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 2.1. Створити простий нейрон

```
import numpy as np
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
class Neuron:
    def __init__(self, weights, bias):
        self.weights = weights
        self.bias = bias
    def feedforward(self, inputs):
        total = np.dot(self.weights, inputs) + self.bias
        return sigmoid(total)
if __name__ == "__main__":
    weights = np.array([0, 1])
    bias = 4
    n = Neuron(weights, bias)
    x = np.array([2, 3])
    print(n.feedforward(x))
```

```
0.9990889488055994
```

Завдання 2.2. Створити просту нейронну мережу для передбачення статі людини

```
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):
    return ((y_true - y_pred) ** 2).mean()
class KrasovskyNeuralNetwork:
    def __init__(self):
        # Вага
        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):
```

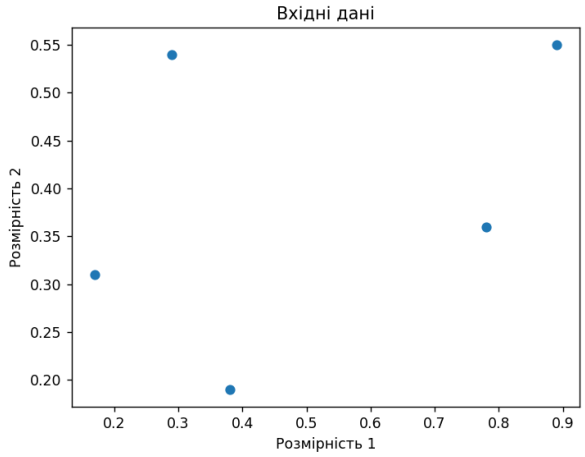
Зм.					
Ро.					
Перевір.	Маєвський О.В.			Звіт з лабораторної роботи	1
Керівник					ФІКТ Гр. КІ-21-1
Н. контр.					
Зав. каф.					

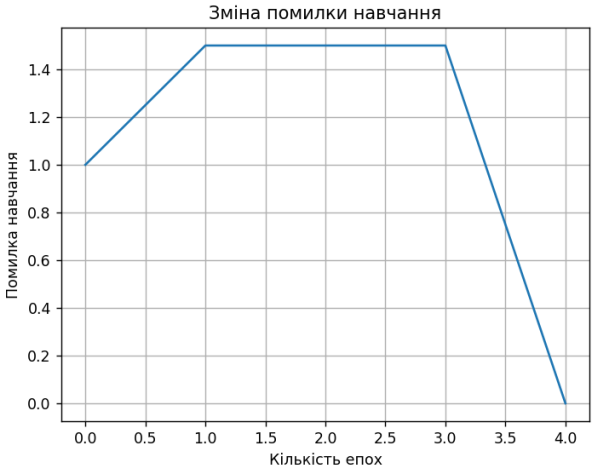
```
                sum_h1 = self.w1 * x[0] + self.w2 * x[1] + self.b1
                h1 = sigmoid(sum_h1)
                sum_h2 = self.w3 * x[0] + self.w4 * x[1] + self.b2
                h2 = sigmoid(sum_h2)
                sum_o1 = self.w5 * h1 + self.w6 * h2 + self.b3
                o1 = sigmoid(sum_o1)
                y_pred = o1
                # --- Підрахунок часткових похідних
                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
            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 = KrasovskyNeuralNetwork()
    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 дюймів
    print("Emily: %.3f" % network.feedforward(emily))  # +-0.966 - F
    print("Frank: %.3f" % network.feedforward(frank))  # +-0.038 - M
```

```
Epoch 720 loss: 0.003
Epoch 730 loss: 0.003
Epoch 740 loss: 0.003
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: 0.002
Epoch 960 loss: 0.002
Epoch 970 loss: 0.002
Epoch 980 loss: 0.002
Epoch 990 loss: 0.002
Emily: 0.949
Frank: 0.040
```

Завдання 2.3. Класифікатор на основі перцептрону з використанням бібліотеки NeuroLab

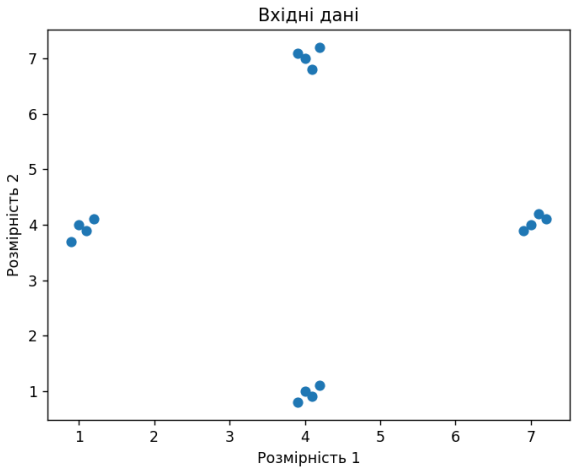
```python
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl
text = np.loadtxt('data_perceptron.txt')
data = text[:, :2]
labels = text[:, 2].reshape((text.shape[0], 1))
plt.figure()
plt.scatter(data[:, 0], data[:, 1])
plt.xlabel('Розмірність 1')
plt.ylabel('Розмірність 2')
plt.title('Вхідні дані')
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('Кількість епох')
plt.ylabel('Помилка навчання')
plt.title('Зміна помилки навчання')
plt.grid()
plt.show()
```

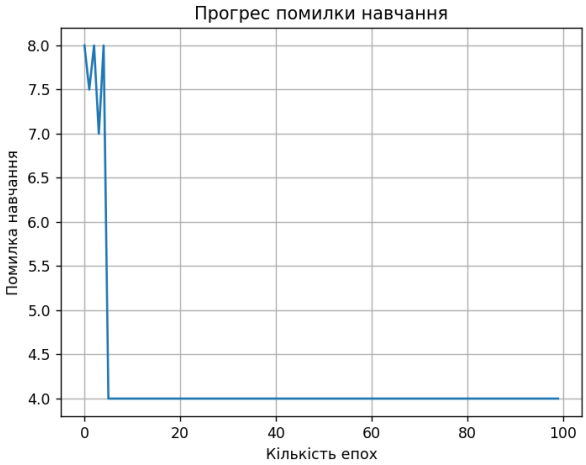




Завдання 2.4. Побудова одношарової нейронної мережі

```
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl
text = np.loadtxt('data_simple_nn.txt')
data = text[:, 0:2]
labels = text[:, 2:]
plt.figure()
plt.scatter(data[:, 0], data[:, 1])
plt.xlabel('Розмірність 1')
plt.ylabel('Розмірність 2')
plt.title('Вхідні дані')
plt.show()
# 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()
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)
# 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()
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])
```



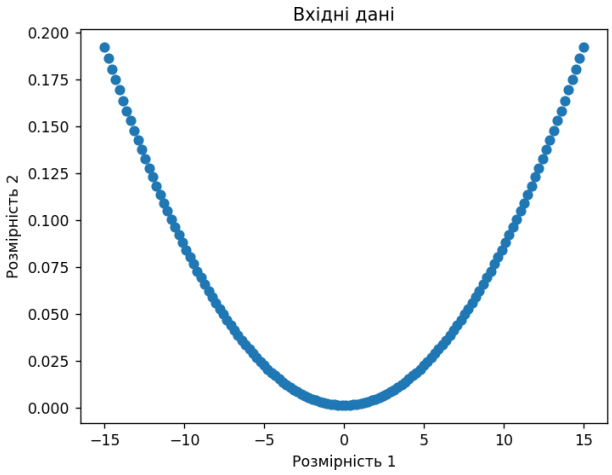


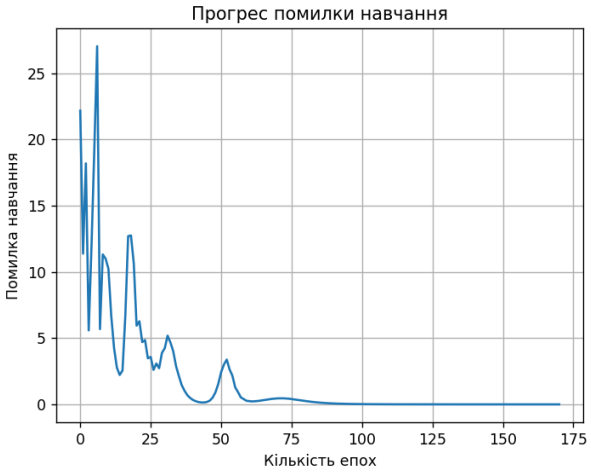

```
Epoch: 20; Error: 4.0;
Epoch: 40; Error: 4.0;
Epoch: 60; Error: 4.0;
Epoch: 80; Error: 4.0;
Epoch: 100; Error: 4.0;
The maximum number of train epochs is reached

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

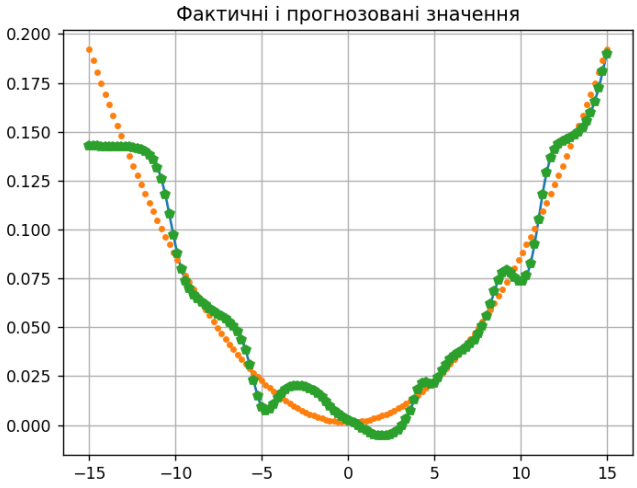
Завдання 2.5. Побудова багатошарової нейронної мережі

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

```
# 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.grid()
plt.show()
```





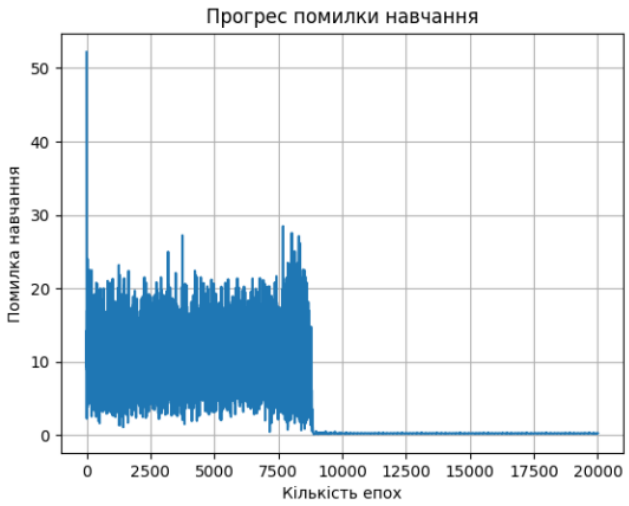


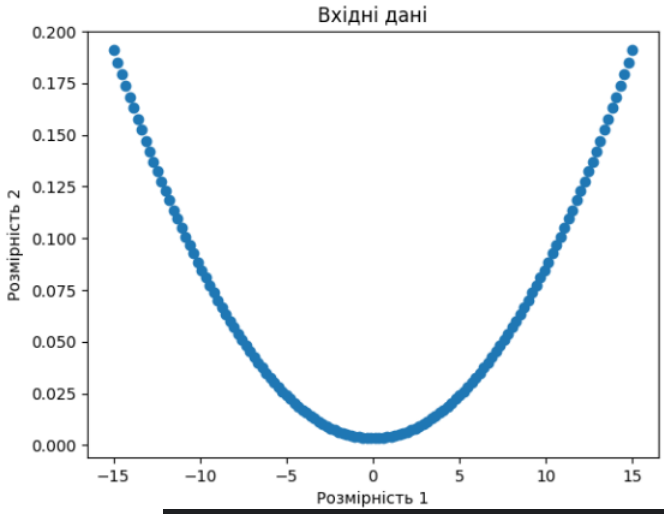

```
Epoch: 100; Error: 0.08732882248943569;
Epoch: 200; Error: 0.030809417061909435;
Epoch: 300; Error: 0.053729694591336516;
Epoch: 400; Error: 0.028947157848686237;
Epoch: 500; Error: 0.03449083259569549;
Epoch: 600; Error: 0.021598542651285385;
Epoch: 700; Error: 0.029292503039546197;
Epoch: 800; Error: 0.014859433595740507;
Epoch: 900; Error: 0.02063754857237577;
Epoch: 1000; Error: 0.014636889094450177;
Epoch: 1100; Error: 0.010801345845027045;
Epoch: 1200; Error: 0.013220060807277466;
Epoch: 1300; Error: 0.011199950506020157;
The goal of learning is reached
```

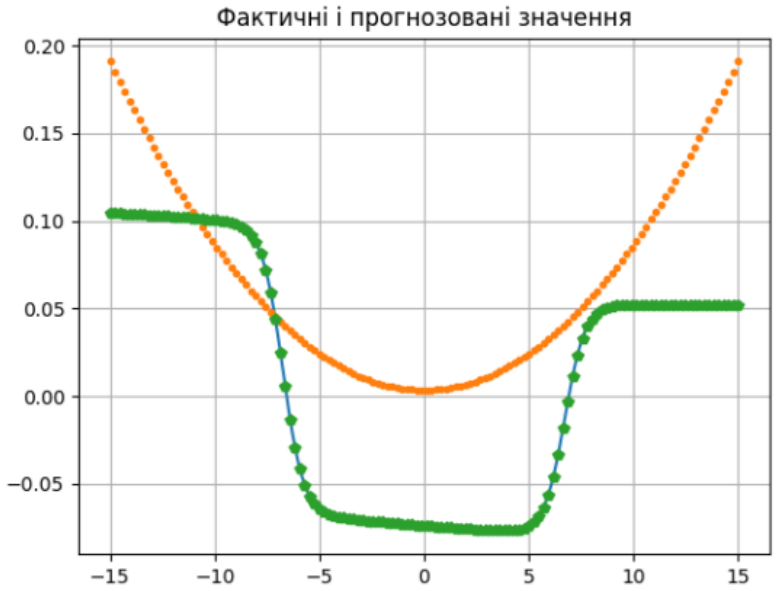
Завдання 2.6. Побудова багатошарової нейронної мережі для свого варіанту

```
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl
min_val = -15
max_val = 15
num_points = 130
x = np.linspace(min_val, max_val, num_points)
y = 2 * np.square(x) + 8
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]], [5, 1])
nn.trainf = nl.train.train_gd
# Тренування нейронної мережі
error_progress = nn.train(data, labels, epochs=20000, show=1000, 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('Прогрес помилки навчання')
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('Фактичні і прогнозовані значення')
plt.grid()
plt.show()
```



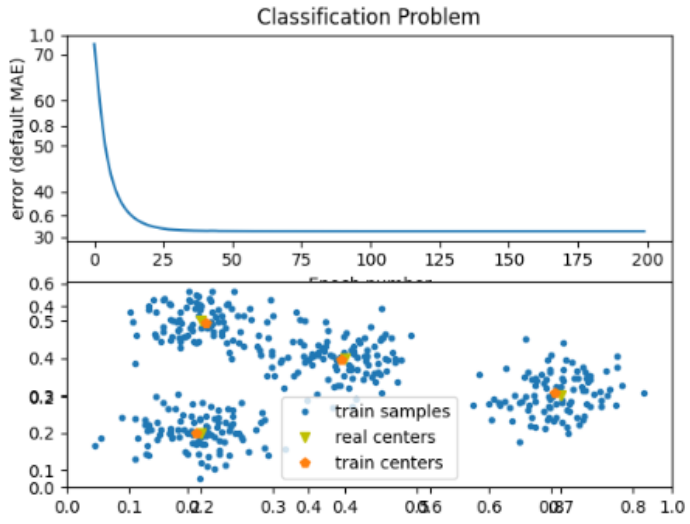





```
Epoch: 1000; Error: 4.061155432569903;
Epoch: 2000; Error: 6.780932339767508;
Epoch: 3000; Error: 6.880188384956554;
Epoch: 4000; Error: 10.605614371063204;
Epoch: 5000; Error: 11.30107817795718;
Epoch: 6000; Error: 15.716544815272957;
Epoch: 7000; Error: 5.99302093117316;
Epoch: 8000; Error: 7.04151551326899;
Epoch: 9000; Error: 0.3727465355245313;
Epoch: 10000; Error: 0.29109274129198487;
Epoch: 11000; Error: 0.25185922808549205;
Epoch: 12000; Error: 0.2770748148462775;
Epoch: 13000; Error: 0.26876271378928496;
Epoch: 14000; Error: 0.24420931004050606;
Epoch: 15000; Error: 0.26764943437177025;
Epoch: 16000; Error: 0.2872016393320364;
Epoch: 17000; Error: 0.24894417273959446;
Epoch: 18000; Error: 0.28900717845522017;
Epoch: 19000; Error: 0.25149099769793404;
Epoch: 20000; Error: 0.31887294306233993;
The maximum number of train epochs is reached
```

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

```
import numpy as np
import neurolab as nl
import numpy.random as rand
skv = 0.05
centr = np.array([[0.2, 0.2], [0.4, 0.4], [0.7, 0.3], [0.2, 0.5]])
rand_norm = skv * rand.randn(100, 4, 2)
inp = np.array([centr + r for r in rand_norm])
inp.shape = (100 * 4, 2)
rand.shuffle(inp)
# 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()
```

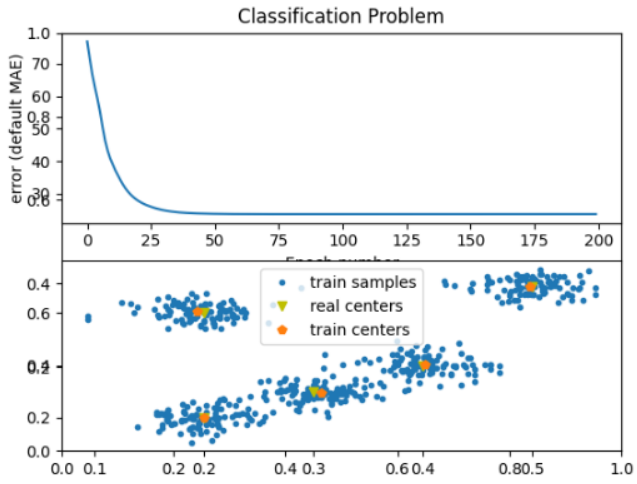



```
Epoch: 20; Error: 32.70262087800296;
Epoch: 40; Error: 31.357890408136065;
Epoch: 60; Error: 31.28775812599054;
Epoch: 80; Error: 31.278140544337628;
Epoch: 100; Error: 31.276665631791193;
Epoch: 120; Error: 31.27643904109247;
Epoch: 140; Error: 31.27640420544947;
Epoch: 160; Error: 31.276398848942158;
Epoch: 180; Error: 31.276398025512222;
Epoch: 200; Error: 31.276397899017077;
The maximum number of train epochs is reached
```

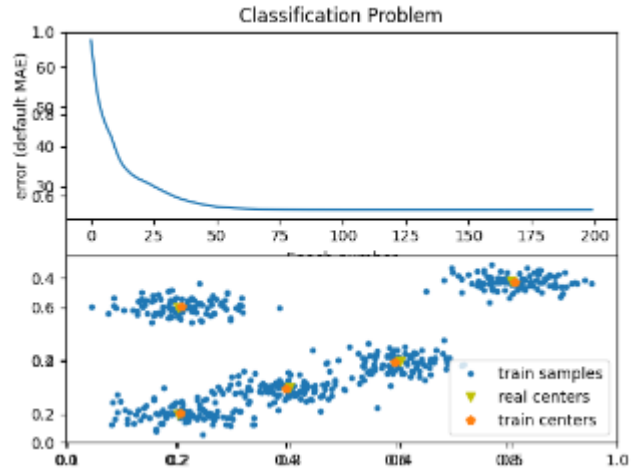
Завдання 2.8. Дослідження нейронної мережі на основі карти Кохонена, що самоорганізується

```
import numpy as np
import neurolab as nl
import numpy.random as rand
skv = 0.03
centr = np.array([[0.2, 0.2], [0.4, 0.4], [0.3, 0.3], [0.2, 0.6], [0.5, 0.7]])
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]], 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()
```




```
Epoch: 20; Error: 28.35141839176814;
Epoch: 40; Error: 24.082671314648913;
Epoch: 60; Error: 23.720960704493983;
Epoch: 80; Error: 23.691387883165785;
Epoch: 100; Error: 23.6991509274219;
Epoch: 120; Error: 23.70223635303263;
Epoch: 140; Error: 23.703249707930368;
Epoch: 160; Error: 23.703560615358562;
Epoch: 180; Error: 23.703655744818334;
Epoch: 200; Error: 23.70368475291959;
The maximum number of train epochs is reached
```




```
Epoch: 20; Error: 31.897557765052493;
Epoch: 40; Error: 26.276807965694594;
Epoch: 60; Error: 24.486178205342615;
Epoch: 80; Error: 24.177610237494576;
Epoch: 100; Error: 24.094775390317604;
Epoch: 120; Error: 24.071292175397005;
Epoch: 140; Error: 24.08388829873744;
Epoch: 160; Error: 24.089018367607814;
Epoch: 180; Error: 24.090450361978547;
Epoch: 200; Error: 24.09087789978174;
The maximum number of train epochs is reached
```

Висновки: в ході виконання лабораторної роботи було досліджено та отримано знання, уміння та навики, щодо особливостей використання спеціалізованих бібліотек та мову програмування Python навчитися створювати та застосовувати прості нейронні мережі.