Лабораторна робота 5

РОЗРОБКА ПРОСТИХ НЕЙРОННИХ МЕРЕЖ

Мета: використовуючи спеціалізовані бібліотеки та мову програмування Python навчитися створювати та застосовувати прості нейронні мережі.

Хід роботи

Завдання 2.1. Створити простий нейрон

```
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
n = Neuron(weights, bias)

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

```
/usr/local/bin/python3.9 /Users/webb/Desktop/CWI/programs/lab5/LR_5_task_1.py
0.9990889488055994

Process finished with exit code 0
```

Рис. 2 Результат виконання програми

Завдання 2.2. Створити просту нейронну мережу для передбачення статі людини

```
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
n = Neuron(weights, bias)

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

class HrishynNeuralNetwork:

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

```
/usr/local/bin/python3.9 /Users/webb/Desktop/CWI/programs/lab5/LR_5_task_2.py
0.7216325609518421

Process finished with exit code 0
```

Рис. 5 Результат виконання програми

```python
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 HrishynNeuralNetwork:

    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)

                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 = HrishynNeuralNetwork()
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))
```

```
/usr/local/bin/python3.9 /Users/webb/Desktop/CWI/programs/lab5/LR_5_task_3.py
Epoch 0 loss: 0.204
Epoch 10 loss: 0.162
Epoch 20 loss: 0.129
Epoch 30 loss: 0.104
Epoch 40 loss: 0.085
Epoch 50 loss: 0.071
Epoch 60 loss: 0.060
Epoch 70 loss: 0.051
Epoch 80 loss: 0.044
Epoch 90 loss: 0.039
Epoch 100 loss: 0.035
Epoch 110 loss: 0.031
Epoch 120 loss: 0.028
Epoch 130 loss: 0.025
Epoch 140 loss: 0.023
Epoch 150 loss: 0.021
Epoch 160 loss: 0.020
Epoch 170 loss: 0.018
Epoch 180 loss: 0.017
Epoch 190 loss: 0.016
Epoch 200 loss: 0.015
Epoch 210 loss: 0.014
Epoch 220 loss: 0.014
Epoch 230 loss: 0.013
Epoch 240 loss: 0.012
Epoch 250 loss: 0.012
Epoch 260 loss: 0.011
Epoch 270 loss: 0.011
Epoch 280 loss: 0.010
Epoch 290 loss: 0.010
Epoch 300 loss: 0.009
Epoch 310 loss: 0.009
Epoch 320 loss: 0.009
Epoch 330 loss: 0.008
Epoch 340 loss: 0.008
Epoch 350 loss: 0.008
Epoch 360 loss: 0.008
Epoch 370 loss: 0.007
Epoch 380 loss: 0.007
Epoch 390 loss: 0.007
Epoch 400 loss: 0.007
Epoch 410 loss: 0.006
```

Рис. 7 Результат виконання програми

Завдання 2.3. Класифікатор на основі перцептрону з використанням бібліотеки NeuroLab

```
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()
```

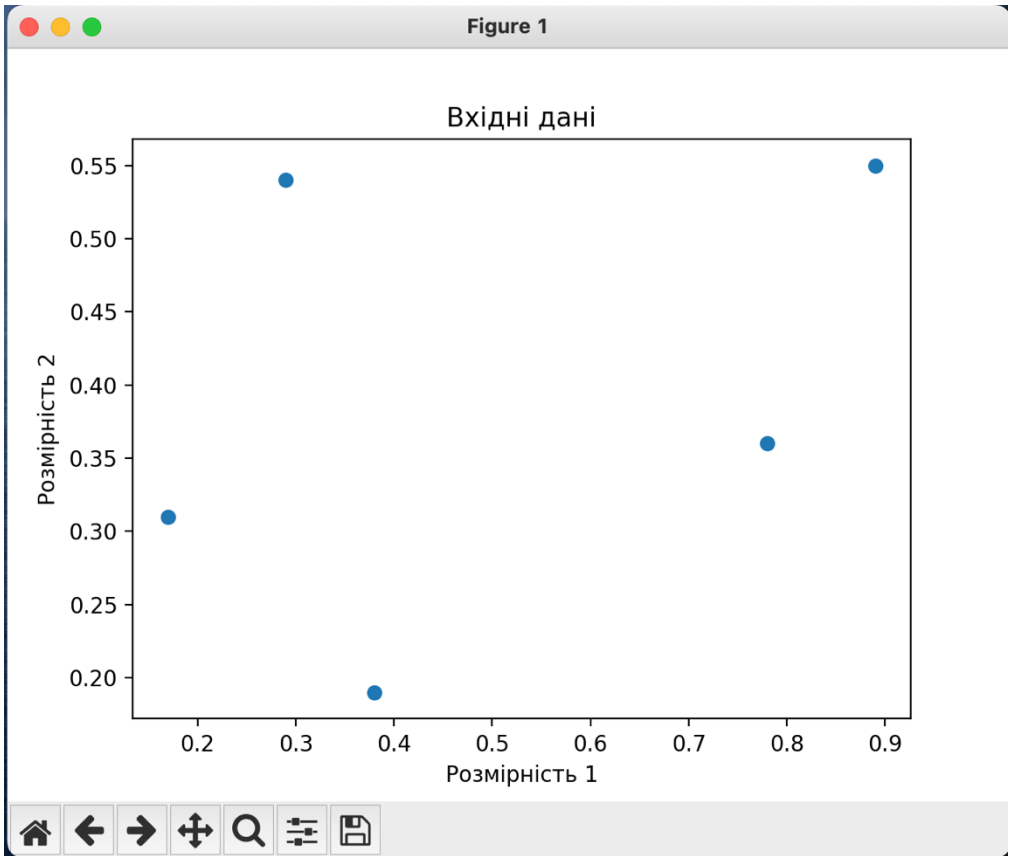

Рис. 12 Графік вхідних даних

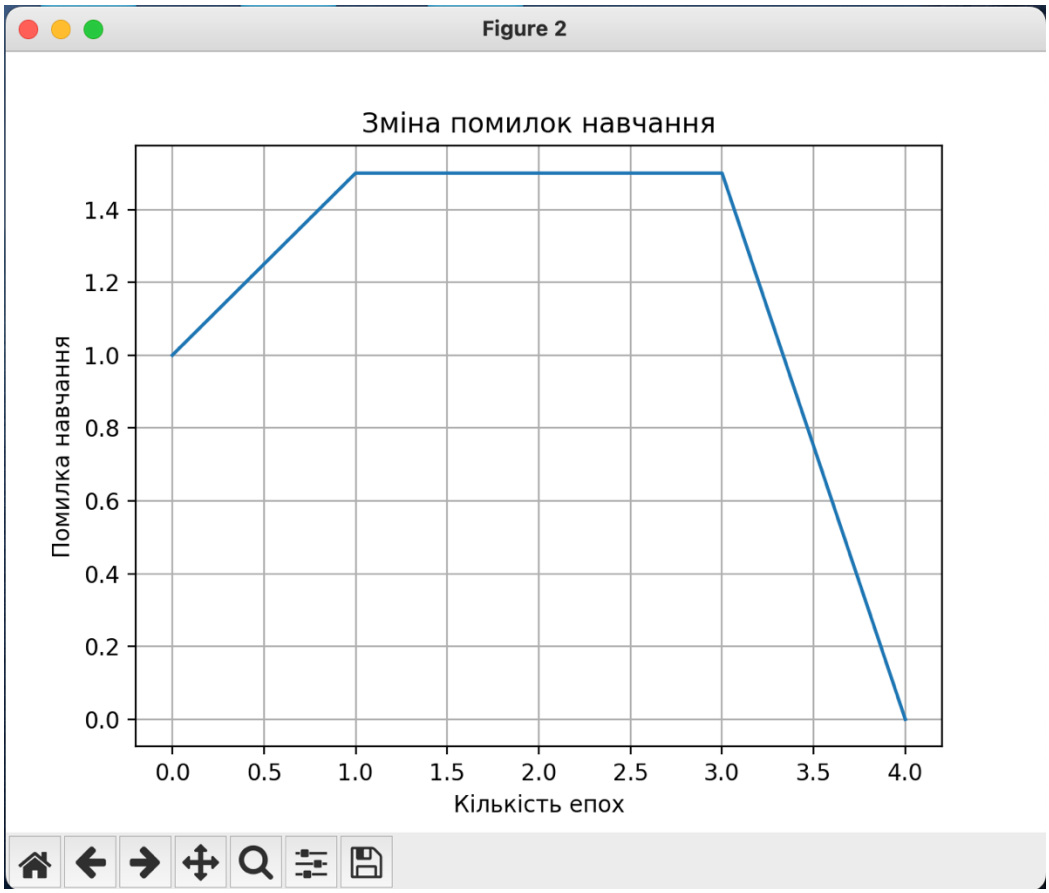

Рис. 13 Графік процесу навчання

Завдання 2.4. Побудова одношарової нейронної мережі

```
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()
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])
```

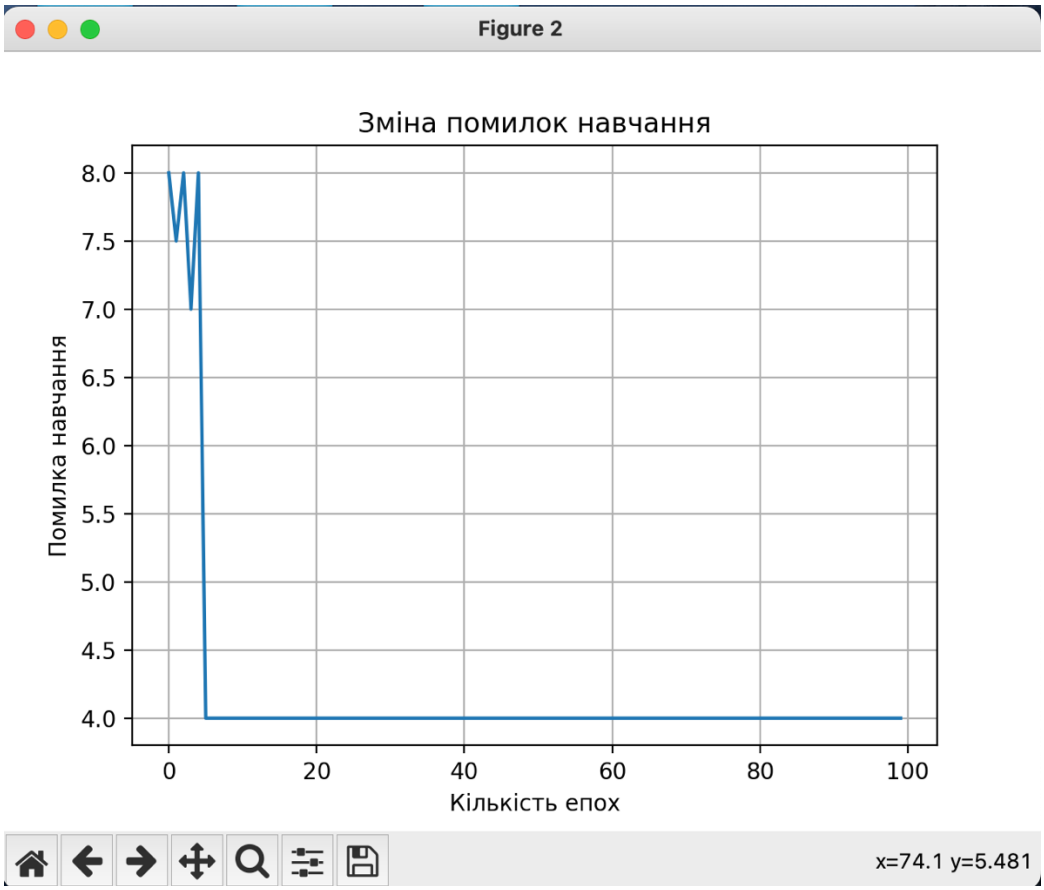

Рис. 15 Графік вхідних даних

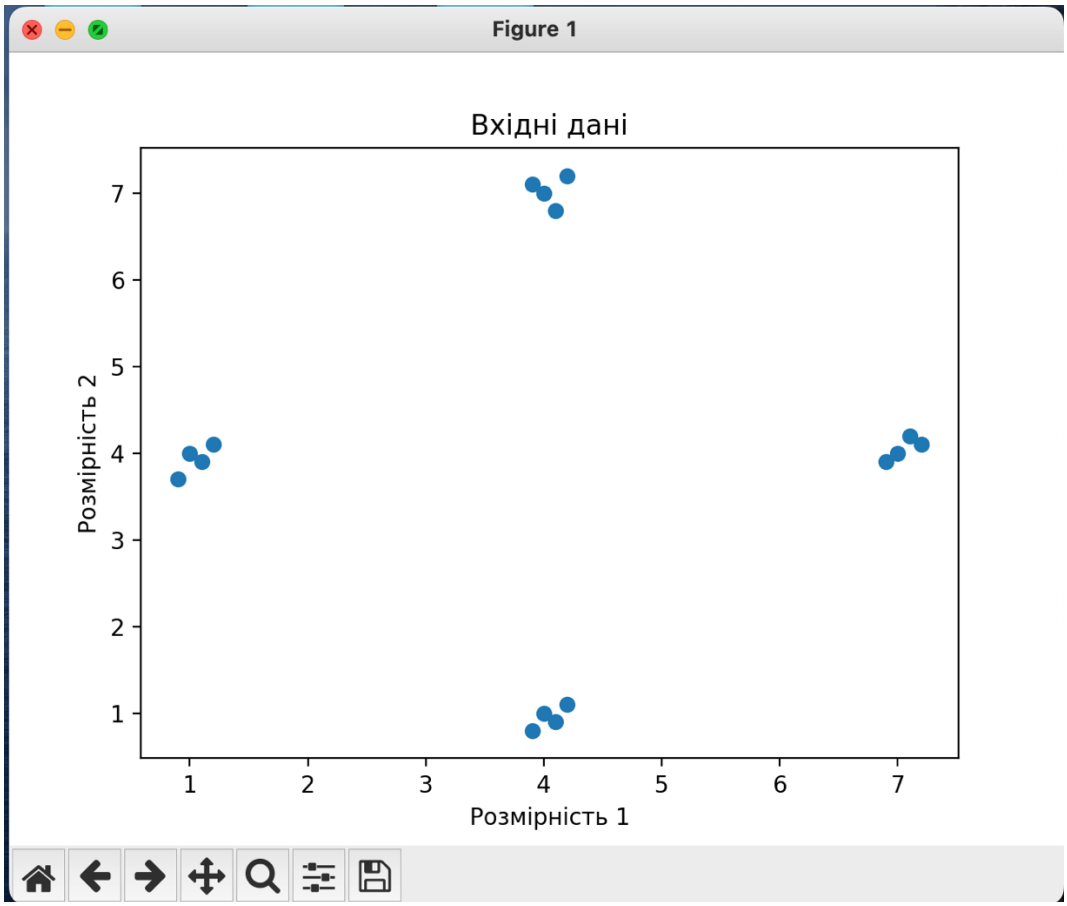

Рис. 16 Графік просування процесу навчання

```
/usr/local/bin/python3.9 /Users/webb/Desktop/СШІ/programs/lab5/LR_5_task_5.py
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.]
```

Рис. 17 Результат виконання програми

Завдання 2.5. Побудова багатошарової нейронної мережі

```
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)
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()
```

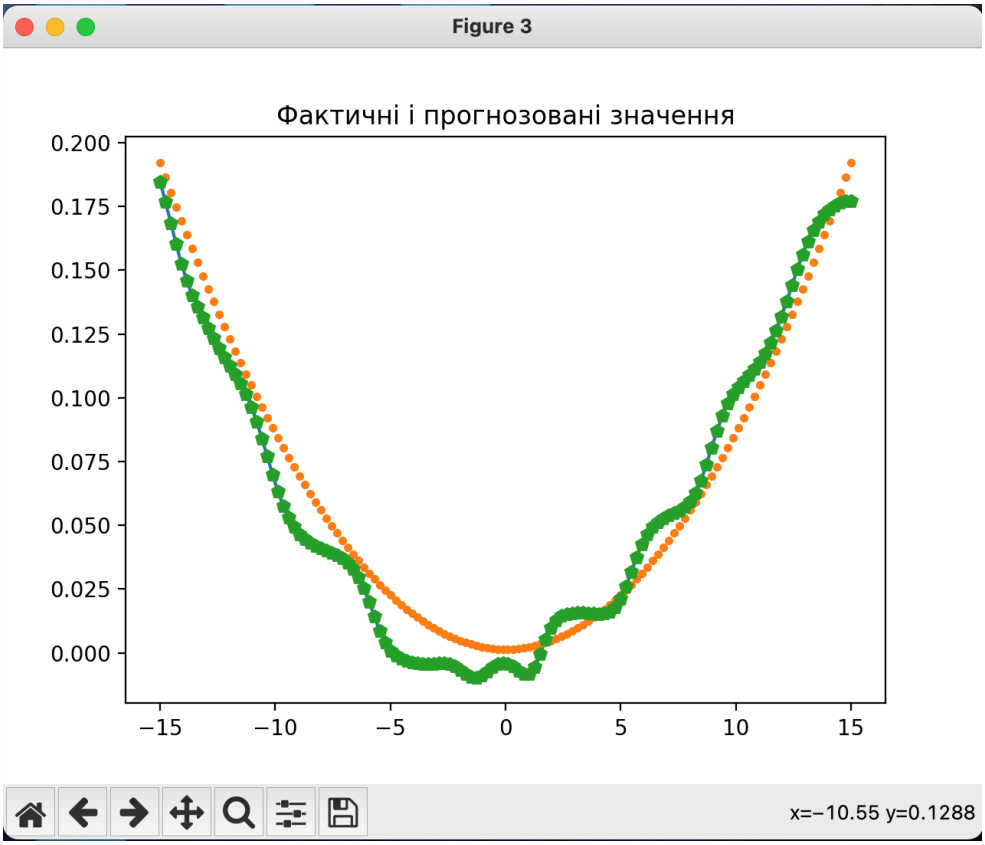

Рис. 19 Результат виконання програми

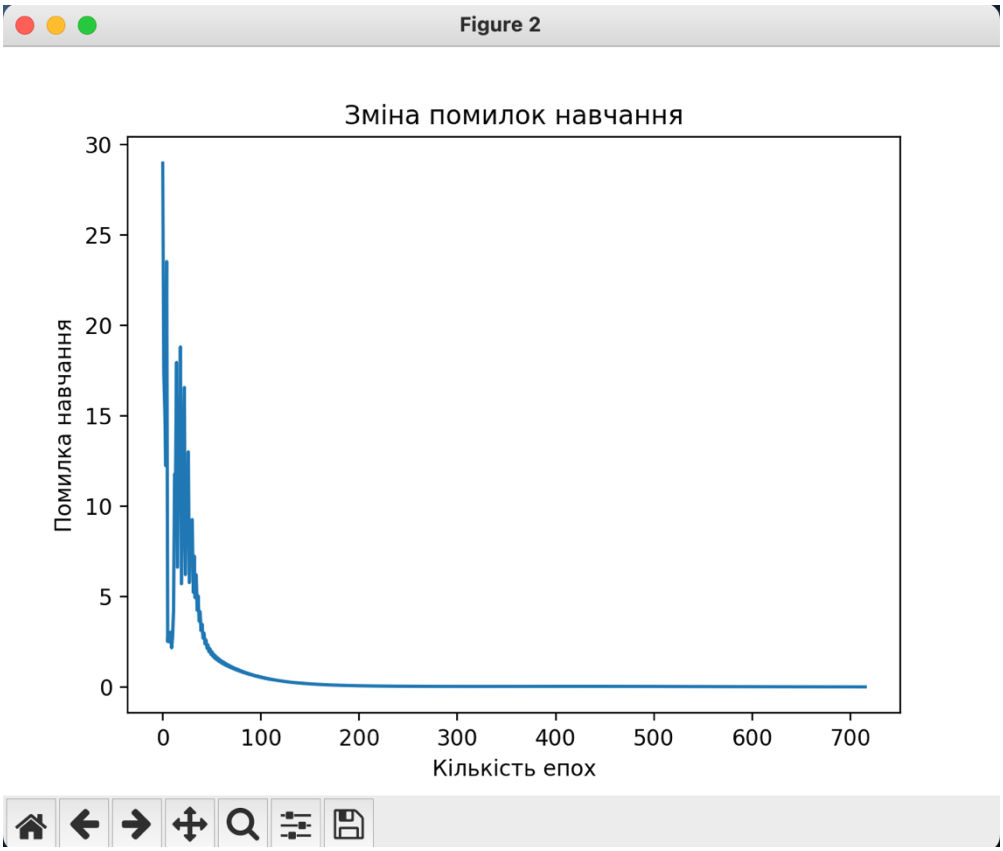

Рис. 20 Результат виконання програми

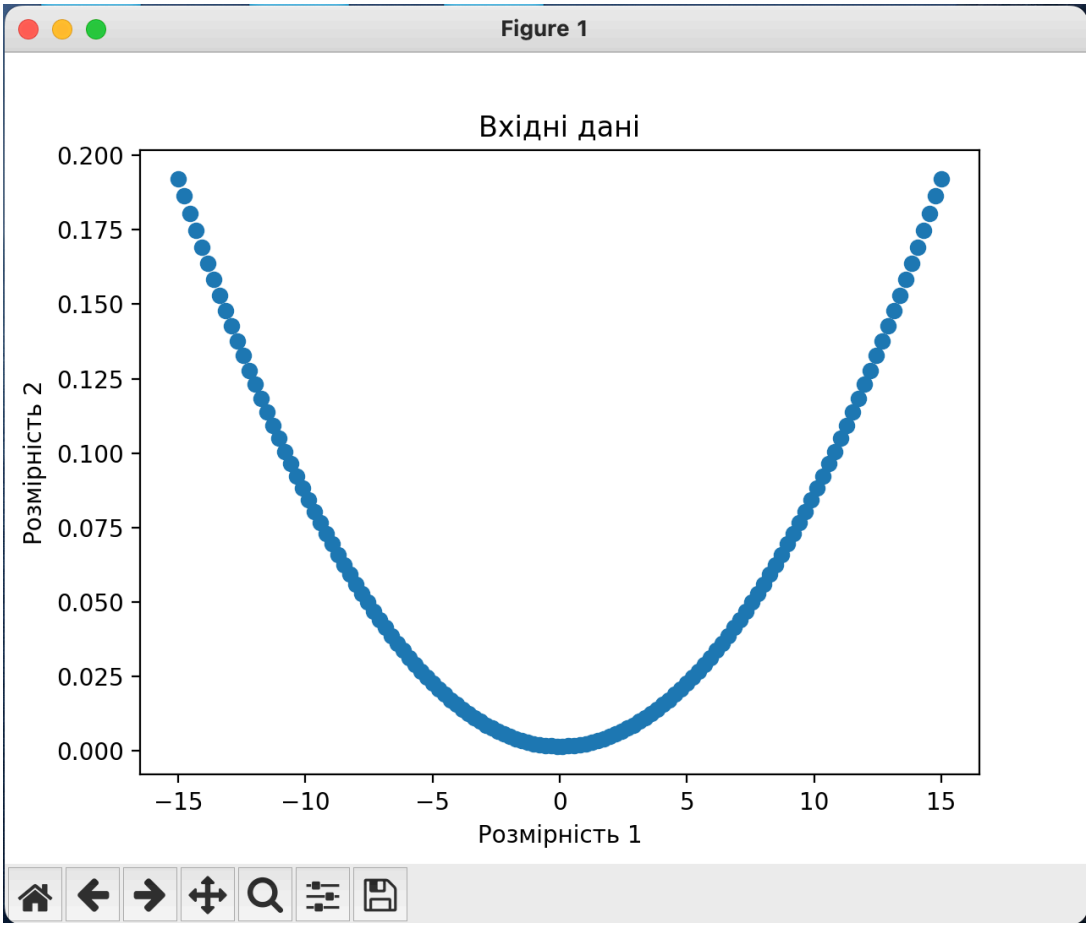

Рис. 21 Результат виконання програми

```
/usr/local/bin/python3.9 /Users/webb/Desktop/CWI/programs/lab5/LR_5_task_6.py
Epoch: 100; Error: 0.5418969790737939;
Epoch: 200; Error: 0.07476159901672944;
Epoch: 300; Error: 0.0367489505279848;
Epoch: 400; Error: 0.03950083749474286;
Epoch: 500; Error: 0.03530659343298223;
Epoch: 600; Error: 0.020818628062709787;
Epoch: 700; Error: 0.010994118334739683;
The goal of learning is reached
```

Рис. 22 Результат виконання програми

Завдання 2.6. Побудова багатошарової нейронної мережі для свого варіанту

Варіант 2	$y = 2x^2+6$	
2	2	2-1

```
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 = 2 * x * x + 6
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]], [2, 2, 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()
```

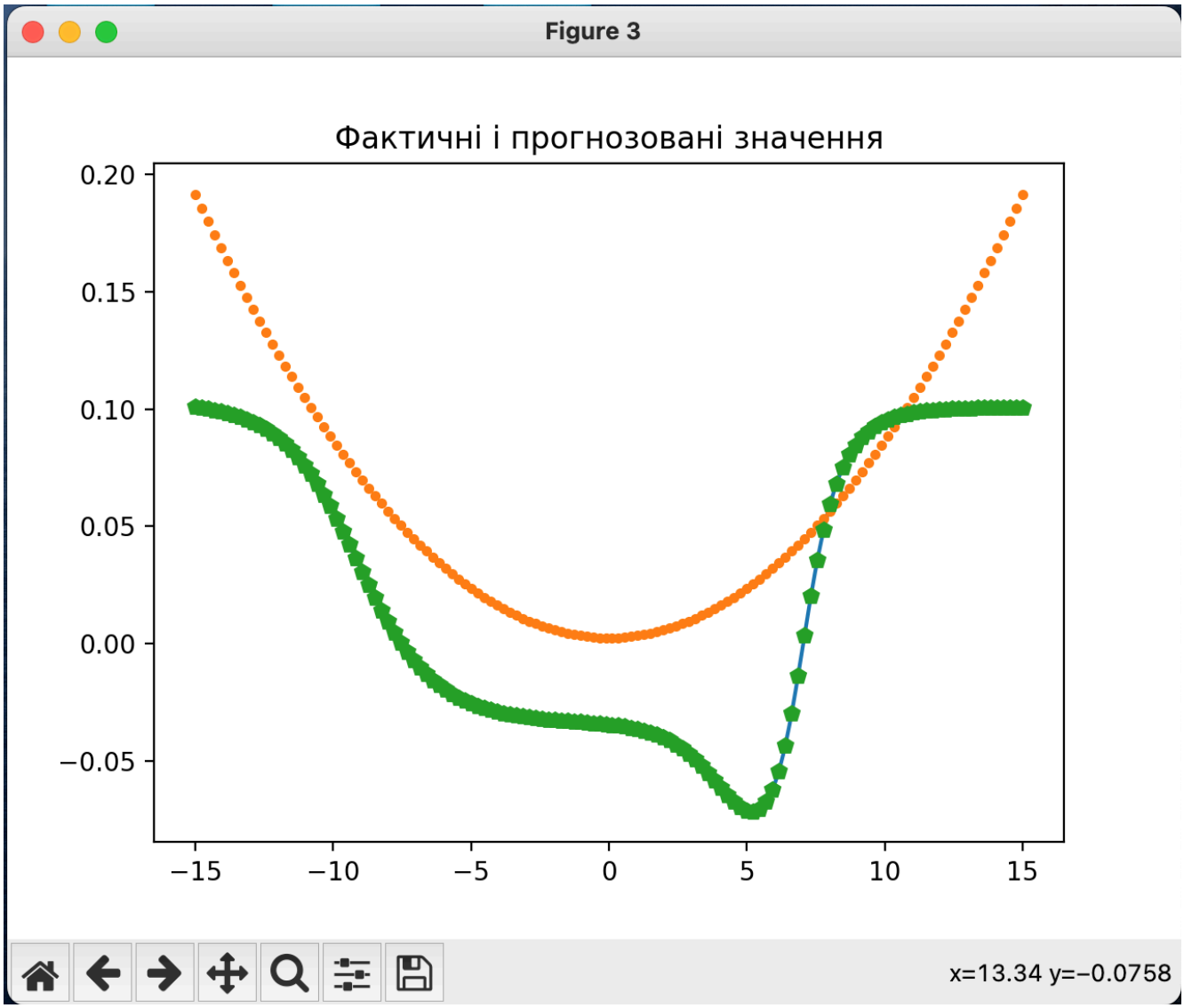

Рис. 24 Результат виконання програми

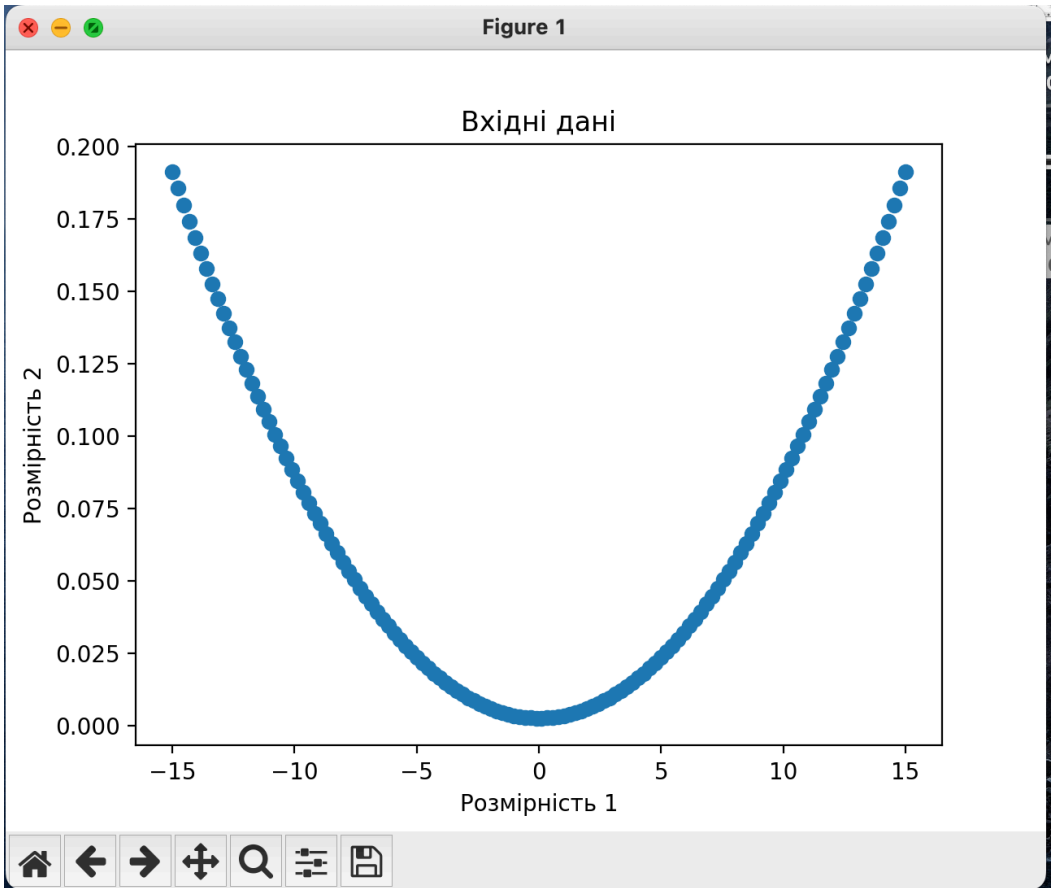

Рис. 25 Результат виконання програми

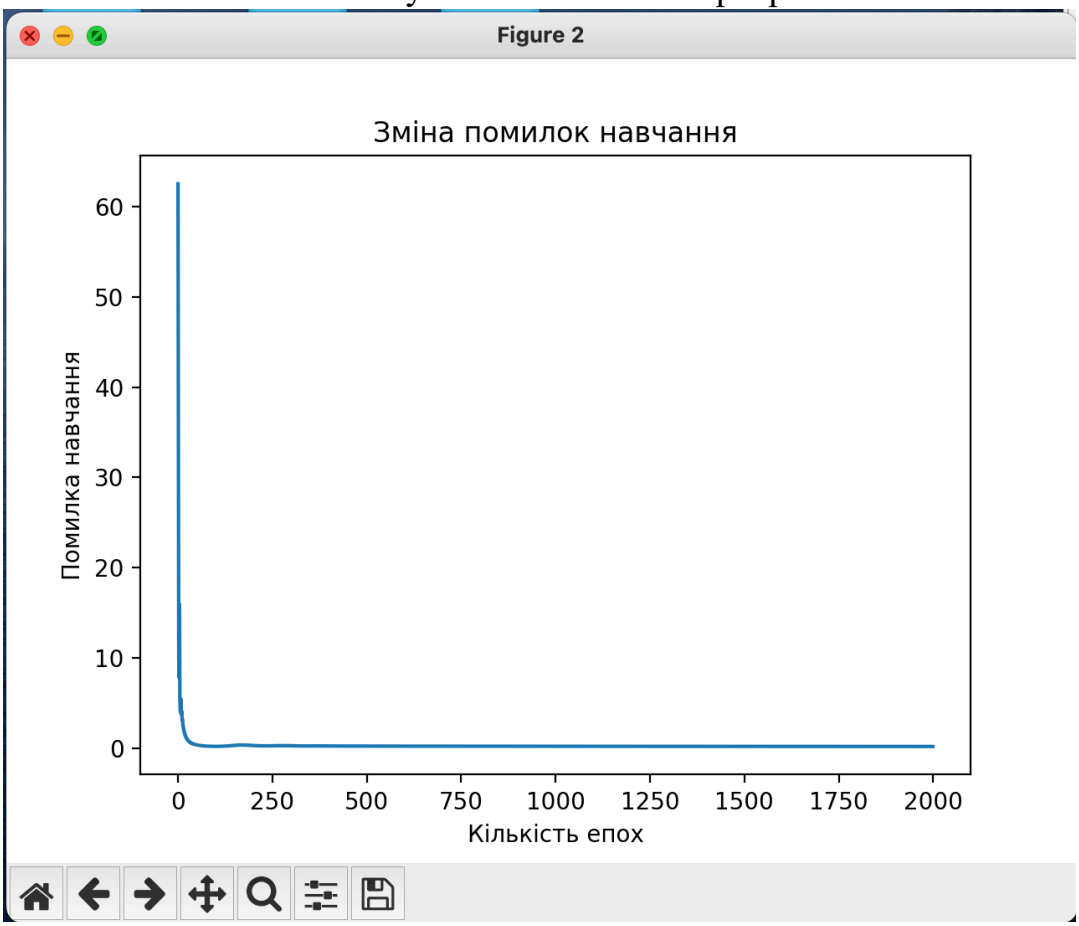

Рис. 26 Результат виконання програми

```
Epoch: 500; Error: 0.21276097759220783;
Epoch: 600; Error: 0.2064268444593929;
Epoch: 700; Error: 0.201834445754156;
Epoch: 800; Error: 0.19796374343910705;
Epoch: 900; Error: 0.19447078354024544;
Epoch: 1000; Error: 0.19125702454730453;
Epoch: 1100; Error: 0.18829402502225695;
Epoch: 1200; Error: 0.18556443334194123;
Epoch: 1300; Error: 0.18304845136027606;
Epoch: 1400; Error: 0.18072360545271982;
Epoch: 1500; Error: 0.17856677600818754;
Epoch: 1600; Error: 0.17655580195716758;
Epoch: 1700; Error: 0.1746703482288018;
Epoch: 1800; Error: 0.17289221449433212;
Epoch: 1900; Error: 0.17120530964364825;
Epoch: 2000; Error: 0.16959546213090337;
The maximum number of train epochs is reached

Process finished with exit code 0
```

Рис. 27 Результат виконання програми

Завдання 2.7. Побудова нейронної мережі на основі карти Кохонена, що самоорганізується

```
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)

net = nl.net.newc([[0.0, 1.0],[0.0, 1.0]], 4)
error = net.train(inp, epochs=200,  show=100)

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()
```

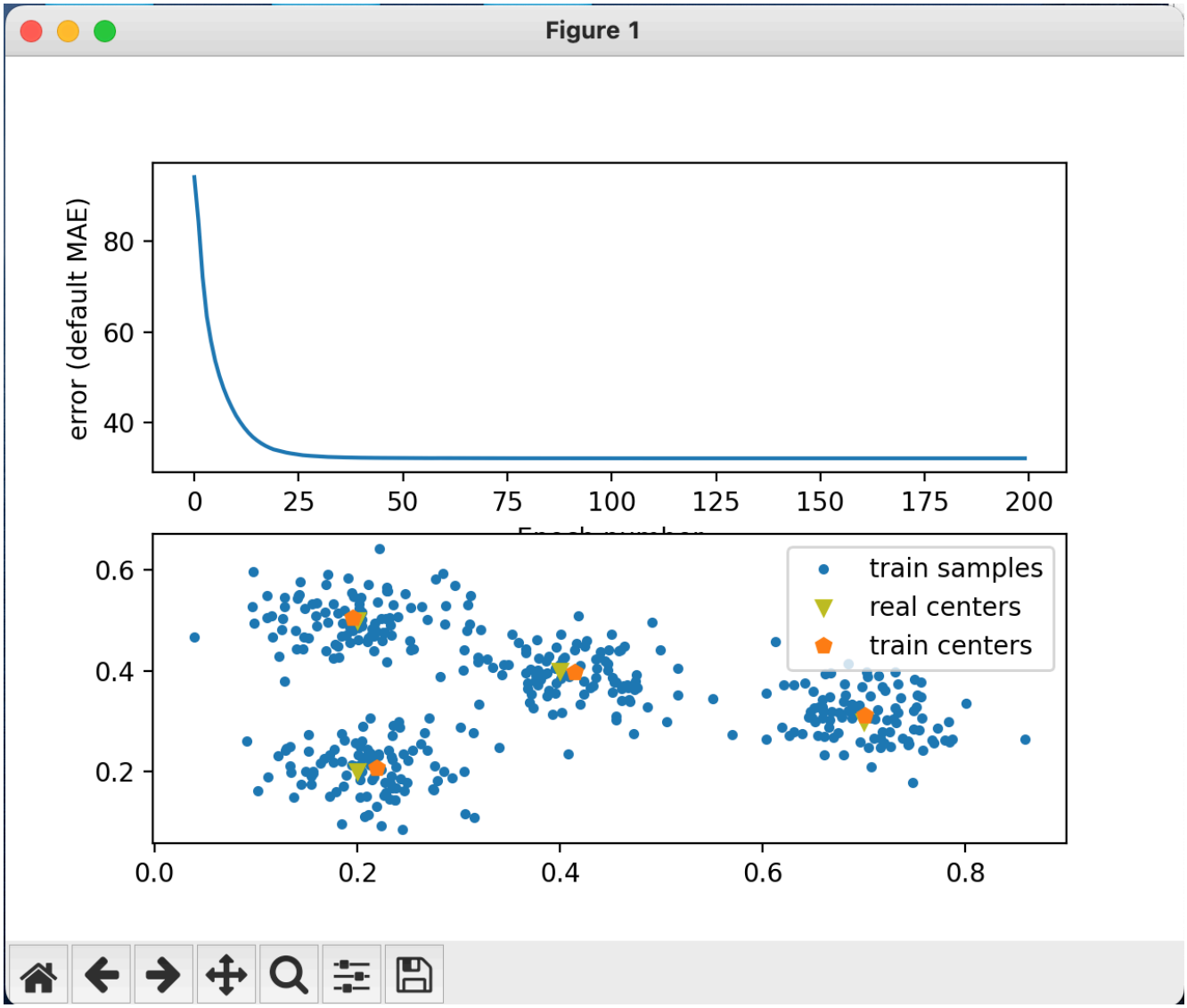

Рис. 29 Результат виконання програми

Завдання 2.8. Дослідження нейронної мережі на основі карти Кохонена, що само організується

Варіант 2	[0.1, 0.2], [0.4, 0.3], [0.7, 0.3], [0.2, 0.5], [0.5, 0.5]	0,03

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

skv = 0.03
centr = np.array([[0.2, 0.3], [0.4, 0.4], [0.7, 0.3], [0.2, 0.5], [0.4, 0.5]])
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 algoritm (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')
pl.legend(['train samples', 'real centers', 'train centers'])
pl.show()
```

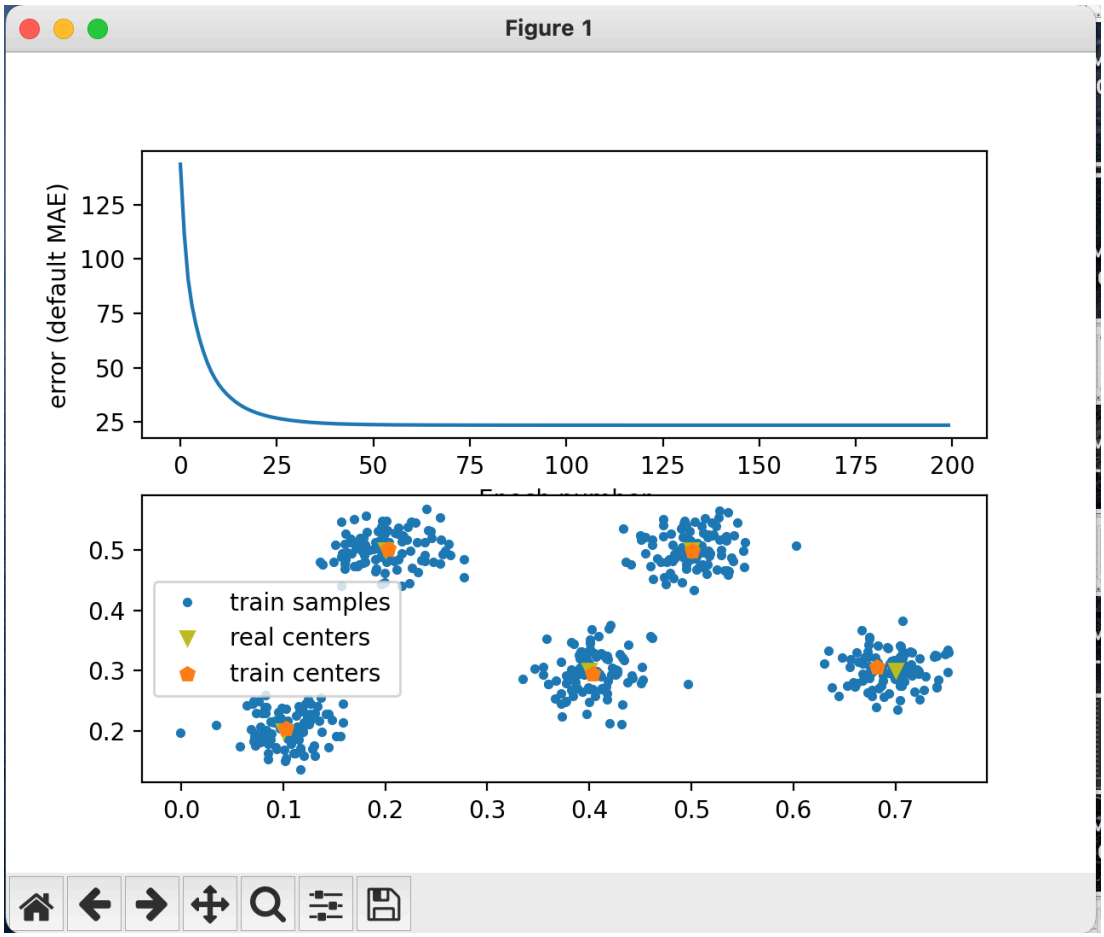

Рис. 31 Результат виконання програми

```
/usr/local/bin/python3.9 /Users/webb/Desktop/CWI/programs/lab5/LR_5_task_9.py
Epoch: 20; Error: 29.80283184938525;
Epoch: 40; Error: 24.384459292998258;
Epoch: 60; Error: 23.751959547809573;
Epoch: 80; Error: 23.658404885434358;
Epoch: 100; Error: 23.642408485616755;
Epoch: 120; Error: 23.639130833543085;
Epoch: 140; Error: 23.638446524092757;
Epoch: 160; Error: 23.63833758445506;
Epoch: 180; Error: 23.638335812166083;
Epoch: 200; Error: 23.638346035037443;
The maximum number of train epochs is reached
```

Рис. 32 Результат виконання програми

ВИСНОВОК: під час виконання лабараторної роботи, використовуючи спеціалізовані бібліотеки та мову програмування Python навчитися створювати та застосовувати прості нейронні мережі.