Міністерство освіти і науки України

Національний університет «Одеська політехніка»

Інститут комп'ютерних систем

Кафедра інформаційних систем

Лабораторна робота №5

З дисципліни: "Інтелектуальний аналіз даних"

Тема: " Введення в нейронні мережі**.**"

Підготувала студентка

Групи АІ – 212

Козуб К.О

Перевірили:

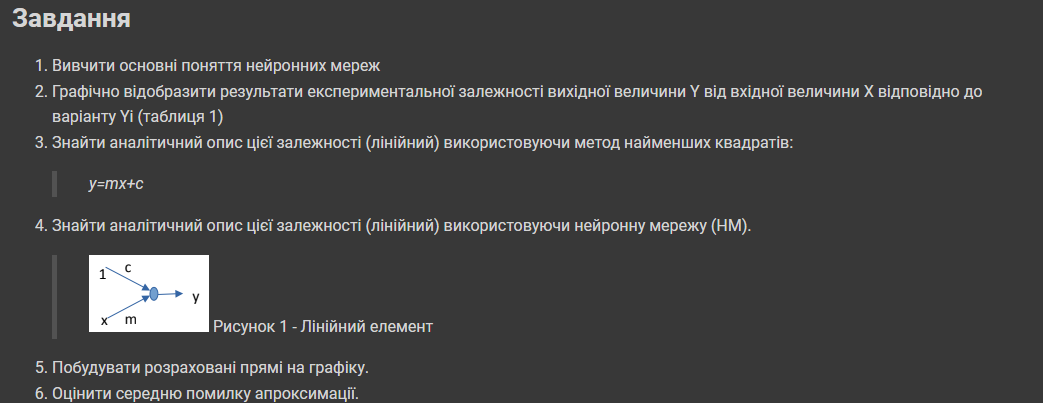
Антощук С.Г

Кошутіна Д.В

Одеса 2023

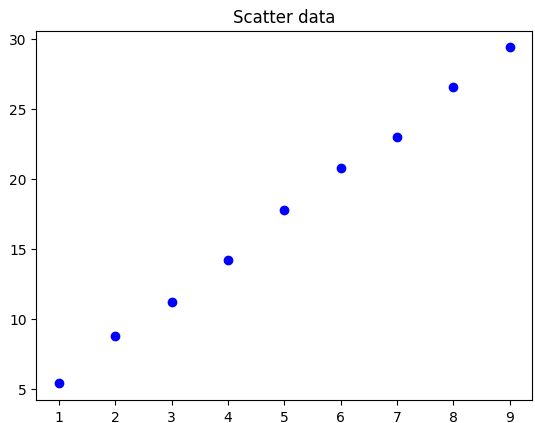
**Мета:** Метою роботи є знайомство з основними поняттями нейронних мереж.

**Завдання**

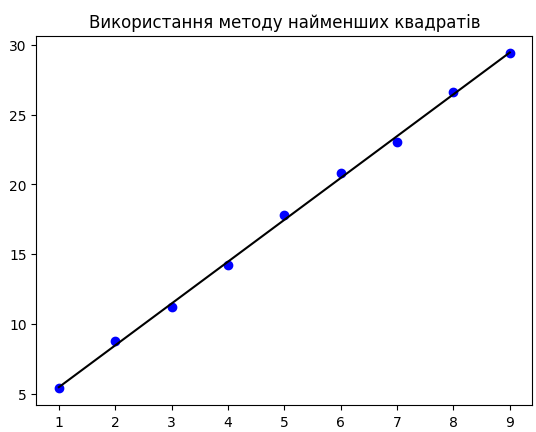
****

<https://colab.research.google.com/drive/1BZcQN_cl1adCKgWPbfNrWdZGs5cT8mkG?usp=sharing>

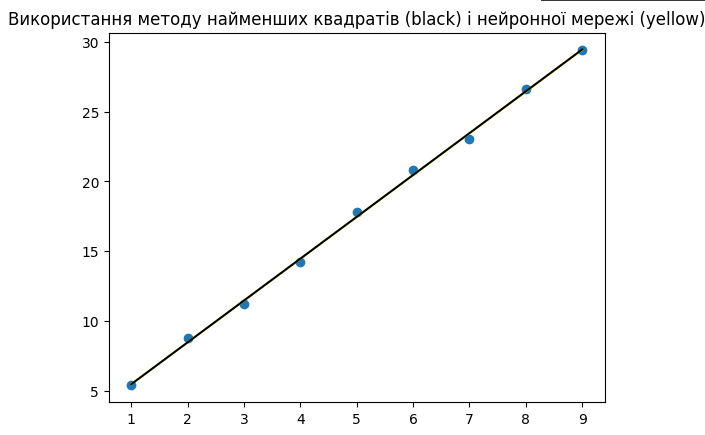
**Хід роботи**



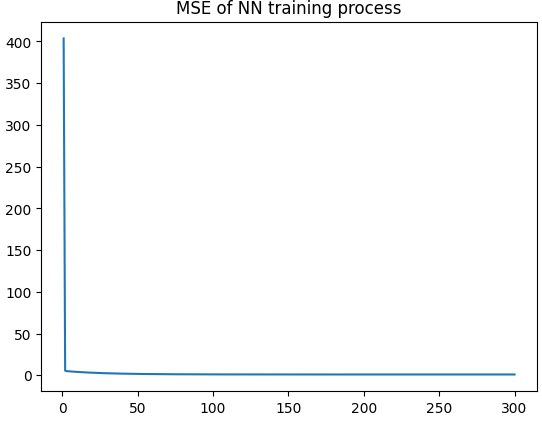
Малюнок 1. Scatter date



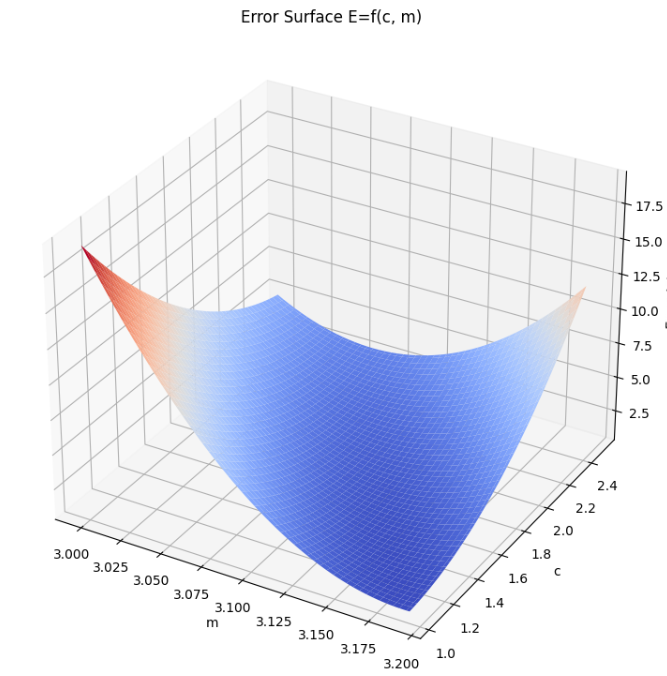
Малюнок 2. Використання методу найменших квадратів



Малюнок 3. Використання методу найменших квадратів(black) і нейронної мережі(yellow)



Малюнок 4. MSE of NN training process



Малюнок 5. Error Surface E=f(c, m)

Висновок: На лабораторній роботі 5. Я знайомилась з основними поняттями нейронних мереж.

Додаток А

import numpy as np

import matplotlib.pyplot as plt

# Створюємо масиви X і Y для даних

X = np.arange(1, 10, 1)

Y = np.array([5.4, 8.8, 11.2, 14.2, 17.8, 20.8, 23, 26.6, 29.4])

# Створюємо діаграму розсіювання з даними X і Y, використовуючи синій колір

plt.figure()

plt.scatter(X, Y, c='blue')

plt.title("Scatter data")

# Оголошення функції для обчислення параметрів методом найменших квадратів

def Least\_Squares\_Method(X, Y):

n = len(X)

sum\_X = np.sum(X)

sum\_Y = np.sum(Y)

sum\_XY = np.sum(X \* Y)

sum\_X\_squared = np.sum(X \*\* 2)

m\_Emin = (n \* sum\_XY - sum\_X \* sum\_Y) / (n \* sum\_X\_squared - sum\_X \*\* 2)

c\_Emin = (sum\_Y - m\_Emin \* sum\_X) / n

Emin = np.sum((Y - (m\_Emin \* X + c\_Emin)) \*\* 2)

return m\_Emin, c\_Emin, Emin

# Оголошення функції для навчання нейронної мережі

def neuron\_network(X, Y, n\_epochs):

new\_m = 0.7

new\_c = 0.5

p = 0.01

E = np.zeros(n\_epochs)

matrix\_m = np.zeros(n\_epochs)

matrix\_c = np.zeros(n\_epochs)

nx = len(X)

for i in range(n\_epochs):

Y\_nn = np.zeros(nx)

for j in range(nx):

m = new\_m

c = new\_c

xj = X[j]

Y\_nn[j] = m \* xj + c

beta = Y[j] - Y\_nn[j]

delta\_m = p \* beta \* xj

delta\_c = p \* beta

new\_m = m + delta\_m

new\_c = c + delta\_c

E[i] = sum((Y - Y\_nn) \*\* 2)

matrix\_m[i] = m

matrix\_c[i] = c

m = new\_m

return m, c, E, matrix\_m, matrix\_c

# Обчислення параметрів методом найменших квадратів

m\_Emin, c\_Emin, Emin = Least\_Squares\_Method(X, Y)

# Вивід результатів методу найменших квадратів

print('\n Метод найменших квадратів')

print('\n m\_Emin', m\_Emin)

print('\n c\_Emin', c\_Emin)

print('\n Emin', Emin, '\n')

plt.figure()

# Створюємо діаграму розсіювання з даними X і Y, використовуючи синій колір

plt.scatter(X, Y, c='blue')

# Будуємо графік методу найменших квадратів, використовуючи чорний колір

plt.plot(X, m\_Emin \* X + c\_Emin, 'black')

# Встановлюємо заголовок для графіку

plt.title("Використання методу найменших квадратів")

# Оголошення кількості епох для навчання нейронної мережі

n\_epochs = 300

# Навчання нейронної мережі та отримання параметрів

m, c, E, matrix\_m, matrix\_c = neuron\_network(X, Y, n\_epochs)

# Вивід результатів нейронної мережі

print("m=", m)

print("c=", c)

print("E=", E)

print("matrix\_m=", matrix\_m)

print("matrix\_c=", matrix\_c)

# Обчислення значень Y з використанням навчених параметрів

Y\_res= np.zeros(len(X))

for i in range(len(X)):

Y\_res[i]=m\*X[i]+c

plt.figure()

# Будуємо графіки методу найменших квадратів (чорний) і нейронної мережі (жовтий)

plt.plot(X, Y\_res, 'yellow')

plt.plot(X, m\_Emin \* X + c\_Emin, 'black')

# Додаємо точки даних на графік

plt.scatter(X, Y)

# Встановлюємо заголовок для графіку

plt.title("Використання методу найменших квадратів (black) і нейронної мережі (yellow)")

# Створюємо нове графічне вікно для відображення зміни помилки навчання нейронної мережі

plt.figure()

xE = np.arange(1, n\_epochs + 1, 1)

# Будуємо графік середньоквадратичної помилки (MSE) під час навчання

plt.plot(xE, E)

# Встановлюємо заголовок для графіку

plt.title('MSE of NN training process')

# Створюємо сітку значень m і c для побудови поверхні помилки

m\_values = np.linspace(min(matrix\_m), max(matrix\_m), 100)

c\_values = np.linspace(min(matrix\_c), max(matrix\_c), 100)

m\_mesh, c\_mesh = np.meshgrid(m\_values, c\_values)

# Створюємо масив для зберігання помилок

E\_mesh = np.zeros\_like(m\_mesh, dtype=np.float64)

# Обчислюємо помилку для кожної комбінації m і c

for i in range(len(m\_values)):

for j in range(len(c\_values)):

Y\_nn\_mesh = m\_values[i] \* X + c\_values[j]

E\_mesh[i, j] = sum((Y - Y\_nn\_mesh) \*\* 2)

# графік

fig = plt.figure(figsize=(9, 9))

ax = fig.add\_subplot(111, projection='3d')

ax.plot\_surface(m\_mesh, c\_mesh, E\_mesh, cmap='coolwarm')

ax.set\_xlabel('m')

ax.set\_ylabel('c')

ax.set\_zlabel('Error (E)')

ax.set\_title('Error Surface E=f(c, m)')

plt.show()