

ЛАБОРАТОРНА РОБОТА № 5

РОЗРОБКА ПРОСТИХ НЕЙРОННИХ МЕРЕЖ

<https://github.com/MaximVengel/AI>

Завдання №1. Створити простий нейрон.

LR_5_task_1.py

```
import numpy as np

def sigmoid(x):
    return 1 / (1 + np.exp(-x))

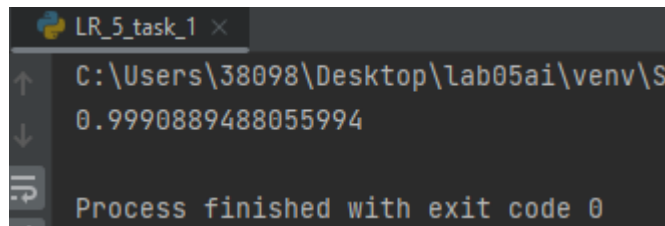
class Neuron:
    def __init__(self, weights, bias):
        self.weights = weights
        self.bias = bias

    def feedforward(self, inputs):
        total = np.dot(self.weights, inputs) + self.bias
        return sigmoid(total)

weights = np.array([0, 1])
bias = 4 # b = 4
n = Neuron(weights, bias)

x = np.array([2, 3])
print(n.feedforward(x))
```

Результат виконання:



```
LR_5_task_1 x
C:\Users\38098\Desktop\lab05ai\venv\S
0.9990889488055994
Process finished with exit code 0
```

Рис. 1. Результат виконання завдання №1.

					ДУ «Житомирська політехніка».23.121.06.000 – Лр5						
Змн.	Арк.	№ докум.	Підпис	Дата							
Розроб.		Венгель М.І.			Звіт з лабораторної роботи			Лім.	Арк.	Аркушів	
Перевір.		Голенко М.Ю.							1	23	
Керівник								ФІКТ Гр. ІПЗ-20-2			
Н. контр.											
Зав. каф.											

Завдання №2. Створити просту нейронну мережу для передбачення статі людини.

LR_5_task_2.py

```
import numpy as np

def sigmoid(x):
    return 1 / (1 + np.exp(-x))

class Neuron:
    def __init__(self, weights, bias):
        self.weights = weights
        self.bias = bias

    def feedforward(self, inputs):
        total = np.dot(self.weights, inputs) + self.bias
        return sigmoid(total)

weights = np.array([0, 1])
bias = 4 # b = 4
n = Neuron(weights, bias)

x = np.array([2, 3])
print(n.feedforward(x))

class PolonevychNeuralNetwork:

    def __init__(self):
        weights = np.array([0, 1])
        bias = 0

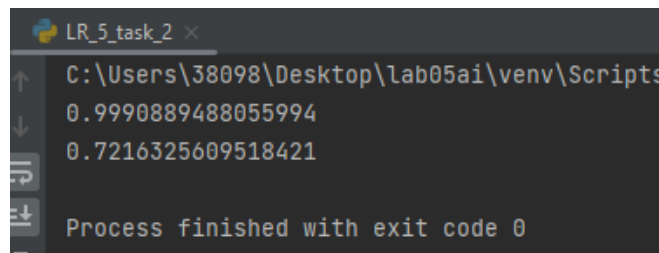
        self.h1 = Neuron(weights, bias)
        self.h2 = Neuron(weights, bias)
        self.o1 = Neuron(weights, bias)

    def feedforward(self, x):
        out_h1 = self.h1.feedforward(x)
        out_h2 = self.h2.feedforward(x)

        out_o1 = self.o1.feedforward(np.array([out_h1, out_h2]))
        return out_o1

network = PolonevychNeuralNetwork()
x = np.array([2, 3])
print(network.feedforward(x)) # 0.7216325609518421
```

Результат виконання:



```
LR_5_task_2 x
C:\Users\38098\Desktop\lab05ai\venv\Scripts
0.9990889488055994
0.7216325609518421
Process finished with exit code 0
```

Рис. 2. Результат виконання завдання №2.

LR_5_task_2_v2.py

		Венгель М.І.			ДУ «Житомирська політехніка».23.121.06.000 – Лр5	Арк.
		Голенко М.Ю.				2
Змн.	Арк.	№ докум.	Підпис	Дата		

```

import numpy as np

def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def deriv_sigmoid(x):
    fx = sigmoid(x)
    return fx * (1 - fx)

def mse_loss(y_true, y_pred):
    return ((y_true - y_pred) ** 2).mean()

class PolonevychNeuralNetwork:
    def __init__(self):
        self.w1 = np.random.normal()
        self.w2 = np.random.normal()
        self.w3 = np.random.normal()
        self.w4 = np.random.normal()
        self.w5 = np.random.normal()
        self.w6 = np.random.normal()

        self.b1 = np.random.normal()
        self.b2 = np.random.normal()
        self.b3 = np.random.normal()

    def feedforward(self, x):
        h1 = sigmoid(self.w1 * x[0] + self.w2 * x[1] + self.b1)
        h2 = sigmoid(self.w3 * x[0] + self.w4 * x[1] + self.b2)
        o1 = sigmoid(self.w5 * h1 + self.w6 * h2 + self.b3)
        return o1

    def train(self, data, all_y_trues):
        learn_rate = 0.1
        epochs = 1000

        for epoch in range(epochs):
            for x, y_true in zip(data, all_y_trues):
                sum_h1 = self.w1 * x[0] + self.w2 * x[1] + self.b1
                h1 = sigmoid(sum_h1)

                sum_h2 = self.w3 * x[0] + self.w4 * x[1] + self.b2
                h2 = sigmoid(sum_h2)

                sum_o1 = self.w5 * h1 + self.w6 * h2 + self.b3
                o1 = sigmoid(sum_o1)
                y_pred = o1

                d_L_d_ypred = -2 * (y_true - y_pred)

                d_ypred_d_w5 = h1 * deriv_sigmoid(sum_o1)
                d_ypred_d_w6 = h2 * deriv_sigmoid(sum_o1)
                d_ypred_d_b3 = deriv_sigmoid(sum_o1)

                d_ypred_d_h1 = self.w5 * deriv_sigmoid(sum_o1)
                d_ypred_d_h2 = self.w6 * deriv_sigmoid(sum_o1)

                d_h1_d_w1 = x[0] * deriv_sigmoid(sum_h1)
                d_h1_d_w2 = x[1] * deriv_sigmoid(sum_h1)
                d_h1_d_b1 = deriv_sigmoid(sum_h1)

                d_h2_d_w3 = x[0] * deriv_sigmoid(sum_h2)
                d_h2_d_w4 = x[1] * deriv_sigmoid(sum_h2)
                d_h2_d_b2 = deriv_sigmoid(sum_h2)

```

		Венгель М.І.			ДУ «Житомирська політехніка».23.121.06.000 – Лр5	Арк.
		Голенко М.Ю.				3
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        self.w1 -= learn_rate * d_L_d_ypred * d_ypred_d_h1 * d_h1_d_w1
        self.w2 -= learn_rate * d_L_d_ypred * d_ypred_d_h1 * d_h1_d_w2
        self.b1 -= learn_rate * d_L_d_ypred * d_ypred_d_h1 * d_h1_d_b1

        self.w3 -= learn_rate * d_L_d_ypred * d_ypred_d_h2 * d_h2_d_w3
        self.w4 -= learn_rate * d_L_d_ypred * d_ypred_d_h2 * d_h2_d_w4
        self.b2 -= learn_rate * d_L_d_ypred * d_ypred_d_h2 * d_h2_d_b2

        self.w5 -= learn_rate * d_L_d_ypred * d_ypred_d_w5
        self.w6 -= learn_rate * d_L_d_ypred * d_ypred_d_w6
        self.b3 -= learn_rate * d_L_d_ypred * d_ypred_d_b3

    if epoch % 10 == 0:
        y_preds = np.apply_along_axis(self.feedforward, 1, data)
        loss = mse_loss(all_y_trues, y_preds)
        print("Epoch %d loss: %.3f" % (epoch, loss))

data = np.array([
    [-2, -1],
    [25, 6],
    [17, 4],
    [-15, -6],
])

all_y_trues = np.array([
    1,
    0,
    0,
    1,
])

network = PolonevychNeuralNetwork()
network.train(data, all_y_trues)
emily = np.array([-7, -3])
frank = np.array([20, 2])
print("Emily: %.3f" % network.feedforward(emily))
print("Frank: %.3f" % network.feedforward(frank))

```

Результат виконання:

		Венгель М.І.			ДУ «Житомирська політехніка».23.121.06.000 – Лр5	Арк.
		Голенко М.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

```
LR_5_task_2_v2 x
C:\Users\38098\Desktop\lab05ai\ver
Epoch 0 loss: 0.227
Epoch 10 loss: 0.091
Epoch 20 loss: 0.072
Epoch 30 loss: 0.060
Epoch 40 loss: 0.051
Epoch 50 loss: 0.044
Epoch 60 loss: 0.039
Epoch 70 loss: 0.034
Epoch 80 loss: 0.031
Epoch 90 loss: 0.028
Epoch 100 loss: 0.025
Epoch 110 loss: 0.023
Epoch 120 loss: 0.021
Epoch 130 loss: 0.020
Epoch 140 loss: 0.018
Epoch 150 loss: 0.017
Epoch 160 loss: 0.016
Epoch 170 loss: 0.015
Epoch 180 loss: 0.014
Epoch 190 loss: 0.013
Epoch 200 loss: 0.013
```

Рис. 3. Результат виконання завдання №2.

		Венгель М.І.			ДУ «Житомирська політехніка».23.121.06.000 – Лр5	Арк.
		Голенко М.Ю.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

```

LR_5_task_2_v2 x
Epoch 200 loss: 0.013
Epoch 210 loss: 0.012
Epoch 220 loss: 0.012
Epoch 230 loss: 0.011
Epoch 240 loss: 0.011
Epoch 250 loss: 0.010
Epoch 260 loss: 0.010
Epoch 270 loss: 0.009
Epoch 280 loss: 0.009
Epoch 290 loss: 0.009
Epoch 300 loss: 0.008
Epoch 310 loss: 0.008
Epoch 320 loss: 0.008
Epoch 330 loss: 0.007
Epoch 340 loss: 0.007
Epoch 350 loss: 0.007
Epoch 360 loss: 0.007
Epoch 370 loss: 0.007
Epoch 380 loss: 0.006
Epoch 390 loss: 0.006
Epoch 400 loss: 0.006
Epoch 410 loss: 0.006
Epoch 420 loss: 0.006

```

Рис. 4. Результат виконання завдання №2.

		Венгель М.І.			ДУ «Житомирська політехніка».23.121.06.000 – Лр5	Арк.
		Голенко М.Ю.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

```

LR_5_task_2_v2 x
Epoch 420 loss: 0.006
Epoch 430 loss: 0.006
Epoch 440 loss: 0.005
Epoch 450 loss: 0.005
Epoch 460 loss: 0.005
Epoch 470 loss: 0.005
Epoch 480 loss: 0.005
Epoch 490 loss: 0.005
Epoch 500 loss: 0.005
Epoch 510 loss: 0.005
Epoch 520 loss: 0.005
Epoch 530 loss: 0.004
Epoch 540 loss: 0.004
Epoch 550 loss: 0.004
Epoch 560 loss: 0.004
Epoch 570 loss: 0.004
Epoch 580 loss: 0.004
Epoch 590 loss: 0.004
Epoch 600 loss: 0.004
Epoch 610 loss: 0.004
Epoch 620 loss: 0.004
Epoch 630 loss: 0.004
Epoch 640 loss: 0.004

```

Рис. 5. Результат виконання завдання №2.

```

LR_5_task_2_v2 x
Epoch 640 loss: 0.004
Epoch 650 loss: 0.004
Epoch 660 loss: 0.004
Epoch 670 loss: 0.003
Epoch 680 loss: 0.003
Epoch 690 loss: 0.003
Epoch 700 loss: 0.003
Epoch 710 loss: 0.003
Epoch 720 loss: 0.003
Epoch 730 loss: 0.003
Epoch 740 loss: 0.003
Epoch 750 loss: 0.003
Epoch 760 loss: 0.003
Epoch 770 loss: 0.003
Epoch 780 loss: 0.003
Epoch 790 loss: 0.003
Epoch 800 loss: 0.003
Epoch 810 loss: 0.003
Epoch 820 loss: 0.003
Epoch 830 loss: 0.003
Epoch 840 loss: 0.003
Epoch 850 loss: 0.003
Epoch 860 loss: 0.003

```

Рис. 6. Результат виконання завдання №2.

```

Epoch 870 loss: 0.003
Epoch 880 loss: 0.003
Epoch 890 loss: 0.003
Epoch 900 loss: 0.003
Epoch 910 loss: 0.002
Epoch 920 loss: 0.002
Epoch 930 loss: 0.002
Epoch 940 loss: 0.002
Epoch 950 loss: 0.002
Epoch 960 loss: 0.002
Epoch 970 loss: 0.002
Epoch 980 loss: 0.002
Epoch 990 loss: 0.002
Emily: 0.950
Frank: 0.039

Process finished with exit code 0

```

Рис. 7. Результат виконання завдання №2.

		Венгель М.І.			ДУ «Житомирська політехніка».23.121.06.000 – Лр5	Арк.
		Голенко М.Ю.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

Висновок до завдання: Функція активації використовується для підключення незв'язаних вхідних даних із виходом, у якого проста та передбачувана форма. Як правило, як функція активації найбільш часто використовується **функція сигмоїди**. Можливості нейронних мереж прямого поширення полягають в тому, що сигнали поширюються в одному напрямку, починаючи від вхідного шару нейронів, через приховані шари до вихідного шару і на вихідних нейронах отримується результат опрацювання сигналу.

В мережах такого виду **немає** зворотніх зв'язків.

Нейронні мережі прямого поширення знаходять своє застосування в задачах комп'ютерного бачення та розпізнаванні мовлення, де класифікація цільових класів ускладнюється. Такі типи нейронних мереж добре справляються із зашумленими даними.

Завдання №3. Класифікатор на основі перцептрону з використанням бібліотеки NeuroLab.

LR_5_task_3.py

```
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl

text = np.loadtxt('data_perceptron.txt')
data = text[:, :2]
labels = text[:, 2].reshape((text.shape[0], 1))
plt.figure()
plt.scatter(data[:, 0], data[:, 1])
plt.xlabel('Розмірність 1')
plt.ylabel('Розмірність 2')
plt.title('Вхідні дані')
dim1_min, dim1_max, dim2_min, dim2_max = 0, 1, 0, 1
num_output = labels.shape[1]
dim1 = [dim1_min, dim1_max]
dim2 = [dim2_min, dim2_max]
perceptron = nl.net.newp([dim1, dim2], num_output)
error_progress = perceptron.train(data, labels, epochs = 100, show = 20, lr = 0.03)
plt.figure()
plt.plot(error_progress)
plt.xlabel('Кількість епох')
plt.ylabel('Помилка навчання')
plt.title('Зміна помилок навчання')
plt.grid()
plt.show()
```

Результат виконання:

		Венгель М.І.			ДУ «Житомирська політехніка».23.121.06.000 – Пр5	Арк.
		Голенко М.Ю.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

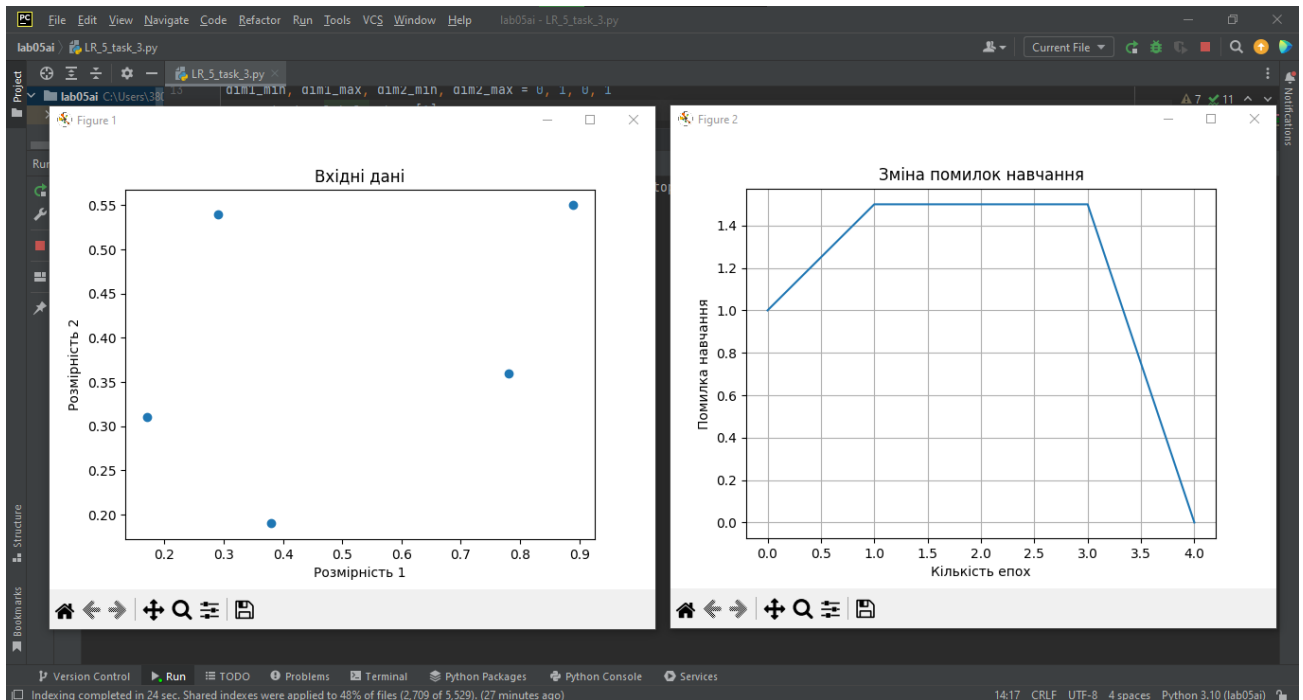


Рис. 8. Графік вхідних даних та процесу навчання.

Висновок до завдання: На другому графіку відображено процес навчання, використовуючи метрику помилки.

Під час першої епохи відбулося від 1.0 до 1.5 помилок, під час наступних двох епох відбулось 1.5 помилок.

Потім під час 4 епохи помилки почались зменшуватись, тому що ми навчили перцептрон за допомогою тренувальних даних.

Завдання №4. Побудова одношарової нейронної мережі.

LR_5_task_4.py

```
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl

text = np.loadtxt('data_simple_nn.txt')
data = text[:, 0:2]
labels = text[:, 2:]
plt.figure()
plt.scatter(data[:, 0], data[:, 1])
plt.xlabel('Розмірність 1')
plt.ylabel('Розмірність 2')
plt.title('Вхідні дані')
dim1_min, dim1_max = data[:, 0].min(), data[:, 0].max()
dim2_min, dim2_max = data[:, 1].min(), data[:, 1].max()
num_output = labels.shape[1]
dim1 = [dim1_min, dim1_max]
dim2 = [dim2_min, dim2_max]
nn = nl.net.newp([dim1, dim2], num_output)
error_progress = nn.train(data, labels, epochs = 100, show = 20, lr = 0.03)
plt.figure()
```

		Венгелі М.І.			ДУ «Житомирська політехніка».23.121.06.000 – Лр5	Арк.
		Голенко М.Ю.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

```
plt.plot(error_progress)
plt.xlabel('Кількість епох')
plt.ylabel('Помилка навчання')
plt.title('Зміна помилок навчання')
plt.grid()
plt.show()
print('\nTest results:')
data_test = [[0.4, 4.3], [4.4, 0.6], [4.7, 8.1]]
for item in data_test:
    print(item, '-->', nn.sim([item])[0])
```

Результат виконання:

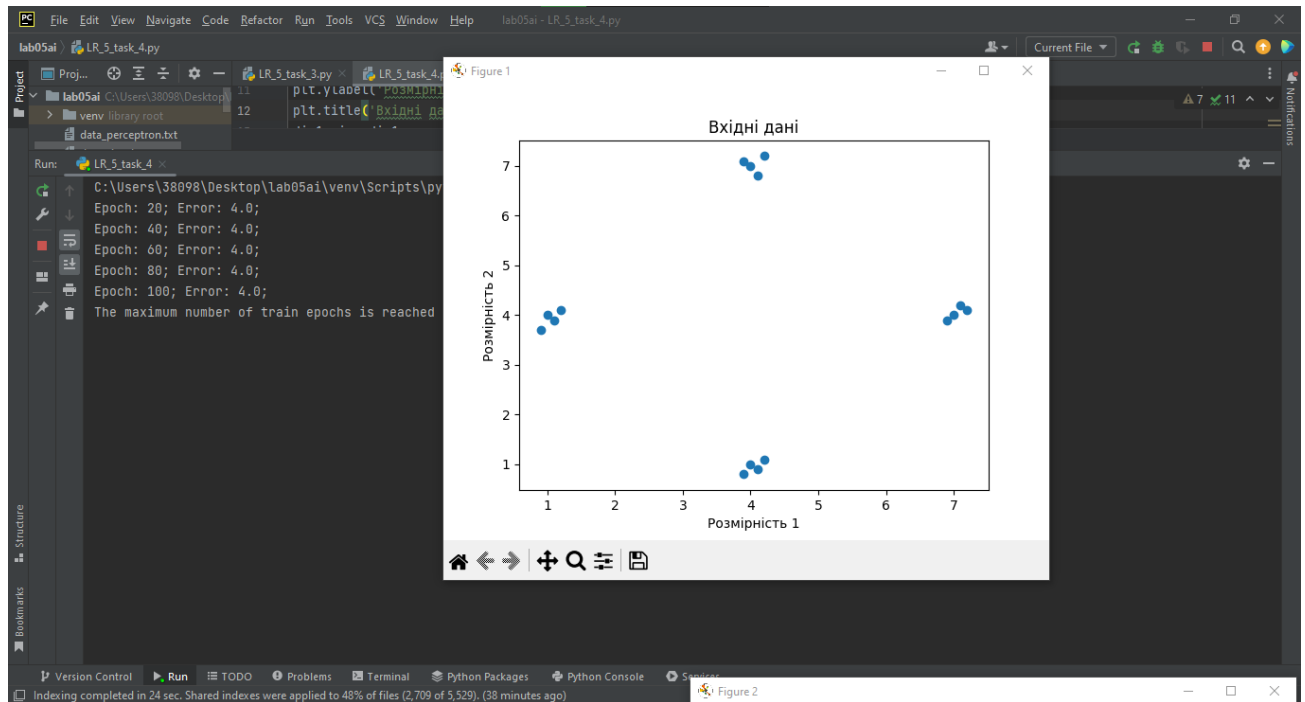
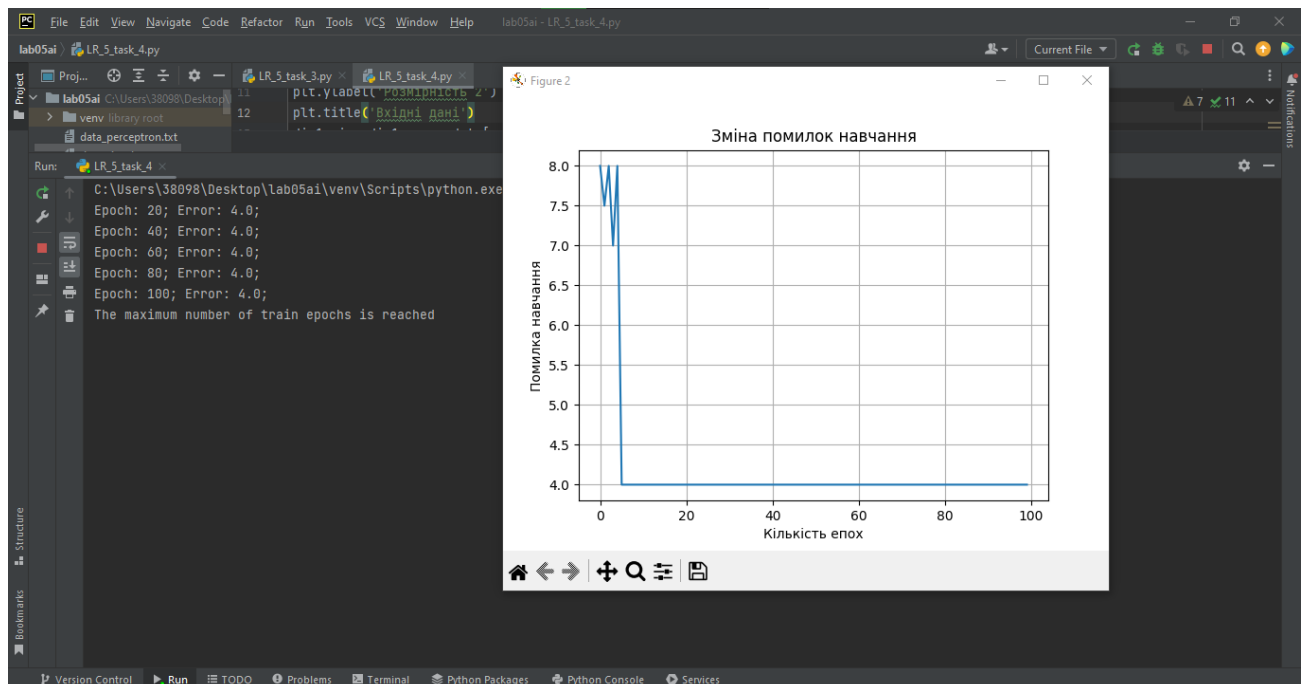


Рис. 9. Графік вхідних даних.



		Венгелі М.І.			ДУ «Житомирська політехніка».23.121.06.000 – Лр5	Арк.
		Голенко М.Ю.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

Рис. 10. Графік просування процесу навчання.

```
LR_5_task_4 x
C:\Users\38098\Desktop\lab05ai\venv\Scripts\python.
Epoch: 20; Error: 4.0;
Epoch: 40; Error: 4.0;
Epoch: 60; Error: 4.0;
Epoch: 80; Error: 4.0;
Epoch: 100; Error: 4.0;
The maximum number of train epochs is reached

Test results:
[0.4, 4.3] --> [0. 0.]
[4.4, 0.6] --> [1. 0.]
[4.7, 8.1] --> [1. 1.]

Process finished with exit code 0
```

Рис. 11. Результат виконання завдання №4.

Висновок до завдання: На рис. 11 зображено процес навчання мережі. На 20-ому епосі відбулось 4 помилки, аналогічно на 40, 60, 80 та 100. Потім вивелось повідомлення, що ми досягли максимальної кількості епох для тренування. Ми вирішили визначити вибірккові тестові точки даних та запустили для них нейронну мережу.

Завдання №5. Побудова багатошарової нейронної мережі.

LR_5_task_5.py

```
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl
min_val = -15
max_val = 15
num_points = 130
x = np.linspace(min_val, max_val, num_points)
y = 3 * np.square(x) + 5
y /= np.linalg.norm(y)
data = x.reshape(num_points, 1)
labels = y.reshape(num_points, 1)
plt.figure()
plt.scatter(data, labels)
plt.xlabel('Розмірність 1')
plt.ylabel('Розмірність 2')
plt.title('Вхідні дані')
nn = nl.net.newff([[min_val, max_val]], [10, 6, 1])
nn.trainf = nl.train.train_gd
error_progress = nn.train(data, labels, epochs=2000, show = 100, goal = 0.01)
output = nn.sim(data)
y_pred = output.reshape(num_points)
```

		Венгель М.І.			ДУ «Житомирська політехніка».23.121.06.000 – Лр5	Арк.
		Голенко М.Ю.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

```
plt.figure()
plt.plot(error_progress)
plt.xlabel('Кількість епох')
plt.ylabel('Помилка навчання')
plt.title('Зміна помилок навчання')
x_dense = np.linspace(min_val, max_val, num_points * 2)
y_dense_pred = nn.sim(x_dense.reshape(x_dense.size, 1)).reshape(x_dense.size)
plt.figure()
plt.plot(x_dense, y_dense_pred, '-', x, y, '.', x, y_pred, 'p')
plt.title('Фактичні і прогнозовані значення')
plt.show()
```

Результат виконання:

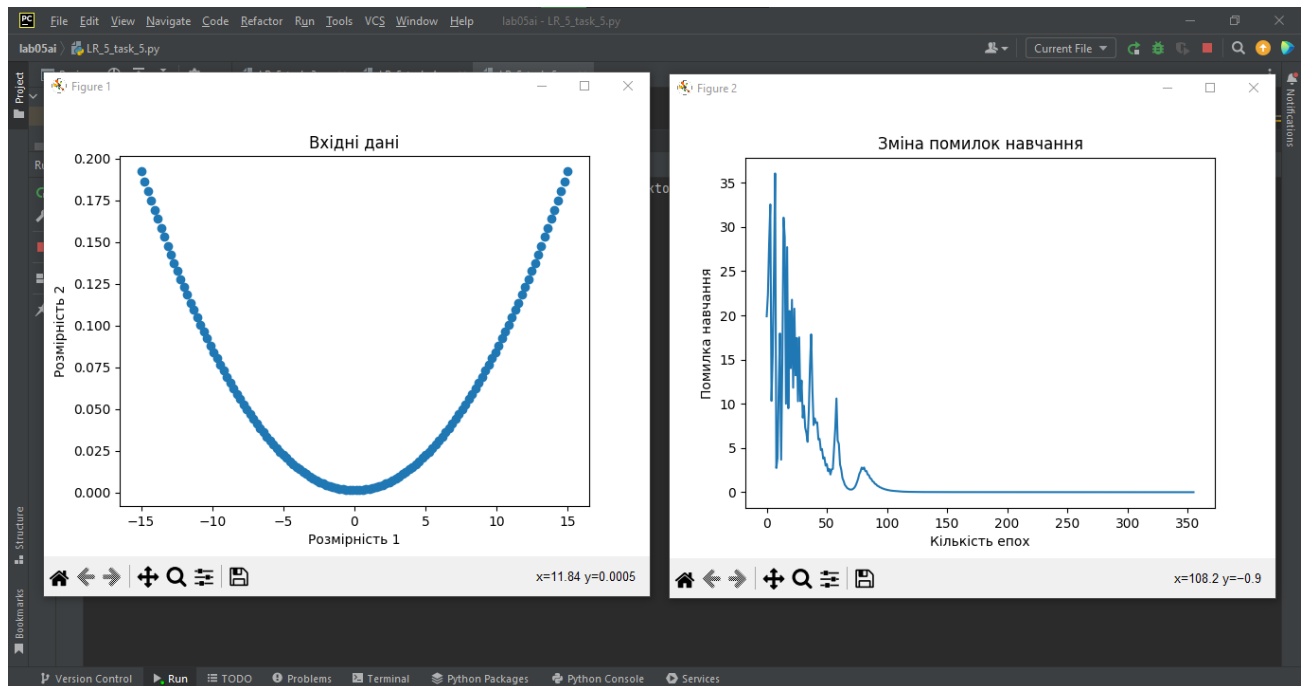
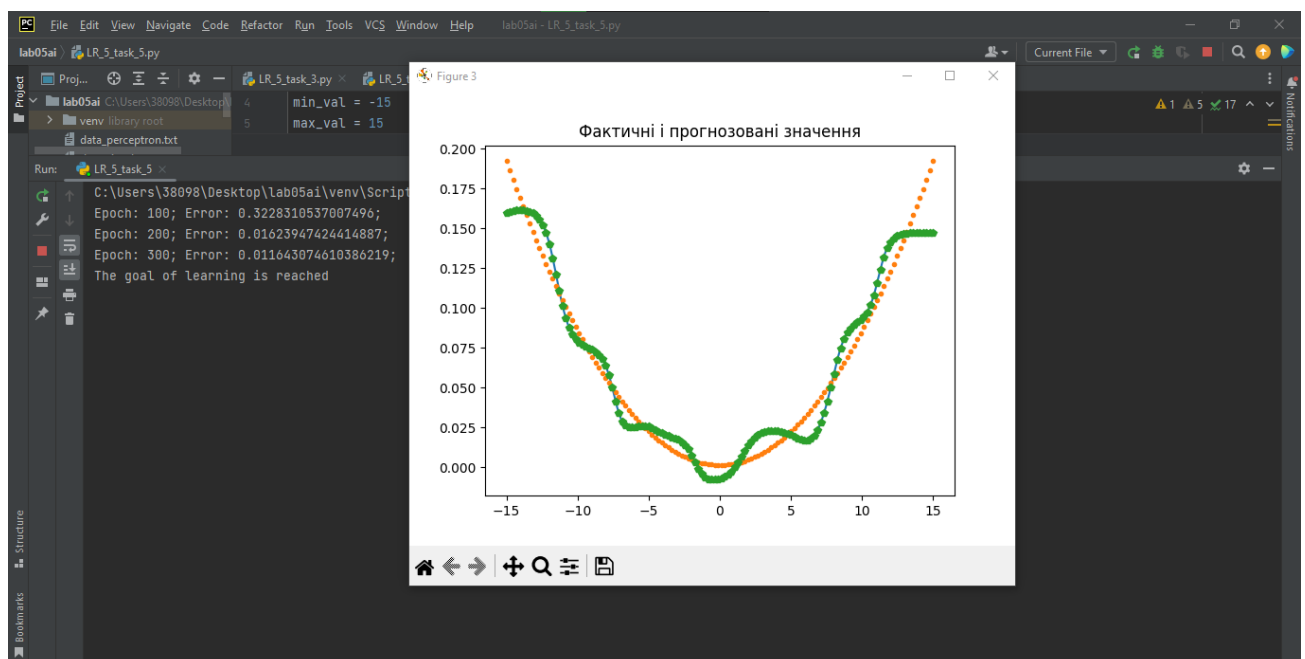


Рис. 12. Результат виконання завдання №5.



		Венгелі М.І.			ДУ «Житомирська політехніка».23.121.06.000 – Лр5	Арк.
		Голенко М.Ю.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

Рис. 13. Результат виконання завдання №5.

```
LR_5_task_5 x
C:\Users\38098\Desktop\lab05ai\venv\Scripts\python
Epoch: 100; Error: 0.3228310537007496;
Epoch: 200; Error: 0.01623947424414887;
Epoch: 300; Error: 0.011643074610386219;
The goal of learning is reached
Process finished with exit code 0
```

Рис. 14. Результат виконання завдання №5.

Висновок до завдання:

На рис. 14 зображено процес навчання мережі. На 100 епосі відбулось 0.32 помилки, на 200 епосі відбулось 0.16 помилки, на 300 епосі відбулось 0.11 помилки. Потім вивелось повідомлення, що ми досягли цілі навчання.

Завдання №6. Побудова багатошарової нейронної мережі для свого варіанту.

№ варіанта	Тестові дані
Варіант 16	$y = 5x^2 + 7$

Номер варіанта	Багатошаровий персептрон	
	Кількість шарів	Кількості нейронів у шарах
16	2	7-1

LR_5_task_6.py

```
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl
min_val = -15
max_val = 15
num_points = 130
x = np.linspace(min_val, max_val, num_points)
y = 5 * x * x + 7
y /= np.linalg.norm(y)
data = x.reshape(num_points, 1)
labels = y.reshape(num_points, 1)
plt.figure()
plt.scatter(data, labels)
plt.xlabel('Розмірність 1')
plt.ylabel('Розмірність 2')
plt.title('Вхідні дані')
nn = nl.net.newff([[min_val, max_val]], [7, 1])
nn.trainf = nl.train.train_gd
error_progress = nn.train(data, labels, epochs=2000, show = 100, goal = 0.01)
output = nn.sim(data)
y_pred = output.reshape(num_points)
plt.figure()
plt.plot(error_progress)
```

```
plt.xlabel('Кількість епох')
plt.ylabel('Помилка навчання')
plt.title('Зміна помилок навчання')
x_dense = np.linspace(min_val, max_val, num_points * 2)
y_dense_pred = nn.sim(x_dense.reshape(x_dense.size, 1)).reshape(x_dense.size)
plt.figure()
plt.plot(x_dense, y_dense_pred, '-', x, y, '.', x, y_pred, 'p')
plt.title('Фактичні і прогнозовані значення')
plt.show()
```

Результат виконання:

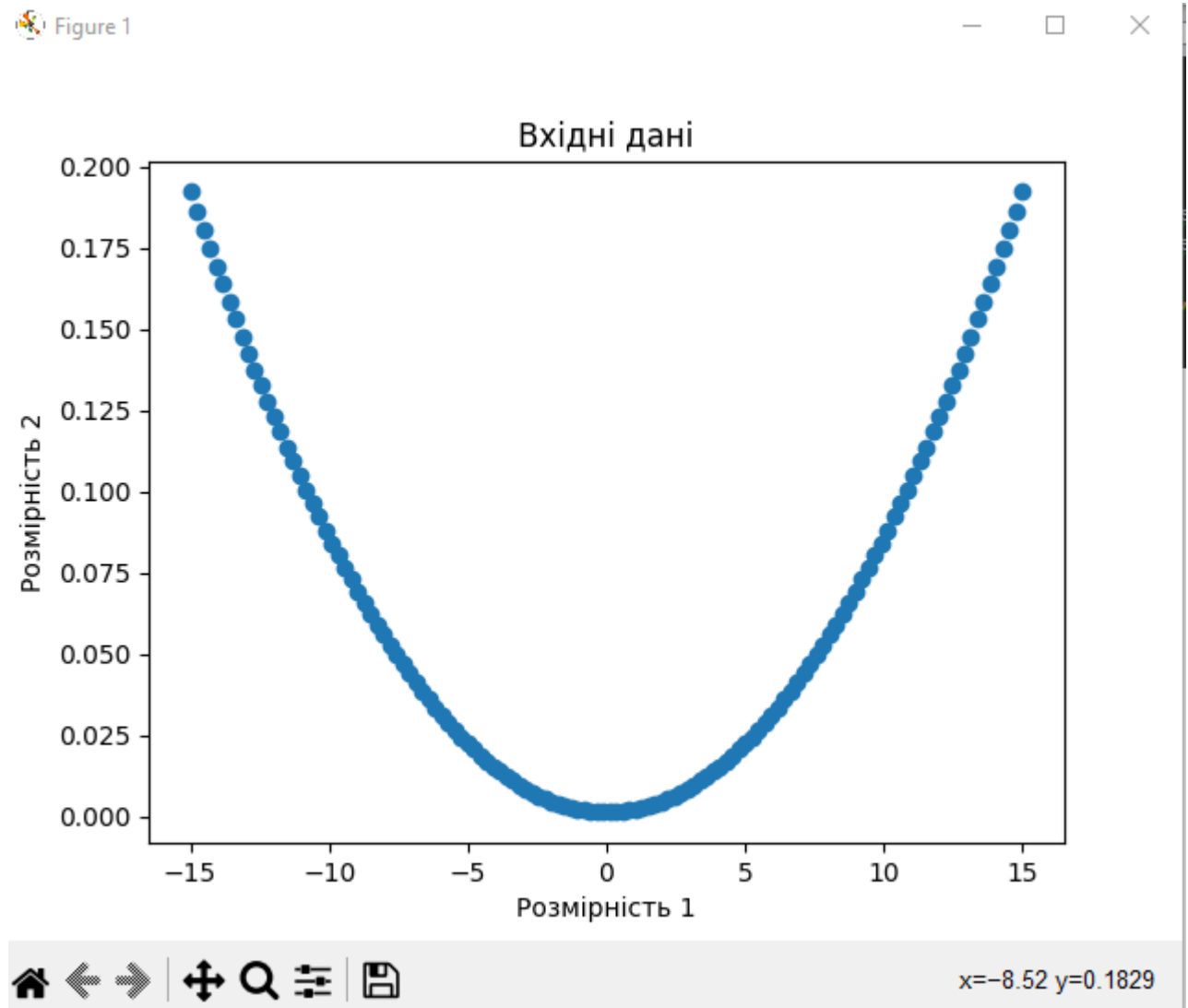


Рис. 16. Результат виконання завдання №6.

		Венгель М.І.			ДУ «Житомирська політехніка».23.121.06.000 – Лр5	Арк.
		Голенко М.Ю.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

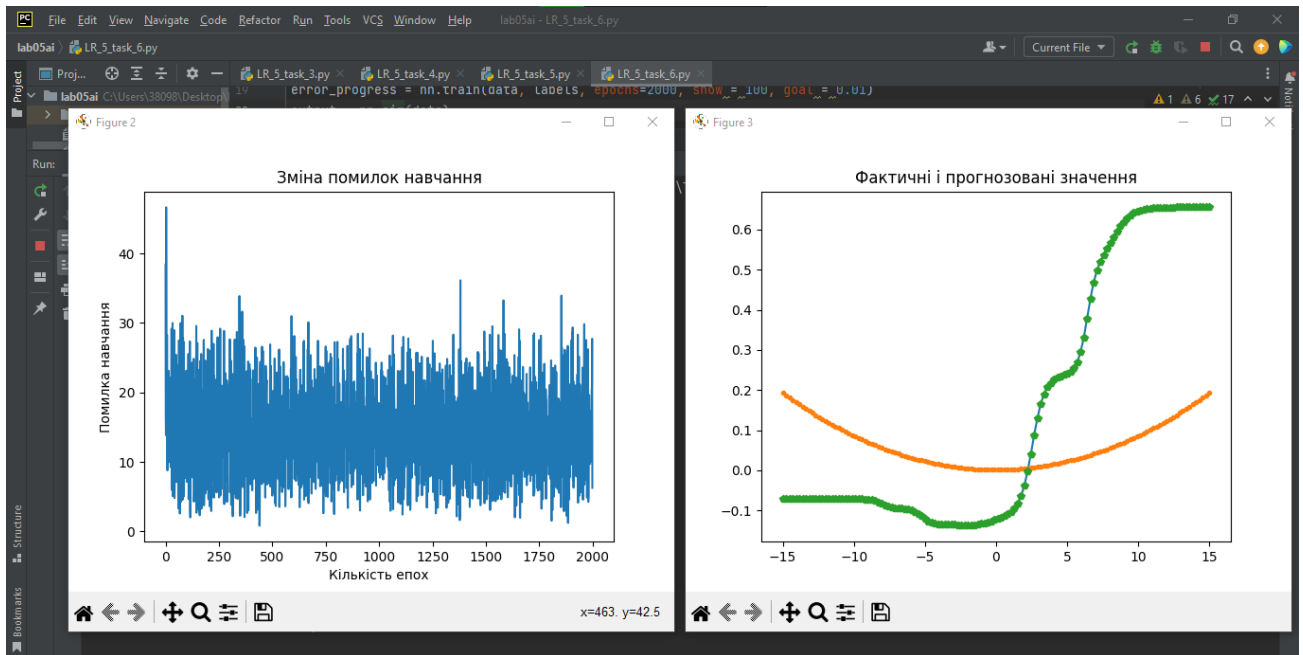


Рис. 17. Результат виконання завдання №6.

```

LR_5_task_6 x
C:\Users\38098\Desktop\lab05ai\venv\Scripts\python.exe C:\Users\38098\Desktop\lab05ai\venv\Scripts\python.exe C:\Users\38098\Desktop\lab05ai\venv\Scripts\python.exe
Epoch: 100; Error: 19.04427623235089;
Epoch: 200; Error: 18.31523494107495;
Epoch: 300; Error: 21.475341073403794;
Epoch: 400; Error: 10.82587568689041;
Epoch: 500; Error: 19.236524975089768;
Epoch: 600; Error: 12.09877980121597;
Epoch: 700; Error: 11.841514419109012;
Epoch: 800; Error: 8.624569559486837;
Epoch: 900; Error: 4.600578404857191;
Epoch: 1000; Error: 18.10832600308324;
Epoch: 1100; Error: 23.082337894047463;
Epoch: 1200; Error: 12.403471793457289;
Epoch: 1300; Error: 11.059856556033955;
Epoch: 1400; Error: 15.716938139229676;
Epoch: 1500; Error: 16.377006750229125;
Epoch: 1600; Error: 21.776547869089974;
Epoch: 1700; Error: 8.564467610305666;
Epoch: 1800; Error: 21.084291334133177;
Epoch: 1900; Error: 11.00089928396124;
Epoch: 2000; Error: 6.285465275612012;
The maximum number of train epochs is reached

Process finished with exit code 0

```


Рис. 18. Результат виконання завдання №6.

Висновок до завдання: На рис. 18 зображено процес навчання мережі. На 100 епосі відбулось 19.04 помилки, на 200 епосі відбулось 18.31 помилки, на 300 епосі відбулось 21.47 помилки і так далі, на 2000 епосі відбулось 6.28 помилки,. Потім вивелось повідомлення, що ми досягли максимальної кількості епох для тренування.

Завдання №7. Побудова нейронної мережі на основі карти Кохонена, що самоорганізується.

LR_5_task_7.py

```
import numpy as np
import neurolab as nl
import numpy.random as rand

skv = 0.05
centr = np.array([[0.2, 0.2], [0.4, 0.4], [0.7, 0.3], [0.2, 0.5]])
rand_norm = skv * rand.randn(100, 4, 2)
inp = np.array([centr + r for r in rand_norm])
inp.shape = (100 * 4, 2)
rand.shuffle(inp)

# Create net with 2 inputs and 4 neurons
net = nl.net.newc([[0.0, 1.0], [0.0, 1.0]], 4)
# train with rule: Conscience Winner Take All algorithm (CWTA)
error = net.train(inp, epochs=200, show=100)

# Plot results:
import pylab as pl
pl.title('Classification Problem')
pl.subplot(211)
pl.plot(error)
pl.xlabel('Epoch number')
pl.ylabel('error (default MAE)')
w = net.layers[0].np['w']

pl.subplot(212)
pl.plot(inp[:,0], inp[:,1], '.', \
        centr[:,0], centr[:,1], 'yv', \
        w[:,0], w[:,1], 'p')
pl.legend(['train samples', 'real centers', 'train centers'])
pl.show()
```

Результат виконання:

		Венгель М.І.			ДУ «Житомирська політехніка».23.121.06.000 – Лр5	Арк.
		Голенко М.Ю.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

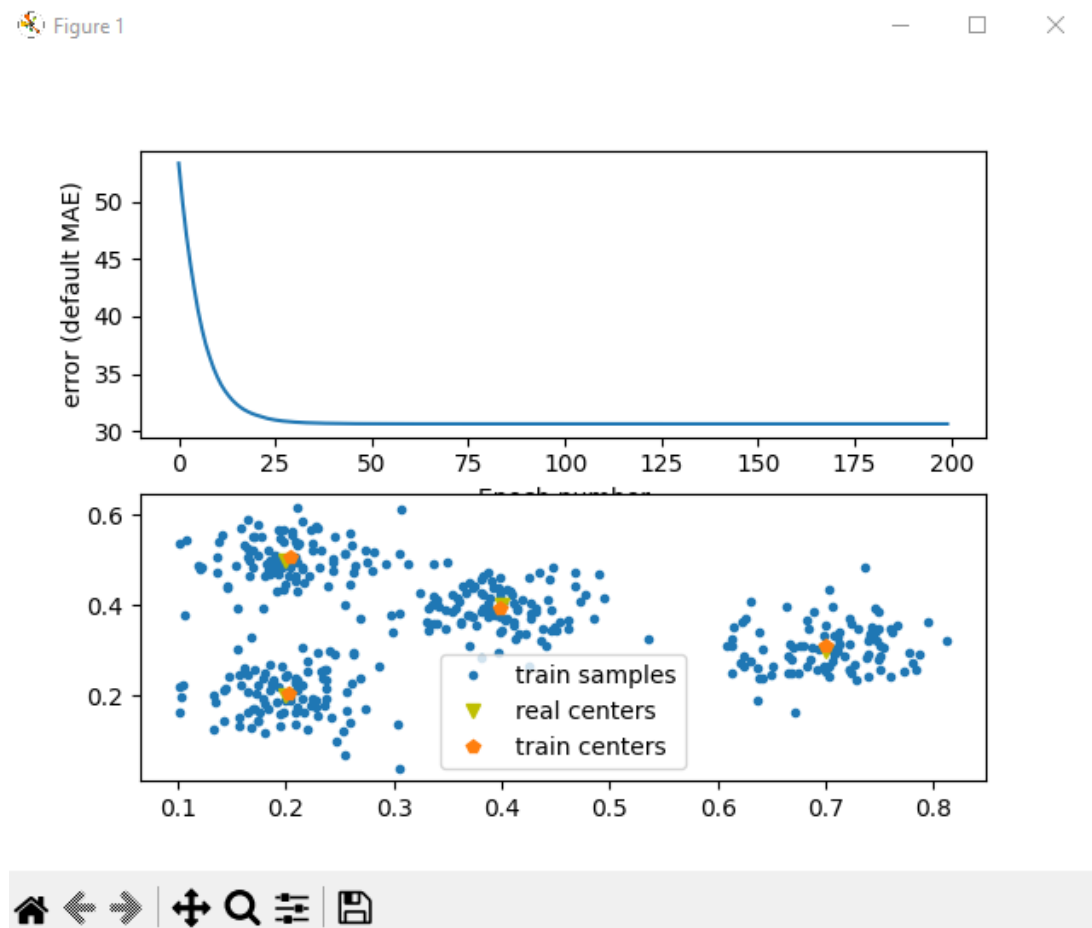


Рис. 19. Результат виконання завдання №7.

Помилка MAE - [Средня абсолютна помилка \(Mean Absolute Error\). Середньою абсолютною похибкою називають середнє арифметичне з абсолютних похибок усіх вимірювань.](#)

Завдання №8. Дослідження нейронної мережі на основі карти Кохонена, що само зорганізується.

Варіант 16	[0.2, 0.2], [0.4, 0.4], [0.3, 0.3], [0.3, 0.6], [0.5, 0.7]	0,05
------------	--	------

LR_5_task_8_v1.py

```
import numpy as np
import neurolab as nl
import numpy.random as rand

skv = 0.05
centr = np.array([[0.2, 0.2], [0.4, 0.4], [0.3, 0.3], [0.3, 0.6], [0.5, 0.7]])
rand_norm = skv * rand.randn(100, 5, 2)
inp = np.array([centr + r for r in rand_norm])
inp.shape = (100 * 5, 2)
rand.shuffle(inp)

# Create net with 2 inputs and 4 neurons
net = nl.net.newc([[0.0, 1.0], [0.0, 1.0]], 4)
# train with rule: Conscience Winner Take All algorithm (CWTA)
error = net.train(inp, epochs=200, show=20)
```

		Венгель М.І.			ДУ «Житомирська політехніка».23.121.06.000 – Лр5	Арк.
		Голенко М.Ю.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

```
# Plot results:
import pylab as pl
pl.title('Classification Problem')
pl.subplot(211)
pl.plot(error)
pl.xlabel('Epoch number')
pl.ylabel('error (default MAE)')
w = net.layers[0].np['w']

pl.subplot(212)
pl.plot(inp[:,0], inp[:,1], '.', \
        centr[:,0], centr[:,1], 'yv', \
        w[:,0], w[:,1], 'p')
pl.legend(['train samples', 'real centers', 'train centers'])
pl.show()
```

Результат виконання:

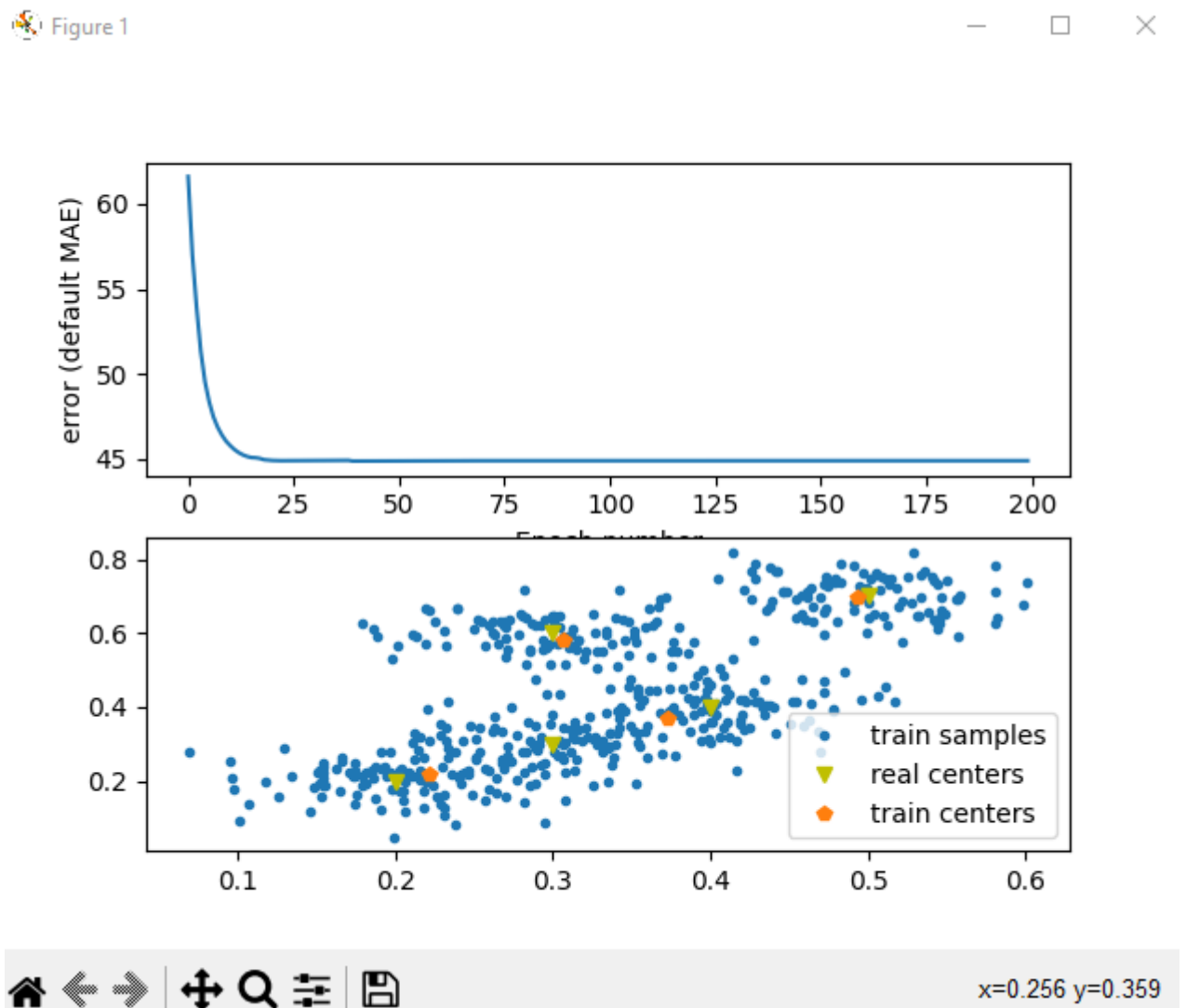


Рис. 20. Результат виконання завдання №8.

		Венгель М.І.			ДУ «Житомирська політехніка».23.121.06.000 – Лр5	Арк.
		Голенко М.Ю.				19
Змн.	Арк.	№ докум.	Підпис	Дата		

```

LR_5_task_8_v1 x
C:\Users\38098\Desktop\lab05ai\venv\Scripts\pyth
Epoch: 20; Error: 44.94552359204139;
Epoch: 40; Error: 44.893494054261126;
Epoch: 60; Error: 44.91039742098229;
Epoch: 80; Error: 44.913941257803245;
Epoch: 100; Error: 44.914187963187686;
Epoch: 120; Error: 44.914211520885544;
Epoch: 140; Error: 44.91421435555742;
Epoch: 160; Error: 44.9142147380467;
Epoch: 180; Error: 44.914214791918646;
Epoch: 200; Error: 44.914214799599414;
The maximum number of train epochs is reached

```

Рис. 21. Результат виконання завдання №8.

На рис. 21 зображено процес навчання мережі. На 20 епосі відбулось 44.9 помилки, на 40 епосі відбулось 44.89 помилки, на 60 епосі відбулось 44.91 помилки і так далі, на 200 епосі відбулось 44.91 помилки. Потім вивелось повідомлення, що ми досягли максимальної кількості епох для тренування.

LR_5_task_8_v2.py

```

import numpy as np
import neurolab as nl
import numpy.random as rand

skv = 0.05
centr = np.array([[0.2, 0.2], [0.4, 0.4], [0.3, 0.3], [0.3, 0.6], [0.5, 0.7]])
rand_norm = skv * rand.randn(100, 5, 2)
inp = np.array([centr + r for r in rand_norm])
inp.shape = (100 * 5, 2)
rand.shuffle(inp)

# Create net with 2 inputs and 5 neurons
net = nl.net.newc([[0.0, 1.0], [0.0, 1.0]], 5)
# train with rule: Conscience Winner Take All algorithm (CWTA)
error = net.train(inp, epochs=200, show=20)

# Plot results:
import pylab as pl
pl.title('Classification Problem')
pl.subplot(211)
pl.plot(error)
pl.xlabel('Epoch number')
pl.ylabel('error (default MAE)')
w = net.layers[0].np['w']

pl.subplot(212)
pl.plot(inp[:,0], inp[:,1], '.', \
        centr[:,0], centr[:,1], 'yv', \
        w[:,0], w[:,1], 'p')

```

		Венгель М.І.			ДУ «Житомирська політехніка».23.121.06.000 – Лр5	Арк.
		Голенко М.Ю.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

```
pl.legend(['train samples', 'real centers', 'train centers'])
pl.show()
```

Figure 1

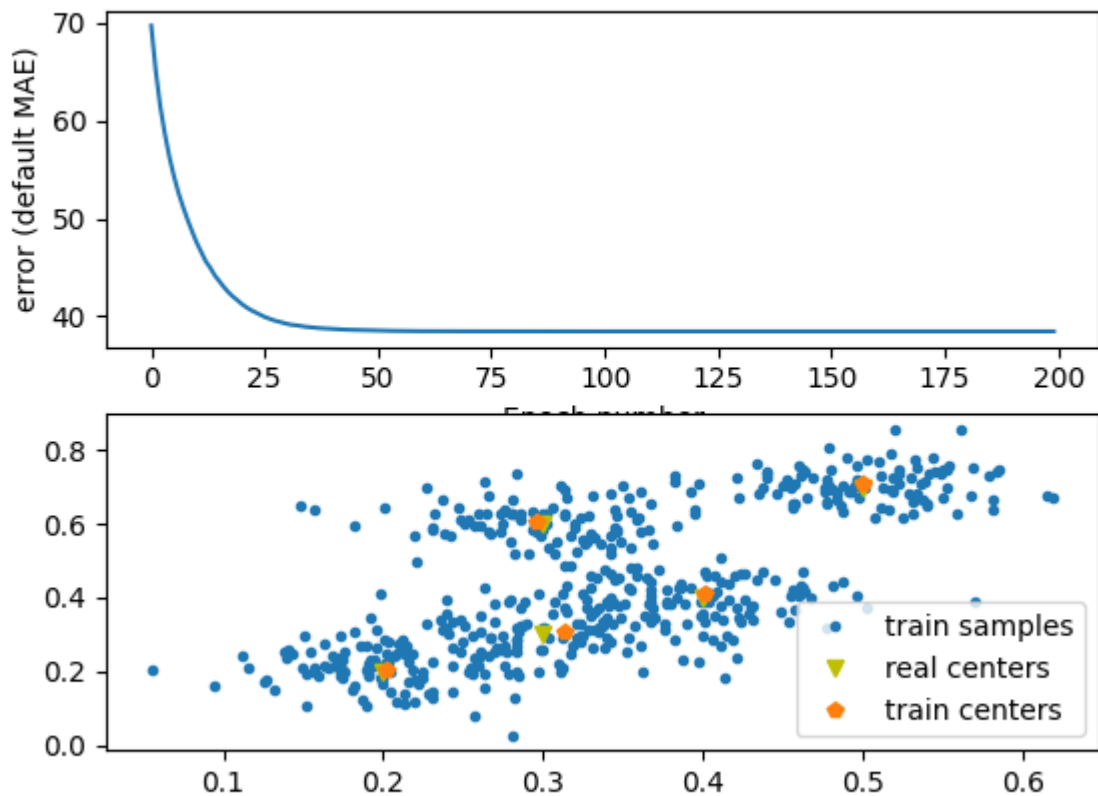


Рис. 22. Результат виконання завдання №8.

```
LR_5_task_8_v2 x
C:\Users\38098\Desktop\lab05ai\venv\Scripts\python.exe
Epoch: 20; Error: 41.65286880128481;
Epoch: 40; Error: 38.73602657076628;
Epoch: 60; Error: 38.489329410542226;
Epoch: 80; Error: 38.46758995455735;
Epoch: 100; Error: 38.46492228850646;
Epoch: 120; Error: 38.46459211173243;
Epoch: 140; Error: 38.46456015822811;
Epoch: 160; Error: 38.46455887029467;
Epoch: 180; Error: 38.464558309215846;
Epoch: 200; Error: 38.46455814238061;
The maximum number of train epochs is reached
```

		Венгель М.І.			ДУ «Житомирська політехніка».23.121.06.000 – Лр5	Арк.
		Голенко М.Ю.				21
Змн.	Арк.	№ докум.	Підпис	Дата		

Рис. 23. Результат виконання завдання №8.

На рис. 23 зображено процес навчання мережі. На 20 епосі відбулось 41.65 помилки, на 40 епосі відбулось 38.73 помилки, на 60 епосі відбулось 38.48 помилки і так далі, на 200 епосі відбулось 38.46 помилки,. Потім вивелось повідомлення, що ми досягли максимальної кількості епох для тренування.

Висновок до завдання: Якщо порівнювати нейронну мережу Кохонена з 4 нейронами та 5 нейронами, можна зробити такі висновки. При 4 нейронах Помилка MAE повільніше зменшується, ніж з 5 нейронами, також з 5 нейронами ця помилка нижча. З 5 нейронами обоє центрів збігаються майже в одні точці. Число нейронів в шарі Кохонена має відповідати числу класів вхідних сигналів. Тобто в нашому випадку нам давалось 5 вхідних сигналів, значить у нас має бути 5 нейронів, а не 4. **Отже, невірний вибір кількості нейронів числу кластерів впливає на величину помилки ускладнюючи навчання мережі і швидкості**, тому на рис. 21 гірші результати, ніж на рис. 23.

Висновок: Під час виконання лабораторної роботи, використовуючи спеціалізовані бібліотеки та мову програмування Python навчитися створювати та застосовувати прості нейронні мережі.

		Венгель М.І.			ДУ «Житомирська політехніка».23.121.06.000 – Лр5	Арк.
		Голенко М.Ю.				22
Змн.	Арк.	№ докум.	Підпис	Дата		