# Department of Computing

# CS370: Artificial Intelligence

# Class: BSCS-8C

# Lab 02: Optimization

# Date: 05-03-2021

# Time: 10:00-1:00

# Instructor: Dr. Hashir Kiani

**LAB TASKS**

**Task 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)]

def F(w):

return sum((w.dot(x)-y)\*\*2 for x,y in points)/len(points)

def dF(w):

return sum(2\*(w.dot(x)-y)\*x for x,y in points)/len(points)

def gradientDescent(F,dF):

w=np.zeros(3)

step=0.01

for i in range(1000):

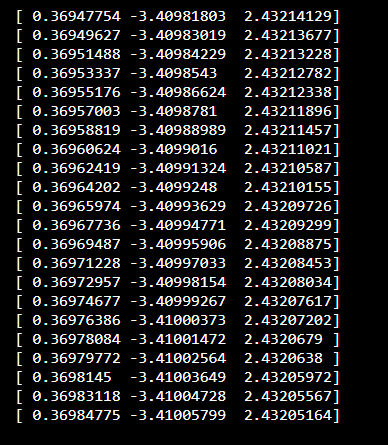
value=F(w)

gradient=dF(w)

w=w-step\*gradient

print(w)

gradientDescent(F,dF)



**Task 2:**

import numpy as np

import random

w1 = [2,4,5,6,8,9,4,2]

x\_values = np.random.rand(10000, 8)

points = []

for x in x\_values:

point = (np.array(x), np.dot(x,w1) + random.random())

points.append(point)

def loss\_function(w):

return sum((w.dot(x) - y)\*\*2 for x,y in points)/len(points)

def derivative\_function(w):

return sum((2\*(w.dot(x) - y))\*x for x,y in points)/len(points)

def gradientDescent(F, dF) :

w = np.zeros(8)

step = 0.001

for \_ in range(10000):

value = F(w)

gradient = dF(w)

w = w - step \* gradient

# print("error now is: " , value)

print(w)

gradientDescent(loss\_function, derivative\_function)

**Text

Description automatically generated**

**Task 3:**

import numpy as np

import random

w1 = [2,4,5,6,8,9,4,2]

x\_values = np.random.rand(10000, 8)

points = []

for x in x\_values:

point = (np.array(x), np.dot(x,w1) + random.random())

points.append(point)

def loss\_function(w):

return sum((w.dot(x) - y)\*\*2 for x,y in points)/len(points)

def gradientDescent(F) :

w = np.zeros(8)

step = 0.001

for x, y in points:

value = F(w)

gradient = (2\*(w.dot(x) - y))\*x

w = w - step \* gradient

# print("error now is: " , value)

print(w)

gradientDescent(loss\_function)

Text

Description automatically generated