Convex Optimization

Tutorial 6

Tanmay Garg CS20BTECH11063

In [ ]:
```
#Importing required Libraries
import numpy as np
import matplotlib.pyplot as plt
import cvxpy as cp
```

In [ ]:
```
#Vector to denote the amount ideal production in entire day for each product
IdealProduceable = np.matrix('6000.0 5000.0 3000.0')
IdealProduceable = IdealProduceable.T

#Vector to denote the profit made on each
Profits = np.matrix('4 6 10')
Profits = Profits.T

#Storage space for each product
Storage = np.matrix('40 45 210')
Storage = (Storage.T)/1000

#Vector to denote the minimum amount of demand for each product
MinDelivered = np.matrix('5000 0 4000')
MinDelivered = MinDelivered.T

#Vector to denote the maximum amount of demand for each product
MaxDemanded = np.matrix('10000 15000 8000')
MaxDemanded = MaxDemanded.T

TotalSpace = 6000
NumWorkingDays = 5
NumWorkingHours = 8
```

In [ ]:
```
# Part a

X = cp.Variable((3, 1))

IdealProducedWeek = 5 * IdealProduceable
TotalStorable = cp.multiply(IdealProducedWeek, Storage)

# print(IdealProducedWeek)
# print(TotalStorable.value)
# print(cp.multiply(IdealProducedWeek, Profits).value)

MyObjective_A = cp.Maximize(X.T@cp.multiply(IdealProducedWeek, Profits))

MyConstraints_A = [
    X >= 0,
    X.T@TotalStorable <= 6000,
    cp.multiply(X, IdealProducedWeek) >= MinDelivered,
    cp.multiply(X, IdealProducedWeek) <= MaxDemanded,
    X.T@np.ones((3,1)) <= 1,
    X <= np.ones((3,1))
]
```

In [ ]:
```
MyProblem_A = cp.Problem(MyObjective_A, MyConstraints_A)
value = MyProblem_A.solve()
```

```
# print(value)
print("The maximum profit is: {} dollars".format(np.round(value, 2)))
print("The proportion of time of each product produced is: ")
print(X.value)
```

```
The maximum profit is: 145000.0 dollars
The proportion of time of each product produced is:
[[0.16666667]
 [0.35335573]
 [0.47997761]]
```

In [ ]:

```
#Part b

Y = cp.Variable((3, 1))
IdealProduceable_Reciprocal = np.reciprocal(IdealProduceable)
# print(IdealProduceable_Reciprocal)

MyObjective_B = cp.Maximize(Y.T@Profits)

MyConstraints_B = [
    Y >= 0,
    Y.T@Storage <= TotalSpace,
    Y >= MinDelivered,
    Y <= MaxDemanded,
    Y.T@IdealProduceable_Reciprocal <= 5
]
```

In [ ]:

```
MyProblem_B = cp.Problem(MyObjective_B, MyConstraints_B)
value = MyProblem_B.solve()
# print(value)
print("The maximum profit is: {} dollars".format(np.round(value, 2)))
print("The amount of each product produced is: ")
print(Y.value)
```

```
The maximum profit is: 145000.0 dollars
The amount of each product produced is:
[[5000.00001178]
 [7621.54643947]
 [7927.07212906]]
```

In [ ]:

```
#Part c

Z = cp.Variable((3, 1))
PerHourProducable = IdealProduceable/NumWorkingHours

MyObjective_C = cp.Maximize((Z.T@np.multiply(PerHourProducable, Profits)) / 1000)

# print(PerHourProducable)
# print(Profits)
# print(np.multiply(PerHourProducable, Profits))

MyConstraints_C = [
    Z >= 0,
    Z <= 40 * np.ones((3, 1)),
    Z.T@cp.multiply(PerHourProducable, Storage) <= TotalSpace,
    cp.multiply(PerHourProducable, Z) >= MinDelivered,
    cp.multiply(PerHourProducable, Z) <= MaxDemanded,
    Z.T@np.ones((3,1)) <= 40
]
```

In [ ]:

```
MyProblem_C = cp.Problem(MyObjective_C, MyConstraints_C)
value = MyProblem_C.solve()
# print(value)
print("The maximum profit is: {} thousand dollars".format(np.round(value, 2)))
```

```
print("The number of hours for each product produced is: ")
print(Z.value)
```

```
The maximum profit is: 145.0 thousand dollars
The number of hours for each product produced is:
[[ 6.66666667]
 [13.40351759]
 [19.92981573]]
```

In [ ]:

```
print("The relation between part a and part b: ")
print("X * 40 = Z (approx)")
print("X * 40: ")
print((X*40).value)
print("Z: ")
print(Z.value)
print("Error between both the matrices")
error_vec = (X*40).value - Z.value
print(error_vec)
```

```
The relation between part a and part b:
X * 40 = Z (approx)
X * 40:
[[ 6.66666666]
 [14.13422914]
 [19.19910421]]
Z:
[[ 6.66666667]
 [13.40351759]
 [19.92981573]]
Error between both the matrices
[[-1.19386057e-08]
 [ 7.30711550e-01]
 [-7.30711517e-01]]
```