Homework 5

Kyle Hadley

In [1]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import matplotlib.lines as mlines
import cvxpy as cp
from mpl_toolkits.mplot3d.art3d import Poly3DCollection
```

In [2]:
```
import warnings
warnings.simplefilter('ignore')
```

1. Linear Program Duality

Given the linear program $p^* = \min_x c^T x$ s.t. $Ax = b, Cx \geq d$.

(a)

We can write the linear program, p^*, in it's game form such that we are minimizing x and maximizing (v, w) of our Lagrangian. Solving for our lagrangian,

$$\mathcal{L}(x, v, w) = f(x) + v^T g(x) + w^T h(x) = c^T x + v^T(Ax - b) + w^T(Cx - d)$$

$$\mathcal{L}(x, v, w) = (c^T + v^T A + w^T C)x - v^T b - w^T d$$

Thus, we find that

$$p^* = \min_x \max_{v,w} \mathcal{L}(x, v, w)$$

$$\text{s.t. } \mathcal{L}(x, v, w) = (c^T + v^T A + w^T C)x - v^T b - w^T d$$

(b)

To find a relationship between p^* and d^*, we can start with the relationship of

$$\mathcal{L}(x, v, w) \mid \forall x, \forall v \forall w \leq \max_x \mathcal{L}(x, v, w) \mid \forall v \forall w$$

If we minimize v and w on both sides the relationship still holds true that

$$\min_{v,w} \mathcal{L}(x, v, w) \mid \forall x \leq \min_{v,w}(\max_x \mathcal{L}(x, v, w))$$

Next, we can see that for any x selected for the left-hand side of our equation, the relationship holds true. Thus we can pick an x that maximizes the left-hand side (i.e. $\max_x$) such that

$$\max_x(\min_{v,w} \mathcal{L}(x, v, w)) \leq \min_{v,w}(\max_x \mathcal{L}(x, v, w))$$

Thus, we can see that from this relationship and our definitions for p^* and d^*, $p^* \leq d^*$.

(c)

Taking the derivative of the Lagrangian, as found in part (a), with respect to x we find and equate the relationship to 0, as we are minimizing x,

$$\frac{\partial \mathcal{L}}{\partial x} = c^T + v^T A + \mu^T C = 0$$

Thus, we find the condition $c^T + v^T A + \mu^T C = 0$.

(d)

Given our constraint as found in part (c), we can write the appropriate objective function of v and w, $l(v, w)$ as follows,

$$p^* = \max_{v,w}(-v^T b - \mu^T d)$$

s.t. $g(v, w) = 0$ and $h(v, w) \geq 0$.

(e)

Given the values for c^T, A, b, C, and d, we can solve both the primal and dual versions using cvxpy as shown below.

In [3]:

```
# Establish our values for input parameters
A = np.matrix([[1, 1, 1, 1, 1], [1, 1, -1, 0, 0], [0, 0, 0, 1, -1]])
b = np.matrix([[1], [0], [0]])
c = np.matrix([1, 2, 4, 5, 6])
C_prime = np.matrix([[-1, -1, -1, 0, 0], [0, 0, 0, -1, -1]])
C = np.concatenate((np.identity(5), C_prime), axis=0)
d = np.matrix([[0], [0], [0], [0], [0], [-0.5], [-0.5]])

#print(A, b, c, C, d)

# Define and solve the CVXPY problem.
x = cp.Variable(shape=(5, 1))
prob = cp.Problem(cp.Minimize(c @ x), [A @ x == b, C @ x >= d])
prob.solve()

# Print result.
print("The optimal value is", prob.value)
print("\nA solution x is:")
print(x.value)
print("\nA dual solution is: ")
print(prob.constraints[0].dual_value, '\nand \n', prob.constraints[1].dual_value)
```

```
The optimal value is 3.9999999998358136

A solution x is:
[[2.50000000e-01]
```

```
 [2.41013141e-11]
 [2.50000000e-01]
 [2.50000000e-01]
 [2.50000000e-01]]

A dual solution is:
[[-5.80414113]
 [ 1.5       ]
 [ 0.5       ]]
and
 [[4.61842945e-10]
 [1.00000000e+00]
 [5.79071862e-10]
 [1.22463856e-10]
 [1.22463889e-10]
 [3.30414113e+00]
 [3.04141131e-01]]
```

(f)

2. Quadratic Program Duality

Consider the quadratic program $p^* = max_x \frac{1}{2} x^T Q x + r^T x$ s.t. $Ax = b, Cx \geq d$.

(a)

We can write the quadratic program, p^*, in it's game form such that we are minimizing x and maximizing (v, w) of our Lagrangian. Solving for our lagrangian,

$$\mathcal{L}(x, v, w) = f(x) + v^T g(x) + w^T h(x) = \frac{1}{2} x^T Q x + r^T x + v^T (Ax - b) + w^T (Cx - d)$$

$$\mathcal{L}(x, v, w) = \frac{1}{2} x^T Q x + (r^T + v^T A + w^T C)x - v^T b - w^T d$$

Thus, we find that

$$p^* = \max_x \min_{v,w} \mathcal{L}(x, v, w)$$

$$\text{s.t. } \mathcal{L}(x, v, w) = \frac{1}{2} x^T Q x + (r^T + v^T A + w^T C)x - v^T b - w^T d$$

(b)

The logic as described from part (b) still holds true for this problem, however rather than minimizing x we are instead maximizing x. Thus, we have the opposite relationship between p^* and d^* as we know that

$$\max_x (\min_{v,w} \mathcal{L}(x, v, w)) \leq \min_{v,w} (\max_x \mathcal{L}(x, v, w))$$

Thus, we can see that from this relationship and our definitions for p^* and d^*, $p^* \geq d^*$.

(c)

Taking the derivative of the Lagrangian, as found in part (a), with respect to x we find and equate the relationship to 0, as we are minimizing x,

$$\frac{\partial \mathcal{L}}{\partial x} = x^TQ + r^T + v^Ta + w^TC = 0$$

Solving for x^T we find that

$x^T = -(r^T + v^Ta + w^TC)Q^{-1}

We can a z^T such that $z^T = -x^TQ = r^T + v^Ta + w^TC$.

(d)

Given our constraint as found in part (c), we can solve for $l(v,w)$ by substituting our relationship for z^T in for x^T such that,

$$l(v,w) = \frac{1}{2}z^TQ^{-1}QQ^{-1} - z^TQ^{-1}z - v^Tb - w^Td$$

$$l(v,w) = \frac{1}{2}z^TQ^{-1} - z^TQ^{-1}z - v^Tb - w^Td$$

Thus, we can write the appropriate objective function of v and w, $l(v,w)$

$$p^* = \min_{v,w} l(v,w)$$

s.t. $l(v,w) = \frac{1}{2}z^TQ^{-1} - z^TQ^{-1}z - v^Tb - w^Td$, $z^T = r^T + v^Ta + w^TC$, $g(v,w) = 0$, and $h(v,w) \geq 0$.

(e)

Given the values for Q, r^T, A, b, C, and d, we can solve both the primal and dual versions using cvxpy as shown below.

In [4]:

```
# Establish our values for input parameters
Q = -np.diag([1, 2, 3, 4, 5])
A = np.matrix([[1, 1, 1, 1, 1], [1, 1, -1, 0, 0], [0, 0, 0, 1, -1]])
b = np.matrix([[1], [0], [0]])
r = np.matrix([1, 2, 3, 4, 5])
C_prime = np.matrix([[-1, -1, -1, 0, 0], [0, 0, 0, -1, -1]])
C = np.concatenate((np.identity(5), C_prime), axis=0)
d = np.matrix([[0], [0], [0], [0], [0], [-0.5], [-0.5]])

#print(Q, A, b, r, C, d)

# Define and solve the CVXPY problem.
x = cp.Variable(shape=(5, 1))
prob = cp.Problem(cp.Maximize((1/2) * cp.quad_form(x, Q) + r @ x), [A @ x == b, C @ x >
prob.solve()

# Print result.
print("The optimal value is", prob.value)
print("\nA solution x is:")
```

```python
print(x.value)
print("\nA dual solution is: ")
print(prob.constraints[0].dual_value, '\nand \n', prob.constraints[1].dual_value)
```

```
The optimal value is 3.0625000000000004

A solution x is:
[[5.1634942e-24]
 [2.5000000e-01]
 [2.5000000e-01]
 [2.5000000e-01]
 [2.5000000e-01]]

A dual solution is:
[[ 1.425]
 [-0.375]
 [-0.375]]
and
 [[0.5 ]
 [0.  ]
 [0.  ]
 [0.  ]
 [0.  ]
 [0.45]
 [1.95]]
```

3. Simplex optimization

(a)

a - $\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T$

b - $\begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T$

c - $\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$

d - $\begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix}^T$

e - $\begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}^T$

f - $\begin{bmatrix} \frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix}^T$

g - $\begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix}^T$

h - $x_2 = 1 - x_1$

i - $x_3 = 1 - x_2$

j - $x_1 = 1 - x_3$

k - $r^T + \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$

l - $r^T + \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix}^T$

m - $r^T + \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T$

n - $\begin{bmatrix} \frac{1}{3} & \frac{1}{3} & 0 \end{bmatrix}^T$

o - $\begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix}^T$

p - r^T

(b)

a - r_1

b - r_2

c - r_3

d - μ_1

e - μ_2

f - μ_3

g - λ

h - μ_1^*

i - μ_3^*

j - λ^*

k - x_2

4. Dynamic Programming

(a)

Assuming that in our diagram our states are numbered from top to bottom ($state = \{1, 2, 3\}$) and time is numbered from left to right ($time = \{0, 1, 2, 3, 4\}$ - time 5 is ignored because their is our final "red" node), the optimal "cost-to-go" for each node to the end is calculated as follows:

$time = 4$, $state = 1$ - 2

$time = 4$, $state = 2$ - 2

$time = 4$, $state = 3$ - 1

$time = 3$, $state = 1$ - 3

$time = 3$, $state = 2$ - 2

$time = 3$, $state = 3$ - 3

$time = 2$, $state = 1$ - 5

$time = 2$, $state = 2$ - 4

$time = 2$, $state = 3$ - 4

$time = 1$, $state = 1$ - 5

$time = 1$, $state = 2$ - 6

$time = 1$, $state = 3$ - 6

$time = 0$, $state = 2$ - 6

(b)

The shortest path is start node ("blue" node) $\longrightarrow$ time = 1, state 1 $\longrightarrow$ time = 2, state = 2 $\longrightarrow$ time = 3, state = 2 $\longrightarrow$ time = 4, state = 3 $\longrightarrow$ end node ("red" node).