05]
 [1.45454523e+02]
 [1.23489946e-06]
 [1.89723319e+02]]
```