

## ЛАБОРАТОРНА РОБОТА №5

### РОЗРОБКА ПРОСТИХ НЕЙРОННИХ МЕРЕЖ

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

Посилання на проект: <https://github.com/ipz202-rev/AI-lab5>

#### Хід роботи:

**Завдання №5.1.** Створити простий нейрон.

Лістинг файлу LR\_5\_task\_1.py:

```
import numpy as np

def sigmoid(x):
    # Наша функція активації:  $f(x) = 1 / (1 + e^{-x})$ 
    return 1 / (1 + np.exp(-x))

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

    def feedforward(self, inputs):
        # Вхідні дані про вагу, додавання зміщення
        # і подальше використання функції активації
        total = np.dot(self.weights, inputs) + self.bias
        return sigmoid(total)

weights = np.array([0, 1]) # w1 = 0, w2 = 1
bias = 4 # b = 4
n = Neuron(weights, bias)

x = np.array([2, 3]) # x1 = 2, x2 = 3
print(n.feedforward(x))
```

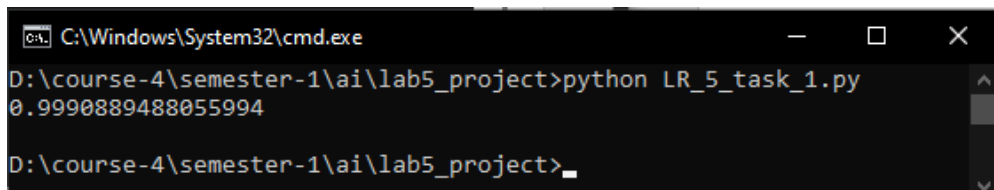


Рис.5.1.1. Результат виконання завдання.

					ДУ«Житомирська політехніка».23.121.23.000 – Лр5			
Змн.	Арк.	№ докум.	Підпис	Дата	Звіт з лабораторної роботи  ФІКТ Гр. ІПЗ-20-2[2]			
Розроб.	Рябова Є.В.							
Перевір.	Голенко М.Ю.							
Керівник								
Н. контр.								
Зав. каф.								
					Лім.	Арк.	Аркушів	
						1	19	

**Завдання №5.2.** Створити просту нейронну мережу для передбачення статі людини.

Лістинг файлу LR\_5\_task\_2.py:

```
import numpy as np
from LR_5_task_1 import Neuron, sigmoid

def deriv_sigmoid(x):
    # Похідна від sigmoid: f'(x) = f(x) * (1 - f(x))
    fx = sigmoid(x)
    return fx * (1 - fx)

def mse_loss(y_true, y_pred):
    # y_true и y_pred є масивами numpy з однаковою довжиною
    return ((y_true - y_pred) ** 2).mean()

class RyabovaNeuralNetwork:
    """
    Нейронна мережа, у якій:
        - 2 входи
        - прихований шар з двома нейронами (h1, h2)
        - шар виходу з одним нейроном (o1)

    *** ВАЖЛИВО ***:
    Код нижче написаний як простий, навчальний. НЕ оптимальний.
    Справжній код нейронної мережі виглядає не так. НЕ ВИКОРИСТОВУЙТЕ цей код у
    подальшому.
    Замість цього, прочитайте та запустіть його, щоб зрозуміти, як працює ця мере-
    жа.
    """

    def __init__(self):
        # Ваги
        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()

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

    def feedforward(self, x):
        # x є масивом numpy з двома елементами
        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):
        """
        - data is a (n x 2) numpy array, n = # of samples in the dataset.
        - all_y_trues is a numpy array with n elements.
          Elements in all_y_trues correspond to those in data.
        """
```

		Рябова Є.В.			ДУ«Житомирська політехніка».23.121.23.000 – Лр5	Арк.
		Голенко М.Ю.				2
Змн.	Арк.	№ докум.	Підпис	Дата		

```

learn_rate = 0.1
epochs = 1000 # кількість циклів у всьому наборі даних

for epoch in range(epochs):
    for x, y_true in zip(data, all_y_trues):
        # --- Виконуємо зворотній зв'язок (ці значання нам потрібні в по-
дальшому )

        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

        # --- Підрахунок часткових похідних
        # --- Найменування: d_L_d_w1 означає "частково L / частково w1"
        d_L_d_ypred = -2 * (y_true - y_pred)

        # Нейрон o1
        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)

        # Нейрон h1
        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)

        # Нейрон h2
        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)

        # --- Оновлюємо вагу і зміщення
        # Нейрон h1
        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

        # Нейрон h2
        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

        # Нейрон o1
        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

        # --- Підраховуємо загальні втрати в кінці кожної фази
        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))

# Задання набору даних

```

		Рябова Є.В.			ДУ«Житомирська політехніка».23.121.23.000 – Лр5	Арк.
		Голенко М.Ю.				3
Змн.	Арк.	№ докум.	Підпис	Дата		

```

data = np.array([
    [-2, -1], # Alice
    [25, 6], # Bob
    [17, 4], # Charlie
    [-15, -6], # Diana
])

all_y_trues = np.array([
    1, # Alice
    0, # Bob
    0, # Charlie
    1, # Diana
])

# Тренуємо вашу нейронну мережу!
network = RyabovaNeuralNetwork()
network.train(data, all_y_trues)

# Робимо передбачення
emily = np.array([-7, -3]) # 128 фунтов, 63 дюйма
frank = np.array([20, 2]) # 155 фунтов, 68 дюймов
print("Emily: %.3f" % network.feedforward(emily)) # 0.951 - F
print("Frank: %.3f" % network.feedforward(frank)) # 0.039 - M

```

```

C:\Windows\System32\cmd.exe
D:\course-4\semester-1\ai\lab5_project>python LR_5_task_2.py
0.9990889488055994
Epoch 0 loss: 0.064
Epoch 10 loss: 0.049
Epoch 20 loss: 0.040
Epoch 30 loss: 0.033
Epoch 40 loss: 0.028
Epoch 50 loss: 0.024
Epoch 60 loss: 0.021
Epoch 70 loss: 0.019
Epoch 80 loss: 0.017
Epoch 90 loss: 0.015
Epoch 100 loss: 0.014
Epoch 110 loss: 0.013
Epoch 120 loss: 0.012
Epoch 130 loss: 0.011
Epoch 140 loss: 0.010
Epoch 150 loss: 0.010
Epoch 160 loss: 0.009
Epoch 170 loss: 0.009
Epoch 180 loss: 0.008
Epoch 190 loss: 0.008
Epoch 200 loss: 0.007
Epoch 210 loss: 0.007
Epoch 220 loss: 0.007
Epoch 230 loss: 0.006
Epoch 240 loss: 0.006
Epoch 250 loss: 0.006
Epoch 260 loss: 0.006
Epoch 270 loss: 0.006
Epoch 280 loss: 0.005
Epoch 290 loss: 0.005
Epoch 300 loss: 0.005
Epoch 310 loss: 0.005
Epoch 320 loss: 0.005
Epoch 330 loss: 0.005
Epoch 340 loss: 0.004
Epoch 350 loss: 0.004
Epoch 360 loss: 0.004
Epoch 370 loss: 0.004
Epoch 380 loss: 0.004
Epoch 390 loss: 0.004
Epoch 400 loss: 0.004
Epoch 410 loss: 0.004
Epoch 420 loss: 0.004
Epoch 430 loss: 0.003
Epoch 440 loss: 0.003
Epoch 450 loss: 0.003
Epoch 460 loss: 0.003
Epoch 470 loss: 0.003
Epoch 480 loss: 0.003
Epoch 490 loss: 0.003
Epoch 500 loss: 0.003
Epoch 500 loss: 0.003
Epoch 510 loss: 0.003
Epoch 520 loss: 0.003
Epoch 530 loss: 0.003
Epoch 540 loss: 0.003
Epoch 550 loss: 0.003
Epoch 560 loss: 0.003
Epoch 570 loss: 0.003
Epoch 580 loss: 0.003
Epoch 590 loss: 0.002
Epoch 600 loss: 0.002
Epoch 610 loss: 0.002
Epoch 620 loss: 0.002
Epoch 630 loss: 0.002
Epoch 640 loss: 0.002
Epoch 650 loss: 0.002
Epoch 660 loss: 0.002
Epoch 670 loss: 0.002
Epoch 680 loss: 0.002
Epoch 690 loss: 0.002
Epoch 700 loss: 0.002
Epoch 710 loss: 0.002
Epoch 720 loss: 0.002
Epoch 730 loss: 0.002
Epoch 740 loss: 0.002
Epoch 750 loss: 0.002
Epoch 760 loss: 0.002
Epoch 770 loss: 0.002
Epoch 780 loss: 0.002
Epoch 790 loss: 0.002
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: 0.001
Epoch 970 loss: 0.001
Epoch 980 loss: 0.001
Epoch 990 loss: 0.001
Emily: 0.964
Frank: 0.038
D:\course-4\semester-1\ai\lab5_project>

```

Рис.5.2.1 – 5.2.2. Результат виконання завдання.

		Рябова Є.В.			ДУ«Житомирська політехніка».23.121.23.000 – Лр5	Арк.
		Голенко М.Ю.				4
Змн.	Арк.	№ докум.	Підпис	Дата		

Функція активації додає нелінійність до вихідних значень нейронів, дозволяючи мережі виявляти та навчатися складним нелінійним взаємозв'язкам в даних, що дає можливість моделювати складні структури. Функції активації, такі як сигмоїда, вводять необхідний елемент гнучкості для ефективного вирішення завдань.

Нейронні мережі прямого поширення є ефективними для класифікації та регресії, дозволяючи точно визначати категорії об'єктів та прогнозувати числові значення. Вони також успішно використовуються для розпізнавання образів в зображеннях та вирішення завдань мовного розпізнавання, надаючи гнучкість та широкий спектр застосувань у різних областях, включаючи фінансову аналітику. Ці мережі взаємодіють з даними, передаючи інформацію від входу до виходу через шари нейронів без зворотного зв'язку.

На основі навчання нейронна мережа здатна з високою точністю передбачати стать осіб. Наприклад, для вхідних даних Emily отримано ймовірність 0.964 для жіночої статі, тоді як для Frank отримано ймовірність 0.038 для чоловічої статі.

**Завдання №5.3.** Класифікатор на основі перцептрону з використанням бібліотеки NeuroLab.

Лістинг файлу LR\_5\_task\_3.py:

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

# Завантаження вхідних даних
text = np.loadtxt('data_perceptron.txt')

# Поділ точок даних та міток
data = text[:, :2]
labels = text[:, 2].reshape((text.shape[0], 1))

# Побудова графіка вхідних даних
plt.figure()
plt.scatter(data[:, 0], data[:, 1])
plt.xlabel('Розмірність 1')
plt.ylabel('Розмірність 2')
plt.title('Вхідні дані')

# Визначення максимального та мінімального значень для кожного виміру
dim1_min, dim1_max, dim2_min, dim2_max = 0, 1, 0, 1

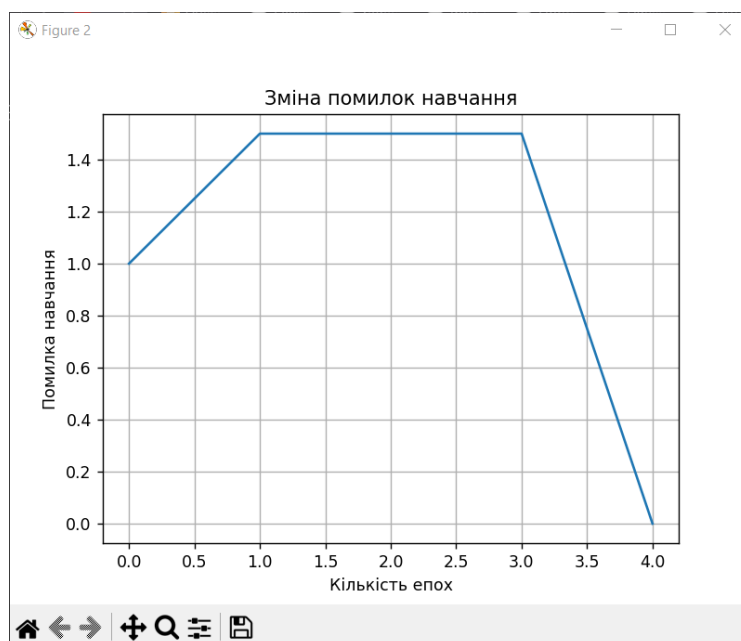
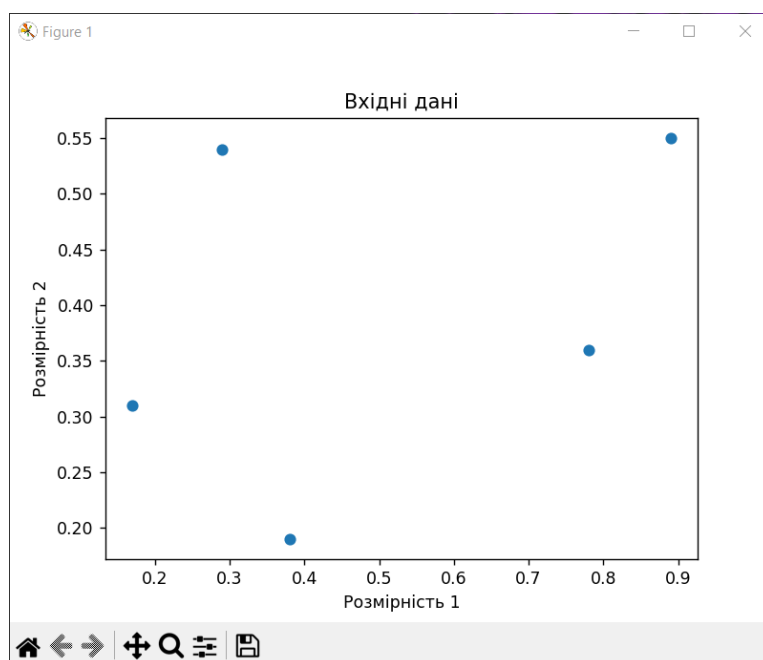
# Кількість нейронів у вихідному шарі
num_output = labels.shape[1]
```

		Рябова Є.В.			ДУ«Житомирська політехніка».23.121.23.000 – Лр5	Арк.
		Голенко М.Ю.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

```
# Визначення перцептрону з двома вхідними нейронами (оскільки вхідні дані - двовимірні)
dim1 = [dim1_min, dim1_max]
dim2 = [dim2_min, dim2_max]
perceptron = nl.net.newp([dim1, dim2], num_output)

# Тренування перцептрону з використанням наших даних
error_progress = perceptron.train(data, labels, epochs=100, show=20, lr=0.03)

# Побудова графіка процесу навчання
plt.figure()
plt.plot(error_progress)
plt.xlabel('Кількість епох')
plt.ylabel('Помилка навчання')
plt.title('Зміна помилок навчання')
plt.grid()
plt.show()
```



		Рябова Є.В.			ДУ«Житомирська політехніка».23.121.23.000 – Лр5	Арк.
		Голенко М.Ю.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

```
D:\course-4\semester-1\ai\lab5_project\venv\Scripts\python.exe
The goal of learning is reached

Process finished with exit code 0
```

Рис.5.3.1 – 5.3.3. Результат виконання завдання.

Цей графік відображає процес навчання перцептрону. “Кількість епох” відноситься до кількості повних проходів через навчальний набір даних, а “Помилка навчання” - це помилка, яку модель робить на навчальному наборі даних. За мірою збільшення кількості епох помилка навчання зменшується, що означає, що модель навчається і покращує свою здатність передбачувати правильні відповіді. По графіку можна побачити, що перцептрон успішно навчився розділяти дані.

#### Завдання №5.4. Побудова одношарової нейронної мережі.

Лістинг файлу LR\_5\_task\_4.py:

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

# Завантаження вхідних даних
text = np.loadtxt('data_simple_nn.txt')

# Поділ даних на точки даних та мітки
data = text[:, 0:2]
labels = text[:, 2:]

# Побудова графіка вхідних даних
plt.figure()
plt.scatter(data[:, 0], data[:, 1])
plt.xlabel('Розмірність 1')
plt.ylabel('Розмірність 2')
plt.title('Вхідні дані')

# Мінімальне та максимальне значення для кожного виміру
dim1_min, dim1_max = data[:, 0].min(), data[:, 0].max()
dim2_min, dim2_max = data[:, 1].min(), data[:, 1].max()

# Визначення кількості нейронів у вихідному шарі
num_output = labels.shape[1]

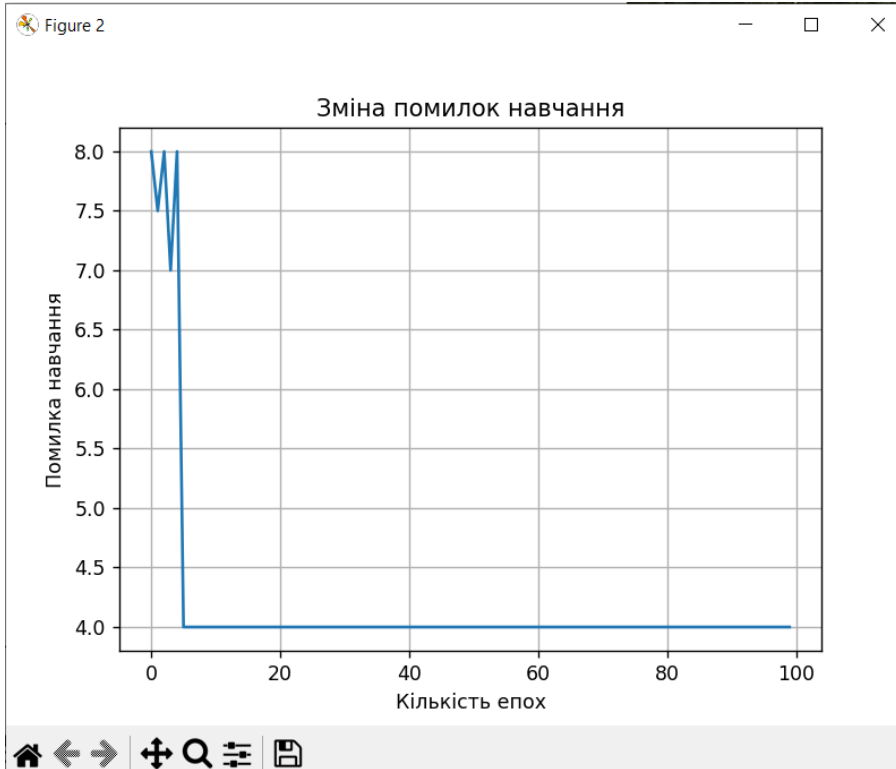
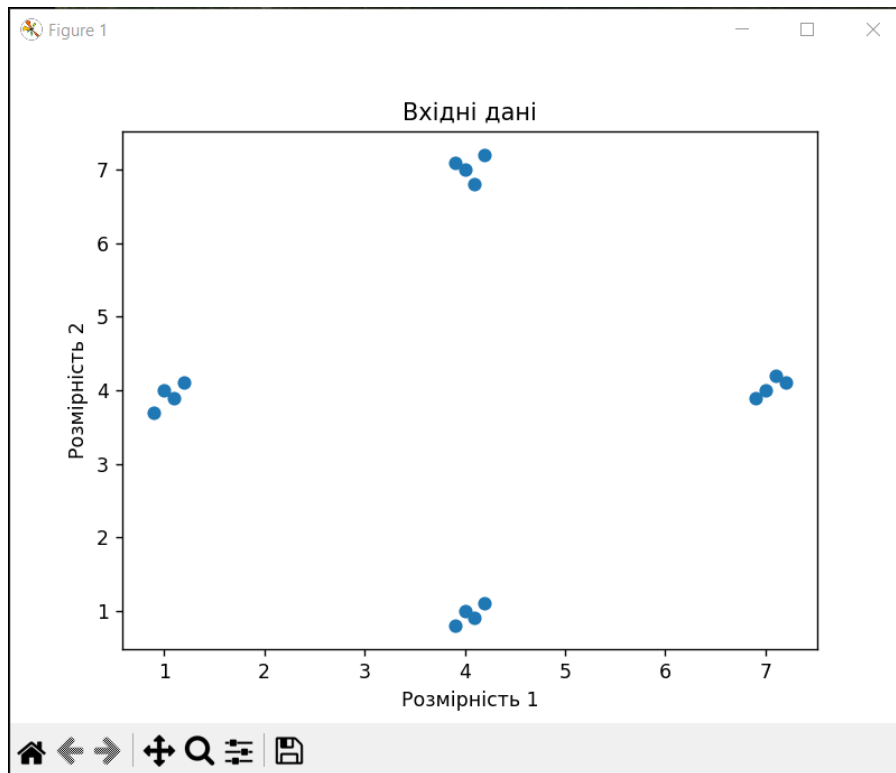
# Визначення одношарової нейронної мережі
dim1 = [dim1_min, dim1_max]
dim2 = [dim2_min, dim2_max]
nn = nl.net.newp([dim1, dim2], num_output)
error_progress = nn.train(data, labels, epochs=100, show=20, lr=0.03)

# Побудова графіка просування процесу навчання
plt.figure()
plt.plot(error_progress)
plt.xlabel('Кількість епох')
plt.ylabel('Помилка навчання')
```

		Рябова Є.В.			ДУ«Житомирська політехніка».23.121.23.000 – Лр5	Арк.
		Голенко М.Ю.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

```
plt.title('Зміна помилок навчання')
plt.grid()
plt.show()

# Виконання класифікатора на тестових точках даних
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])
```



		Рябова Є.В.			ДУ«Житомирська політехніка».23.121.23.000 – Лр5	Арк.
		Голенко М.Ю.				8
Змн.	Арк.	№ докум.	Підпис	Дата		



```

D:\course-4\semester-1\ai\lab5_project\venv\Scripts\python.exe
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.5.4.1 – 5.4.3. Результат виконання завдання.

Перший графік показує вхідні дані, які були використані для навчання нейронної мережі. Він має дві розмірності, "Розмірність 1" та "Розмірність 2", і точки даних представлені синіми крапками, згруповані в чотири кластери. Другий графік показує зміну помилок навчання протягом 100 епох. За результатами, помилка навчання залишалася стабільною після 5-10 епохи і дорівнювала 4.0 протягом всього процесу навчання. Після навчання нейронна мережа була використана для класифікації трьох тестових точок даних. Результати показали, що перша точка була класифікована як клас [0. 0.], друга – [1. 0.], а третя – [1. 1.].

### Завдання №5.5. Побудова багатошарової нейронної мережі.

Лістинг файлу LR\_5\_task\_5.py:

```

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

# Генерація тренувальних даних
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)

# Створення даних та міток
data = x.reshape(num_points, 1)
labels = y.reshape(num_points, 1)

# Побудова графіка вхідних даних
plt.figure()

```

		Рябова Є.В.			ДУ«Житомирська політехніка».23.121.23.000 – Лр5	Арк.
		Голенко М.Ю.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

```

plt.scatter(data, labels)
plt.xlabel('Розмірність 1')
plt.ylabel('Розмірність 2')
plt.title('Вхідні дані')

# Визначення багатoshарової нейронної мережі з двома прихованими
# шарами. Перший прихований шар складається із десяти нейронів.
# Другий прихований шар складається з шести нейронів.
# Вихідний шар складається з одного нейрона.
nn = nl.net.newff([[min_val, max_val]], [10, 6, 1])

# Завдання градієнтного спуску як навчального алгоритму
nn.trainf = nl.train.train_gd

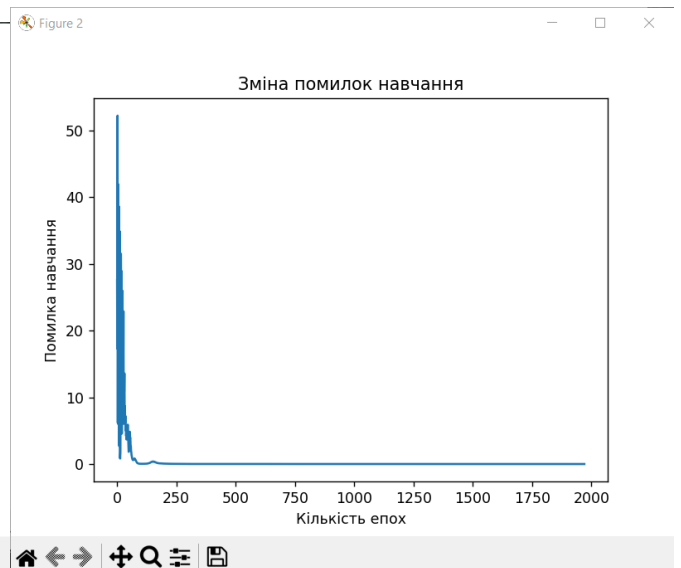
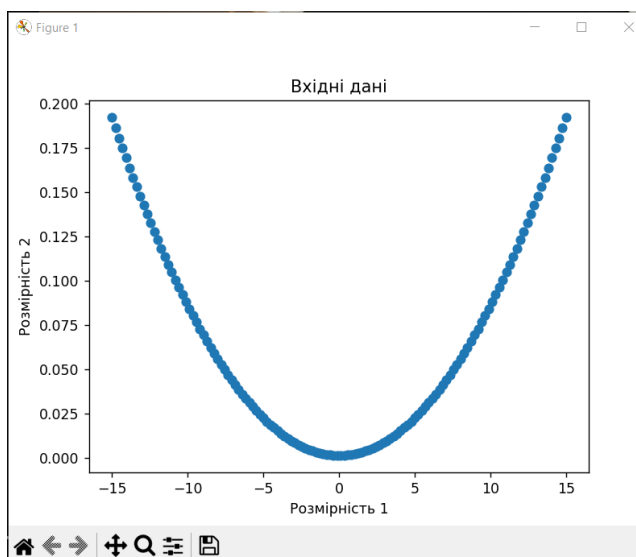
# Тренування нейронної мережі
error_progress = nn.train(data, labels, epochs=2000, show=100, goal=0.01)

# Виконання нейронної мережі на тренувальних даних
output = nn.sim(data)
y_pred = output.reshape(num_points)

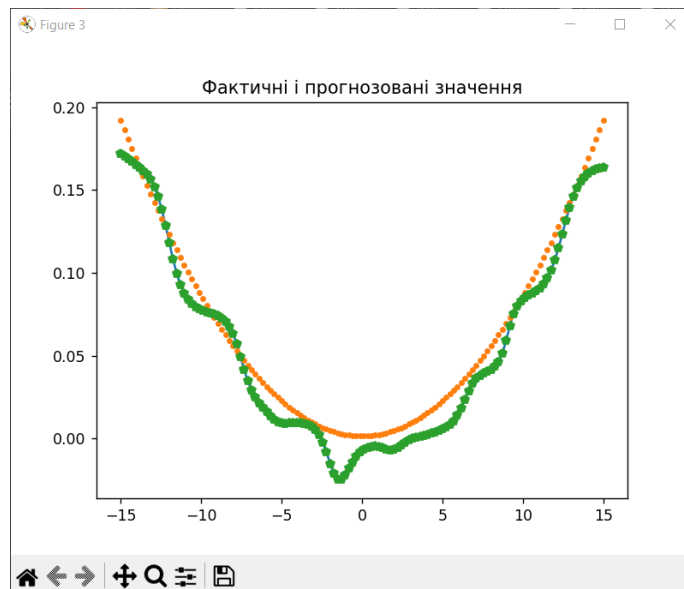
# Побудова графіка помилки навчання
plt.figure()
plt.plot(error_progress)
plt.xlabel('Кількість епох')
plt.ylabel('Помилка навчання')
plt.title('Зміна помилок навчання')

# Побудова графіка результатів
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()

```



		Рябова Є.В.			ДУ«Житомирська політехніка».23.121.23.000 – Лр5	Арк.
		Голенко М.Ю.				10
Змн.	Арк.	№ докум.	Підпис	Дата		



```
D:\course-4\semester-1\ai\lab5_project\venv\Scripts\python.exe
Epoch: 100; Error: 0.045701565001542814;
Epoch: 200; Error: 0.06971852467552621;
Epoch: 300; Error: 0.020418723633166814;
Epoch: 400; Error: 0.01693876971835017;
Epoch: 500; Error: 0.01858297732203288;
Epoch: 600; Error: 0.01906197437080181;
Epoch: 700; Error: 0.0158695258225786;
Epoch: 800; Error: 0.012595780584128361;
Epoch: 900; Error: 0.011554500755920665;
Epoch: 1000; Error: 0.01236432414999313;
Epoch: 1100; Error: 0.013649664385528023;
Epoch: 1200; Error: 0.013478564934872182;
Epoch: 1300; Error: 0.012117667805018498;
Epoch: 1400; Error: 0.011044755242555004;
Epoch: 1500; Error: 0.010783559037924998;
Epoch: 1600; Error: 0.011067063514797151;
Epoch: 1700; Error: 0.011277079313020948;
Epoch: 1800; Error: 0.010988733282169986;
Epoch: 1900; Error: 0.010398516757447575;
The goal of learning is reached
Process finished with exit code 0
```

Рис.5.5.1 – 5.5.4. Результат виконання завдання.

У результаті виконання коду була створена багатошарова нейронна мережа, яка успішно навчилася відтворювати квадратичну залежність вхідних даних. Помилка тренування систематично зменшувалась протягом 2000 епох, досягнувши мети навчання. Графік фактичних та прогнозованих значень підтверджує високу точність мережі у відтворенні залежності між вхідними та вихідними даними.

		Рябова Є.В.			ДУ«Житомирська політехніка».23.121.23.000 – Лр5	Арк.
		Голенко М.Ю.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

**Завдання №5.6.** Побудова багатошарової нейронної мережі для свого варіанту.

Варіант	Тестові дані	Багатошаровий персептрон	
		Кількість шарів	Кількості нейронів у шарах
23	$y = 2x^2 + 2x + 1$	3	3-3-1

Лістинг файлу LR\_5\_task\_6.py:

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

# Генерація тренувальних даних
min_val = -15
max_val = 15
num_points = 130
x = np.linspace(min_val, max_val, num_points)
y = 2 * np.square(x) + 2 * x + 1
y /= np.linalg.norm(y)

# Створення даних та міток
data = x.reshape(num_points, 1)
labels = y.reshape(num_points, 1)

# Побудова графіка вхідних даних
plt.figure()
plt.scatter(data, labels)
plt.xlabel('Розмірність 1')
plt.ylabel('Розмірність 2')
plt.title('Вхідні дані')

# Визначення багатошарової нейронної мережі з двома прихованими
# шарами. Перший прихований шар складається із десяти нейронів.
# Другий прихований шар складається з шести нейронів.
# Вихідний шар складається з одного нейрона.
nn = nl.net.newff([[min_val, max_val]], [3, 3, 1])

# Завдання градієнтного спуску як навчального алгоритму
nn.trainf = nl.train.train_gd

# Тренування нейронної мережі
error_progress = nn.train(data, labels, epochs=2000, show=100, goal=0.01)

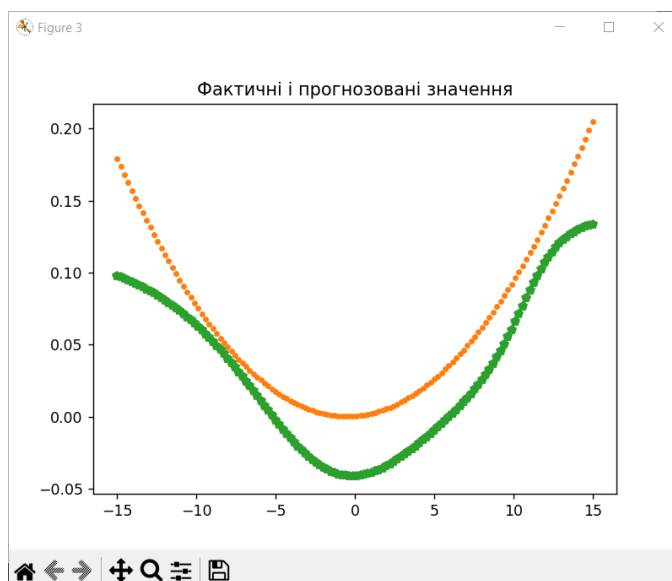
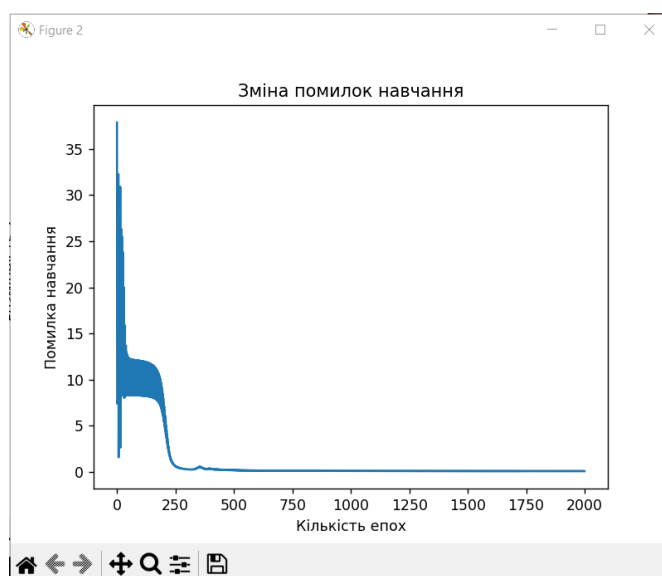
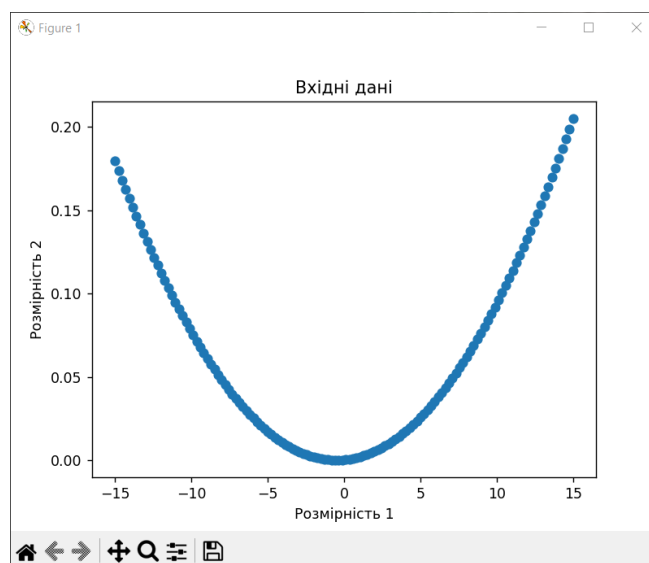
# Виконання нейронної мережі на тренувальних даних
output = nn.sim(data)
y_pred = output.reshape(num_points)

# Побудова графіка помилки навчання
plt.figure()
plt.plot(error_progress)
plt.xlabel('Кількість епох')
plt.ylabel('Помилка навчання')
plt.title('Зміна помилок навчання')

# Побудова графіка результатів
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')
```

		Рябова Є.В.			ДУ«Житомирська політехніка».23.121.23.000 – Лр5	Арк.
		Голенко М.Ю.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

```
plt.title('Фактичні і прогнозовані значення')
plt.show()
```



		Рябова Є.В.			ДУ«Житомирська політехніка».23.121.23.000 – Лр5	Арк.
		Голенко М.Ю.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

```

D:\course-4\semester-1\ai\lab5_project\venv\Scripts\python.exe
Epoch: 100; Error: 8.275639890233528;
Epoch: 200; Error: 5.726036446708101;
Epoch: 300; Error: 0.28572334231457464;
Epoch: 400; Error: 0.33053155384369004;
Epoch: 500; Error: 0.18075920767761794;
Epoch: 600; Error: 0.14123568791580077;
Epoch: 700; Error: 0.12460520836286362;
Epoch: 800; Error: 0.11546879001957047;
Epoch: 900; Error: 0.10951125763667773;
Epoch: 1000; Error: 0.10516780769112824;
Epoch: 1100; Error: 0.10176523962335285;
Epoch: 1200; Error: 0.09897491849409622;
Epoch: 1300; Error: 0.0966177753559106;
Epoch: 1400; Error: 0.0945856798587647;
Epoch: 1500; Error: 0.09280721409364731;
Epoch: 1600; Error: 0.0912318158513458;
Epoch: 1700; Error: 0.08982183857149019;
Epoch: 1800; Error: 0.0885482541451679;
Epoch: 1900; Error: 0.08738811852123521;
Epoch: 2000; Error: 0.08632296168371265;
The maximum number of train epochs is reached

Process finished with exit code 0

```

Рис.5.6.1 – 5.6.4. Результат виконання завдання.

У результаті виконання коду була побудована багатошарова нейронна мережа для апроксимації функції, яка має квадратичну та лінійну залежності від вхідних даних. Помилка навчання показує ефективність тренування, зменшуючись від 8.28 до 0.086 протягом 2000 епох. Графік фактичних та прогнозованих значень відображає відмінну узгодженість мережі з оригінальними даними.

**Завдання №5.7.** Побудова нейронної мережі на основі карти Кохонена, що самоорганізується.

Лістинг файлу LR\_5\_task\_7.py:

```

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

# Генерація даних
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)

# Створення мережі з 2 входами і 4 нейронами
net = nl.net.newc([[0.0, 1.0], [0.0, 1.0]], 4)

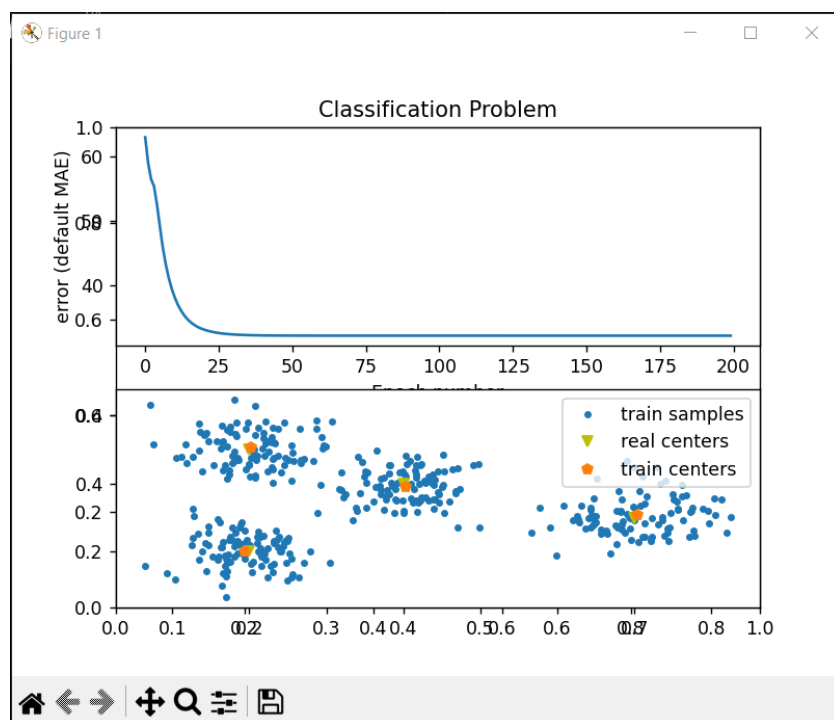
```

		Рябова Є.В.			ДУ«Житомирська політехніка».23.121.23.000 – Лр5	Арк.
		Голенко М.Ю.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

```
# Тренування за алгоритмом «Переможець отримує все» (CWTA) на 200 ітерацій
# та виводення помилки кожних 20 епох
error = net.train(inp, epochs=200, show=20)

# Створення графіків
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()
```



```
D:\course-4\semester-1\ai\lab5_project\venv\Scripts\python.exe
Epoch: 20; Error: 33.3365542487744;
Epoch: 40; Error: 32.27236320913694;
Epoch: 60; Error: 32.22976689388744;
Epoch: 80; Error: 32.22728552576839;
Epoch: 100; Error: 32.22702699416939;
Epoch: 120; Error: 32.226979763816345;
Epoch: 140; Error: 32.226970899321614;
Epoch: 160; Error: 32.22696922203053;
Epoch: 180; Error: 32.22696890510687;
Epoch: 200; Error: 32.22696884555205;
The maximum number of train epochs is reached

Process finished with exit code 0
```

Рис.5.7.1 – 5.7.2. Результат виконання завдання.

		Рябова Є.В.			ДУ«Житомирська політехніка».23.121.23.000 – Лр5	Арк.
		Голенко М.Ю.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

У результаті виконання завдання була побудована нейронна мережа на основі карти Кохонена, яка самоорганізується для класифікації вхідних даних. Помітне зменшення помилки тренування вказує на успішне самоорганізуюче навчання мережі. Графіки відображають розташування центрів нейронів та їхнє наближення до реальних центрів даних під час тренування.

**Завдання №5.8.** Дослідження нейронної мережі на основі карти Кохонена, що самоорганізується.

№ варіанту	Центри кластера	skv
Варіант 23	[0.1, 0.1], [0.2, 0.4], [0.5, 0.3], [0.2, 0.7], [0.6, 0.5]	0,07

Створіть нейронну мережу Кохонена з 2 входами та 4 нейронами

Лістинг файлу LR\_5\_task\_8.py:

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

# Генерація даних
skv = 0.07
centr = np.array([[0.1, 0.1], [0.2, 0.4], [0.5, 0.3], [0.2, 0.7], [0.6, 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)

# Створення мережі з 2 входами і 4 нейронами
net = nl.net.newc([[0.0, 1.0], [0.0, 1.0]], 4)

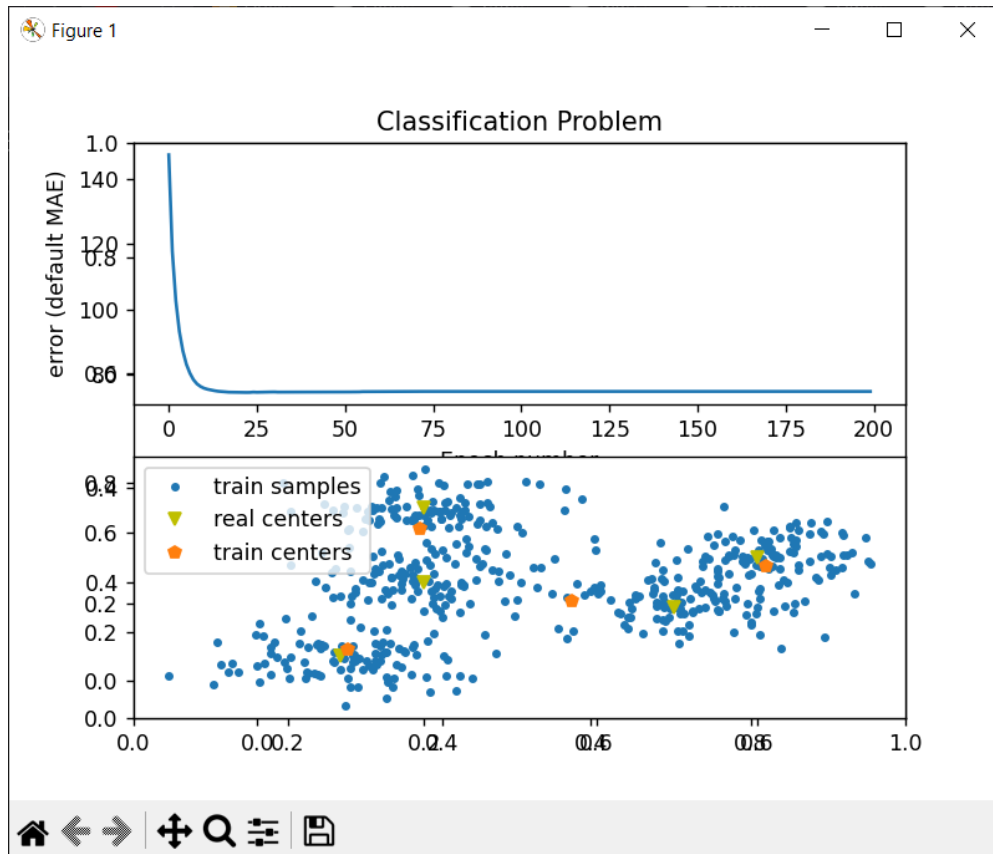
# Тренування за алгоритмом «Переможець отримує все» (CWTA) на 200 ітерацій
# та виводення помилки кожних 20 епох
error = net.train(inp, epochs=200, show=20)

# Створення графіків
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()
```

		Рябова Є.В.			ДУ«Житомирська політехніка».23.121.23.000 – Лр5	Арк.
		Голенко М.Ю.				16
Змн.	Арк.	№ докум.	Підпис	Дата		





```
D:\course-4\semester-1\ai\lab5_project\venv\Scripts\python.exe
Epoch: 20; Error: 74.5189659800019;
Epoch: 40; Error: 74.58638082842512;
Epoch: 60; Error: 74.7265688371713;
Epoch: 80; Error: 74.77418410760926;
Epoch: 100; Error: 74.77471374571869;
Epoch: 120; Error: 74.77472088360042;
Epoch: 140; Error: 74.77472097884959;
Epoch: 160; Error: 74.77472097964878;
Epoch: 180; Error: 74.77472097963641;
Epoch: 200; Error: 74.77472097963543;
The maximum number of train epochs is reached

Process finished with exit code 0
```

Рис.5.8.1 – 5.8.2. Результат виконання завдання.

Створіть нейронну мережу Кохонена з 2 входами та 5 нейронами

Лістинг файлу LR\_5\_task\_8.py:

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

# Генерація даних
skv = 0.07
```

		Рябова Є.В.			ДУ«Житомирська політехніка».23.121.23.000 – Лр5	Арк.
		Голенко М.Ю.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

```

centr = np.array([[0.1, 0.1], [0.2, 0.4], [0.5, 0.3], [0.2, 0.7], [0.6, 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)

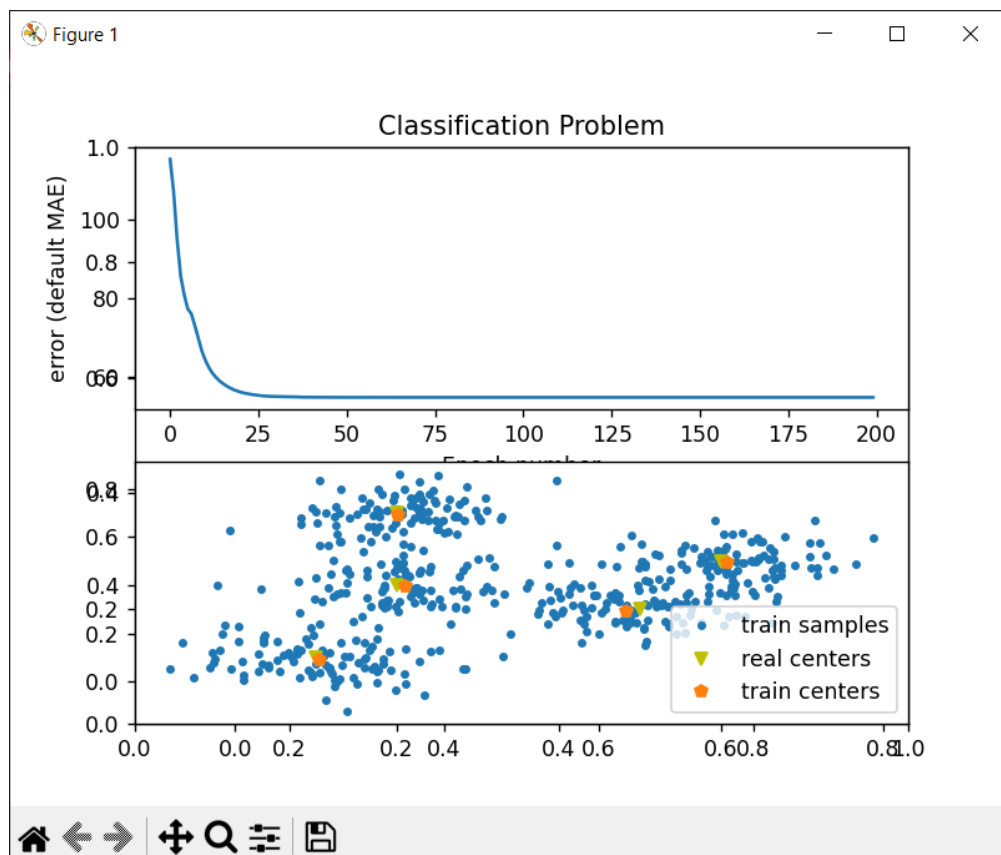
# Створення мережі з 2 входами і 5 нейронами
net = nl.net.newcc([[0.0, 1.0], [0.0, 1.0]], 5)

# Тренування за алгоритмом «Переможець отримує все» (CWTA) на 200 ітерацій
# та виводення помилки кожних 20 епох
error = net.train(inp, epochs=200, show=20)

# Створення графіків
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()

```



		Рябова Є.В.			ДУ«Житомирська політехніка».23.121.23.000 – Лр5	Арк.
		Голенко М.Ю.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

```

D:\course-4\semester-1\ai\lab5_project\venv\Scripts\python.exe
Epoch: 20; Error: 56.35796983116502;
Epoch: 40; Error: 54.77648725792997;
Epoch: 60; Error: 54.736554873402994;
Epoch: 80; Error: 54.73435533610572;
Epoch: 100; Error: 54.73419854148791;
Epoch: 120; Error: 54.73418630688211;
Epoch: 140; Error: 54.73418531943968;
Epoch: 160; Error: 54.734185237569186;
Epoch: 180; Error: 54.734185230641046;
Epoch: 200; Error: 54.73418523004591;
The maximum number of train epochs is reached

Process finished with exit code 0

```

Рис.5.8.3 – 5.8.4. Результат виконання завдання.

У результаті виконання завдання можна зробити висновок, що мережа Кохонена з 5 нейронами дала меншу помилку, ніж з 4 нейронами. Такий результат можна обґрунтувати тим, що у завданні було дано саме 5 центрів кластера. Тому правильний вибір кількості нейронів грає велику роль у зменшенні помилки.

Якщо порівняти результати 7 і 8 завдань, можна сказати, що збільшення  $skv$  призводить до збільшення помилки. У попередньому завданні центри кластерів були визначені точніше, ніж у цьому.

**Висновки:** в ході виконання лабораторної роботи було створено та застосовано прості нейронні мережі, використовуючи спеціалізовані бібліотеки та мову програмування Python.

		Рябова Є.В.			ДУ«Житомирська політехніка».23.121.23.000 – Лр5	Арк.
		Голенко М.Ю.				19
Змн.	Арк.	№ докум.	Підпис	Дата		