

## ЛАБОРАТОРНА РОБОТА № 5

### РОЗРОБКА ПРОСТИХ НЕЙРОННИХ МЕРЕЖ

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

#### Хід роботи

**Посилання на GitHub:** [https://github.com/Max2002/AI\\_IPZ-19-3\\_LMV](https://github.com/Max2002/AI_IPZ-19-3_LMV)

**Завдання 1.** Створити простий нейрон

Лістинг коду файлу LR\_5\_Task\_1.py:

```
import numpy as np

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

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

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

if __name__ == "__main__":
    weights = np.array([0, 1])
    bias = 4
    n = Neuron(weights, bias)

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

```
C:\Users\Max\AppData\Local\Programs\Python\Python311\python.exe "E:\Лабораторні\4 курс\Системи штучного інтелекту\Lab - 5\LR_5_Task_1.py"
0.9990889488055994
```

```
Process finished with exit code 0
```

Рис.1. Результат роботи нейрона

					Державний університет «Житомирська політехніка». 22.121.09.000 – Лр5		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Ляшук М.В.			Звіт з лабораторної роботи	Літ.	Арк.
Перевір.		Філіпов В.О					1
Керівник							15
Н. контр.						ФІКТ Гр. ІПЗ-19-3	
Зав. каф.							

## Завдання 2. Створити просту нейронну мережу для передбачення статі людини

Лістинг коду файлу LR\_5\_Task\_2.py:

```
import numpy as np
from LR_5_Task_1 import Neuron, sigmoid

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 = 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)

                # 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)
```

		Ляшук М.В.			Державний університет «Житомирська політехніка». 22.121.09.000 – Лр5	Арк.
		Філіпов В.О.				2
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        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)
        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

```

```

Epoch 750 loss: 0.003
Epoch 760 loss: 0.003
Epoch 770 loss: 0.003
Epoch 780 loss: 0.003
Epoch 790 loss: 0.003
Epoch 800 loss: 0.003
Epoch 810 loss: 0.003
Epoch 820 loss: 0.003
Epoch 830 loss: 0.003
Epoch 840 loss: 0.003
Epoch 850 loss: 0.003
Epoch 860 loss: 0.003
Epoch 870 loss: 0.003
Epoch 880 loss: 0.003
Epoch 890 loss: 0.003
Epoch 900 loss: 0.003
Epoch 910 loss: 0.003
Epoch 920 loss: 0.003
Epoch 930 loss: 0.003
Epoch 940 loss: 0.003
Epoch 950 loss: 0.003
Epoch 960 loss: 0.002
Epoch 970 loss: 0.002
Epoch 980 loss: 0.002
Epoch 990 loss: 0.002
Emily: 0.963
Frank: 0.056

Process finished with exit code 0

```

Рис.2. Результат навчання нейронної мережі

Функція активації використовується для підключення незв'язаних вхідних даних із виходом з простою та передбачуваною формою.

Нейронні мережі прямого поширення дозволяють, використовуючи функції активації, передбачати відповідь (класифікувати).

**Завдання 3.** Класифікатор на основі перцептрону з використанням бібліотеки NeuroLab

Лістинг коду файлу LR\_5\_Task\_3.py:

		Ляшук М.В.			Державний університет «Житомирська політехніка». 22.121.09.000 – Лр5	Арк.
		Філіпов В.О.				4
Змн.	Арк.	№ докум.	Підпис	Дата		

```

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('Dimension 1')
plt.ylabel('Dimension 2')
plt.title('Input data')
plt.show()

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('Number of epochs')
plt.ylabel('Training error')
plt.title('Training error progress')
plt.grid()
plt.show()

```

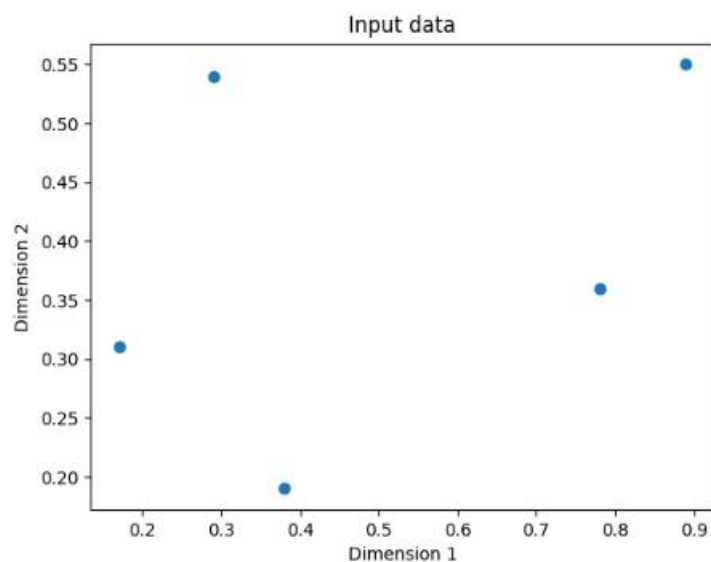


Рис.3. Вхідні дані до перцептрону

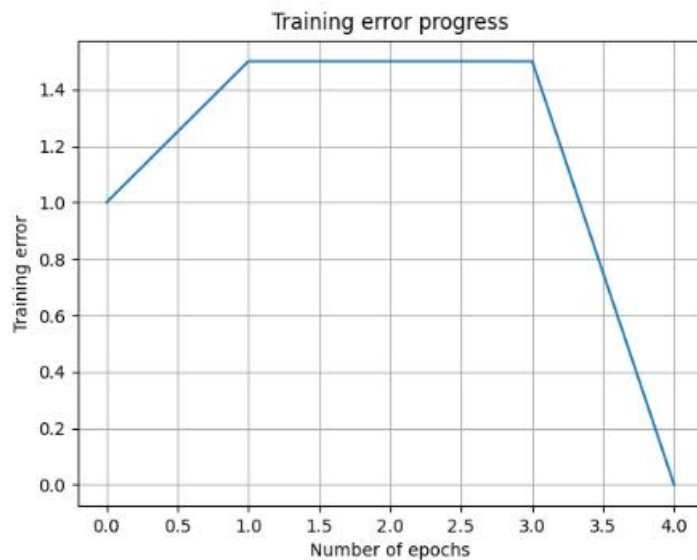


Рис.4. Навчання перцептрону

#### Завдання 4. Побудова одношарової нейронної мережі

Лістинг коду файлу 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('Dimension 1')
plt.ylabel('Dimension 2')
plt.title('Input data')
plt.show()

dim1 = [data[:, 0].min(), data[:, 0].max()]
dim2 = [data[:, 1].min(), data[:, 1].max()]
num_output = labels.shape[1]

nn = nl.net.newff([dim1, dim2], [3, num_output])
error_progress = nn.train(data, labels, epochs=1000, show=100, goal=0.02)

plt.figure()
plt.plot(error_progress)
plt.xlabel('Number of epochs')
plt.ylabel('Training error')
plt.title('Training error progress')
plt.grid()
plt.show()

print('Test 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])
```

		Ляшук М.В.			Державний університет «Житомирська політехніка». 22.121.09.000 – Лр5	Арк.
		Філіпов В.О.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

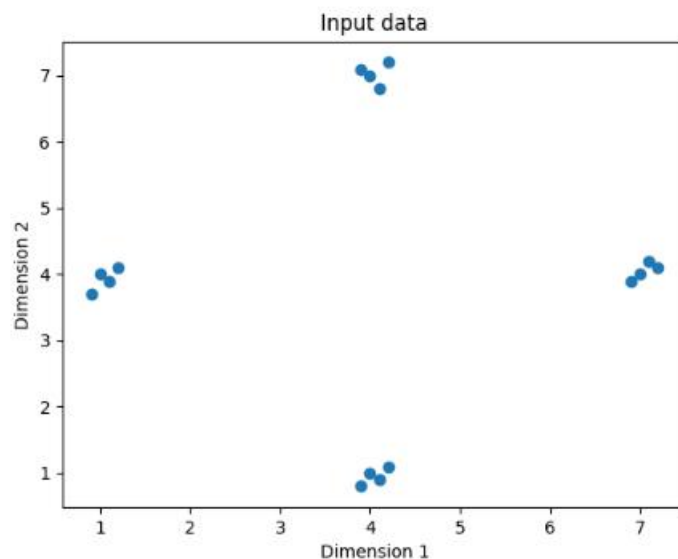


Рис.5. Вхідні дані до нейронної мережі

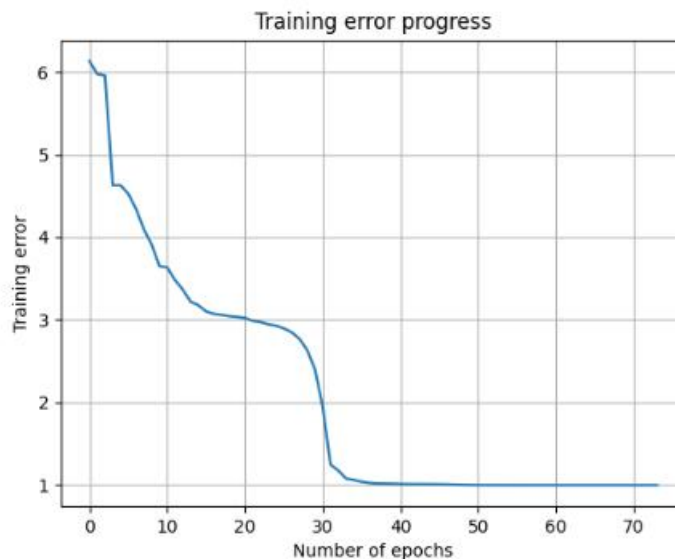


Рис.6. Навчання мережі

```
C:\Users\Max\AppData\Local\Programs\Python\Python311\python.exe "E:\Лабораторні\4 курс\Системи штучного інтелекту\Lab - 5\LR_5_Task_4.py"
Test results:
[0.4, 4.3] --> [-5.83029234e-06 -1.93835747e-05]
[4.4, 0.6] --> [ 1.00000000e+00 -2.13799583e-07]
[4.7, 8.1] --> [0.50000004 1.          ]
Process finished with exit code 0
```

Рис.7.Тестові результати

## Завдання 5. Побудова багатошарової нейронної мережі

Лістинг коду файлу LR\_5\_Task\_5.py:

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

		Ляшук М.В.			Державний університет «Житомирська політехніка». 22.121.09.000 – Лр5	Арк.
		Філіпов В.О				7
Змн.	Арк.	№ докум.	Підпис	Дата		

```

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('Dimension 1')
plt.ylabel('Dimension 2')
plt.title('Data-points')
plt.show()

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('Number of epochs')
plt.ylabel('Error')
plt.title('Training error progress')
plt.grid()
plt.show()

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('Actual vs predicted')
plt.show()

```

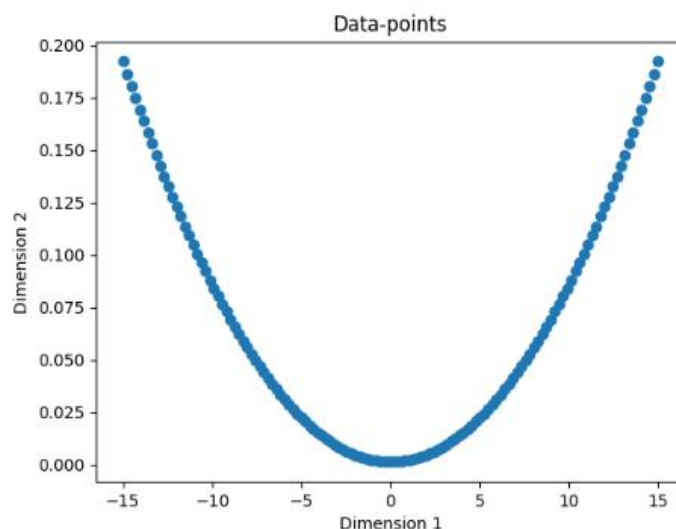


Рис.8. Дані рівняння  $3x^2+5$

		Ляшук М.В.			Державний університет «Житомирська політехніка». 22.121.09.000 – Лр5	Арк.
		Філіпов В.О.				8
Змн.	Арк.	№ докум.	Підпис	Дата		



```

C:\Users\Max\AppData\Local\Programs\Python\Python311\python.exe "E:\Лабораторні\4 курс\Системи штучного інтелекту\Lab - 5\LR_5_Task_5.py"
Epoch: 100; Error: 0.028043501184368717;
Epoch: 200; Error: 0.01223050276868899;
The goal of learning is reached

Process finished with exit code 0

```

Рис.9. Звітність про навчання по епохам

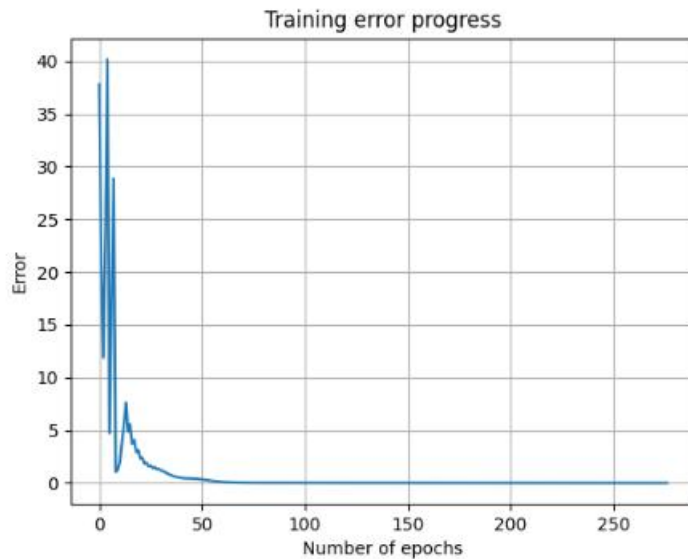


Рис.10. Графік навчання мережі

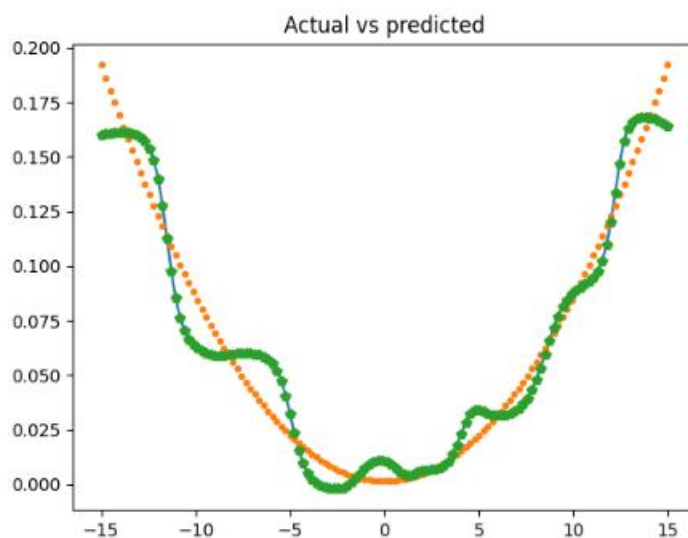


Рис.11. Графік-порівняння істинних та отриманих даних

**Завдання 6.** Побудова багат шарової нейронної мережі для свого варіанту

Варіант 9, дані:  $y = 3x^2 + 9$ , кількість шарів: 3, кількість нейронів: 3-5-1

Лістинг коду файлу LR\_5\_Task\_6.py:

		Ляшук М.В.			Державний університет «Житомирська політехніка». 22.121.09.000 – Лр5	Арк.
		Філіпов В.О				9
Змн.	Арк.	№ докум.	Підпис	Дата		

```

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) + 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('Dimension 1')
plt.ylabel('Dimension 2')
plt.title('Data-points')
plt.show()

nn = nl.net.newff([[min_val, max_val]], [3, 5, 1])
nn.trainf = nl.train.train_gd
error_progress = nn.train(data, labels, epochs=3000, show=100, goal=0.01)

output = nn.sim(data)
y_pred = output.reshape(num_points)

plt.figure()
plt.plot(error_progress)
plt.xlabel('Number of epochs')
plt.ylabel('Error')
plt.title('Training error progress')
plt.grid()
plt.show()

x_dense = np.linspace(min_val, max_val, num_points * 3)
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('Actual vs predicted')
plt.show()

```

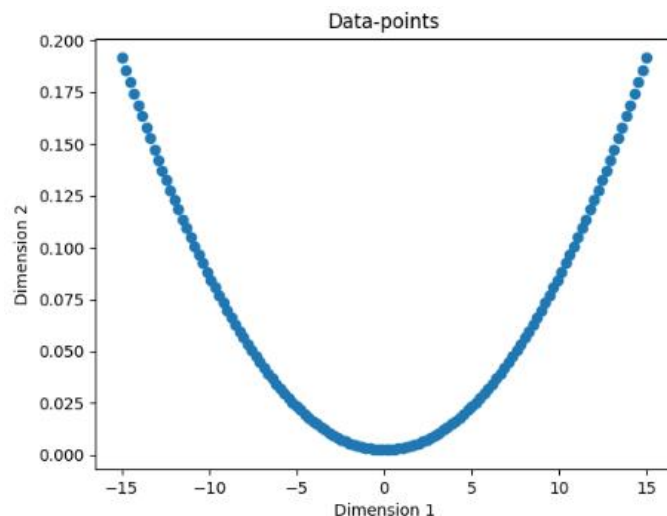


Рис.12. Графік вхідних даних по варіанту

		Ляшук М.В.			Державний університет «Житомирська політехніка». 22.121.09.000 – Лр5	Арк.
		Філіпов В.О.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

```

Epoch: 500; Error: 0.25013912522605386;
Epoch: 600; Error: 0.25532256930752234;
Epoch: 700; Error: 0.33611222315651257;
Epoch: 800; Error: 0.3797589437899146;
Epoch: 900; Error: 0.44973044473799256;
Epoch: 1000; Error: 0.2769297413020857;
Epoch: 1100; Error: 0.3679258244837218;
Epoch: 1200; Error: 0.3076299202657165;
Epoch: 1300; Error: 0.3555539148835636;
Epoch: 1400; Error: 0.266245143094972;
Epoch: 1500; Error: 0.2791360807063537;
Epoch: 1600; Error: 0.289720415350038;
Epoch: 1700; Error: 0.32297101976291254;
Epoch: 1800; Error: 0.363873660285242;
Epoch: 1900; Error: 0.29600371485394494;
Epoch: 2000; Error: 0.27580137278269323;
Epoch: 2100; Error: 0.3419335717604159;
Epoch: 2200; Error: 0.32170597168214365;
Epoch: 2300; Error: 0.2677525019721497;
Epoch: 2400; Error: 0.3257315835957031;
Epoch: 2500; Error: 0.3184419434144783;
Epoch: 2600; Error: 0.26249611799128414;
Epoch: 2700; Error: 0.32919826708318517;
Epoch: 2800; Error: 0.29942828771483476;
Epoch: 2900; Error: 0.2606007469854883;
Epoch: 3000; Error: 0.33611379373578193;
The maximum number of train epochs is reached

Process finished with exit code 0

```

Рис.13. Звітність навчання по епохам (вивід 100х епох)

		Ляшук М.В.			Державний університет «Житомирська політехніка». 22.121.09.000 – Лр5	Арк.
		Філіпов В.О				11
Змн.	Арк.	№ докум.	Підпис	Дата		

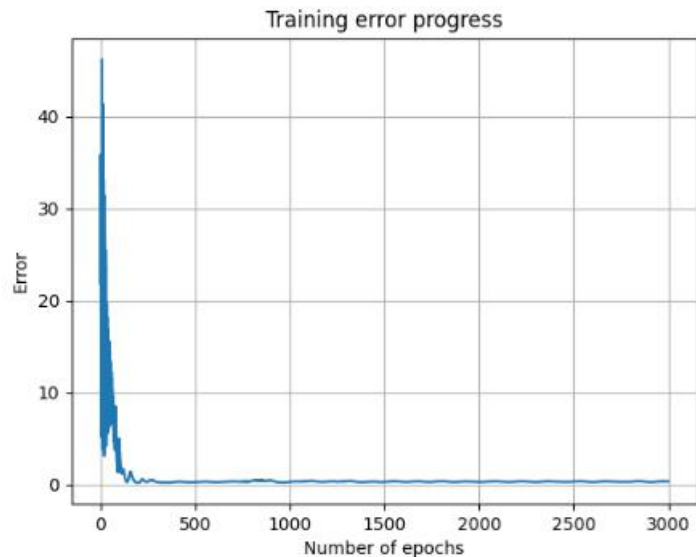


Рис.14. Прогрес помилковості при навчанні

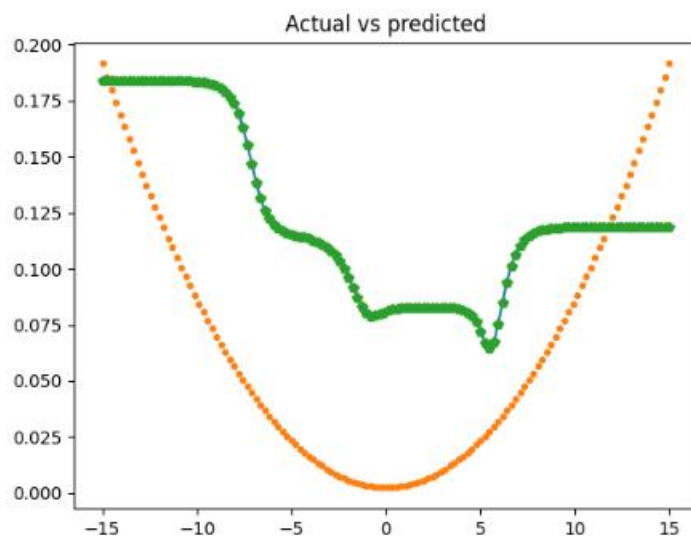


Рис.15. Графік-порівняння дійсних та передбачених даних

У результаті навчання точність нейронної мережі є досить низькою, що може бути пов'язано з кількістю шарів або нейронів у шарах.

**Завдання 7.** Побудова нейронної мережі на основі карти Кохонена, що самоорганізується

Лістинг коду файлу LR\_5\_Task\_7.py:

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

skv = .05
```

```

center = np.array([[.2, .2], [.4, .4], [.7, .3], [.2, .5]])
random_norm = skv * rand.randn(100, 4, 2)
inp = np.array([center + r for r in random_norm])
inp = inp.reshape(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=20)

pl.title('Classification problem')
pl.subplot(211)
pl.plot(error)
pl.xlabel('Epoch number')
pl.ylabel('error (default SSE)')
w = net.layers[0].np['w']

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

```

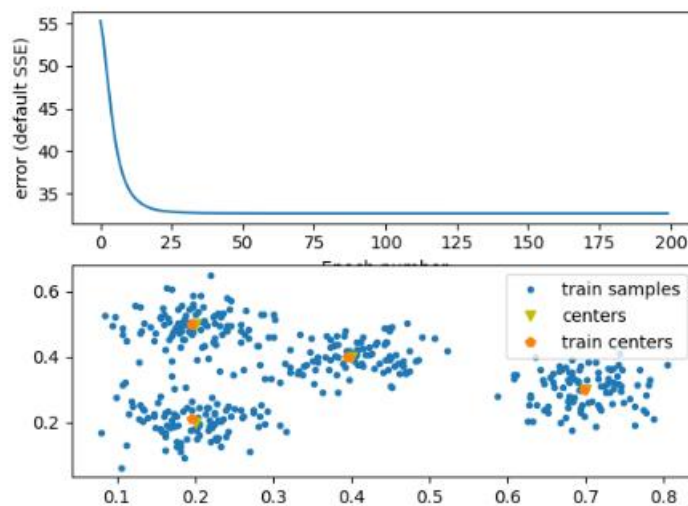


Рис.18. Графік помилковості по епохам та класифікація центрів

```

C:\Users\Max\AppData\Local\Programs\Python\Python311\python.exe "E:\Лабораторні\4 курс\Системи штучного інтелекту\Lab - 5\LR_5_Task_7.py"
Epoch: 20; Error: 33.149235417013486;
Epoch: 40; Error: 32.6986458011785;
Epoch: 60; Error: 32.67234076431852;
Epoch: 80; Error: 32.66982626768639;
Epoch: 100; Error: 32.66953671690182;
Epoch: 120; Error: 32.66950333836992;
Epoch: 140; Error: 32.66949948118511;
Epoch: 160; Error: 32.66949903306823;
Epoch: 180; Error: 32.66949898048166;
Epoch: 200; Error: 32.669498974203734;

```

Рис.17. Звітність навчання

**Завдання 8.** Дослідження нейронної мережі на основі карти Кохонена, що самоорганізується

		Ляшук М.В.			Державний університет «Житомирська політехніка». 22.121.09.000 – Лр5	Арк.
		Філіпов В.О.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

Варіант 9, центри: [0.2, 0.3], [0.4, 0.4], [0.7, 0.3], [0.1, 0.5], [0.4, 0.5], skv = 0.04.

Лістинг коду файлу LR\_5\_Task\_8.py:

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

skv = .04
center = np.array([[0.2, 0.3], [0.4, 0.4], [0.7, 0.3], [0.1, 0.5], [0.4, 0.5]])
random_norm = skv * rand.randn(100, 5, 2)
inp = np.array([center + r for r in random_norm])
inp = inp.reshape(100 * 5, 2)
rand.shuffle(inp)

net = nl.net.newc([[0.0, 1.0], [0.0, 1.0]], 5)
error = net.train(inp, epochs=200, show=20)

pl.title('Classification problem')
pl.subplot(211)
pl.plot(error)
pl.xlabel('Epoch number')
pl.ylabel('error (default SSE)')
w = net.layers[0].np['w']

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

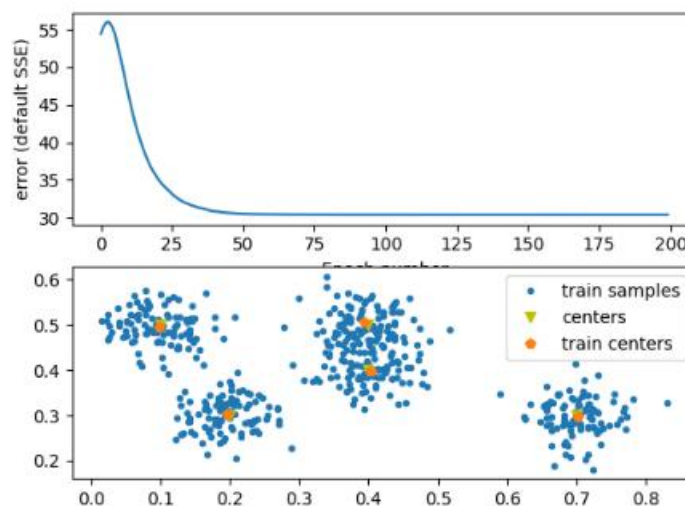


Рис.18. Графік навчання та класифікації за 4 нейрона



```
C:\Users\Max\AppData\Local\Programs\Python\Python311\python.exe "E:\Лабораторні\4 курс\Системи штучного інтелекту\Lab - 5\LR_5_Task_8.py"
Epoch: 20; Error: 35.773520490273775;
Epoch: 40; Error: 30.883345277112408;
Epoch: 60; Error: 30.42078891545894;
Epoch: 80; Error: 30.3731030192955;
Epoch: 100; Error: 30.371976502792492;
Epoch: 120; Error: 30.36823218281319;
Epoch: 140; Error: 30.369347639105474;
Epoch: 160; Error: 30.370052009821894;
Epoch: 180; Error: 30.370268899296327;
Epoch: 200; Error: 30.37032485715726;
```

Рис.19. Звітність за 4х нейронів

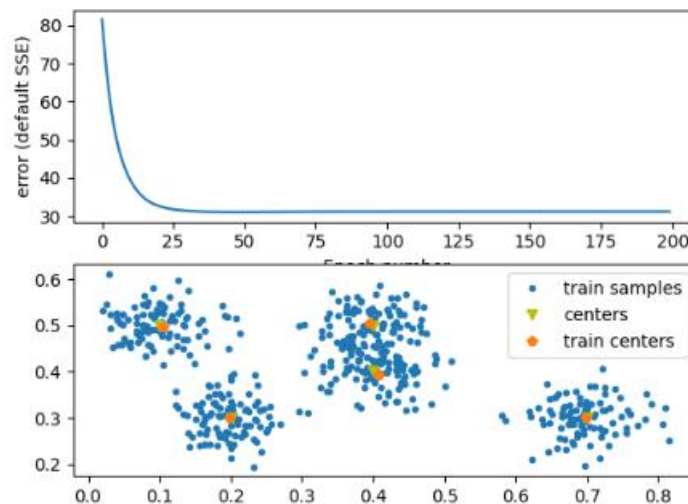


Рис.22. Графік навчання та класифікація за 5ти нейронів

```
C:\Users\Max\AppData\Local\Programs\Python\Python311\python.exe "E:\Лабораторні\4 курс\Системи штучного інтелекту\Lab - 5\LR_5_Task_8.py"
Epoch: 20; Error: 32.94170108620209;
Epoch: 40; Error: 31.154043331963262;
Epoch: 60; Error: 31.156143497182157;
Epoch: 80; Error: 31.229074314102757;
Epoch: 100; Error: 31.24530807626473;
Epoch: 120; Error: 31.248686181462077;
Epoch: 140; Error: 31.249393419787104;
Epoch: 160; Error: 31.249546871127492;
Epoch: 180; Error: 31.249581549234314;
Epoch: 200; Error: 31.249589728385175;
```

Рис.21. Звітність за 5ти нейронів

**Висновок:** було отримано навички зі створення та застосовування простих нейронних мереж використовуючи спеціалізовані бібліотеки та мову програмування Python.

		Ляшук М.В.			Державний університет «Житомирська політехніка». 22.121.09.000 – Лр5	Арк.
		Філіпов В.О				15
Змн.	Арк.	№ докум.	Підпис	Дата		