ЛАБОРАТОРНА РОБОТА № 5

РОЗРОБКА ПРОСТИХ НЕЙРОННИХ МЕРЕЖ

Мета роботи: використовуючи спеціалізовані бібліотеки та мову програмування Python навчитися створювати та застосовувати прості нейронні мережі.
Посилання на GitHub: https://github.com/enot1k666/labsOAI.git

2. ЗАВДАННЯ НА ЛАБОРАТОРНУ РОБОТУ ТА МЕТОДИЧНІ РЕКОМЕНДАЦІЇ ДО ЙОГО ВИКОНАННЯ

Завдання 2.1. Створити простий нейрон

Лістинг програми:

```
import numpy as np
import self

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

    # Вхідні дані про вагу, додавання зміщення і подальше використання функції ак-
тивації
    def feedforward(self, inputs):
        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))
```

Результат виконання:

```
C:\Users\skoki\PycharmProjects\labsOAI\.venv\Scripts\python.exe C:\Users\skoki\PycharmProjects\labsOAI\lab5\task1.py
0.9990889488055994
```

					ДУ «Житомирська політехніка».22.121.18.000 – Лр5		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Скоківський В.			Звіт з лабораторної роботи	Літ.	Арк. 1 / Аркушів
Перевір.		Голенко М.Ю.					
Керівник						ФІКТ Гр. ІПЗ-20-3	
Н. контр.							
Зав. каф.							

Завдання 2.2. Створити просту нейронну мережу для передбачення статі людини

Лістинг програми:

```
import numpy as np
import self

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

    # Вхідні дані про вагу, додавання зміщення і подальше використання функції ак-
тивації
    def feedforward(self, inputs):
        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))


class SkokivskyNeuralNetwork:
    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 = SkokivskyNeuralNetwork()
x= np.array([2, 3])
print(network.feedforward(x))
```

Результат виконання:

```
C:\Users\skoki\PycharmProjects\labsOAI\.venv\Scripts\python.exe C:\Users\skoki\PycharmProjects\labsOAI\lab5\task2.py
0.9990889488055994
0.7216325609518421
```

Висновки:

Функція Sigmoid – перетворює ваговані суми вхідних сигналів в діапазон значень між 0 і 1.

Mean Squared Error - визначає середньоквадратичну помилку між прогнозованими значеннями та фактичними значеннями вихідної змінної.

Можливості нейронних мереж прямого поширення:

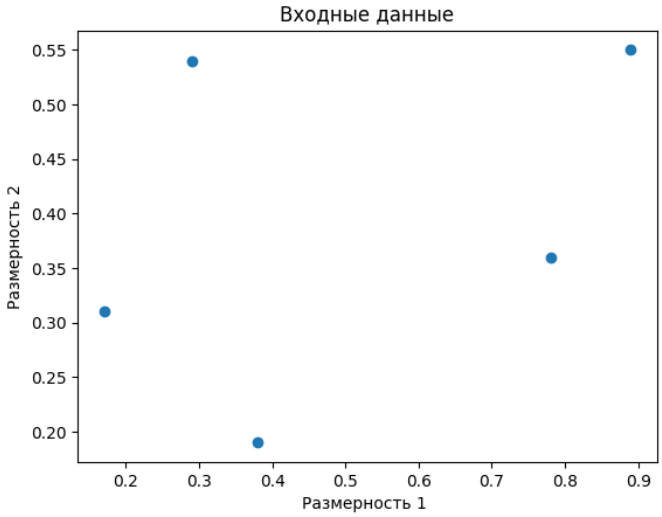
1. Вони можуть бути використані для вирішення багатьох типів завдань, включаючи класифікацію, регресію, розпізнавання образів та багато інших.
2. Вони можуть навчатися на прикладах і вдосконалювати свої параметри, щоб наближати вихід до бажаного результату (наприклад, мінімізувати втрати).
3. Нейронні мережі можуть автоматично визначати ваги та зміщення для вирішення конкретних завдань, що робить їх потужними і універсальними інструментами для багатьох додатків.

Завдання 2.3. Класифікатор на основі перцептрону з використанням бібліотеки NeuroLab

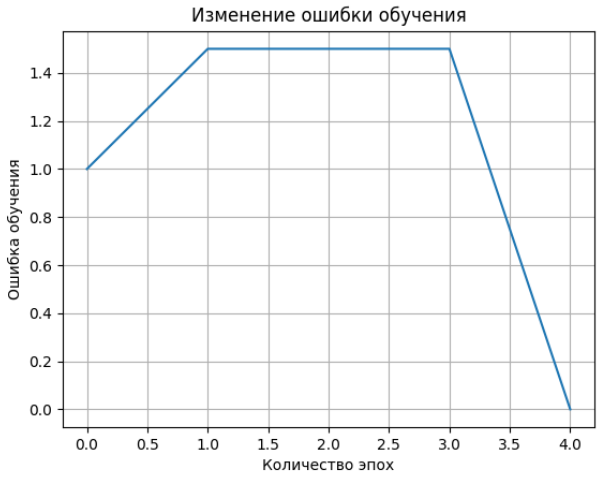
Лістинг програми:

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

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

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

plt. figure()
plt.scatter(data[:, 0], data[:, 1])
plt. xlabel('Размерность 1 ')
plt.ylabel('Размерность 2')
plt. title('Входные данные')

# 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('Количество эпох')
plt.ylabel('Ошибка обучения')
plt.title('Изменение ошибки обучения')
plt.grid()
plt.show()
```

Результат виконання:



Графік №1. Вхідні дані.



Графік №2. Класифікування вхідних даних.

Висновок: Другий графік показує зміну помилки навчання впродовж епох під час навчання перцептрону.З часом помилка навчання зменшується і стає близькою до нуля. Це означає, що перцептрон зміг навчитися правильно класифікувати вхідні дані.

Завдання 2.4. Побудова одношарової нейронної мережі

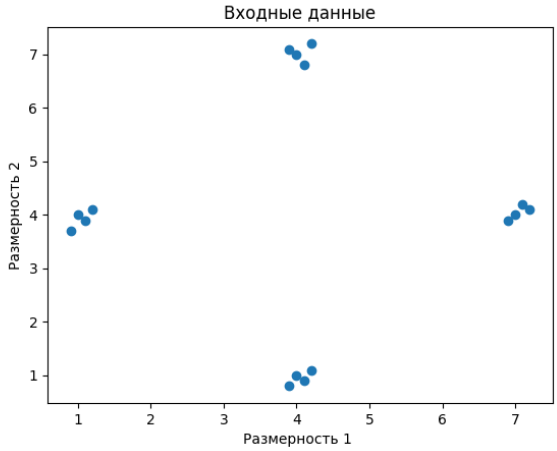
Створіть одношарову нейронну мережу, що складається з незалежних нейронів, для вхідного файлу `data_simple_nn.txt`.

Лістинг програми:

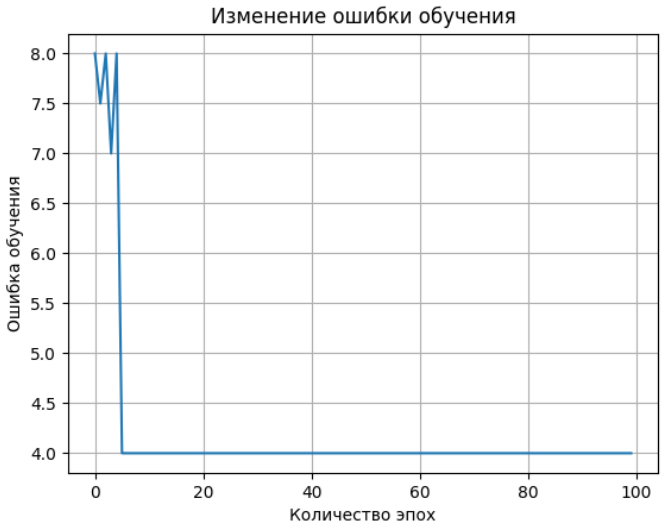
```
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl

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

# Поділ точок даних та міток
data = text[:, 0:2]
labels = text[:, 2:]

# Побудова графіка вхідних даних
plt. figure()
plt.scatter(data[:, 0], data[:, 1])
plt. xlabel('Размерность 1 ')
plt.ylabel('Размерность 2')
plt. title('Входные данные')

# 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)

#Навчимо мережу на тренувальних даних
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('Количество эпох')
plt.ylabel('Ошибка обучения')
plt.title('Изменение ошибки обучения')
plt.grid()
plt.show()
# Виконання класифікатора на тестових точках даних
print('\n 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])
```

Результат виконання:

```
C:\Users\skoki\PycharmProjects\labsOAI\.venv\Scripts\python.exe C:\Users\skoki\PycharmProjects\labsOAI\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.]
```



Графік №1. Вхідні дані.



Графік №2. Класифікування вхідних даних.

Висновок: Другий графік відображає зміну помилки під час навчання нейронної мережі. Рівень помилки зменшився з 8.0 до 4.0 (приблизно за 5 епох),але значення 4.0 не змінилось за 100 епох,можна зробити висновок, що мережа здатна вдосконалити своє рішення під час навчання,але вона не досягла задовільної точності класифікації даних.
З тестових результатів можна зробити висновок,що мережа видає виведення [0, 0] для першої тестової точки, [1, 0] для другої точки і [1, 1] для третьої точки.

Ці виведення вказують на класифікацію точок, але вони є некоректними, оскільки вони не відповідають очікуваним класам.

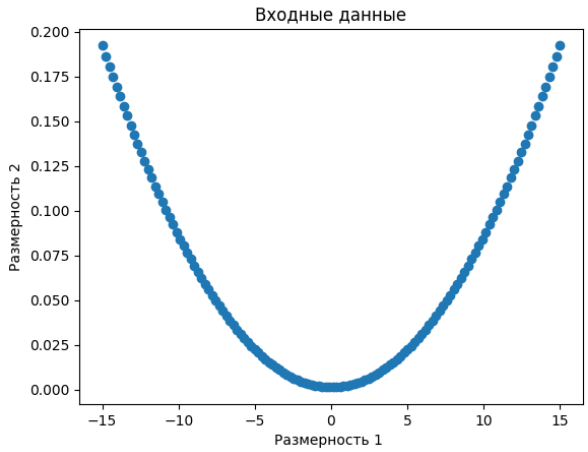
Завдання 2.5. Побудова багатошарової нейронної мережі

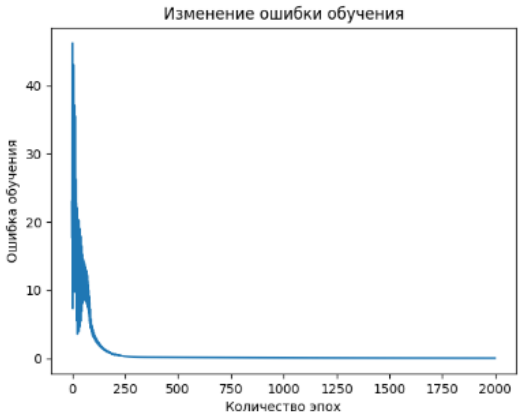
Лістинг програми:

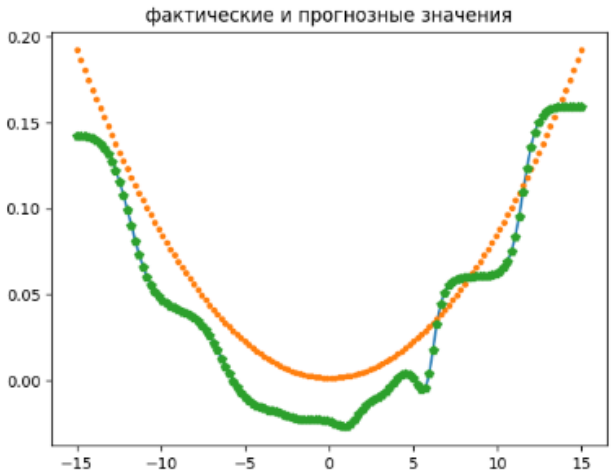
```
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl

# 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)

#Побудуємо графік вхідних даних.
plt. figure()
plt.scatter(data,labels)
plt. xlabel('Размерность 1 ')
plt.ylabel('Размерность 2')
plt. title('Входные данные')

# 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 мереж
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('Количество эпох')
plt.ylabel('Ошибка обучения')
plt.title('Изменение ошибки обучения')

# 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('фактические и прогнозные значения')
plt.show()
```

Результат виконання:

```
C:\Users\skoki\PycharmProjects\labsOAI\.venv\Scripts\python.exe C:\Users\skoki\PycharmProjects\labsOAI\lab5\task5.py
Epoch: 100; Error: 3.3279150454681767;
Epoch: 200; Error: 0.5904466735072104;
Epoch: 300; Error: 0.22375594178770664;
Epoch: 400; Error: 0.1802609154095195;
Epoch: 500; Error: 0.16063931454683525;
Epoch: 600; Error: 0.14363100762843883;
Epoch: 700; Error: 0.12813941622975136;
Epoch: 800; Error: 0.11270942311375966;
Epoch: 900; Error: 0.09859214118670911;
Epoch: 1000; Error: 0.0866389934745466;
Epoch: 1100; Error: 0.0770491160610457;
Epoch: 1200; Error: 0.06946277095455822;
Epoch: 1300; Error: 0.06336554378612864;
Epoch: 1400; Error: 0.05830813535592029;
Epoch: 1500; Error: 0.05397633325606394;
Epoch: 1600; Error: 0.050179236612388455;
Epoch: 1700; Error: 0.04680987270527913;
Epoch: 1800; Error: 0.04380906593825513;
Epoch: 1900; Error: 0.04114161554882166;
Epoch: 2000; Error: 0.03878304264963466;
The maximum number of train epochs is reached
```







Висновок: В терміналі було показано навчання мережі(номер епохи та значення її помилки),навчання відбулось протягом 2000 епох, найкраща досягнута помилка становила приблизно 0.0387(при початковій – 3.32).

Завдання 2.6. Побудова багатошарової нейронної мережі для свого варіанту

№ варіанта	Тестові дані
Варіант 18	$y = 5x^2+9$

Номер варіанта	Багатошаровий персептрон	
	Кількість шарів	Кількості нейронів у шарах
18	3	7-4-1

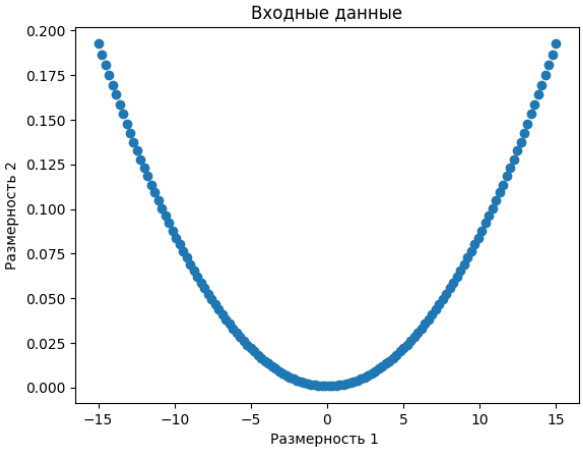
Лістинг програми:

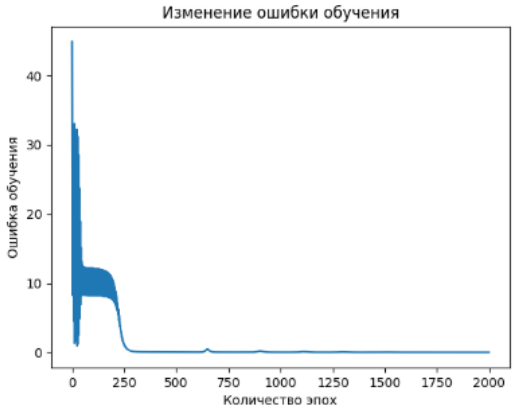
```
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl

# Генерація тренувальних даних
min_val = -15
max_val = 15
num_points = 130
x = np. linspace(min_val, max_val, num_points)
y = 5 * 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)

#Побудуємо графік вхідних даних.
plt. figure()
plt.scatter(data,labels)
plt. xlabel('Размерность 1 ')
plt.ylabel('Размерность 2')
plt. title('Входные данные')

# Вихідний шар складається з одного нейрона.
nn = nl.net.newff([[min_val, max_val]],[7,4,1])

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

# Тренування нейронної мереж
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('Количество эпох')
plt.ylabel('Ошибка обучения')
plt.title('Изменение ошибки обучения')

# Побудова графіка результатів
x_dense = np. linspace(min_val, max_val, num_points * 2)
```

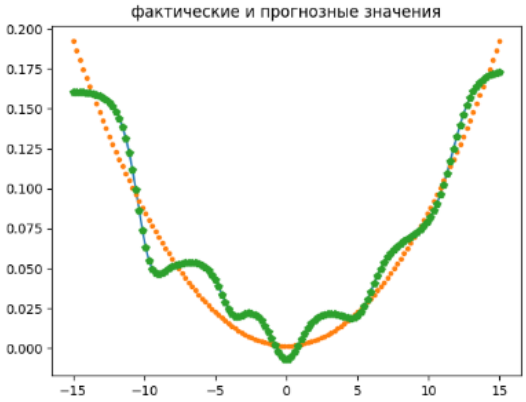
```
y_dense_pred = nn.sim(x_dense.reshape(x_dense.size, 1)).reshape(x_dense.size)
plt. figure ()
plt.plot(x_dense, y_dense_pred, '-',x, y,'.',x,y_pred,'p')
plt.title('фактические и прогнозные значения')
plt.show()
```

Результат виконання:

```
C:\Users\skoki\PycharmProjects\labsOAI\.venv\Scripts\python.exe C:\Users\skoki\PycharmProjects\labsOAI\lab5\task6.py
Epoch: 100; Error: 12.135955557632712;
Epoch: 200; Error: 10.019281192246956;
Epoch: 300; Error: 0.11933600852743462;
Epoch: 400; Error: 0.06398732540883006;
Epoch: 500; Error: 0.05256562844906525;
Epoch: 600; Error: 0.04380432897934352;
Epoch: 700; Error: 0.04142826668962155;
Epoch: 800; Error: 0.03397845609364439;
Epoch: 900; Error: 0.1890161703700312;
Epoch: 1000; Error: 0.026466817166251416;
Epoch: 1100; Error: 0.08744145147119382;
Epoch: 1200; Error: 0.021488593293024896;
Epoch: 1300; Error: 0.06942552479915343;
Epoch: 1400; Error: 0.017915911901576015;
Epoch: 1500; Error: 0.033157565733187976;
Epoch: 1600; Error: 0.015850623965647036;
Epoch: 1700; Error: 0.013158343358626834;
Epoch: 1800; Error: 0.013725385907307647;
Epoch: 1900; Error: 0.01269208456319922;
Epoch: 2000; Error: 0.010837521114976647;
The maximum number of train epochs is reached
```







Висновок: В терміналі було показано навчання мережі(номер епохи та значення її помилки),навчання відбулось протягом 2000 епох, найкраща досягнута помилка становила приблизно 0.01(при початковій – 12.1).

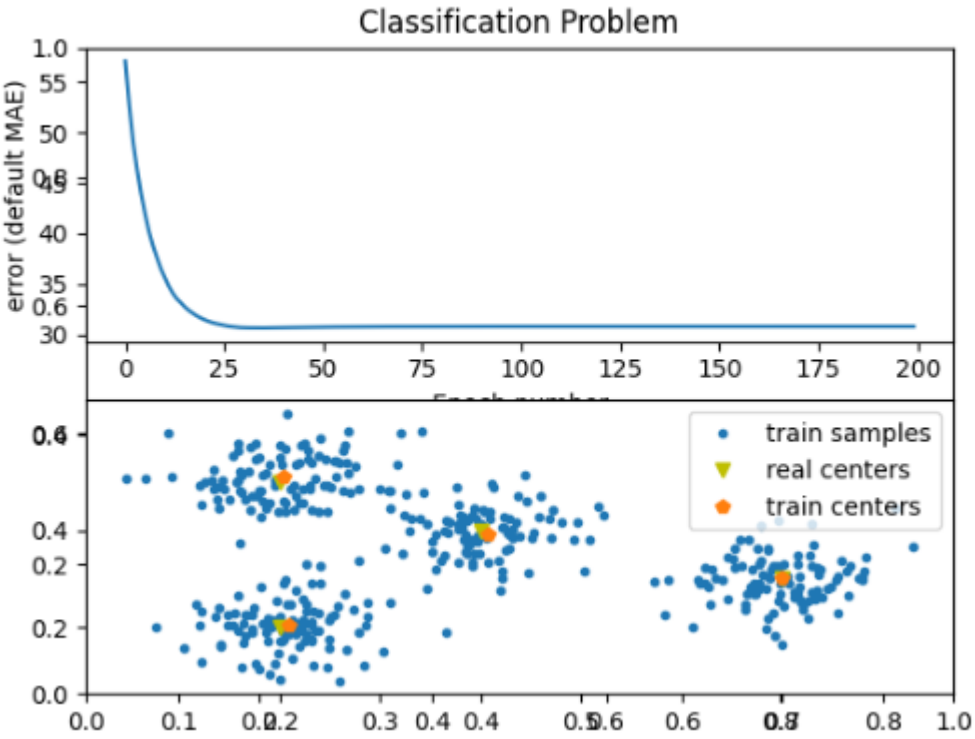
Завдання 2.7. Побудова нейронної мережі на основі карти Кохонена, що самоорганізується

Лістинг програми:

```
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=20)

# 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:

```
C:\Users\skoki\PycharmProjects\labsOAI\.venv\Scripts\python.exe C:\Users\skoki\PycharmProjects\labsOAI\lab5\task7.py
Epoch: 20; Error: 31.63918168120306;
Epoch: 40; Error: 30.661703765008387;
Epoch: 60; Error: 30.76039042967288;
Epoch: 80; Error: 30.7847406162754;
Epoch: 100; Error: 30.789354851671554;
Epoch: 120; Error: 30.79017110714269;
Epoch: 140; Error: 30.790316110999925;
Epoch: 160; Error: 30.790341956118308;
Epoch: 180; Error: 30.790346574757468;
Epoch: 200; Error: 30.790347401881775;
The maximum number of train epochs is reached
```



Висновок: Ми побудували нейрону мережу на основі карти Кохонена,модель не справилась з завданням(не покращувала свою точність),значення через 200 епох зменшилось на 0.84.Помилка MAE – вимірює наскільки середньоквадратична похибка змінюється під час навчання нейромережі.

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

№ варіанту	Центри кластера	skv
Варіант 18	[0.2, 0.3], [0.4, 0.4], [0.7, 0.3], [0.2, 0.5], [0.5, 0.5]	0,06

Створіть нейронну мережу Кохонена з 2 входами та 4 нейронами

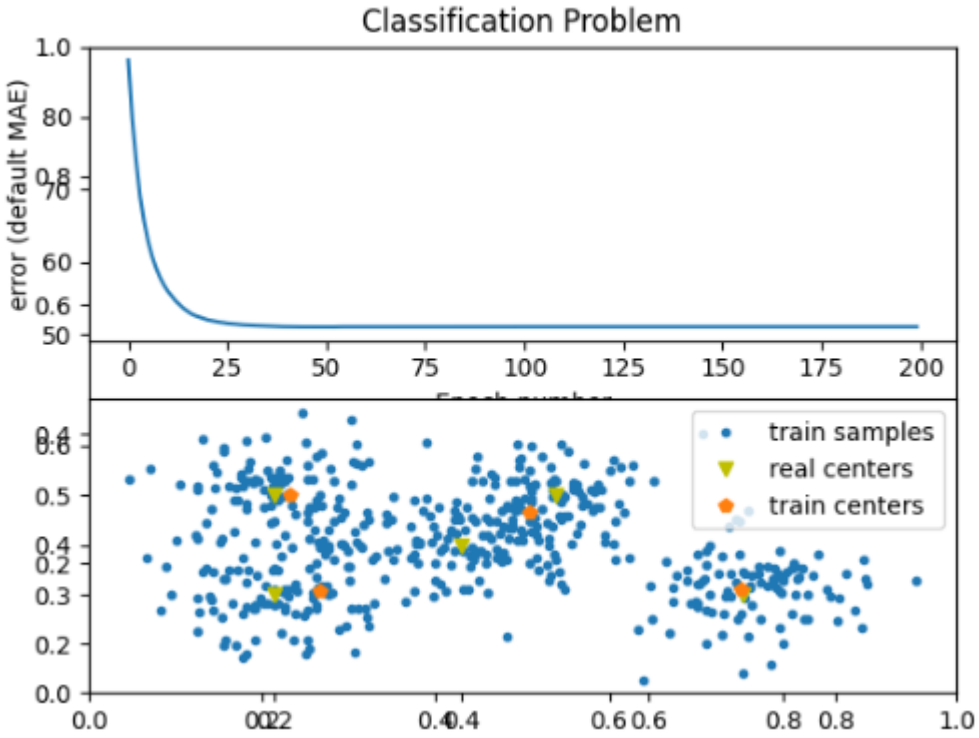
Лістинг програми:

```
import numpy as np
import neurolab as nl
import numpy.random as rand
import pylab as pl
skv = 0.06
centr = np.array([[0.2, 0.3], [0.4, 0.4], [0.7, 0.3], [0.2, 0.5],[0.5, 0.5]])
rand_norm = skv * rand.randn(100, 5, 2) )#змінив значення з 4 на 5
inp = np.array([centr + r for r in rand_norm])
inp.shape = (100 * 5, 2)#змінив значення з 4 на 5
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:
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:

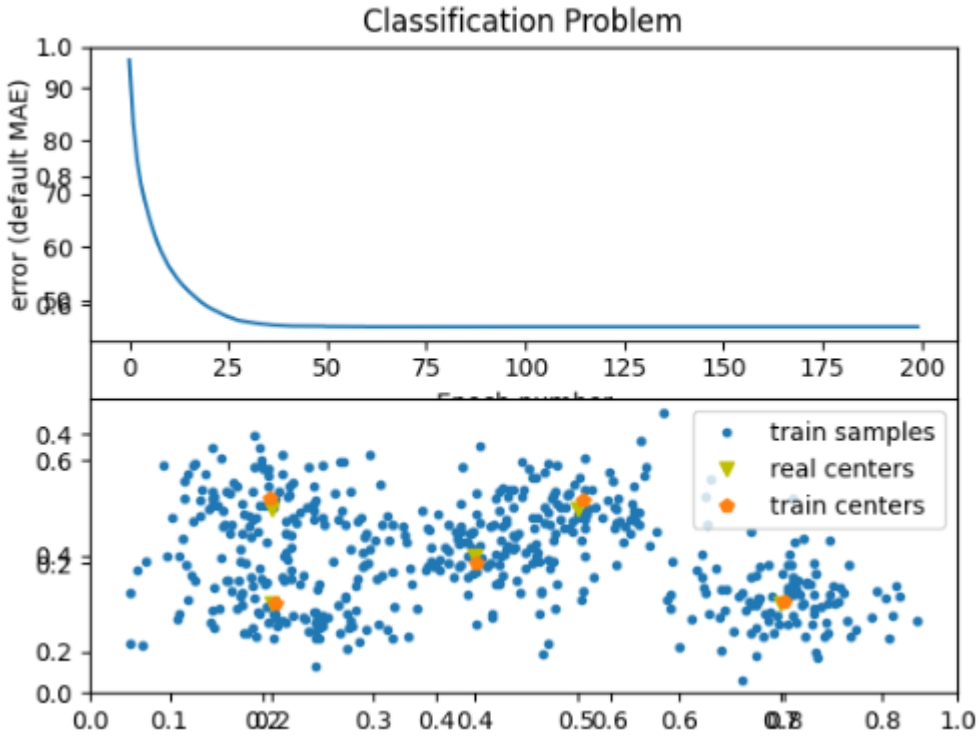
```
C:\Users\skoki\PycharmProjects\labsOAI\.venv\Scripts\python.exe C:\Users\skoki\PycharmProjects\labsOAI\lab5\task8.py
Epoch: 20; Error: 52.21609243947757;
Epoch: 40; Error: 51.0901395765834;
Epoch: 60; Error: 51.07198450125132;
Epoch: 80; Error: 51.076771512559944;
Epoch: 100; Error: 51.07675257627356;
Epoch: 120; Error: 51.07673548140801;
Epoch: 140; Error: 51.07673423441359;
Epoch: 160; Error: 51.07673417708807;
Epoch: 180; Error: 51.07673417678711;
Epoch: 200; Error: 51.076734177143756;
The maximum number of train epochs is reached
```



Створіть нейронну мережу Кохонена з 2 входами та 5 нейронами

Результат виконання:

```
C:\Users\skoki\PycharmProjects\labsOAI\.venv\Scripts\python.exe C:\Users\skoki\PycharmProjects\labsOAI\lab5\task8.py
Epoch: 20; Error: 49.17846257597526;
Epoch: 40; Error: 45.22138528307414;
Epoch: 60; Error: 45.004735085592046;
Epoch: 80; Error: 44.972122439326895;
Epoch: 100; Error: 44.978232316604604;
Epoch: 120; Error: 44.97916873303316;
Epoch: 140; Error: 44.97930405825683;
Epoch: 160; Error: 44.97932362445584;
Epoch: 180; Error: 44.97932644278234;
Epoch: 200; Error: 44.97932684675703;
The maximum number of train epochs is reached
```



Висновок: Зменшення кількості кластерів при незмінній кількості нейронів може поліпшити точність моделі, оскільки відображає кількість класів або кластерів, які модель намагається розрізнити. У другому випадку (5 нейронів і 4 кластери), зменшення кількості кластерів знизило помилку MAE. Таким чином, вибір кількості нейронів і кластерів повинен враховувати природу даних і завдання, яке ви намагаєтеся вирішити. При 5 кластерах і 5 нейронах було отримано значення MAE – 44.97, при 5 кластерах і 4 нейронах – 51,07. Якщо порівнювать з попереднім завданням,то в цьому нейрона мережа відпрацювала краще,т.к., значення помилки значно зменшилось.