Міністерство освіти та науки України

Львівський національний університет імені Івана Франка

факультет електроніки та комп’ютерних технологій

кафедра системного проектування

ЗВІТ

про виконання лабораторної роботи №2

«Градієнтний спуск»

виконала студентка групи ФЕС-32

Леськів Лілія Романівна

перевірив ст.викладач Рибак А.В.

ЛЬВІВ-2019

**ТЕОРЕТИЧНІ ВІДОМОСТІ**

Градієнтний спуск – ітераційний алгоритм оптимізації для знаходження мінімуму або максимуму функкії.

Метод градієнтного спуску : коли у нас є певний простір коефіцієнтів і ми, рухаючись ітерація за ітерацією в напрямку зменшення помилки( а у випадку логістичної регресії – у напрямку максимізації правильної класифікації), крок за кроком наближаємось до глобального мінімуму функції, якій відповідає дана логістична регресія.

В результаті ми отримуємо певний вектор коефіцієнтів, який при підставлянні в рівняння логістичної регресії максимізує правильність класифікації для випадкової точки.

**ХІД РОБОТИ**

Створити програму на мові програмування Python, для обчислення градієнтного спуску, яка включає в себе клас GRADIUNT, який містить метод f(self, x)- задана функція, fp(self, x, eps) - похідна від заданої функції, compute\_gd(self, alpha, eps, epoch) – метод градієнтного спуску, show\_graph(self) – графік, на якою представлено саму функції, вихідні точки градієнтного спуску і точку мінімуму; і точку входу, де ми створюємо об’єкт класу і викликаємо ці методи.

Використані бібліотеки – math, numpy, scipy.misc matplotlib.pyplot

**Лістинг програми**

import math  
import numpy as np  
from scipy.misc import derivative  
from matplotlib import pyplot as plt  
  
  
class GRADIUNT:  
 def f(self, x):  
 return (x - 5) \*\* 2  
  
 def fp(self,x,eps):  
 return (self.f(x+eps)-self.f(x))/eps  
  
 def compute\_gd(self, alpha=0.1, eps=0.00001, epoch=1000):  
 x = []  
 y = []  
 data = {}  
 x.append(0)  
 y.append(self.f(x[0]))  
 for i in range(1, epoch):  
 #x.append(x[i - 1] - alpha \* self.fp(x[i-1],eps))  
 x.append(x[i - 1] - alpha \* derivative(self.f, x[i - 1]))  
 y.append(self.f(x[i]))  
 data[x[i]] = y[i]  
 if abs(x[i] - x[i - 1]) <= eps:  
 return data  
 return data  
  
 def show\_graph(self):  
 t = np.linspace(0, 10, 20)  
 yt = self.f(t)  
 data\_gd = self.compute\_gd()  
 x = data\_gd.keys()  
 y = data\_gd.values()  
 plt.plot(t, yt, 'g^',  
 x, y, 'ro--')  
 plt.legend(['y = (x - 5)\*\*2',  
 'GD method'])  
 plt.annotate('min', xy=(5, 0), xytext=(5, 2),  
 arrowprops=dict(facecolor='yellow', shrink=1))  
 plt.grid(True)  
 plt.show()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 a = GRADIUNT()  
 data = a.compute\_gd()  
 a.show\_graph()  
 print('data=', data)

**Результати роботи**

data= {1.0: 16.0, 1.8: 10.240000000000002, 2.4400000000000004: 6.553599999999998, 2.9520000000000004: 4.194303999999998, 3.3616: 2.6843545599999996, 3.68928: 1.7179869183999996, 3.9514240000000003: 1.0995116277759995, 4.1611392: 0.7036874417766399, 4.32891136: 0.4503599627370493, 4.4631290880000005: 0.2882303761517112, 4.570503270400001: 0.184467440737095, 4.65640261632: 0.11805916207174093, 4.725122093056: 0.07555786372591429, 4.7800976744448: 0.04835703278458515, 4.82407813955584: 0.030948500982134555, 4.859262511644672: 0.019807040628566166, 4.8874100093157375: 0.012676506002282305, 4.90992800745259: 0.008112963841460612, 4.927942405962073: 0.005192296858534741, 4.942353924769658: 0.0033230699894622544, 4.953883139815726: 0.0021267647932558427, 4.9631065118525814: 0.0013611294676837131, 4.9704852094820655: 0.0008711228593175555, 4.976388167585652: 0.0005575186299632523, 4.981110534068522: 0.000356811923176468, 4.984888427254818: 0.0002283596308329288, 4.987910741803854: 0.00014615016373307872, 4.990328593443083: 9.353610478917726e-05, 4.992262874754466: 5.986310706507345e-05, 4.993810299803573: 3.83123885216492e-05, 4.995048239842858: 2.451992865385725e-05, 4.996038591874287: 1.5692754338465825e-05, 4.996830873499429: 1.0043362776619255e-05, 4.9974646987995435: 6.427752177036323e-06, 4.997971759039634: 4.113761393304687e-06, 4.9983774072317075: 2.632807291715e-06, 4.998701925785366: 1.6849966666971388e-06, 4.998961540628293: 1.0783978666865378e-06, 4.999169232502634: 6.901746346793842e-07, 4.999335386002107: 4.417117661952781e-07, 4.999468308801686: 2.826955303647891e-07, 4.9995746470413485: 1.8092513943361614e-07, 4.999659717633079: 1.1579208923763523e-07, 4.9997277741064625: 7.410693711227997e-08, 4.99978221928517: 4.7428439751781807e-08, 4.999825775428136: 3.035420144107846e-08, 4.999860620342509: 1.9426688922339734e-08, 4.999888496274007: 1.243308091029743e-08, 4.999910797019206: 7.957171782526972e-09, 4.999928637615365: 5.092589940842615e-09, 4.999942910092292: 3.2592575620987087e-09, 4.999954328073834: 2.0859248397431735e-09, 4.999963462459067: 1.3349918974486118e-09}

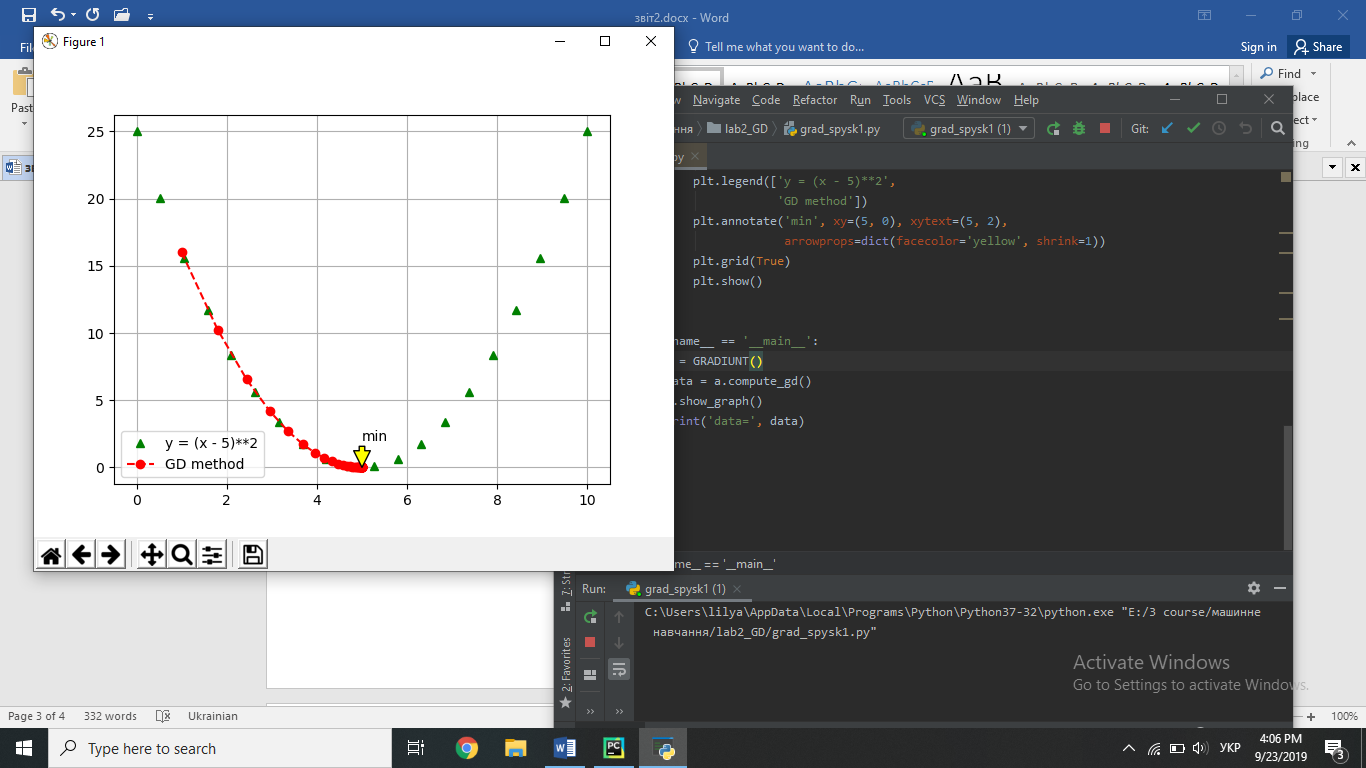


рис.1 Графік функції і точки градієнтного спуску

**ВИСНОВОК**

На даній лабораторній роботі я реалізувала метод градієнтного спуску, який використовують для мінімізує помилку.