


Name : Mukul Shingwani

Roll : B20AI023

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
df = pd.read_csv('/content/drive/MyDrive/Optimization_in_ML_labs/lab6/2_column_dataset[linear
```

df

	x	f(x)	
<b>0</b>	0.641677	1.094105	
<b>1</b>	-0.076840	-0.140966	
<b>2</b>	-0.145054	0.287691	
<b>3</b>	-0.101638	-4.788898	
<b>4</b>	-0.829183	-7.958363	
...	...	...	
<b>1995</b>	-0.406379	-3.081566	
<b>1996</b>	-1.690617	-4.626408	
<b>1997</b>	-2.205103	-8.354884	
<b>1998</b>	0.494990	0.665459	
<b>1999</b>	-0.500474	1.292592	

2000 rows × 2 columns

```
data = np.array(df)
```

```
data[0][0]
```

0.6416766584030617

▼ Q1

```
def fun(data, m, c):
    f = 0
    for i in range(len(data)):
        f = f + pow(m*data[i][0] + c - data[i][1], 2)
    f = f/(2*len(data))
    return f
```

```
def gradient_f(data, m, c):
    N = len(data[:,0])
    grad = np.zeros((2, 1), dtype = float)
    grad[0] = 0.0
    grad[1] = 0.0

    for i in range(len(data)):
        grad[0] = grad[0] + data[i][0]*(m*data[i][0] + c - data[i][1])
    grad[0] = grad[0]/N

    for i in range(len(data)):
        grad[1] = grad[1] + m*data[i][0] + c - data[i][1]
    grad[1] = grad[1]/N
    return grad
```

```
m = 0.0
c = 0.0
beta_1 = 1e-4
beta_2 = 0.9
alpha = 1
r = 0.5
k = 0
epsilon = 1e-2
```

```
f0 = fun(data, m, c)
g0 = gradient_f(data, m, c)

while(np.linalg.norm(g0) > epsilon and k < 1000):
    d0 = -1*g0
    alpha = 1
    f1 = fun(data, m + alpha*d0[0], c + alpha*d0[1])
    g1 = gradient_f(data, m + alpha*d0[0], c + alpha*d0[1])

    while f1 > f0 + alpha*beta_1*np.dot(g0.T, d0) or np.dot(g1.T, d0) < beta_2*np.dot(g0.T, d0):
        alpha = alpha*r
        f1 = fun(data, m + alpha*d0[0], c + alpha*d0[1])
        g1 = gradient_f(data, m + alpha*d0[0], c + alpha*d0[1])
    k = k + 1
    print(alpha, f1 - f0)
    m = m + alpha*d0[0]
    c = c + alpha*d0[1]
```

```

f0 = fun(data, m, c)
g0 = gradient_f(data, m, c)

print(m, c, fun(data, m, c), k)

1 [-7.08231009]
1 [-0.02035331]
1 [-5.89773345e-05]
[3.67217187] [0.0716981] [2.00275395] 3

```

## ▼ Q2

```

m = 0.0
c = 0.0
beta_1 = 1e-4
beta_2 = 0.9
alpha = 1
r = 0.5
k = 0
epsilon = 1e-3

```

```

f0 = fun(data, m, c)

while(f0 > 2.003 and k < 1000):
    temp_data = np.array(df.sample(frac = 0.01))
    g0 = gradient_f(temp_data, m, c)
    d0 = -1*g0
    alpha = 1/(k + 1)
    f1 = fun(data, m + alpha*d0[0], c + alpha*d0[1])
    if(f1 < f0):
        m = m + alpha*d0[0]
        c = c + alpha*d0[1]
        f0 = f1
    k = k + 1
print(m, c, fun(data, m, c), k)

```

```
[3.68110591] [0.08878402] [2.00294596] 19
```

## ▼ Q3

```

def fun(data, m1, m2, c):
    f = 0
    for i in range(len(data)):
        f = f + pow(m1*pow(data[i][0], 2) + m2*data[i][0] + c - data[i][1], 2)
    f = f/(2*len(data))

```

```
return f
```

```
def gradient_f(data, m1, m2, c):
    N = len(data[:,0])
    grad = np.zeros((3, 1), dtype = float)

    for i in range(len(data)):
        grad[0] = grad[0] + pow(data[i][0], 2)*(m1*pow(data[i][0], 2) + m2*data[i][0] + c - d
    grad[0] = grad[0]/N

    for i in range(len(data)):
        grad[1] = grad[1] + data[i][0]*(m1*pow(data[i][0], 2) + m2*data[i][0] + c - data[i][1]
    grad[1] = grad[1]/N

    for i in range(len(data)):
        grad[2] = grad[2] + (m1*pow(data[i][0], 2) + m2*data[i][0] + c - data[i][1])
    grad[2] = grad[2]/N
    return grad
```

```
m1 = 0.0
m2 = 0.0
c = 0.0
beta_1 = 1e-4
beta_2 = 0.9
alpha = 1
r = 0.5
epsilon = 1e-2
k = 0
```

```
f0 = fun(data, m1, m2, c)
g0 = gradient_f(data, m1, m2, c)

while(np.linalg.norm(g0) > epsilon and k < 1000):
    d0 = -1*g0
    alpha = 1
    f1 = fun(data, m1 + alpha*d0[0], m2 + alpha*d0[1], c + alpha*d0[2])
    g1 = gradient_f(data, m1 + alpha*d0[0], m2 + alpha*d0[1], c + alpha*d0[2])

    while f1 > f0 + alpha*beta_1*np.dot(g0.T, d0) or np.dot(g1.T, d0) < beta_2*np.dot(g0.T, d
        alpha = alpha*r
        f1 = fun(data, m1 + alpha*d0[0], m2 + alpha*d0[1], c + alpha*d0[2])
        g1 = gradient_f(data, m1 + alpha*d0[0], m2 + alpha*d0[1], c + alpha*d0[2])
    k = k + 1
    print(alpha, f1 - f0, f0)
    m1 = m1 + alpha*d0[0]
    m2 = m2 + alpha*d0[1]
    c = c + alpha*d0[2]
    f0 = fun(data, m1, m2, c)
```

```
g0 = gradient_f(data, m1, m2, c)

print(m1, m2, c, fun(data, m1, m2, c), k)
```

```
0.5 [-0.00183856] [2.01829512]
0.5 [-0.00162116] [2.01645655]
0.5 [-0.00142947] [2.01483539]
0.5 [-0.00126044] [2.01340592]
0.5 [-0.00111114] [2.01214548]
0.5 [-0.00097999] [2.01103407]
0.5 [-0.00086411] [2.01005408]
0.5 [-0.00076193] [2.00918997]
0.5 [-0.00067184] [2.00842804]
0.5 [-0.0005924] [2.0077562]
0.5 [-0.00052235] [2.0071638]
0.5 [-0.00046059] [2.00664145]
0.5 [-0.00040613] [2.00618086]
0.5 [-0.0003581] [2.00577473]
0.5 [-0.00031576] [2.00541663]
0.5 [-0.00027842] [2.00510087]
0.5 [-0.0002455] [2.00482245]
0.5 [-0.00021647] [2.00457695]
0.5 [-0.00019088] [2.00436047]
0.5 [-0.00016831] [2.0041696]
0.5 [-0.0001484] [2.00400129]
0.5 [-0.00013086] [2.00385289]
0.5 [-0.00011538] [2.00372203]
0.5 [-0.00010174] [2.00360665]
0.5 [-8.97102535e-05] [2.00350491]
0.5 [-7.91025673e-05] [2.0034152]
0.5 [-6.97491749e-05] [2.00333609]
0.5 [-6.15017636e-05] [2.00326634]
0.5 [-5.4229558e-05] [2.00320484]
0.5 [-4.7817246e-05] [2.00315061]
0.5 [-4.21631506e-05] [2.00310279]
0.5 [-3.71776171e-05] [2.00306063]
0.5 [-3.27815924e-05] [2.00302345]
0.5 [-2.89053705e-05] [2.00299067]
0.5 [-2.54874881e-05] [2.00296177]
0.5 [-2.24737493e-05] [2.00293628]
0.5 [-1.98163665e-05] [2.00291381]
0.5 [-1.7473203e-05] [2.00289399]
0.5 [-1.54071041e-05] [2.00287652]
0.5 [-1.35853087e-05] [2.00286111]
0.5 [-1.19789295e-05] [2.00284752]
0.5 [-1.05624947e-05] [2.00283554]
0.5 [-9.3135446e-06] [2.00282498]
0.5 [-8.21227518e-06] [2.00281567]
0.5 [-7.24122411e-06] [2.00280746]
0.5 [-6.38499386e-06] [2.00280022]
0.5 [-5.63000758e-06] [2.00279383]
0.5 [-4.96429378e-06] [2.0027882]
0.5 [-4.37729655e-06] [2.00278324]
0.5 [-3.85970813e-06] [2.00277886]
0.5 [-3.40332137e-06] [2.002775]
0.5 [-3.00089953e-06] [2.0027716]
0.5 [-2.6160616e-06] [2.00276859]
```

```
0.5 [-2.0400010e-06] [2.00270859]
0.5 [-2.33318108e-06] [2.00276595]
0.5 [-2.05729675e-06] [2.00276362]
0.5 [-1.81403406e-06] [2.00276156]
0.5 [-1.59953568e-06] [2.00275974]
[-0.00485504] [3.67176844] [0.07355137] [2.00275814] 82
```

## ▼ Q4

```
m1 = 0.0
m2 = 0.0
c = 0.0
beta_1 = 1e-4
beta_2 = 0.9
alpha = 1
r = 0.5
epsilon = 1e-2
k = 0
```

```
f0 = fun(data, m1, m2, c)
g0 = gradient_f(data, m1, m2, c)

while(f0 > 2.003 and k < 1000):
    temp_data = np.array(df.sample(frac = 0.01))
    g0 = gradient_f(temp_data, m1, m2, c)
    d0 = -1*g0
    alpha = 1/(k + 1)
    f1 = fun(data, m1 + alpha*d0[0], m2 + alpha*d0[1], c + alpha*d0[2])
    if(f1 < f0):
        m1 = m1 + alpha*d0[0]
        m2 = m2 + alpha*d0[1]
        c = c + alpha*d0[2]
    k = k + 1
    #print(alpha, f1 - f0)

print(m1, m2, c, fun(data, m1, m2, c), k)
```

```
[0.00277583] [3.67208192] [0.11373576] [2.00378865] 1000
```

---

END OF CODE

---

[Colab paid products](#) - [Cancel contracts here](#)

---

✓ 20s    completed at 10:49 PM ● ✕