OPTIMIZATION AND OPERATIONAL RESAERCH PROJECT

Hamed RAHIMI and Rediet Gebretsion TADESSE

Université de Lyon, France

COURSE NAME:	**Optimization and Operational Research (OOP)**
SUPERVISORS:	**Prof. Redko**

DATE OF SUBMISSION: March 2020

Q1.

In this project, first, we have installed a python library named cvxpy by typing ***pip install cvxpy*** in our comment prompt. Then, we used this library to solve our optimization problems. The structure of problems is as follows. In question 2, we have worked on convex optimization and comparing our theoretical results with the output of the code. In the third part of this question, we used the log-barrier function to optimize one of the problems. In question 3, we solved 3 read-world questions and in the last part we propose another question and we formalized it with cvxpy and solved it. Finally, we used cvxpy in the linear regression method as a data analysis algorithm.

We have seen how to use different usage of cvxpy package for optimization problems and practiced how to use it properly. We started from Declaring Variables as scalar as well as a vector, and then we practiced how to declare constraints involved in the optimization problem. We have also exercised how to express the objective function using different standard expressions from the package. This helped us to solve different types of optimization problems with a short time. These exercises give us the first foundation and basic things of the CVXPY package. The general structure of using cvxpy is as follows:

```
import cvxpy as cp

# Create optimization variables.
x = cp.Variable()
y = cp.Variable() # cp.Variable(n), Here n is the dimensionality of the variable
as vector

# Create constraints.
constraints = [x + y == 1, x-y >= 1]

# Form objective.
obj = cp.Minimize((x - y)**2) # define objective function

# Form and solve problem.
prob = cp.Problem(obj, constraints) # Define Optimization Problem
prob.solve() # Returns the optimal value

print("status:", prob.status)
print("optimal value", prob.value) # Optimal value
print("optimal var", x.value, y.value) # Optimal value of x and y
```

Q2.1.

Is $F = ((X_1-4)^2 + 7*(X_2-4)^2 + 4*X_2)$ Convex?

Checking determinant (H) >=0 and trace (H)>0

$$H = \begin{bmatrix} \partial F/\partial X1\,\partial X1 & \partial F/\partial X1\,\partial X2 \\ \partial F/\partial X2\,\partial X1 & \partial F/\partial X2\,\partial X2 \end{bmatrix} \qquad H = \begin{bmatrix} 2 & 0 \\ 0 & 14 \end{bmatrix}$$

Determinant (H) = 28 and Trace (H) = 16

Therefore, Yes! It is Convex!

Since it is convex, we can find the minimum value using Euler's equation.

The Jacobian of the function is:

$$\partial F/\partial X_1=2*(X_1-4)$$

$$\partial F/\partial X_2=14*(X_2-4) + 4$$

Euler's Equation (Setting Jacobian to 0)

$$2*(X_1-4) = 0 \dashrightarrow X_1=4$$

$$14*(X_2-4) + 4 =0 \dashrightarrow X_2=3.715$$

Now, we code it in Python using CVXPY library

```
# Q2.1
# We want to formulize  Min (x1-4)^2+7*(x2-4)^2+4*x2 in CVPXY and solve it

import cvxpy as cp

# we reate two variables for x1 and x2.
x1 =cp.Variable()
x2 =cp.Variable()

# There is no constrain so:
constraints = None

# we formulate our problem.
F=(x1 - 4)**2+ 7*((x2 - 4)**2) + 4*x2
```

```
#we define our objectvie funtion
obj = cp.Minimize(F)

# insert Cons and Obj in cvxpy to solve the problem.
prob = cp.Problem(obj, constraints)
prob.solve() # Returns the optimal value.

print("The status of the problem is:", prob.status)
print("optimal value of minimization is:", prob.value)
print("optimal value of variable x1 is :", x1.value)
print("optimal value of variable x2 is :", x2.value)
```

Result:

```
The status of the problem is: optimal
optimal value of minimization is: 15.428571428571429
optimal value of variable x1 is : 4.0
optimal value of variable x2 is : 3.7142857142857144
```

As you can see, the result is same as what we calculate with hand.Now, we code it again by using a vector variable.

```
# Q2.1.2 (Using Vector Variable)
# We want to formulize  Min (x1-4)^2+7*(x2-4)^2+4*x2 in CVPXY and solve it

import cvxpy as cp

# we reate two Vectors with shape 2.
x =cp.Variable(2)

# There is no constrain so:
constraints = None

# we formulate our problem.
F=(x[0] - 4)**2+ 7*((x[1] - 4)**2) + 4*x[1]

#we define our objectvie funtion
#wrap up the vectors we use cp.sum
obj = cp.Minimize(cp.sum(F))

# insert Cons and Obj in cvxpy to solve the problem.
```

```python
prob = cp.Problem(obj, constraints)
prob.solve() # Returns the optimal value.
print("The status of the problem is:", prob.status)
print("optimal value of minimization is:", prob.value)
print("optimal value of variable x is :", x.value)
```

Result:

```
The status of the problem is: optimal
optimal value of minimization is: 15.428571428571429
optimal value of variable x is : [4.         3.71428571]
```

As you can see, the result is same as what we calculate with hand.

Q2.2.

Is F= $X_1^3 + (X_2 - X_3)^2 + X_3^3 + 2$ Convex?

Let's check each function separately

$(X1)^3$ **is convex when $X_1 > 0$**

X_1^3 :

Convex if F" > 0 for single variable function

$(X_1^3)'' = 6 * X_1 \rightarrow X_1 > 0$ (1)

$(X1)^3$ **is convex when $X_1 > 0$**

$(X_2 - X_3)^2$:

$$\frac{\partial F}{\partial X2} = 2 * (X2 - X3) \qquad \frac{\partial F}{\partial X3} = -2 * (X2 - X3) \quad H = \begin{bmatrix} 2 & -2 \\ -2 & 2 \end{bmatrix}$$

Checking determinant (H) >=0 and trace (H)>0

Determinant (H) = 0 and Trace (H) = 4

$(X2 - X3)^3$ **is convex**

$(X3)^3$ **is convex when $X_3 > 0$**

Therefore, Yes! It is Convex under constrains x1>0 and x3>0

To find the minimum of the convex function, fin the Jacobian and set it to zero and solve it simultaneously.

The Jacobian of the function is:

(1) $\mathbf{(X1)^2 = 0 \rightarrow X1 = 0}$

(2) $\mathbf{(2 * X2) - (2 * X3) \rightarrow X2 = X3 \rightarrow \quad (4)}$

(3) $\mathbf{3 * (X3)^2 - (2 * X2) + (2 * X3) = 0}$

Substitute (4) to (3) -> $\mathbf{X3 = 0 = X2}$

Therefore the minimum value is (x1,x2,x3) approaches (0,0,0)

Min F= $\mathbf{X_1^3 + (X_2 - X_3)^2 + X_3^3 + 2}$ **is equal to 0+0+0+2= 2**

Now, we code it in Python using CVXPY library.

```
#Q2.2

#We want to formulize  F = Min (x1)**3 + (x2 -
 x3)**2 + (x3)**3 + 2 in CVPXY and solve it

import cvxpy as cp

# Create three variables.
x1 = cp.Variable()
x2 = cp.Variable()
x3 = cp.Variable()

# Create two constraints.
constraints = [x1 >= 0, x3 >= 0]

# Form objective.
obj = cp.Minimize((x1)**3 + (x2 - x3)**2 + (x3)**3 + 2)

# Form and solve problem.
prob = cp.Problem(obj,constraints)
prob.solve() # Returns the optimal value.

print("The status of the problem is:", prob.status)
print("optimal value of minimization is:", prob.value)
print("optimal value of variable x1 is :", x1.value)
print("optimal value of variable x2 is :", x2.value)
print("optimal value of variable x3 is :", x3.value)
```

Result:

```
The status of the problem is: optimal
optimal value of minimization is: 1.9999999993629338
optimal value of variable x1 is : 0.0008258815459611968
optimal value of variable x2 is : 0.0008258861793499652
optimal value of variable x3 is : 0.0008258861755015717
```

As you can see, the result is same as what we calculate with hand. Now, we code it again by using a Vector variable.

```
#Q2.2.2 (Using Vector Variable)

#We want to formulize  F = Min (x1)**3 + (x2 -
 x3)**2 + (x3)**3 + 2 in CVPXY and solve it

import cvxpy as cp

# Create vector variable with shape 3.
x = cp.Variable(3)

# Create two constraints.
constraints =  [(x[1]) >= 0, (x[2]) >= 0]

# Form objective.
obj = cp.Minimize((x[0])**3 + (x[1] - x[2])**2 + (x[2])**3 + 2)

# Form and solve problem.
prob = cp.Problem(obj,constraints)
prob.solve() # Returns the optimal value.

print("The status of the problem is:", prob.status)
print("optimal value of minimization is:", prob.value)
print("optimal value of variable x1 is :", x.value)
```

Result:

```
The status of the problem is: optimal
optimal value of minimization is: 1.9999999999174138
optimal value of variable x1 is : [0.00024227 0.00045838 0.00045833]
optimal value of variable x3 is : 0.0008258861755015717
```

As you can see, the result is same as what we calculate with hand

Q2.3.

15.1 Introduction

Previously, we looked at Newton's method for minimizing twice differentiable convex functions with equality constraints. One of the limitations of this method is that we cannot deal with inequality constraints. The barrier method is a way to address this issue. Formally, given the following problem,

$$\begin{aligned} \min \quad & f(x) \\ \text{subject to} \quad & h_i(x) \le 0, i = 1, \ldots m \\ & Ax = b \end{aligned}$$

assuming that f, h_i are all convex and twice differentiable functions, all with domain $\mathbb{R}^n$, the log barrier is defined as

$$\phi(x) = -\sum_{i=1}^{m} log(-h_i(x))$$

It can be seen that the domain of the log barrier is the set of strictly feasible points, $\{x : h_i(x) < 0, i = 1 \ldots m\}$. Note that the equality constraints are ignored for the rest of this chapter, because those can be solved using the Newton's method directly.

15.1.1 Approximation of Indicator Functions

The idea behind the definition of the log barrier is that it is a smooth approximation of the indicator functions. More precisely, given that our problem with inequality constraints is to minimize

$$f(x) + \sum_{i=1}^{m} I_{\{h_i(x) \le 0\}}(x)$$

we approximate it as follows

$$f(x) - ({}^1/_t) \sum_{i=1}^{m} log(-h_i(x))$$

As t approaches ∞, the approximation becomes closer to the indicator function, as shown in Figure 15.1. Also, for any value of t, if any of the constraints is violated, the value of the barrier approaches infinity

Figure 1- Berkeley University[1]

Using the formula, we formulate this problem in Python using CVXPY library and solving it. considering that we are increasing T to the find best solution.

[1] https://www.stat.cmu.edu/~ryantibs/convexopt-S15/scribes/15-barr-method-scribed.pdf

```python
#Q2.3

import cvxpy as cp

# Create two variables and one parameter.
x = cp.Variable()
y = cp.Variable()
t = cp.Parameter(nonneg=True, value=1)

# no constraints.
constraints = None

sqr=cp.square(x-2)
log=cp.log(-4+x+y)
log=log/t

F= sqr  + (3*y)  - log

# Form objective
obj = cp.Minimize(F)
# Form and solve problem.
prob = cp.Problem(obj, constraints)
prob.solve() # Returns the optimal value.

new_v=prob.value
old_v=6
print("it takes few sec to find best value of t")
# it takes few sec to find best t

while(new_v<old_v):
  t.value=t.value+100
  sqr=cp.square(x-2)
  log=cp.log(-4+x+y)
  log=log/t
  F= sqr  + (3*y)  - log
# Form objective
  obj = cp.Minimize(F)
# Form and solve problem.
  prob = cp.Problem(obj, constraints)
  prob.solve() # Returns the optimal value.
  old_v=new_v
  new_v=prob.value

print("otimal value of t is:", t.value)
print("The status of the problem is:", prob.status)
print("optimal value of minimization is:", prob.value)
```

```
print("optimal value of variable x is :", x.value)
print("optimal value of variable y is :", y.value)
```

Result:

```
it takes few sec to find best value of t
otimal value of t is: 180301
The status of the problem is: optimal
optimal value of minimization is: 3.7500787481363
optimal value of variable x is : 3.5000027798119273
optimal value of variable y is : 0.4999990596912537
```

As you can see, the result is same as what we calculate with hand. Now, we code it again by using a Vector variable.

```
#Q2.3.2 (Using Vector Variable)

import cvxpy as cp

# Create vector with two shape and one parameter.
x = cp.Variable(2)
t = cp.Parameter(nonneg=True, value=1)

# no constraints.
constraints = None

sqr=cp.square(x[0]-2)
log=cp.log(-4+x[0]+x[1])
log=log/t

F= sqr  + (3*x[1])  - log

# Form objective
obj = cp.Minimize(cp.sum(F))
# Form and solve problem.
prob = cp.Problem(obj, constraints)
prob.solve() # Returns the optimal value.

new_v=prob.value
old_v=6


print("it takes few secs to find the best value of t")


while(new_v<old_v): # it tries to increase t to find the optimal t
```

```
  t.value=t.value+100
  sqr=cp.square(x[0]-2)
  log=cp.log(-4+x[0]+x[1])
  log=log/t
  F= sqr  + (3*x[1])  - log
# Form objective
  obj = cp.Minimize(F)
# Form and solve problem.
  prob = cp.Problem(obj, constraints)
  prob.solve() # Returns the optimal value.
  old_v=new_v
  new_v=prob.value

print("optimal value of t is:", t.value)
print("The status of the problem is:", prob.status)
print("optimal value of minimization is:", prob.value)
print("optimal value of variable x is :", x.value)
```

Result:

```
it takes few sec to find best value of t
optimal value of t is: 180301
The status of the problem is: optimal
optimal value of minimization is: 3.7500787481363
optimal value of variable x is : [3.50000278 0.49999906]
```

Q3.1

First, we model this problem as a constrained optimization problem. Considering S represents stream and R represent Reservoir. The objective function is to minimize the cost which is:

Objective Function:

C= 100*R + 50 S

Constrains are:

A city needs 500,000 liters of water per day → **S+R <= 100**

No more than 100,000 liters per day can be drawn from the stream → **S <= 100**

The concentration of pollutants in the water served to the city cannot exceed 100 ppm →
((50S+250R)/(S+R)) <= 100 is equal to **150R-50S<=0**

Now, we formulate the question in Python using CVXPY library

```
#Question 3.1

# s= stream r= reservoir
# Goal is to minimizae Cost which is equal to 100R+50S
# S+R <= 100
# S<= 100
# ((50S+250R)/(S+R)) <= 100 which is equal to 150R-50S<=0


import cvxpy as cp
# Create two scalar optimization variables.
r = cp.Variable()
s = cp.Variable()
# Create two constraints.
constraints = [r + s == 500,
150*s-50*r <= 100, s <= 100]
# Form objective.
obj = cp.Minimize(100*r + 50*s)
# Form and solve problem.
prob = cp.Problem(obj, constraints)
prob.solve(verbose=True) # Returns the optimal value.
print("status:", prob.status)
print("optimal value", prob.value)
print("optimal var", r.value, s.value)
```

Result:

```
status: optimal
optimal value 44999.999996477294
optimal var 399.99999992954594 100.00000007045402
```

Q3.2

First, we model this problem as a constrained optimization problem. Given:

b1 is the composition of Blend 1 in the perfume

b2 is the composition of Blend 2 in the perfume

b2 is the composition of Blend 3 in the perfume

b3 is the composition of Blend 4 in the perfume

T be the Total of all the blends which is equal to 1

Objective function:

Min Z = 55*b1 + 65*b2 + 35*b3 + 85*b4

Constraints:

((0.35*b1) + (0.6*b2) + (0.35*b3) + (0.4*b4)) <= 0.5

((0.15*b1)+(0.05*b2)+(0.2*b3)+(0.1*b4)) >= 0.08

((0.15*b1)+(0.05*b2)+(0.2*b3)+(0.1*b4)) <= 0.13

((0.3*b1)+(0.2*b2)+(0.4*b3)+(0.2*b4)) <= 0.35

((0.2*b1)+(0.15*b2)+(0.05*b3)+(0.3*b4)) >= 0.19,

b1 >= 0.1, b1 <= 0.25, b2 >= 0.05, b2 <= 0.2, b3 >= 0.3, b1+b2+b3+b4 == 1

Now, we formulate the question in Python using CVXPY library

```
#Question 3.2 by Hamed Rahimi

import cvxpy as cp
# Create two scalar optimization variables.
b1 = cp.Variable()
b2 = cp.Variable()
b3 = cp.Variable()
b4 = cp.Variable()

# Create two constraints.
constraints = [((0.35*b1)+(0.6*b2)+(0.35*b3)+(0.4*b4)) <= 0.5,
               ((0.15*b1)+(0.05*b2)+(0.2*b3)+(0.1*b4)) >= 0.08,
               ((0.15*b1)+(0.05*b2)+(0.2*b3)+(0.1*b4)) <= 0.13,
               ((0.3*b1)+(0.2*b2)+(0.4*b3)+(0.2*b4)) <= 0.35,
               ((0.2*b1)+(0.15*b2)+(0.05*b3)+(0.3*b4)) >= 0.19,
               b1 >= 0.1, b1 <= 0.25, b2 >= 0.05, b2 <= 0.2, b3 >= 0.3, b1+b2+b3+b
4 == 1]

# Form objective.
obj = cp.Minimize((55*b1) + (65*b2) + (35*b3) + (85*b4)) # Declare objective funct
ion
# Form and solve problem.
prob = cp.Problem(obj, constraints)
prob.solve()

print("status:", prob.status)
print("optimal value", prob.value)
print("optimal var", b1.value, b2.value, b3.value, b4.value)
status: optimal
optimal value 62.99999999836229
optimal var 0.140000000009318 0.140000000014373 0.3000000000218449 0.4199999999547174
```

Q3.3

First, we model this problem as a constrained optimization problem. Considering r is rural, u is urban Bu is profit of urban and Br is profit of rural

Objective Function:

Maximize net profit that is equal to Bu+Br-r-u

Constrains are:

r+u = 200

Now, we formulate the question in Python using CVXPY library

```
#Question 3.3 by Hamed Rahimi

# r= rural u= urban Bu= profit of urban Br= profit of rural

# Goal is to maximizae net profit which is equal to Bu+Br-r-u

# r+u = 200


import cvxpy as cp
# Create two scalar optimization variables.
r = cp.Variable()
u = cp.Variable()

Bu= 5000*cp.atoms.elementwise.log1p.log1p(u)
Br= 7000*cp.atoms.elementwise.log1p.log1p(r)
# Create two constraints.
constraints = [r + u == 200]

# Form objective.
obj = cp.Maximize(Bu+Br-r-u)
# Form and solve problem.
prob = cp.Problem(obj, constraints)
prob.solve() # Returns the optimal value.
print("status:", prob.status)
print("optimal value", prob.value)
print("optimal var", r.value, u.value)

#############################
```

```
status: optimal
optimal value 55348.893106737625
optimal var 116.83330317497843 83.16669632982155
```

Q3.4

The factory RadioIn builds two types of radios A and B. Every radio is produced by the work of three specia lists Pierre, Paul and Jacques. Pierre works at most 24 hours per week. Paul works at most 45 hours per wee k.Jacques works at most 30 hours per week.The resources necessary to build each type of radio and their sell ing prices as well are given in the following table:

	Radio A	Radio B
Pierre	1h	2h
Paul	2h	1h
Jacques	1h	3h
Selling prices	15e	10e

We assume that the company has no problem to sell its production, whichever it is.
a) Model the problem of finding a weekly production plan maximizing the revenue of RadioIn as a linear pr ogramme. Write precisely what are the decision variables, the objective function and the constraints.
b) Solve the linear programme using the geometric method and give the optimal production plan.

First, we model this problem as a constrained optimization problem. Considering r is rural, u is urban Bu is profit of urban and Br is profit of rural

Objective Function:

Maximize profit which is 15*A+10*B
Constrains are

A+2*B<=24, 2*A+B<=45, A+3*B<=30

Now, we formulate the question in Python using CVXPY library

```
# Question 3.4 by Hamed Rahimi

#The factory RadioIn builds two types of radios A and B.
#Every radio is produced by the work of three specialists Pierre, Paul and Jacque
#Pierre works at most 24 hours per week. Paul works at most 45 hours per week.
#Jacques works at most 30 hours per week.
#The resources necessary to build each type of radio and their selling prices as w
ell are given in the following table:
#Radio A Radio B
#Pierre 1h 2h
#Paul 2h 1h
#Jacques 1h 3h
#Selling prices 15 euros 10 euros
```

```
#We assume that the company has no problem to sell its production, whichever it is
#a) Model the problem of finding a weekly production plan maximizing the revenue o
f RadioIn as a linear programme.
#Write precisely what are the decision variables, the objective function and the c
onstraints.
#b) Solve the linear programme using the geometric method and give the optimal pro
duction plan.

import cvxpy as cp
# Create two scalar optimization variables.
A = cp.Variable()
B = cp.Variable()
# Create two constraints.
constraints = [A+2*B<=24, 2*A+B<=45, A+3*B<=30]

# Form objective.
obj = cp.Maximize(15*A+10*B)
# Form and solve problem.
prob = cp.Problem(obj, constraints)
prob.solve() # Returns the optimal value.
print("status:", prob.status)
print("optimal value", prob.value)
print("optimal var", A.value, B.value)
```

```
status: optimal
optimal value 340.0
optimal var 22.0 0.9999999999999976
```

Q4.
Linear regression Method

Here we used linear regression approach for predicting a **response** using a **single feature**.We assumed that the two variables are linearly related. Hence, we try to find a linear function that predicts the response value(y) as accurately as possible as a function of the feature or independent variable(x) and is to find a **line which fits best** for the dataset.

The equation of regression line is represented as:

$$h(xi)= \beta_o+ \beta_1 Xi$$

we used Least Squares technique

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i = h(x_i) + \varepsilon_i \Rightarrow \varepsilon_i = y_i - h(x_i)$$

Here, ε_i is **residual error** in i_{th} observation.

So, our aim is to minimize the total residual error.
We define the squared error or cost function J as:

$$J(\beta_0, \beta_1) = \frac{1}{2n} \sum_{i=1}^{n} \varepsilon_i^2$$

and our task is to find the value of b_0 and b_1 for which J(b_0,b_1) is minimum!

Without going into the mathematical details, we present the result here:

$$\beta_1 = \frac{SS_{xy}}{SS_{xx}}$$

$$\beta_0 = \bar{y} - \beta_1 \bar{x}$$

where SS_xy is the sum of cross-deviations of y and x:

$$SS_{xy} = \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) = \sum_{i=1}^{n} y_i x_i - n\bar{x}\bar{y}$$

and SS_xx is the sum of squared deviations of x:

$$SS_{xx} = \sum_{i=1}^{n} (x_i - \bar{x})^2 = \sum_{i=1}^{n} x_i^2 - n(\bar{x})^2$$

We implemented it using both CVXPM and other python method ,we consider data set to work on it

```
#Q4
import cvxpy as cp

import matplotlib.pyplot as plt
import numpy as np


x = np.arange(40)
y = 0.3 * x + 5 + np.random.standard_normal(40)
plt.scatter(x, y)

def estimate_coef(x, y):
    # number of observations/points
    n = np.size(x)

    # mean of x and y vector
    m_x, m_y = np.mean(x), np.mean(y)
```

```
    # calculating cross-deviation and deviation about x
    SS_xy = np.sum(y*x) - n*m_y*m_x
    SS_xx = np.sum(x*x) - n*m_x*m_x

    # calculating regression coefficients
    b_1 = SS_xy / SS_xx
    b_0 = m_y - b_1*m_x

    return(b_0, b_1)

def plot_regression_line(x, y, b):
    # plotting the actual points as scatter plot
    plt.scatter(x, y, color = "m",
               marker = "o", s = 30)

    # predicted response vector
    y_pred = b[0] + b[1]*x

    # plotting the regression line
    plt.plot(x, y_pred, color = "g")

    # putting labels
    plt.xlabel('x')
    plt.ylabel('y')

    # function to show plot
    plt.show()
def plot_regression_with_CVXY():
    b0 = cp.Variable()
    b1 = cp.Variable()
    obj = 0
    for i in range(40):
        obj += (b0 * x[i] + b1 - y[i]) ** 2
    cp.Problem(cp.Minimize(obj), []).solve()
    b0 = b0.value; b1 = b1.value
    plt.scatter(x, y)
    plt.plot(x, b0 * x + b1)

def main():
    # observations

    # estimating coefficients
    b = estimate_coef(x, y)

    print("Estimated coefficients:\nb_0 = {}  \ nb_1 = {}".format(b[0], b[1]))
```

```
    # plotting regression line
    plot_regression_line(x, y, b)
    plot_regression_with_CVXY()

if __name__ == "__main__":
    main()
```

Results:

```
Estimated coefficients:
b_0 = 5.04349604212707  \ nb_1 = 0.30472438488408676
```

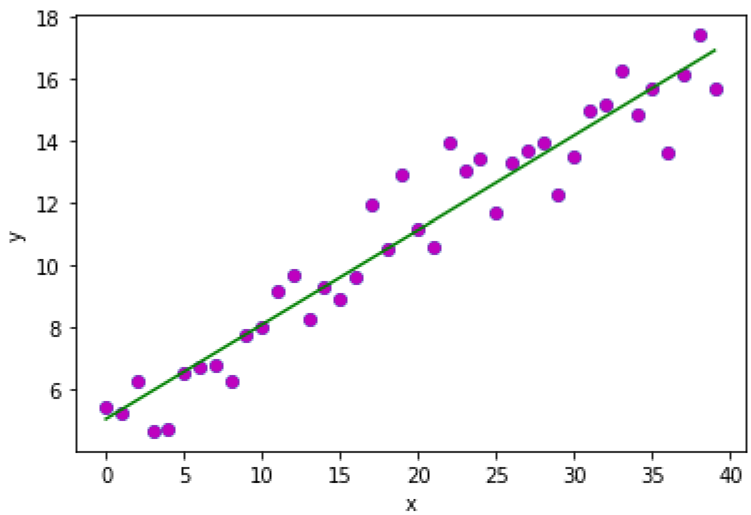

Figure 2- Linear Regression using matplotlib

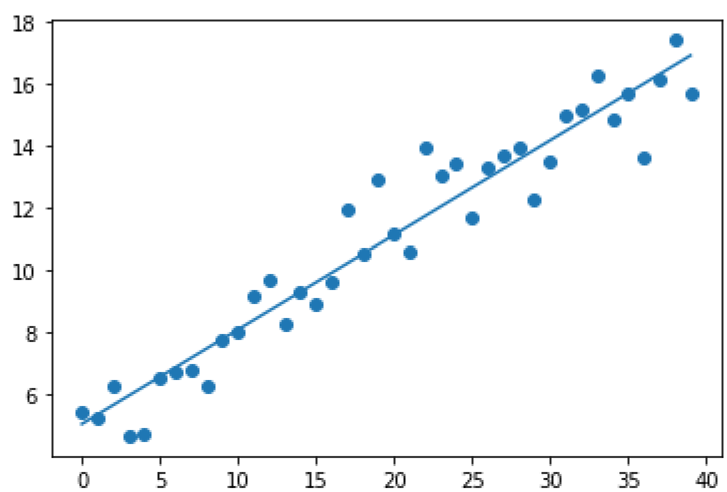

Figure 3- Figure 4- Linear Regression using cvxpy