**CPEG 585 – COMPUTER VISION**

**HOMEWORK 4**

**Name:** Ivan Sangines Escrig

**ID#:** 968606

**Instructor:** Dr. Mahmood

**Date:** February 28, 2019

**TABLE OF CONTENT**

[**INTRODUCTION**](#_Toc446970371) 3

[**SCREENSHOTS:**](#_Toc446970373) 4

[**SOURCE CODE:**](#_Toc446970374) 16

[**CONCLUSION:**](#_Toc446970375) 20

**INTRODUCTION**

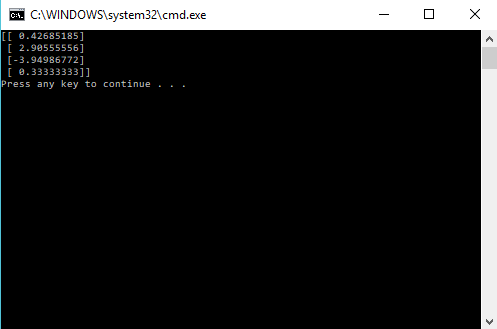
The purpose of this assignment is to get familiar and understand better how linear least squares optimization works. We will be using the linear of squares to find the equation of a line for the given data with least error possible.

**SCREEN SHOTS:**

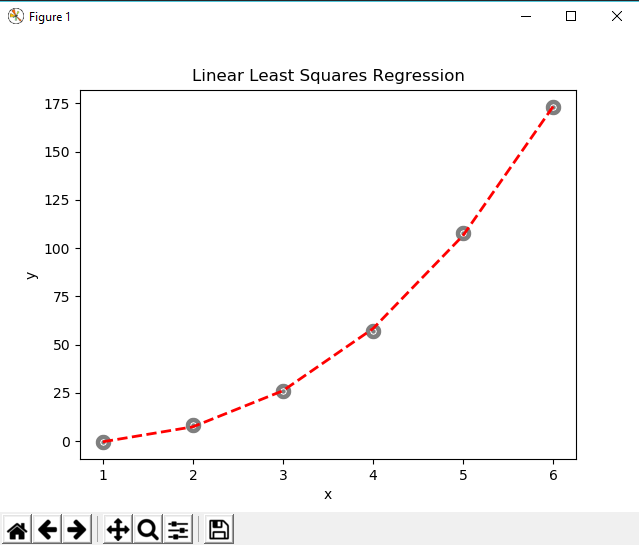
**Problem 1:**

On this example, we are applying the linear of squares optimization for a 3rd order polynomial with the form:

On the following screenshot, we can see the values of a,b,c and d which are the unknown parameters to create the equation of the line.



As we can see on the following picture, the data points are drawed with the black dots. The dashed red line is drawed using the equation of the line obtained by the linear of squares optimization. As we can see the error is minimum since the line contain all the data points.

****

**SOURCE CODE:**

**Problem 2:**

import sys

import numpy as np

import matplotlib.pyplot as plt

def main():

x = np.ndarray((6,1))

y = np.ndarray((6,1))

x[0,0] = 1

x[1,0] = 2

x[2,0] = 3

x[3,0] = 4

x[4,0] = 5

x[5,0] = 6

y[0,0] = -0.6

y[1,0] = 8.3

y[2,0] = 26

y[3,0] = 57

y[4,0] = 108

y[5,0] = 173

a=b=c=d=0

a1=b1=c1=d1=0

a2=b2=c2=d2=0

a3=b3=c3=d3=0

r=r1=r2=r3=0 #these are the y results of the function

for i in range(len(x)):

a += x[i,0] \* x[i,0] \* x[i,0] \* x[i,0] \* x[i,0] \* x[i,0]

b += x[i,0] \* x[i,0] \* x[i,0] \* x[i,0] \* x[i,0]

c += x[i,0] \* x[i,0] \* x[i,0] \* x[i,0]

d += x[i,0] \* x[i,0] \* x[i,0]

a1 += x[i,0] \* x[i,0] \* x[i,0] \* x[i,0] \* x[i,0]

b1 += x[i,0] \* x[i,0] \* x[i,0] \* x[i,0]

c1 += x[i,0] \* x[i,0] \* x[i,0]

d1 += x[i,0] \* x[i,0]

a2 += x[i,0] \* x[i,0] \* x[i,0] \* x[i,0]

b2 += x[i,0] \* x[i,0] \* x[i,0]

c2 += x[i,0] \* x[i,0]

d2 += x[i,0]

a3 += x[i,0] \* x[i,0] \* x[i,0]

b3 += x[i,0] \* x[i,0]

c3 += x[i,0]

d3 += 1

r += y[i,0] \* x[i,0] \* x[i,0] \* x[i,0]

r1 += y[i,0] \* x[i,0] \* x[i,0]

r2 += y[i,0] \* x[i,0]

r3 += y[i,0]

A = np.ndarray((4,4))

A[0,0] = a

A[0,1] = b

A[0,2] = c

A[0,3] = d

A[1,0] = a1

A[1,1] = b1

A[1,2] = c1

A[1,3] = d1

A[2,0] = a2

A[2,1] = b2

A[2,2] = c2

A[2,3] = d2

A[3,0] = a3

A[3,1] = b3

A[3,2] = c3

A[3,3] = d3

ainverse = np.linalg.inv(A)

z = np.ndarray((4,1))

z[0,0] = r

z[1,0] = r1

z[2,0] = r2

z[3,0] = r3

res = np.dot(ainverse,z)

print(res)

area = 10

colors = ['black']

plt.scatter(x, y, s=area, c=colors, alpha=0.5, linewidths=8) #drawing points using X,Y data arrays

plt.title('Linear Least Squares Regression')

plt.xlabel('x')

plt.ylabel('y')

yfitted = x \* x \* x \* res[0,0] + x \* x \* res[1,0] + x \* res[2,0] + res[3,0]

line, = plt.plot(x, yfitted, '--', linewidth=2) #line plot

line.set\_color('red')

plt.show()

if \_\_name\_\_ == "\_\_main\_\_":

sys. exit(int(main() or 0))

**Conclusion:**

After concluding this assignment I have to say I understand better how the linear of squares optimization works, and how can it be applied to find the equation of a line with the least possible error (fitting the max data points). We have practiced this technique in Deep learning however, this assignment gave me more knowledge about the linear of squares by applying it to a polynomial of a higher order.