Optimisation for Machine Learning - Lab 7
Ayush Abrol B20AI052

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

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

Question1: gradient descent of least square problem

```
dt=pd.read_excel("/content/drive/MyDrive/5 sem/optimization in  ml/lab7/car_price.xlsx")
prices = dt.iloc[: , -1]
data = dt.iloc[:, :-1]
```

```
data
```

	wheelbase	carlength	carwidth	carheight	boreratio
0	88.6	168.8	64.1	48.8	3.47
1	88.6	168.8	64.1	48.8	3.47
2	94.5	171.2	65.5	52.4	2.68
3	99.8	176.6	66.2	54.3	3.19
4	99.4	176.6	66.4	54.3	3.19
...	...	...	...	...	...
200	109.1	188.8	68.9	55.5	3.78
201	109.1	188.8	68.8	55.5	3.78
202	109.1	188.8	68.9	55.5	3.58
203	109.1	188.8	68.9	55.5	3.01
204	109.1	188.8	68.9	55.5	3.78

205 rows × 5 columns

```
X=data.values
Y=prices.values
```

```
Y=Y.reshape(X.shape[0],1)
```

```
# f(β) = 1/(2*N) (sigma i to N) (β1xi1+β2xi2+β3xi3+β4xi4+β5xi5+β6-yi)**2
```

```
#N=205
# defining the function here

def fun(a):
  return (1/(2*205))*(sum(((a[0]*X[:,0:1]+a[1]*X[:,1:2]+a[2]*X[:,2:3]+a[3]*X[:,3:4]+a[4]*X[:,4:5]+a[5]-Y)**2))[0])
```

```
fun(np.array([0,0,0,0,0,0]))
```

```
119890739.74557047
```

```
def gradf(a):   # defining the gradient as per gradient descent method


    beta0=(1/(205))*(sum(((a[0]*X[:,0:1]+a[1]*X[:,1:2]+a[2]*X[:,2:3]+a[3]*X[:,3:4]+a[4]*X[:,4:5]+a[5]-Y)*X[:,0:1]))[0])
    beta1=(1/(205))*(sum(((a[0]*X[:,0:1]+a[1]*X[:,1:2]+a[2]*X[:,2:3]+a[3]*X[:,3:4]+a[4]*X[:,4:5]+a[5]-Y)*X[:,1:2]))[0])
    beta2=(1/(205))*(sum(((a[0]*X[:,0:1]+a[1]*X[:,1:2]+a[2]*X[:,2:3]+a[3]*X[:,3:4]+a[4]*X[:,4:5]+a[5]-Y)*X[:,2:3]))[0])
    beta3=(1/(205))*(sum(((a[0]*X[:,0:1]+a[1]*X[:,1:2]+a[2]*X[:,2:3]+a[3]*X[:,3:4]+a[4]*X[:,4:5]+a[5]-Y)*X[:,3:4]))[0])
    beta4=(1/(205))*(sum(((a[0]*X[:,0:1]+a[1]*X[:,1:2]+a[2]*X[:,2:3]+a[3]*X[:,3:4]+a[4]*X[:,4:5]+a[5]-Y)*X[:,4:5]))[0])
    beta5=(1/(205))*(sum(((a[0]*X[:,0:1]+a[1]*X[:,1:2]+a[2]*X[:,2:3]+a[3]*X[:,3:4]+a[4]*X[:,4:5]+a[5]-Y)))[0])

    return np.array([beta0,beta1,beta2,beta3,beta4,beta5])
```

```
gradf(np.array([0,0,0,0,0,0]))
```

```
array([-1338824.02495854, -2377782.54980195,  -887988.47311854,
        -715607.84748293,   -45399.28708639,   -13276.71057073])
```

```
# initially, a0=[0,0,0,0,0,0]

a0,beta1,beta2,r,eps,iter1,countf=np.array([0,0,0,0,0,0]),pow(10,-4),0.9,0.5,pow(10,-3),0,0
print("x",iter1," : ",a0)
print("f",iter1," : ",fun(a0))
```

```
print("g",iter1," : ",gradf(a0))
print("\n==========\n")
while np.linalg.norm(gradf(a0))>0.001 and iter1<1000:
  d0,alpha=-gradf(a0),1   # steepest descent
  while fun(a0+alpha*d0)>fun(a0)+alpha*beta1*np.dot(gradf(a0).T,d0) or np.dot(gradf(a0+alpha*d0).T,d0)<beta2*np.dot(gradf(a0).T,d0) and alpha>pow(10,-5): # until a
    alpha=alpha*r
    countf += 1

  # print(fun(a0+alpha*d0)-fun(a0))
  # print("x",iter1+1," : ",a0+alpha*d0)
  # print("f",iter1+1," : ",fun(a0+alpha*d0))
  # print("g",iter1+1," : ",gradf(a0+alpha*d0))
  # print("\n==========\n")
  a0,iter1=a0+alpha*d0,iter1+1   # updating x
print("\n==========\n")
print("x",iter1," : ",a0)
print("f",iter1," : ",fun(a0))
print("g",iter1," : ",gradf(a0))

print("Iterations: ",iter1)
print("countf: ",countf)
```

```
x 0  :  [0 0 0 0 0 0]
f 0  :  119890739.74557047
g 0  :  [-1338824.02495854 -2377782.54980195  -887988.47311854  -715607.84748293
   -45399.28708639   -13276.71057073]

==========


==========

x 1000  :  [ -86.94078764  343.82938545 -192.82664778 -468.68424383    7.98884996
   -9.71132641]
f 1000  :  19420686.0810273
g 1000  :  [ -3845.00052452 -12768.91930736   -622.3799388    4769.91004341
   -319.3983298     108.67174386]
Iterations:  1000
countf:  14446
```

```
R=12
r=2
a_1=[R*r, 100+R*2*r, (R-1)*r, 54.3, r*R]
output=a0[0]*a_1[0]+a0[1]*a_1[1]+a0[2]*a_1[2]+a0[3]*a_1[3]+a0[4]*a_1[4]+a0[5]
print(output)
```

```
19290.45052503653
```

QUESTION 2: Stochastic gradient with 10 random points in every iterations with 100 iterations

```
data = pd.read_excel("/content/drive/MyDrive/5 sem/optimization in  ml/lab7/Concrete_Data_Yeh.xlsx")
x15,y15=data.values[:,0:-1],data.values[:,-1]
```

```
import random
temp=[]
list1=range(len(x15))
for i in range(10):
    temp.append(random.choice(list1))
x1 ,y1=data.values[temp,0:5],data.values[temp,5]
```

```
def func(x):
    n=len(x1)
    sum=0
    # print(x1[0].T.shape, x.shape, y1[0].shape)
    for i in range(n):
        # print(x1[i].T, x, y1[i])
        temp=(np.dot(x1[i].T, x)- y1[i])**2
        print(temp)
        sum += temp
    sum=sum/(2*n)
    return sum

def grad(x):
    n=len(x1)
    sum=np.array([0.0, 0.0, 0.0, 0.0, 0.0])
    for i in range(n):
        sum+=np.dot(x1[i], (np.dot(x1[i].T, x)- y1[i]))
    sum=sum/n
    return sum
# f=np.array([0, 0, 0, 0, 0])
# print(func(f))
```

```
epsilon=0.01
x0=np.array([0,0,0,0,0],dtype=float)
b1=0.0001
b2=0.9
r=0.5
i=0
a=1
no_itr=0
of_v=[]
of_v.append(func(x0))
```

```
while (np.linalg.norm(grad(x0))>epsilon):
    d0=-grad(x0)

    if(no_itr==100):
        break
    no_itr=no_itr+1
    argimo_val=b1*a*(np.dot(grad(x0).reshape(1,-1),d0.reshape(-1,1)))


    temp=[]
    for i in range(10):
        temp.append(random.choice(list1))

    x1,y1=data.values[temp,0:5],data.values[temp,5]
    a=1
    while((func(x0+a*d0)> func(x0)+b1*a*(np.dot(grad(x0).T,d0))) or (np.dot(grad(x0+a*d0).T,d0)<b2*(np.dot(grad(x0).T,d0)))):

        a=a*r
    x0=x0+a*d0
    print("objective function value : "+ str(func(x0)))
    of_v.append(func(x0))
    print(x0)

print("no of itr : " +str(no_itr))
```

```
51531.03688808712
28018.290706157404
4710.589043903452
4877.026817717713
30467.171810619275
67105.92260341052
2024.561661435349
128740.15064187611
55437.48462903886
32532.226562338714
51531.16271448856
28018.202494206416
4710.564133831395
4877.060389199134
30467.111454156246
67105.83211352225
2024.5845586037958
128739.98535968634
55437.56443486478
32532.15975781967
51531.03688808712
28018.290706157404
4710.589043903452
4877.026817717713
30467.171810619275
67105.92260341052
2024.561661435349
```