Міністерство освіти і науки України

Львівський національний університет імені Івана Франка

факультет електроніки та комп’ютерних технологій

кафедра системного проектування

**ЗВІТ**

про виконання лабораторної роботи №2

“Реалізація методу градієнтного спуску ”

виконала студентка групи ФеС-32

Богоніс Є.А.

перевірив старший викладач

Рибак А.В.

Львів 2019

**Теоретичні відомості**

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

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.

**Завдання:**

1. Побудувати графік функції (функцію обираємо самостійно).

2. Знаходимо похідну за допомогою скінченних різниць, а для тестування використовуємо бібліотеку «derivative».

3. Реалізація градієнтного спуску для пошуку глобального мінімуму функції.

4. Візуалізація результату градієнтного спуску на графіку, як друга легенда відповідно до 1-го пункту завдань.

5. Виведення результату (глобального мінімуму) градієнтного спуску та візуалізація на графіку у вигляді маркера.

6. Тестування методу (підбір параметрів: alpha, eps; порівняння результатів пошуку похідної функції реалізованої з використанням скінченних різниць та бібліотеки) та аналіз результатів методу градієнтного спуску.

**Хід роботи**

У мові Python завдання можна реалізувати таким чином:

*import math as m1* ***#підключаємо необхідні бібліотеки***

*import numpy as np*

*from scipy.misc import derivative*

*from matplotlib import pyplot as plt*

*import pandas as pd*

*import random*

*class GRADIENT:* ***#створюємо клас в якому будуть міститись функції***

*def f(self, x):* ***#функція параболи***

*return (x - 5) \*\* 2*

**#self** - при виклику методу класу йому передається посилання на об'єкт. Перебуваючи всередині методу можна #отримати доступ до нього посиланням через self.

*def fp(self,x,eps):* ***#похідна***

*return (self.f(x+eps)-self.f(x))/eps*

*def compute\_GD(self, alpha=0.1, eps=0.0001, epoch=1000):* ***#градієнтний спуск***

*x = []*

*y = []*

*data = {}*

*x.append(0)*

*y.append(self.f(x[0]))*

*for i in range(1, epoch):*

*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* ***#запис обрахунків по досягненні глобального мінімуму***

*return data*

***#тепер малюємо графік градієнтного списку***

*def graphik(self):*

*t = np.linspace(0, 15, 30)* ***#30 чисел, від 0 до 15***

*yt = self.f(t)* ***#похідні від чисел***

*data\_gd = self.compute\_GD()*

*x = data\_gd.keys()*

*y = data\_gd.values()*

*plt.plot(t, yt, 'bo--',* ***#графік град.спуску, голубий, кружечками***

*x, y, 'co--')* ***#графік параболи, синій, кружечками***

*plt.xlabel('x')*

*plt.ylabel('y')*

*plt.legend(['y = (x - 5)\*\*2',*

*'Gradient Descent'])* ***#позначки, ‘оголошення’***

*plt.annotate('minimum', xy=(5,0), xytext=(10, 2),*

*arrowprops=dict(facecolor='red',width=2, shrink=0.045))* ***#стрілочка***

*plt.show()*

*if \_\_name\_\_ == '\_\_main\_\_':* ***#всі ф-ції перенаправлю в нейм і запускається весь код***

*a = GRADIENT()*

*data = a.compute\_GD()*

*a.graphik()*

*print('data=', data)*

Результат програми:

Графік градієнтного спуску зображений на рис.1

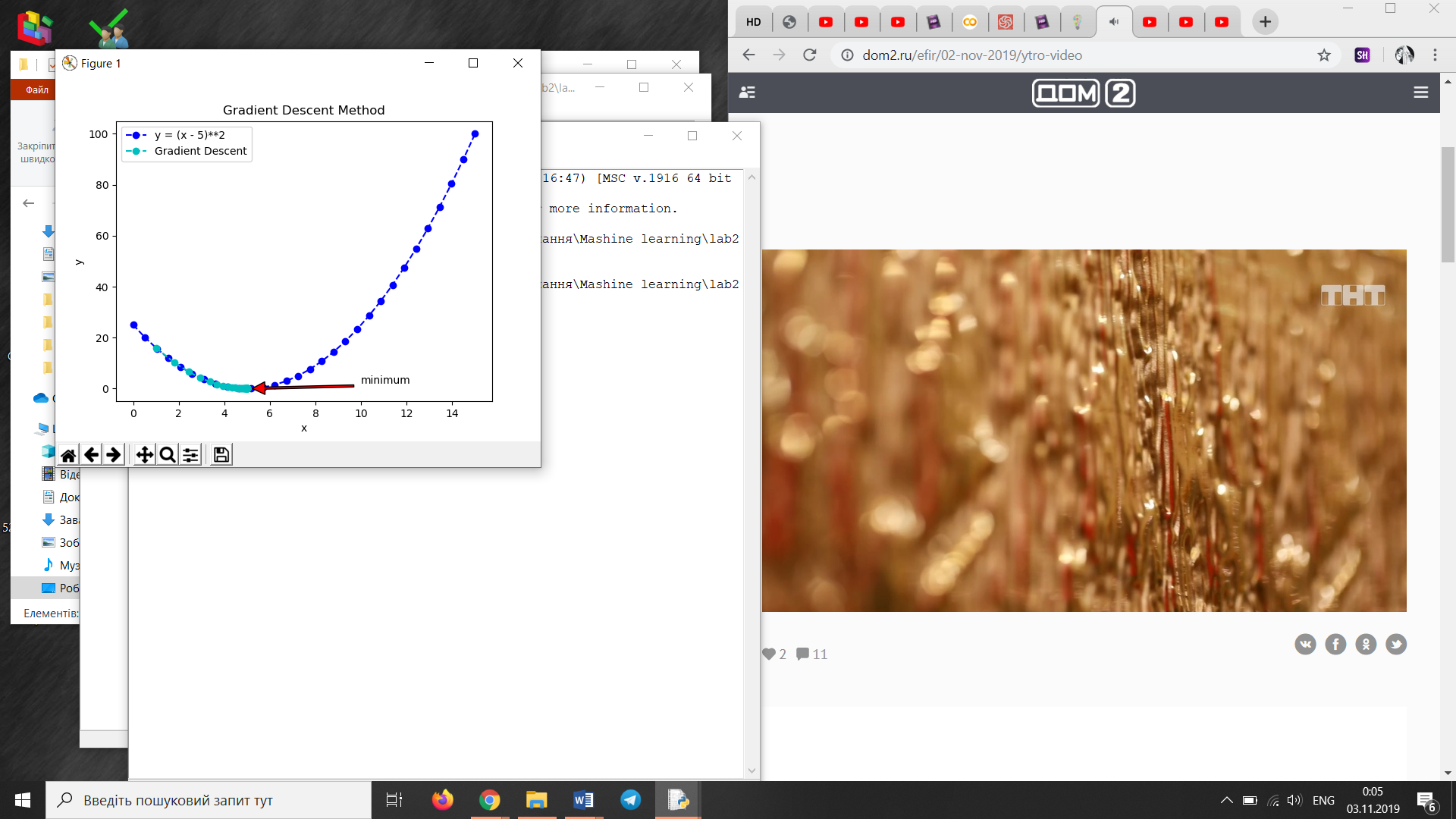


Рис.1 Градієнтний спуск(графік)

Також є такі дані в консолі:

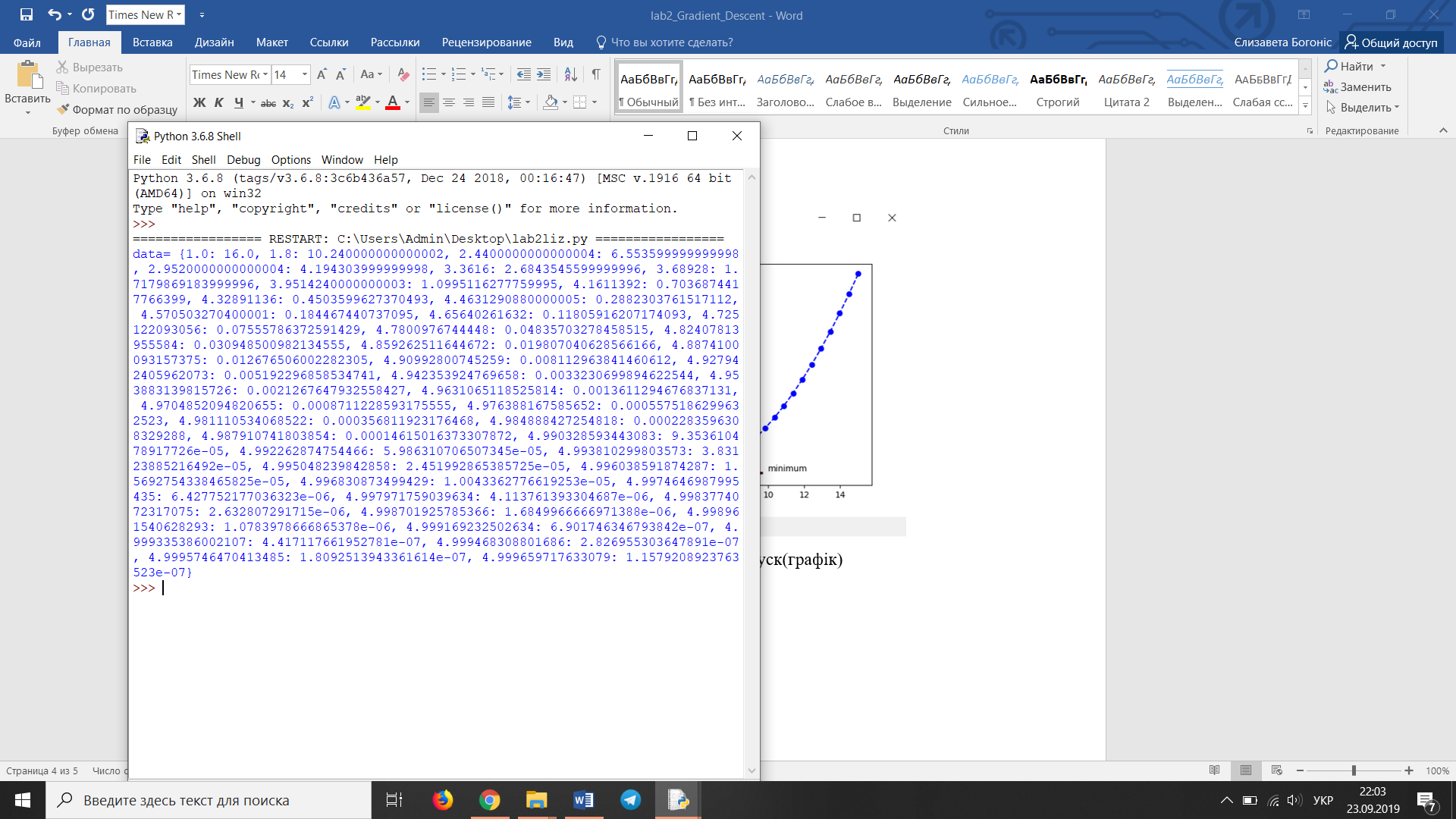


Рис.2 Обрахунки для знаходження глобального мінімуму

**Висновок:** на даній лабораторній роботі я реалізувала метод градієнтного спуску, який використовують для мінімізації помилки і здіснила тестування цього методу підбором параметрів alpha, eps.