

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 №1: Створити простий нейрон.

Лістинг програми:

```
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))

"D:\it\Forth Course\AI\venv\Scripts\python.exe" "D:\it\Forth Course\AI\Lab5\LR_5_task_1.p
0.9990889488055994
```

Рис.1 – Результат виконання програми.

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

Лістинг програми:

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

					ДУ «Житомирська політехніка».23.121.9.000 – Лр5			
Змн.	Арк.	№ докум.	Підпис	Дата	Звіт з лабораторної роботи	Літ.	Арк.	Аркушів
Розроб.		Кормиш Р.І						
Перевір.		Іванов Д.А					1	15
Керівник						ФІКТ Гр.ІПЗ-20-4		
Н. контр.								
Зав. каф.								

```

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

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

class OleksiichukNeuralNetwork:
    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 = 100

        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)

                # Neuron o1
                d_ypred_d_w5 = h1 * derivative_sigmoid(sum_o1)
                d_ypred_d_w6 = h2 * derivative_sigmoid(sum_o1)
                d_ypred_d_b3 = derivative_sigmoid(sum_o1)

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

                # Neuron h1
                d_h1_d_w1 = x[0] * derivative_sigmoid(sum_h1)
                d_h1_d_w2 = x[1] * derivative_sigmoid(sum_h1)
                d_h1_d_b1 = derivative_sigmoid(sum_h1)

                # Neuron h2
                d_h2_d_w3 = x[0] * derivative_sigmoid(sum_h2)
                d_h2_d_w4 = x[1] * derivative_sigmoid(sum_h2)

```

		Кормиш Р.І			ДУ «Житомирська політехніка».23.121.9.000 – Лр5	Арк.
		Іванов Д.А.				2
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        d_h2_d_b2 = derivative_sigmoid(sum_h2)

        # Update weights and biases
        # Neuron 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

        # Neuron 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

        # Neuron 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

    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))

if __name__ == "__main__":
    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
    ])

    network = OleksiichukNeuralNetwork()
    network.train(data, all_y_trues)

    emily = np.array([-7, -3]) # 128 pounds, 63 inches
    frank = np.array([20, 2]) # 155 pounds, 68 inches
    print("Emily: %.3f" % network.feedforward(emily)) # +-0.96 - F
    print("Frank: %.3f" % network.feedforward(frank)) # +-0.039 - M

```

		Кормиш Р.І			ДУ «Житомирська політехніка».23.121.9.000 – Лр5	Арк.
		Іванов Д.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		3

```
"D:\it\Forth Course\AI\venv\Scripts\python.exe" "D:\it\Forth Course\AI\Lab5\LR_5_task_2.py"
0.9990889488055994
Epoch 0 loss: 0.147
Epoch 10 loss: 0.072
Epoch 20 loss: 0.058
Epoch 30 loss: 0.048
Epoch 40 loss: 0.040
Epoch 50 loss: 0.034
Epoch 60 loss: 0.029
Epoch 70 loss: 0.025
Epoch 80 loss: 0.022
Epoch 90 loss: 0.020
Emily: 0.893
Frank: 0.132
```

Рис.2 – Результат виконання програми.

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

Лістинг програми:

```
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl
# Завантаження вхідних даних
text = np.loadtxt('data_perceptron.txt')
# Поділ даних на точки даних та мітки
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]
# 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')
```

		Кормиш Р.І			ДУ «Житомирська політехніка».23.121.9.000 – Лр5	Арк.
		Іванов Д.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

```
plt.title('Прогрес помилки навчання')
plt.grid()
plt.show()
```

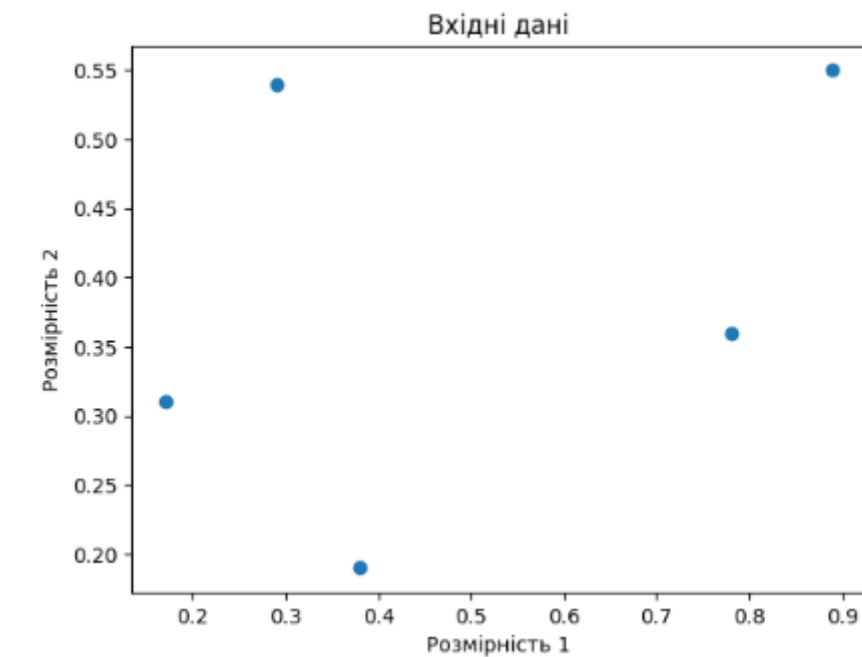


Рис.3 – Результат виконання програми.

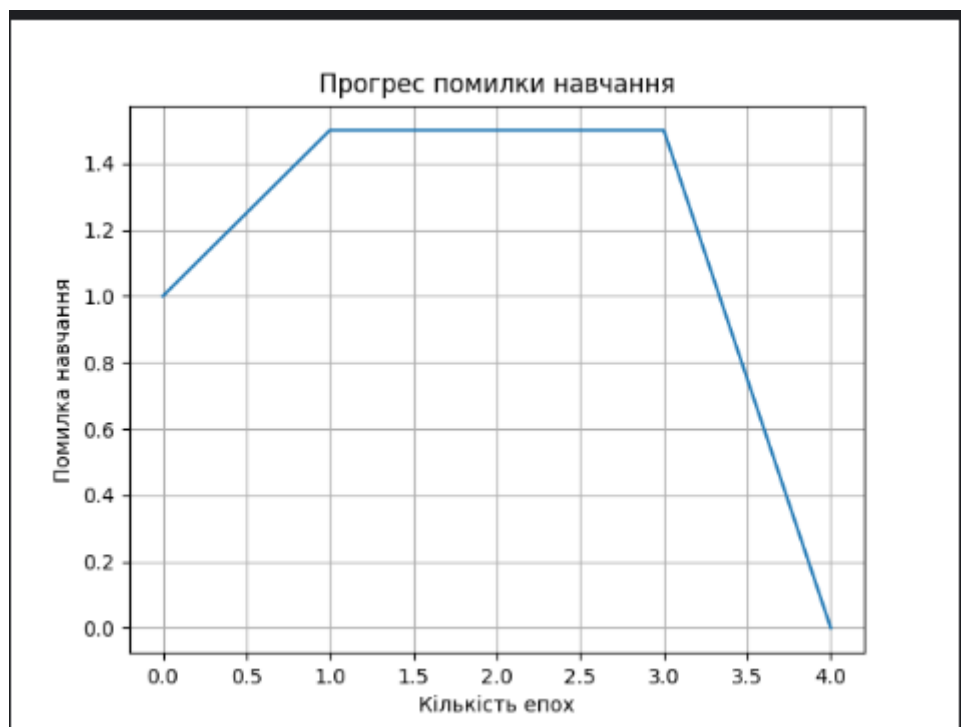


Рис.4 – Результат виконання програми.

Завдання №4: Побудова одношарової нейронної мережі.

Лістинг програми:

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

		Кормиш Р.І			ДУ «Житомирська політехніка».23.121.9.000 – Лр5	Арк.
		Іванов Д.А.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

```

# 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)
# Навчання нейронної мережі
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')
plt.title('Прогрес помилки навчання')
plt.grid()
plt.show()
# Виконання класифікатора на тестових точках даних
print('\nРезультати тесту:')
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])

```

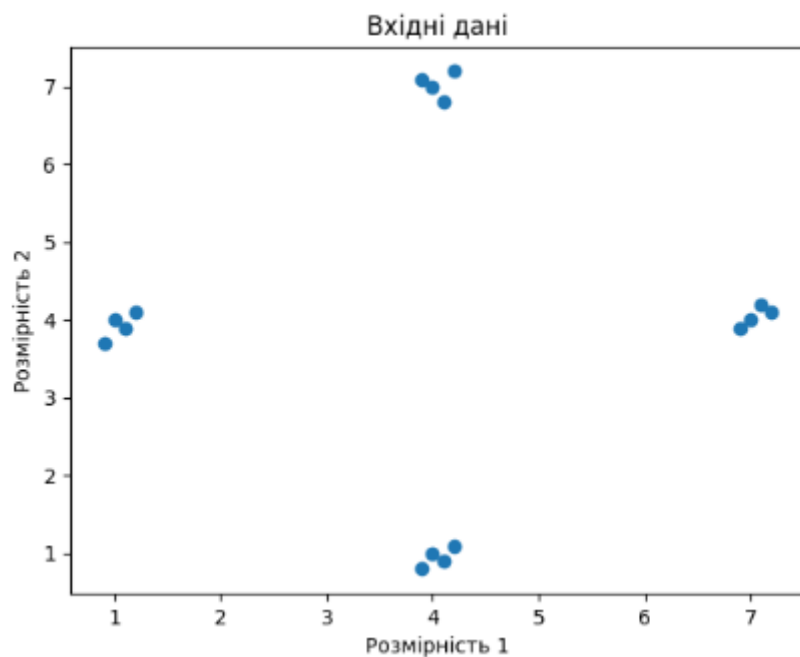


Рис.5 – Результат виконання програми.

		Кормиш Р.І			ДУ «Житомирська політехніка».23.121.9.000 – Лр5	Арк.
		Іванов Д.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

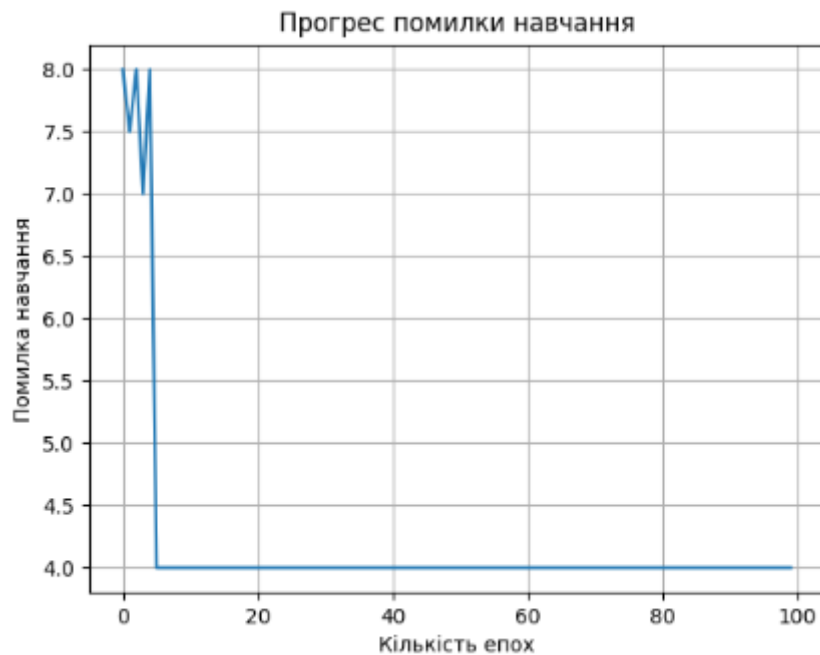


Рис.6 – Результат виконання програми.

```
"D:\it\Forth Course\AI\venv\Scripts\python.exe" "D:\it
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

Результати тесту:
[0.4, 4.3] --> [0. 0.]
[4.4, 0.6] --> [1. 0.]
[4.7, 8.1] --> [1. 1.]
```

Рис.7 – Результат виконання програми.

Завдання №5: Побудова багатошарової нейронної мережі.

Лістинг програми:

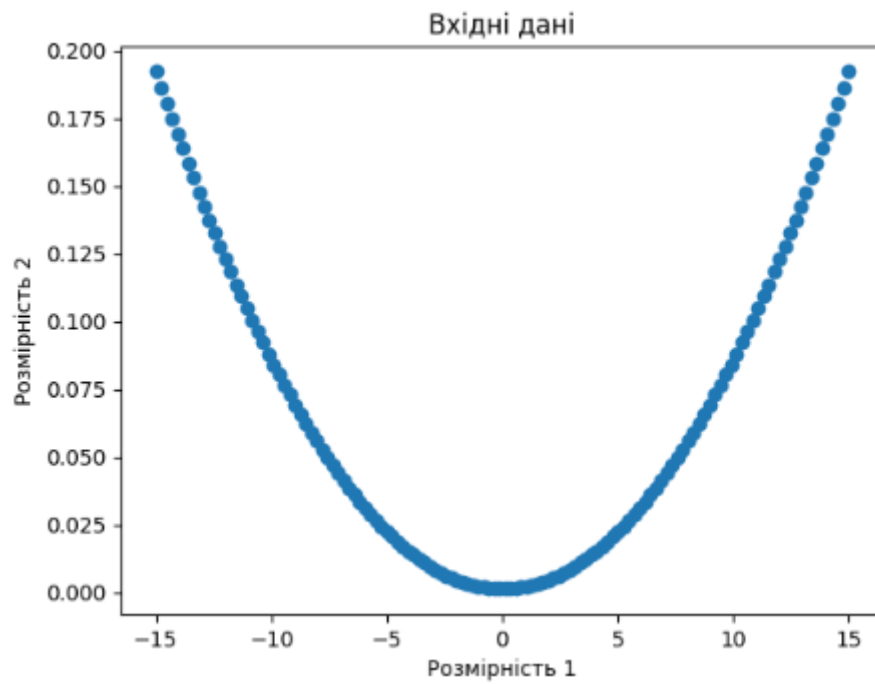


Рис.8 – Результат виконання програми.

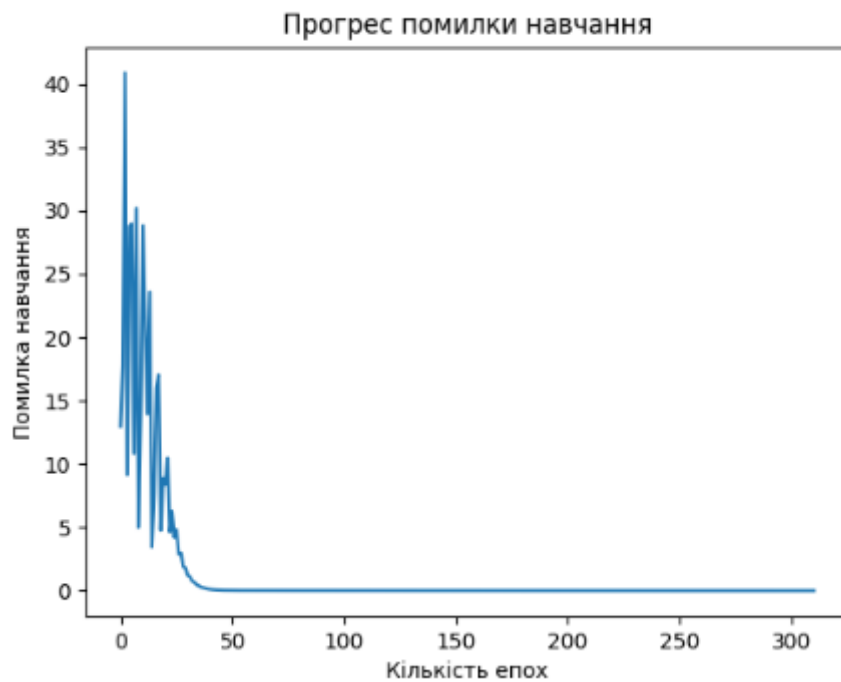


Рис.9 – Результат виконання програми.

		Кормиш Р.І			ДУ «Житомирська політехніка».23.121.9.000 – Лр5	Арк.
		Іванов Д.А.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

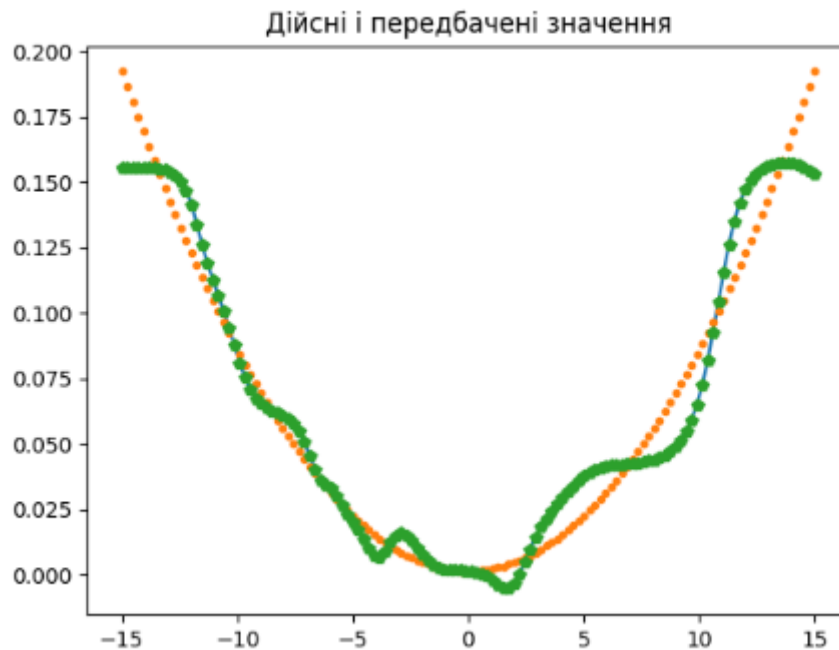


Рис.10 – Результат виконання програми.

```
"D:\it\Forth Course\AI\venv\Scripts\python.exe" "D:\it\Forth Course\AI\venv\Scripts\python.exe"
Epoch: 100; Error: 0.020317802849684007;
Epoch: 200; Error: 0.013326176013376757;
Epoch: 300; Error: 0.010239766434290594;
The goal of learning is reached
```

Рис.11 – Результат виконання програми.

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

Табл. 1

№ варіанта	Тестові дані
Варіант 9	$y = 3x^2 + 9$

Табл. 2

Номер варіанта	Багатошаровий перцептрон	
	Кількість шарів	Кількості нейронів у шарах
9	3	3-5-1

Лістинг програми:

```
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl
# Генерація тренувальних даних
```

		Кормиш Р.І			ДУ «Житомирська політехніка».23.121.9.000 – Лр5	Арк.
		Іванов Д.А.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

```

min_val = -15
max_val = 15
num_points = 130
x = np.linspace(min_val, max_val, num_points)
y = 3 * np.square(x) + 9
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 багатoshарової нейронної мережі з трьома прихованими
# шарами. Перший прихований шар складається із трьох нейронів.
# Другий прихований шар складається з чотирьох нейронів.
# Третій прихований шар складається з одного нейрона.
# Вихідний шар складається з одного нейрона
nn = nl.net.newff([[min_val, max_val]], [3, 3, 5, 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('Прогрес помилки 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('Дійсні і передбачені значення')
plt.show()

```

		Кормиш Р.І			ДУ «Житомирська політехніка».23.121.9.000 – Лр5	Арк.
		Іванов Д.А.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

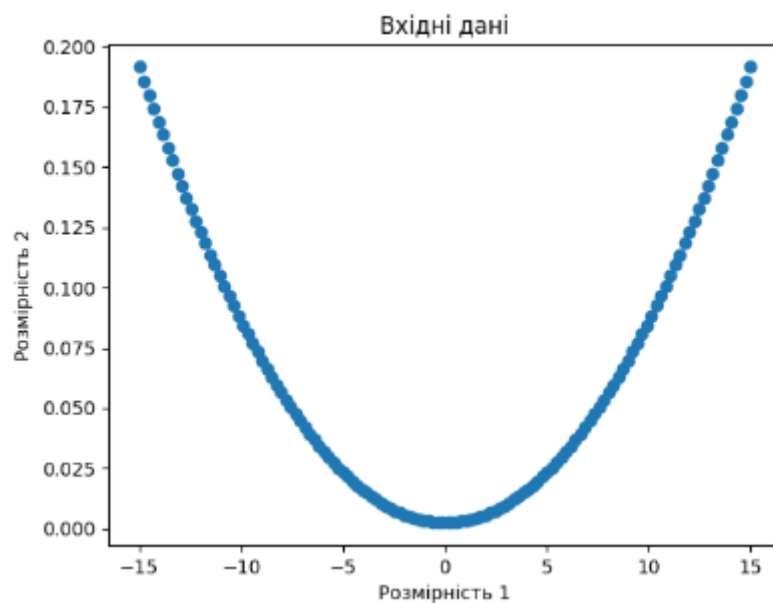


Рис.12 – Результат виконання програми.

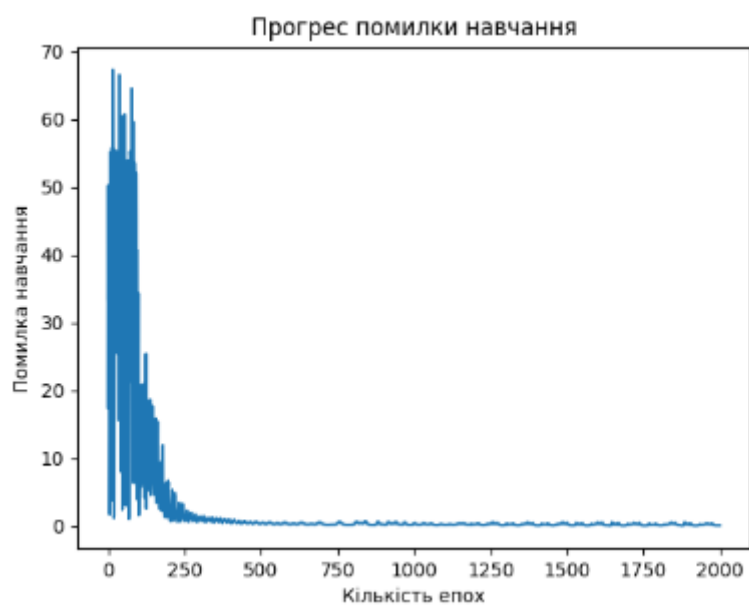


Рис.13 – Результат виконання програми.

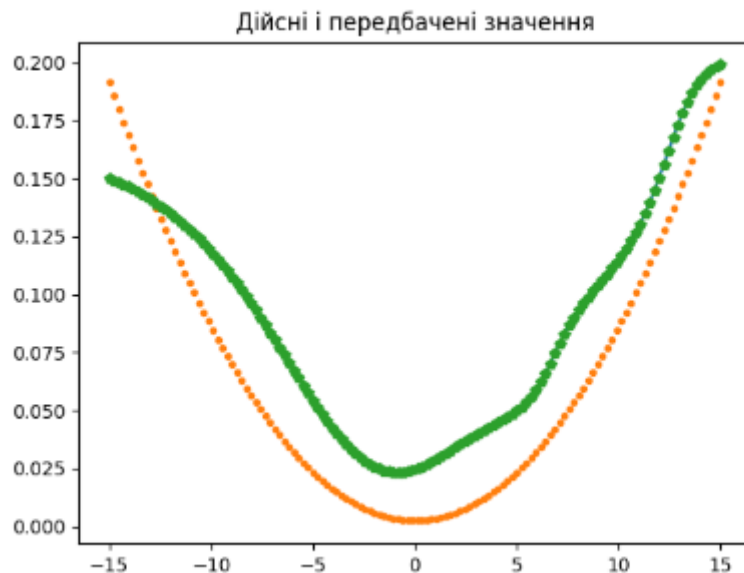


Рис.14 – Результат виконання програми.

```
"D:\it\Forth Course\AI\venv\Scripts\python.exe" "D:\it\Forth Course\AI\Lab5\LR_5_task_6.py"
Epoch: 100; Error: 17.875286179353502;
Epoch: 200; Error: 3.1644282150670264;
Epoch: 300; Error: 1.2547842061580714;
Epoch: 400; Error: 0.6996244984409767;
Epoch: 500; Error: 0.28835967396551737;
Epoch: 600; Error: 0.22116595689900898;
Epoch: 700; Error: 0.3169229864805193;
Epoch: 800; Error: 0.12074045857615799;
Epoch: 900; Error: 0.19358333174087416;
Epoch: 1000; Error: 0.2436753497838561;
Epoch: 1100; Error: 0.12017639652335393;
Epoch: 1200; Error: 0.18293019913758915;
Epoch: 1300; Error: 0.14656785608470596;
Epoch: 1400; Error: 0.05768710048152943;
Epoch: 1500; Error: 0.30126557607757776;
Epoch: 1600; Error: 0.3825120969711301;
Epoch: 1700; Error: 0.1906487000748826;
Epoch: 1800; Error: 0.07391144145555745;
Epoch: 1900; Error: 0.12441559351836393;
Epoch: 2000; Error: 0.05220414629697763;
The maximum number of train epochs is reached
```

Рис.15 – Результат виконання програми.

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

Лістинг програми:

		Кормиш Р.І			ДУ «Житомирська політехніка».23.121.9.000 – Лр5	Арк.
		Іванов Д.А.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

```

import numpy as np
import neurolab as nl
import numpy.random as rand
import pylab as pl
skv = 0.05
center = 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([center + 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=20)
# Plot results:
fig, axs = pl.subplots(2)
fig.suptitle('Classification Problem')
axs.flat[1].set(xlabel='Epoch number', ylabel='error (default MAE)')
axs[1].plot(error)
w = net.layers[0].np['w']
axs[0].plot(inp[:, 0], inp[:, 1], '.', center[:, 0], center[:, 1], 'yv', w[:, 0],
w[:, 1], 'p')
axs[0].legend(['train samples', 'real centers', 'train centers'], loc='upper
left')
pl.show()

```

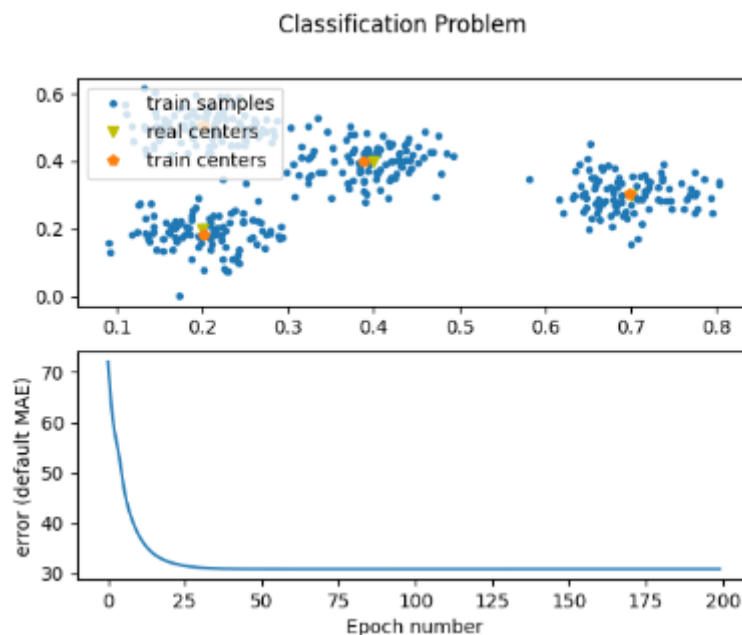


Рис.16 – Результат виконання програми.

		Кормиш Р.І			ДУ «Житомирська політехніка».23.121.9.000 – Лр5	Арк.
		Іванов Д.А.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

```
"D:\it\Forth Course\AI\venv\Scripts\python.exe" "D:\it\Forth Course\AI\Lab5\LR_5_task_7.py"
Epoch: 20; Error: 32.51350127918001;
Epoch: 40; Error: 30.95851912236332;
Epoch: 60; Error: 30.91905596158979;
Epoch: 80; Error: 30.92175734116705;
Epoch: 100; Error: 30.922767887173478;
Epoch: 120; Error: 30.922972903851022;
Epoch: 140; Error: 30.92301391038535;
Epoch: 160; Error: 30.923022032718876;
Epoch: 180; Error: 30.92302363065115;
Epoch: 200; Error: 30.923023943534112;
The maximum number of train epochs is reached
```

Рис.17 – Результат виконання програми.

Завдання №8: Дослідження нейронної мережі на основі карти Кохонена, що самоорганізується.

Табл. 3

№ варіанту	Центри кластера	skv
Варіант 9	[0.2, 0.3], [0.4, 0.4], [0.7, 0.3], [0.1, 0.5], [0.4, 0.5]	0,04

Лістинг програми:

```
import numpy as np
import neurolab as nl
import numpy.random as rand
import pylab as pl
skv = 0.04
center = 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([center + 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:
fig, axs = pl.subplots(2)
fig.suptitle('Classification Problem')
axs.flat[1].set(xlabel='Epoch number', ylabel='error (default MAE)')
axs[1].plot(error)
w = net.layers[0].np['w']
axs[0].plot(inp[:, 0], inp[:, 1], '.', center[:, 0], center[:, 1], 'yv', w[:, 0],
w[:, 1], 'p')
axs[0].legend(['train samples', 'real centers', 'train centers'], loc='upper left')
pl.show()
```

		Кормиш Р.І			ДУ «Житомирська політехніка».23.121.9.000 – Лр5	Арк.
		Іванов Д.А.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

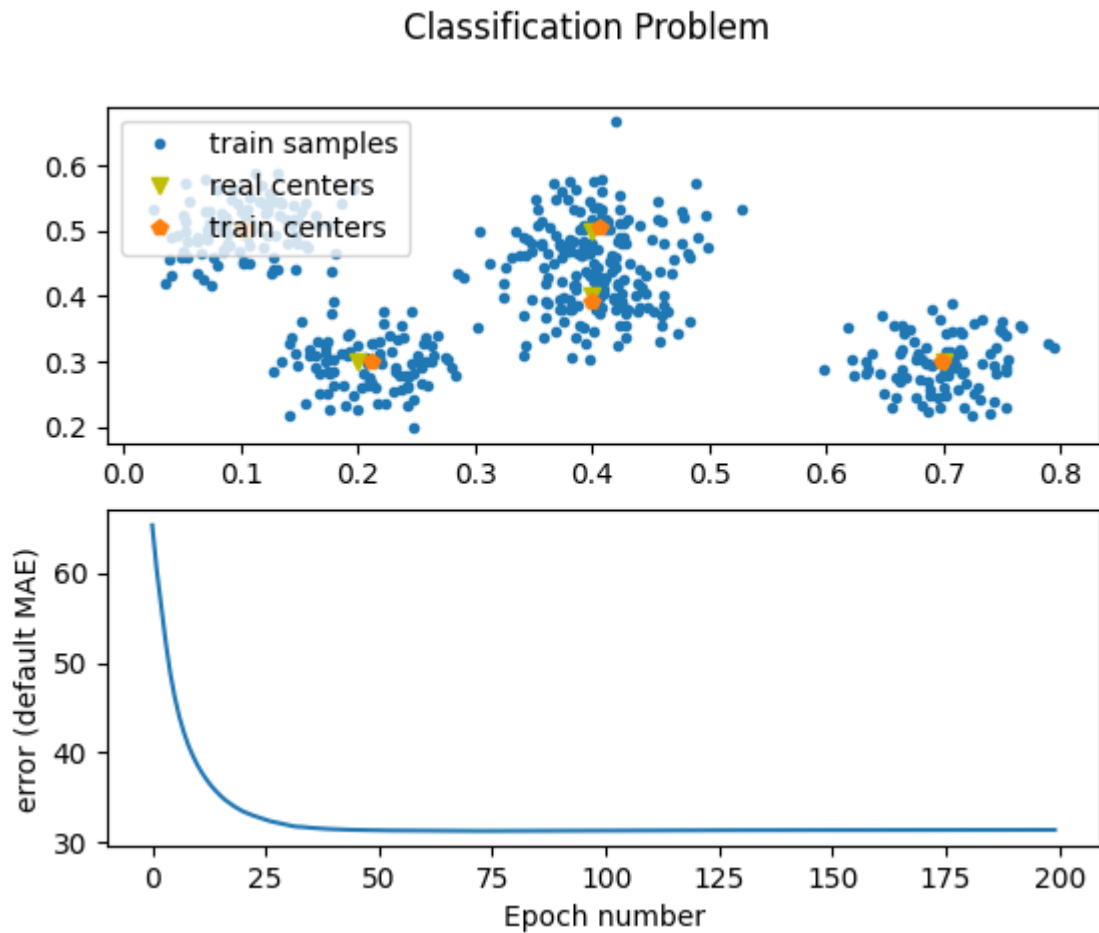


Рис.18 – Результат виконання програми.

```
"D:\it\Forth Course\AI\venv\Scripts\python.exe" "D:\it\Forth Course\AI\Lab5\LR_5_task_8.py"
Epoch: 20; Error: 33.768766959782916;
Epoch: 40; Error: 31.50912178820382;
Epoch: 60; Error: 31.30696882309654;
Epoch: 80; Error: 31.30372256945291;
Epoch: 100; Error: 31.32865822534325;
Epoch: 120; Error: 31.37134480608685;
Epoch: 140; Error: 31.38550870039879;
Epoch: 160; Error: 31.40497194404459;
Epoch: 180; Error: 31.40971749808024;
Epoch: 200; Error: 31.41082903894852;
The maximum number of train epochs is reached
```

Рис.19 – Результат виконання програми.

Висновки: були досліджені прості нейронні мережі, використовуючи спеціалізовані бібліотеки та мову програмування Python. Створено одношарову та багатошарову нейронні мережі.

		Кормиш Р.І			ДУ «Житомирська політехніка».23.121.9.000 – Лр5	Арк.
		Іванов Д.А.				15
Змн.	Арк.	№ докум.	Підпис	Дата		