ЛАБОРАТОРНА РОБОТА №5

РОЗРОБКА ПРОСТИХ НЕЙРОННИХ МЕРЕЖ

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

ХІД РОБОТИ

Завдання №1. Створити простий нейрон.

```python
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:

```
C:\Users\yaros\AppData\Local\Programs\Python\Python311\python.exe "C:\Study\4Course\Системи штучного інтелекту\lab5\task1.py"
0.9990889488055994

Process finished with exit code 0
```

Рис. 1. Результат виконання завдання №1.

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

```python
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 GarbarNeuralNetwork:

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

Результат виконання:

```
C:\Users\yaros\AppData\Local\Programs\Python\Python311\python.exe "C:\Study\4Course\Системи штучного інтелекту\lab5\task2.py"
0.9990889488055994
0.7216325609518421

Process finished with exit code 0
```

Рис. 2. Результат виконання завдання №2.

V2

```
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 GarbarNeuralNetwork:
    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 = GarbarNeuralNetwork()
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:

```
C:\Users\yaros\AppData\Local\Programs\Python\Python311\python.exe "C:\Study\4Course\Системи штучного інтелекту\lab5\task2_v2.py"
Epoch 0 loss: 0.434
Epoch 10 loss: 0.284
Epoch 20 loss: 0.174
Epoch 30 loss: 0.113
Epoch 40 loss: 0.079
Epoch 50 loss: 0.059
Epoch 60 loss: 0.046
Epoch 70 loss: 0.038
Epoch 80 loss: 0.031
Epoch 90 loss: 0.027
Epoch 100 loss: 0.023
Epoch 110 loss: 0.021
Epoch 120 loss: 0.018
Epoch 130 loss: 0.017
Epoch 140 loss: 0.015
Epoch 150 loss: 0.014
Epoch 160 loss: 0.013
Epoch 170 loss: 0.012
Epoch 180 loss: 0.011
Epoch 190 loss: 0.010
Epoch 200 loss: 0.010
```

```
Epoch 800 loss: 0.002
Epoch 810 loss: 0.002
Epoch 820 loss: 0.002
Epoch 830 loss: 0.002
Epoch 840 loss: 0.002
Epoch 850 loss: 0.002
Epoch 860 loss: 0.002
Epoch 870 loss: 0.002
Epoch 880 loss: 0.002
Epoch 890 loss: 0.002
Epoch 900 loss: 0.002
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: 0.964
Frank: 0.039

Process finished with exit code 0
```

Рис. 3. Результат виконання завдання №2.

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 нейроннних 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.

```
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:

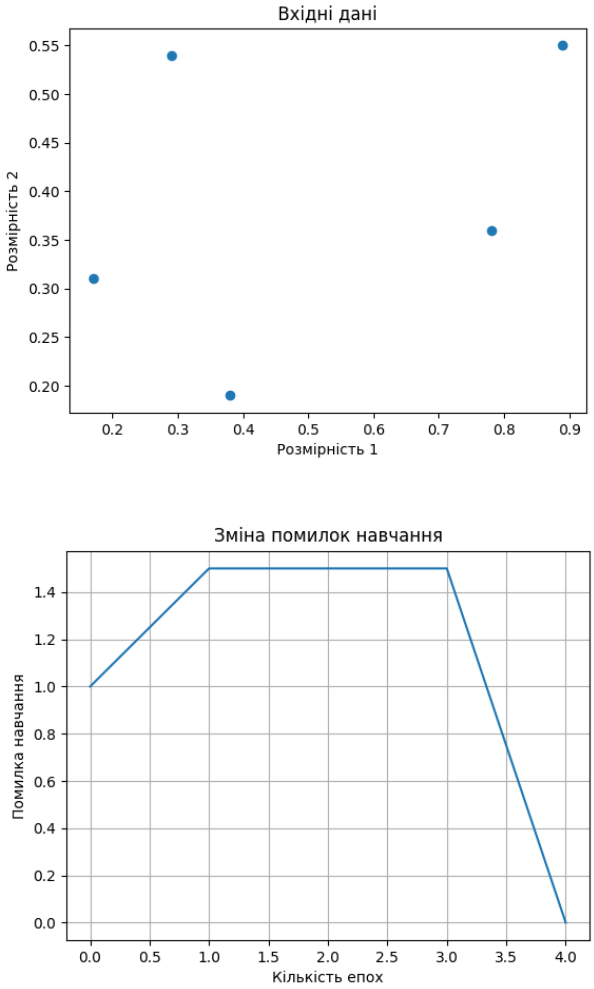


Рис. 4. Графік вхідних даних та процесу навчання.

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()
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:

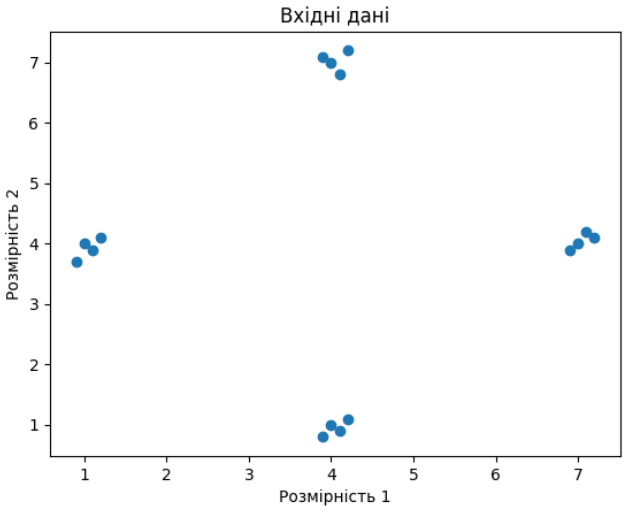


Рис. 5. Графік вхідних даних.

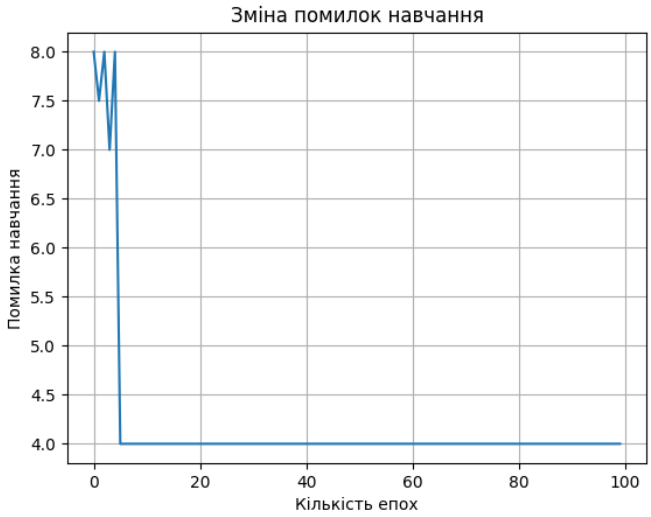


Рис. 6. Графік просування процесу навчання.

```
C:\Users\yaros\AppData\Local\Programs\Python\Python311\python.exe "C:\Study\4Course\Системи штучного інтелекту\lab5\task4.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.]

Process finished with exit code 0
```

Рис. 7. Результат виконання завдання №4.

Висновок до завдання:

На рис. 6 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 вибіркові 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)
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:

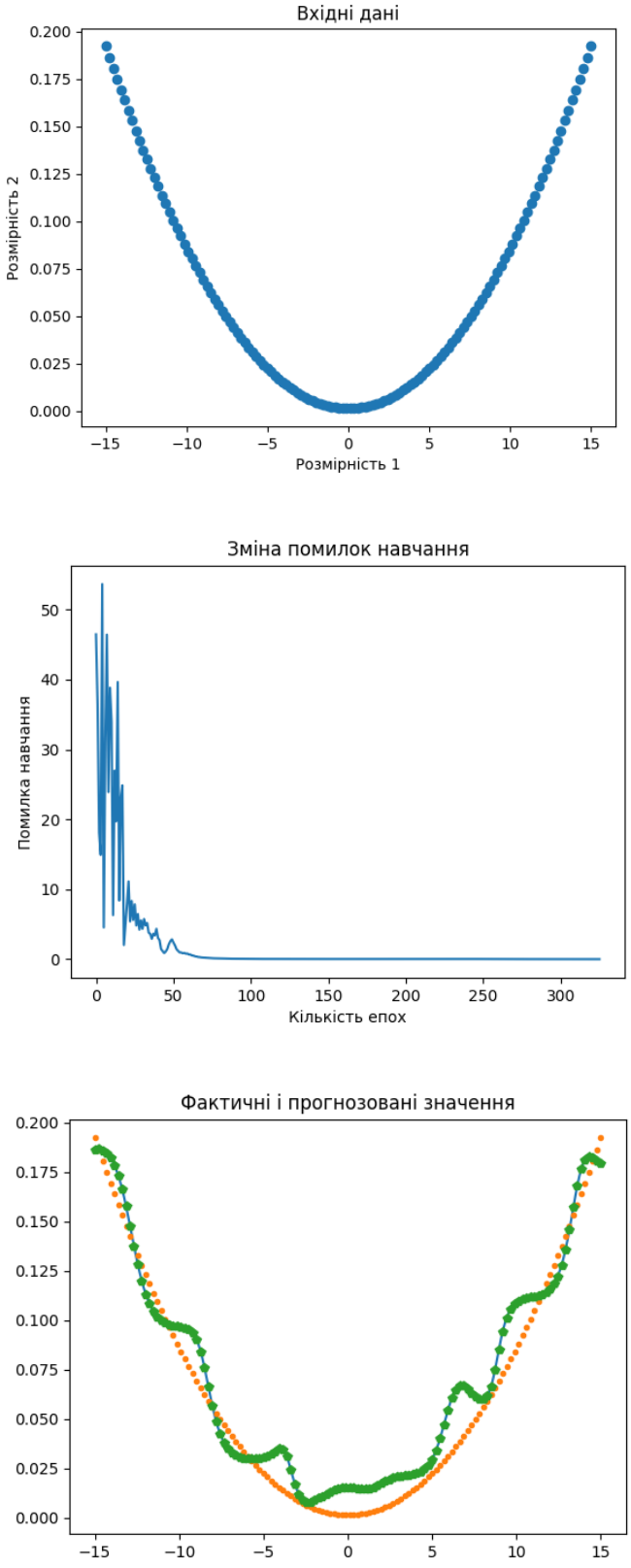


Рис. 8. Результат виконання завдання №5.

```
C:\Users\yaros\AppData\Local\Programs\Python\Python311\python.exe "C:\Study\4Course\Системи штучного інтелекту\lab5\task5.py"
Epoch: 100; Error: 0.054414322995261555;
Epoch: 200; Error: 0.060790000915742895;
Epoch: 300; Error: 0.013180694071081092;
The goal of learning is reached

Process finished with exit code 0
```

Рис. 9. Результат виконання завдання №5.

Висновок до завдання:

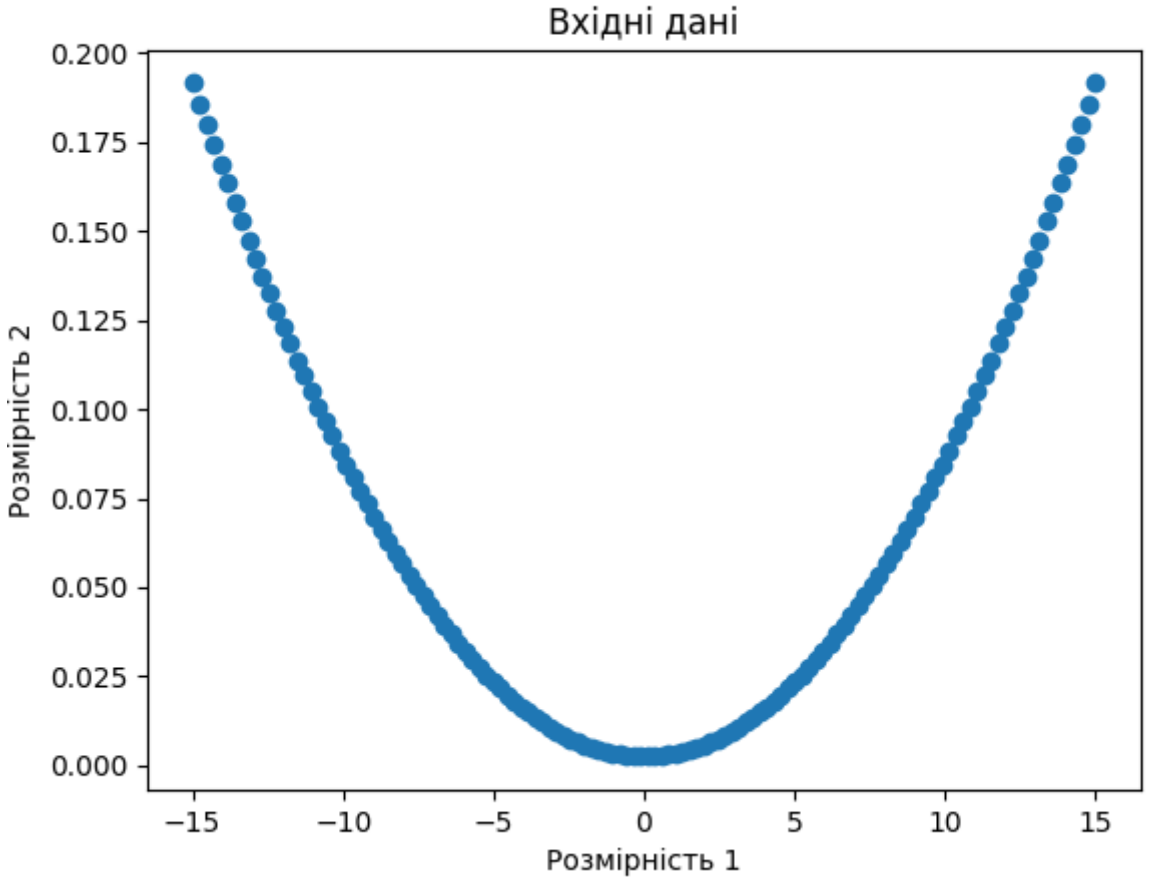
На рис. 9 зображено процес навчання мережі. На 100 епосі відбулось 0.32 помилки, на 200 епосі відбулось 0.16 помилки, на 300 епосі відбулось 0.11 помилки. Потім вивелось повідомлення, що ми досягли цілі навчання.

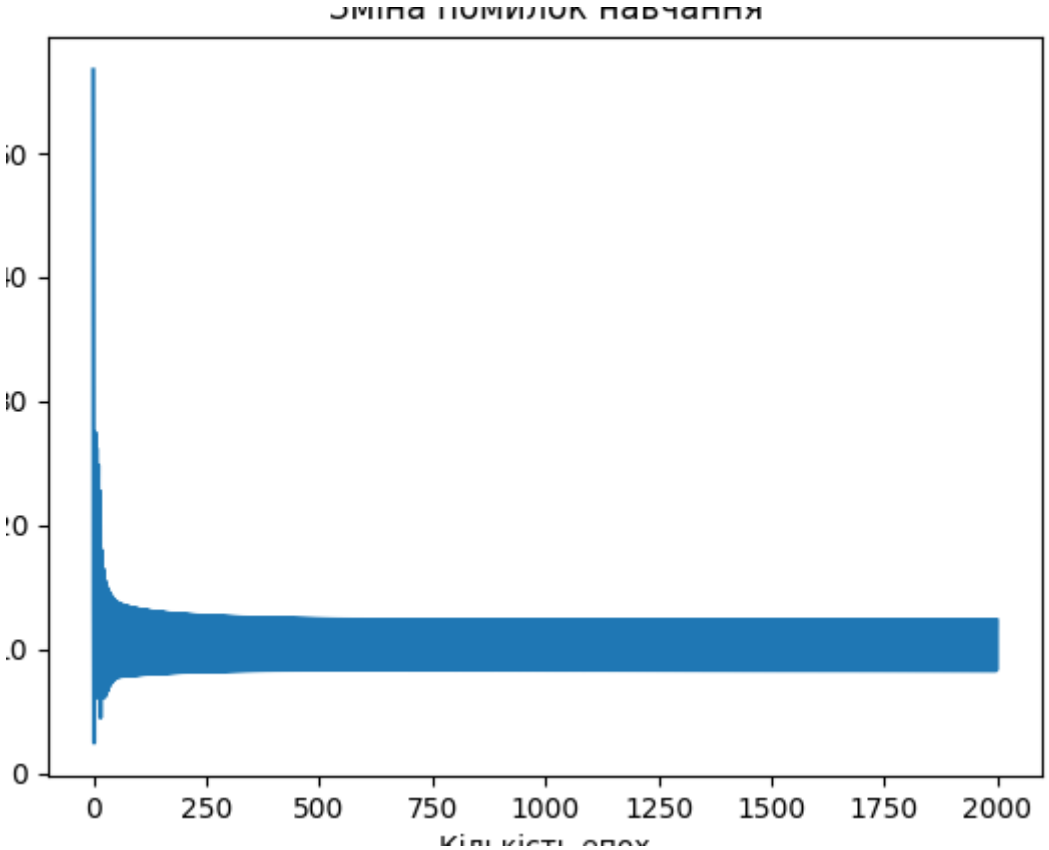
Завдання №6. Побудова багатошарової нейронної мережі для свого варіанту.

Варіант 9	$y = 3x^2+9$	
9	3	3-5-1

```
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 = 3 * x * x + 9
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]], [3, 5, 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:





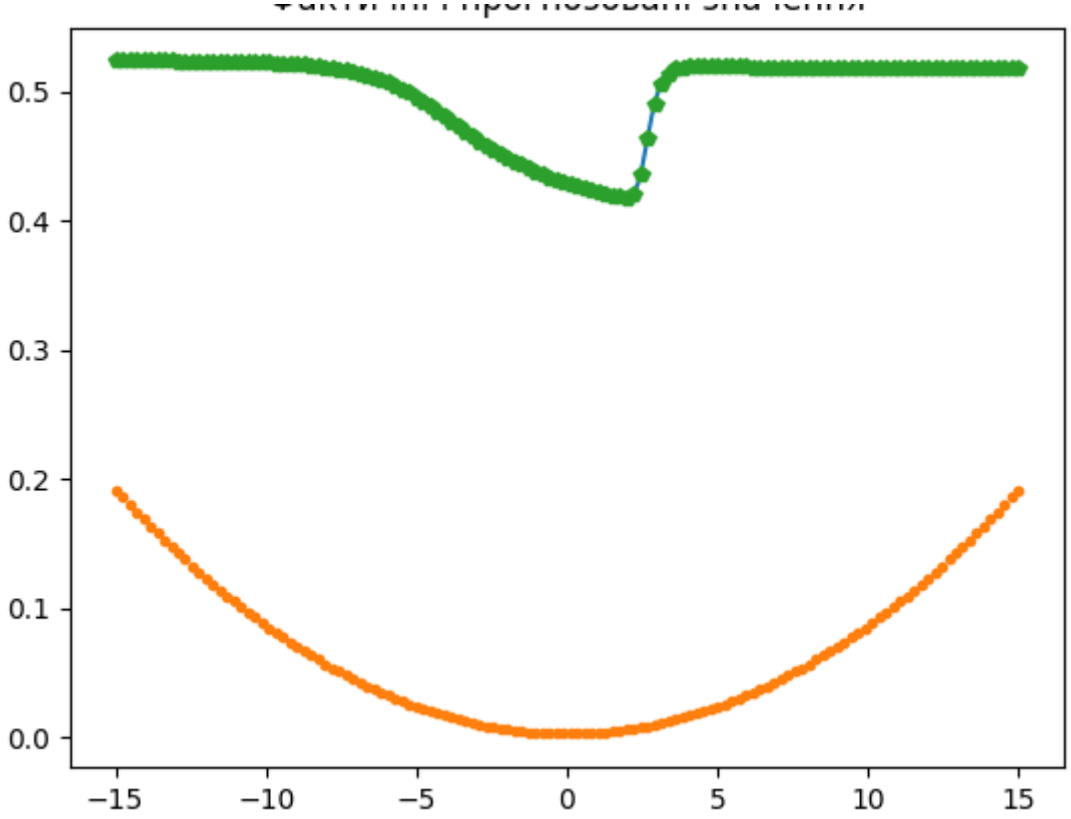


Рис. 10. Результат виконання завдання №6.

```
Epoch: 100; Error: 13.216933303428108;
Epoch: 200; Error: 12.780121441869813;
Epoch: 300; Error: 12.585893558126962;
Epoch: 400; Error: 12.47332241261676;
Epoch: 500; Error: 12.404428090545828;
Epoch: 600; Error: 12.365305193544772;
Epoch: 700; Error: 12.34558533868584;
Epoch: 800; Error: 12.337090683427286;
Epoch: 900; Error: 12.33414953423065;
Epoch: 1000; Error: 12.333268525466265;
Epoch: 1100; Error: 12.332564902142703;
Epoch: 1200; Error: 12.331271466123214;
Epoch: 1300; Error: 12.329755464608326;
Epoch: 1400; Error: 12.330437607765347;
Epoch: 1500; Error: 12.335817733236258;
Epoch: 1600; Error: 12.341302695994969;
Epoch: 1700; Error: 12.342252441457212;
Epoch: 1800; Error: 12.33927080342831;
Epoch: 1900; Error: 12.333960225944214;
Epoch: 2000; Error: 12.327302021892766;
The maximum number of train epochs is reached
```

Рис. 11. Результат виконання завдання №6.

Висновок до завдання:

На рис. 11 зображено процес навчання мережі. На 100 епосі відбулось 13.21 помилки, на 200 епосі відбулось 12.78 помилки, на 300 епосі відбулось 12,58 помилки і так далі, на 2000 епосі відбулось 12,32 помилки,. Потім вивелось повідомлення, що ми досягли максимальної кількості епох для тренування.

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

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

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 algoritm (CWTA)
error = net.train(inp, epochs=200, show=100)

# Plot results:


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:

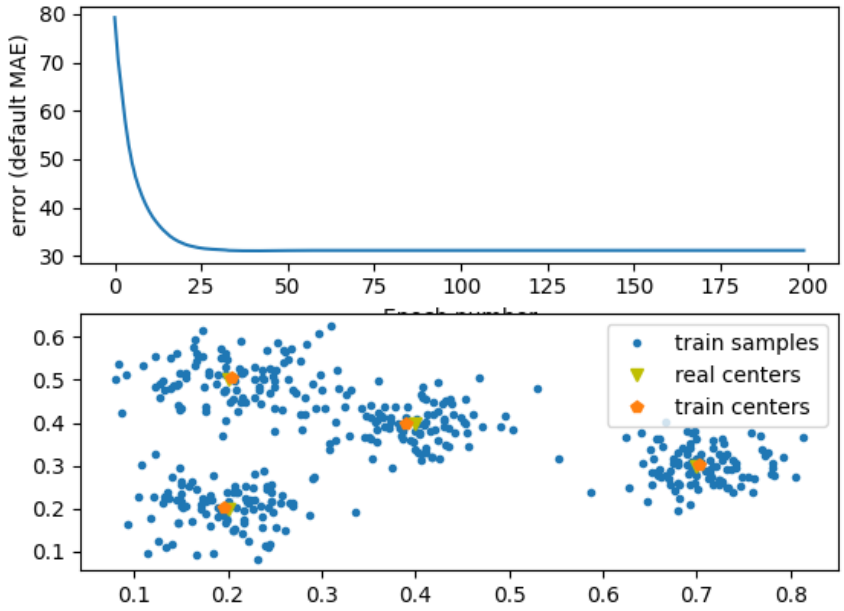


Рис. 12. Результат виконання завдання №7.

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

За результатами можна зазначити, що нейронна мережа була навчена на вхідних даних, а її завданням було навчитися апроксимувати центри даних (центри сконцентрованих точок). Помилка MAE вимірює різницю між передбачуваними центрами та реальними центрами. Нейромережа здатна визначати центри, які є близькими до реальних, але існує деяка помилка, яка може виникнути внаслідок шуму у даних або параметрів навчання мережі.

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

Варіант 9	[0.2, 0.3], [0.4, 0.4], [0.7, 0.3], [0.1, 0.5], [0.4, 0.5]	0,04

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

skv = 0.04
centr = np.array([[0.2, 0.3], [0.4, 0.4], [0.7, 0.3], [0.1, 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 4 neurons
net = nl.net.newc([[0.0, 1.0],[0.0, 1.0]], 4)
# 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()
```

Результат виконання:

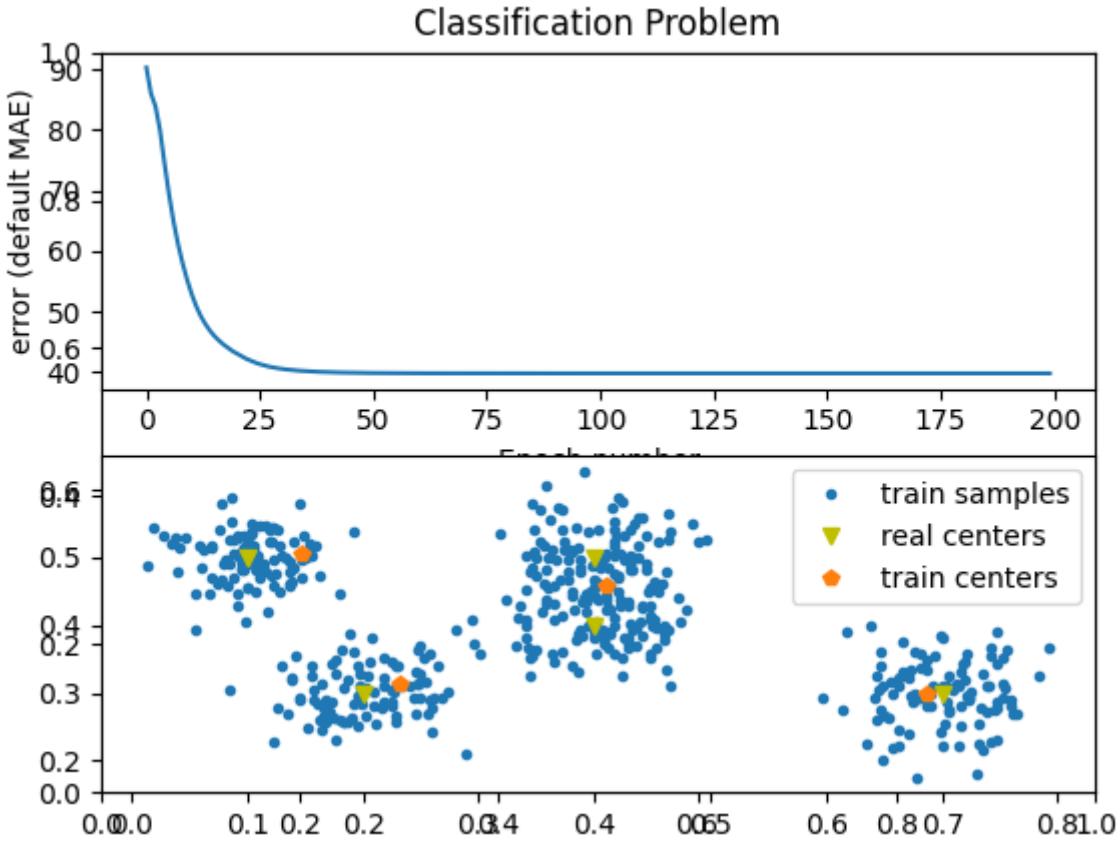


Рис. 13. Результат виконання завдання №8.

```
Epoch: 20; Error: 43.45139896273828;
Epoch: 40; Error: 40.00962191612465;
Epoch: 60; Error: 39.80761745995241;
Epoch: 80; Error: 39.78967618758342;
Epoch: 100; Error: 39.787798032573534;
Epoch: 120; Error: 39.78756894926685;
Epoch: 140; Error: 39.78753815957131;
Epoch: 160; Error: 39.78753377702263;
Epoch: 180; Error: 39.78753313411072;
Epoch: 200; Error: 39.787533038371556;
The maximum number of train epochs is reached
```

Рис. 14. Результат виконання завдання №8.

На рис. 14 зображено процес навчання мережі. На 20 епосі відбулось 43.45 помилки, на 40 епосі відбулось 40.09 помилки, на 60 епосі відбулось 39.80 помилки і так далі, на 200 епосі відбулось 39.78 помилки. Потім вивелось повідомлення, що ми досягли максимальної кількості епох для тренування.

V2

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

skv = 0.06
centr = np.array([[0.2, 0.1], [0.5, 0.4], [0.5, 0.3], [0.2, 0.5], [0.5, 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()
```

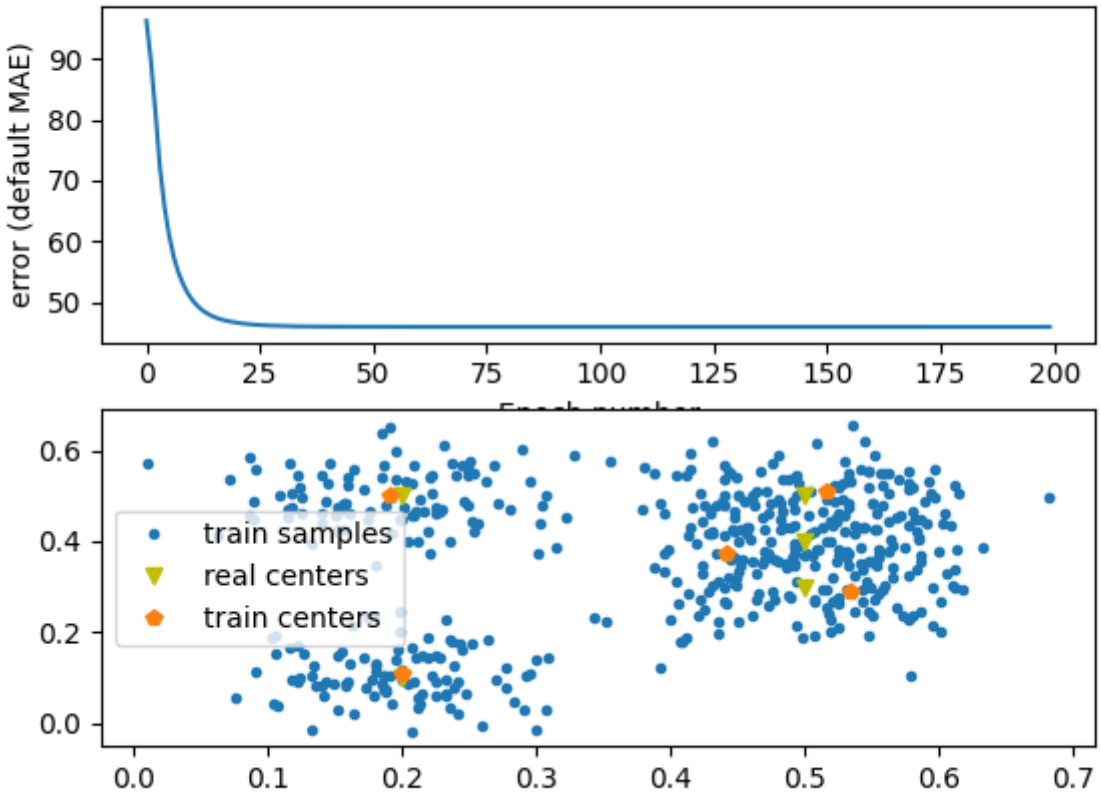


Рис. 15. Результат виконання завдання №8.

```
C:\Users\yaros\AppData\Local\Programs\Python\Python311\python.exe "C:\Study\4Course\Системи штучного інтелекту\lab5\task8_v2.py"
Epoch: 20; Error: 46.76328420190433;
Epoch: 40; Error: 45.99182784674552;
Epoch: 60; Error: 45.963104631113275;
Epoch: 80; Error: 45.960310544239434;
Epoch: 100; Error: 45.960062378562384;
Epoch: 120; Error: 45.96004423356482;
Epoch: 140; Error: 45.960043562014015;
Epoch: 160; Error: 45.96004366712542;
Epoch: 180; Error: 45.96004370083105;
Epoch: 200; Error: 45.960043706889614;
The maximum number of train epochs is reached
```

Рис. 16. Результат виконання завдання №8.

На рис. 16 зображено процес навчання мережі. На 20 епосі відбулось 46.76 помилки, на 40 епосі відбулось 45.99 помилки, на 60 епосі відбулось 45.96 помилки і так далі, на 200 епосі відбулось 45.96 помилки,. Потім вивелось повідомлення, що ми досягли максимальної кількості епох для тренування.

Висновок до завдання:

Порівнюючи нейронну мережу Кохонена із 4 та 5 нейронами, можна зробити наступні висновки. При 4 нейронах помилка MAE зменшується повільніше, ніж при 5 нейронах, і при цьому помилка при 5 нейронах є нижчою. З 5 нейронами обидва центри збігаються майже в одній точці. Важливо відзначити, що кількість нейронів в шарі Кохонена повинна відповідати кількості класів вхідних сигналів. У випадку з 5 вхідними сигналами потрібно мати 5 нейронів, а не 4. Таким чином, невірний вибір кількості нейронів впливає на величину помилки, ускладнюючи навчання мережі та сповільнюючи його. Такий невірний вибір може вплинути на результати, які ви побачили на рисунку 21, де нейронна мережа з 4 нейронами показує гірші результати порівняно з рисунком 23, де використовується 5 нейронів.

Висновок до лабораторної роботи:

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

GitHub: https://github.com/unravee1/AI_labs