**Abdul Moiz Asif**

**CMS:263802**

**BSCS 8C**

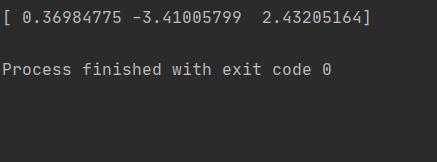
**Lab#2**

**AI**

**Task#1**

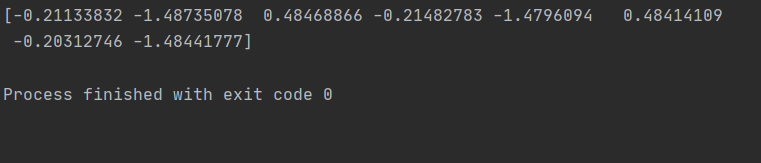
#!/usr/bin/env python  
# coding: utf-8  
  
# In[1]:  
  
  
import numpy as np  
points=[  
 (np.array([1,0,1]),2),  
 (np.array([1,1,0]),-3),  
 (np.array([2,1,3]),5),  
 (np.array([1,2,1]),-4),  
 (np.array([0,1,2]),1),  
 (np.array([0,0,1]),3)  
]  
  
  
# In[3]:  
  
  
def cost\_function(w):  
 return np.sum((w.dot(x)-y)\*\*2 for x,y in points)/len(points)  
  
  
# In[4]:  
  
  
def derivative(w):  
 return np.sum(2\*(w.dot(x)-y)\*x for x,y in points)/len(points)  
  
  
# In[6]:  
  
  
def gradientDescent():  
 w=np.zeros(3)  
 step=0.01  
 for i in range(1000):  
 cost=cost\_function(w)  
 dF=derivative(w)  
 w=w-step\*dF  
 print(w)  
  
  
# In[9]:  
  
  
gradientDescent()  
  
# In[ ]:

**Output**

****

**Task#2**

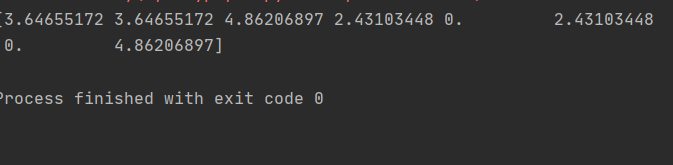
import numpy as np  
from numpy import random  
  
def dF(w):  
 return sum(2\*(w.dot(x[0])-y)\*x[0] for x,y in points)/len(points)  
  
  
def gradientDescent():  
 w=np.zeros(8)  
 step=0.01  
 for i in range(10):  
 dFu=dF(w)  
 w=w-step\*dFu  
 print(w)  
  
  
  
  
  
points = [(np.array([0,0,0,0,0,0,0,0]),0)]  
w1=[0.36984775,-3.41005799,2.43205167,0.36984775,-3.41005799,2.43205167,0.36984775,-3.41005799]  
for i in range(100000):  
 x = [(np.array(random.randint(5,size=(8))))]  
 y=np.sum(np.multiply(x,w1))+0.3  
 point=[x,y]  
 points.append(point)  
  
points= np.delete(points,(0),axis=0)  
print(points)  
gradientDescent()

**Output  
**

**Task#3**

import numpy as np  
from numpy import random  
  
w1 = [2, 4, 8, 3, 9, 1, 4, 3]  
  
points = [(np.array([0, 0, 0, 0, 0, 0, 0, 0]), 0)]  
for i in range(100000):  
 x = [(np.array(random.randint(5, size=(8))))]  
 y\_multi = np.multiply(x, w1)  
 y = np.sum(y\_multi) + 0.5  
 point = [x, y]  
  
 points.append(point)  
  
points = np.delete(points, (0), axis=0)  
  
index = np.random.choice(points.shape[0], 1, replace=False)  
x\_random = points[index]  
  
  
def dF(w):  
 return sum(2 \* (w.dot(x[0]) - y) \* x[0] for x, y in x\_random) / len(x\_random)  
  
  
def gradientDescent(dF):  
 w = np.zeros(8)  
 step = 0.01  
 for i in range(1000):  
 gradient = dF(w)  
 w = w - step \* gradient  
 print(w)  
  
  
gradientDescent(dF)

**Output:**

****