**ЛАБОРАТОРНА РОБОТА № 5**

РОЗРОБКА ПРОСТИХ НЕЙРОННИХ МЕРЕЖ

**Посилання на GitHub: https://github.com/FrancIwanicki/OAI.git**

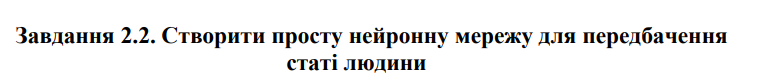
**Мета роботи**: використовуючи спеціалізовані бібліотеки та мову програмування Python навчитися створювати та застосовувати прості нейронні мережі.



**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 активації  
 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))

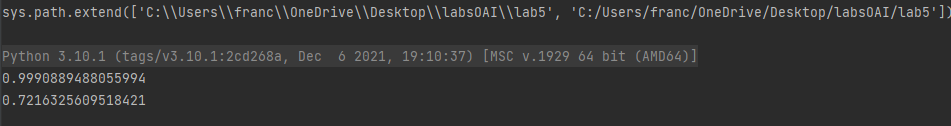
**Результат виконання:  
**

****

**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 активації  
 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 IvanytskyiNeuralNetwork:  
 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 = IvanytskyiNeuralNetwork()  
x= np.array([2, 3])  
print(network.feedforward(x))

**Результат виконання:**

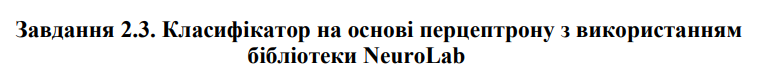
****

**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:

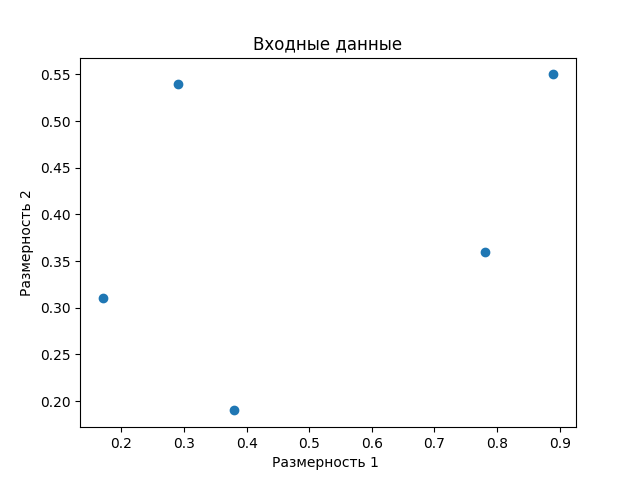
1. Вони можуть бути використані для вирішення багатьох типів завдань, включаючи класифікацію, регресію, розпізнавання образів та багато інших.
2. Вони можуть навчатися на прикладах і вдосконалювати свої параметри, щоб наближати вихід до бажаного результату (наприклад, мінімізувати втрати).
3. Нейронні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



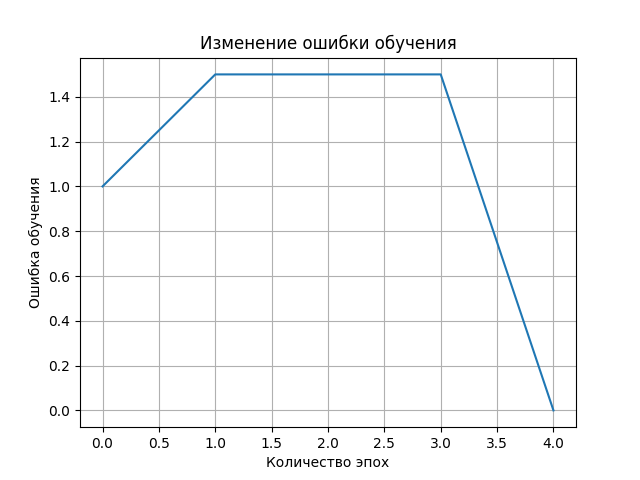
**Лістинг програми:**

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:**

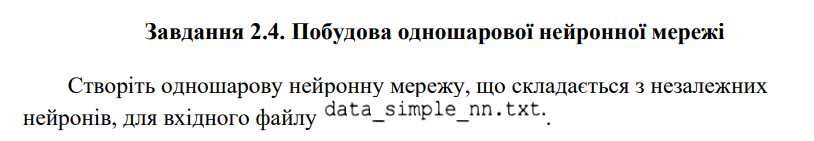
****

Графік №1. Вхідні дані.

****

Графік №2. Класифікування вхідних даних.

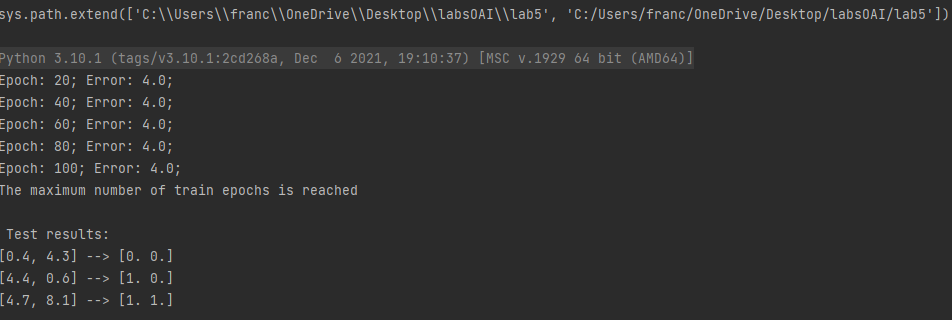
**Висновок:** Другий графік показує зміну помилки навчання впродовж епох під час навчання перцептрону.З часом помилка навчання зменшується і стає близькою до нуля. Це означає, що перцептрон зміг навчитися правильно класифікувати вхідні дані.

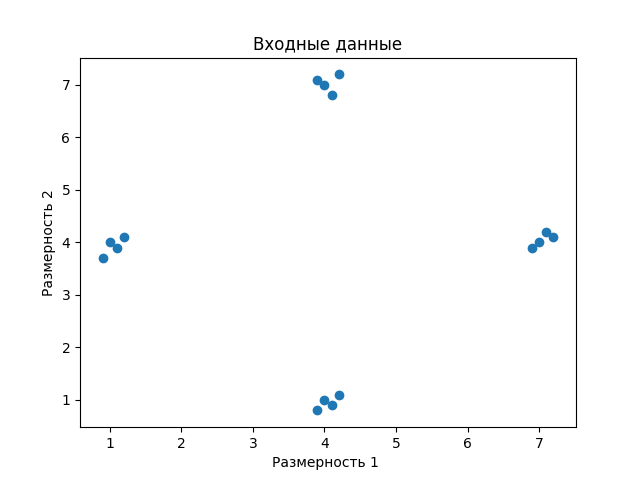
****

**Лістинг програми:**

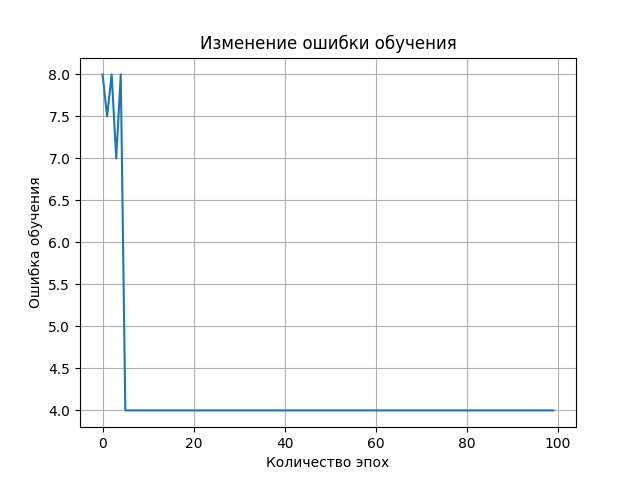
import numpy as np  
import matplotlib.pyplot as plt  
import neurolab as nl  
  
# Завантаження вхідних даних  
text = np.loadtxt('data\_simple\_nn.txt')  
  
# Поділ точок даних та міток  
data = text[:, 0:2]  
labels = text[:, 2:]  
  
# Побудова графіка вхідних даних  
plt. figure()  
plt.scatter(data[:, 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:**

****

****

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.

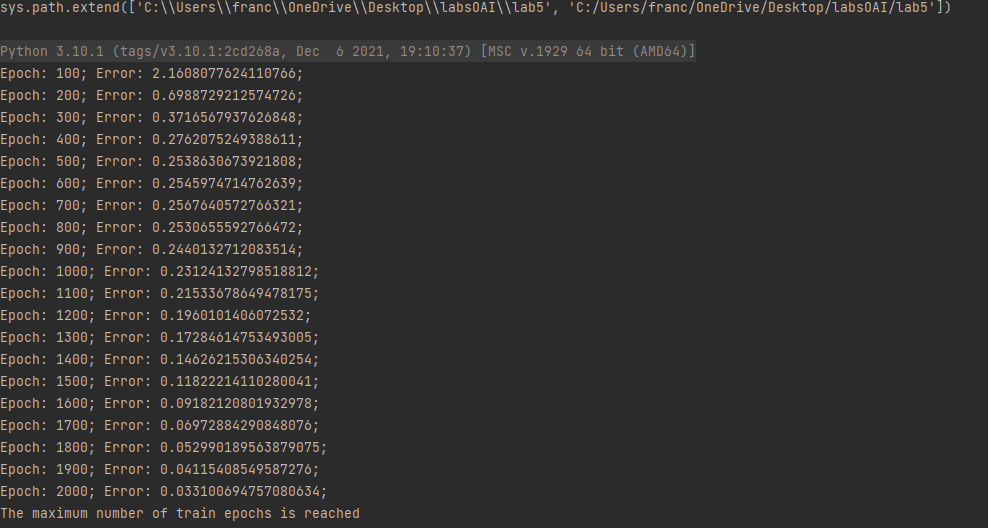
З тестових результатів можна зробити висновок,що мережа видає виведення [0, 0] для першої тестової точки, [1, 0] для другої точки і [1, 1] для третьої точки. Ці виведення вказують на класифікацію точок, але вони є некоректними, оскільки вони не відповідають очікуваним класам.

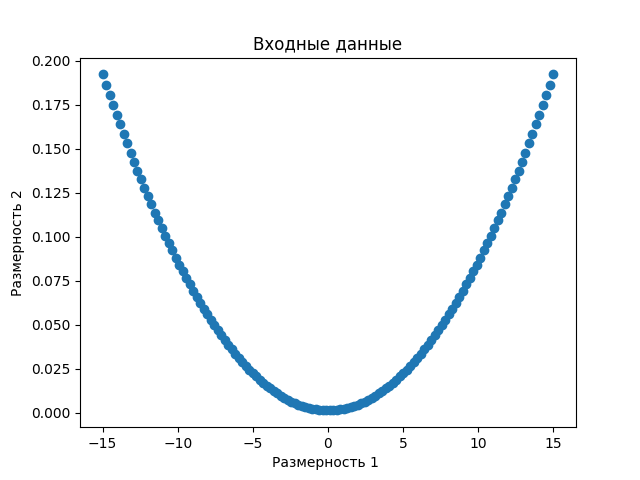
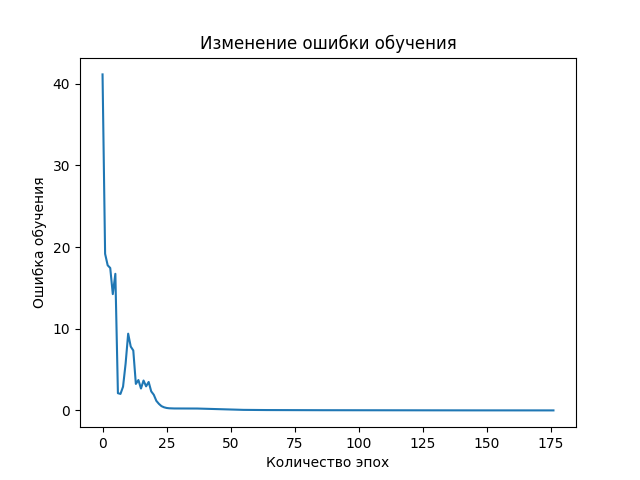


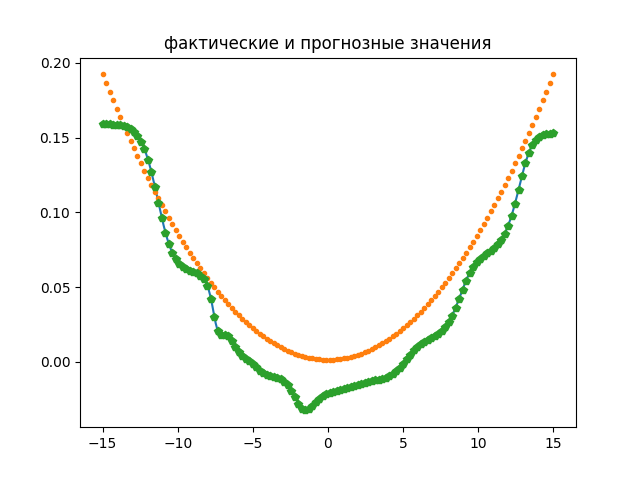
**Лістинг програми:**

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:**

****

**** ****

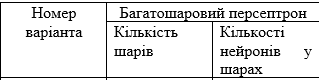
****

**Висновок:** В терміналі було показано навчання мережі(номер епохи та значення її помилки),навчання відбулось протягом 2000 епох, найкраща досягнута помилка становила приблизно 0.0331(при початковій – 2.16).







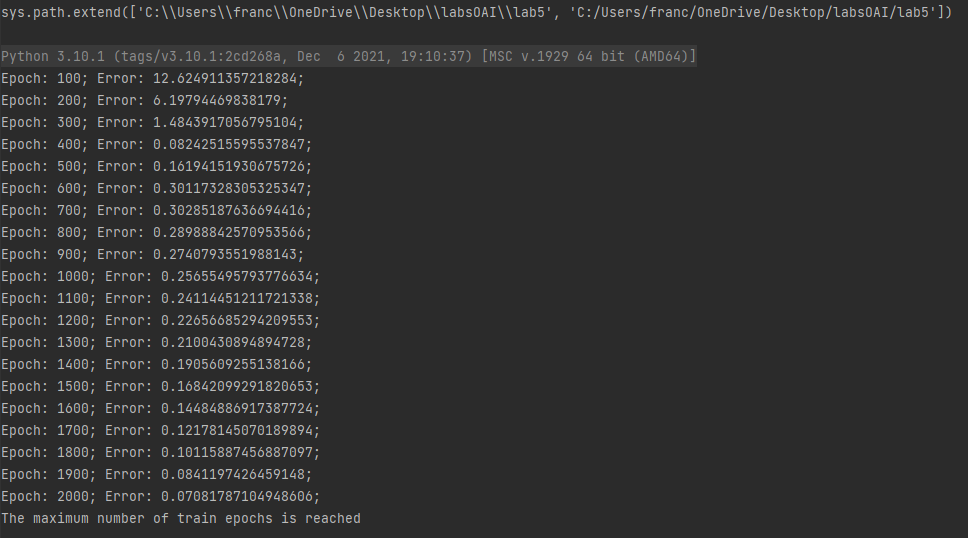


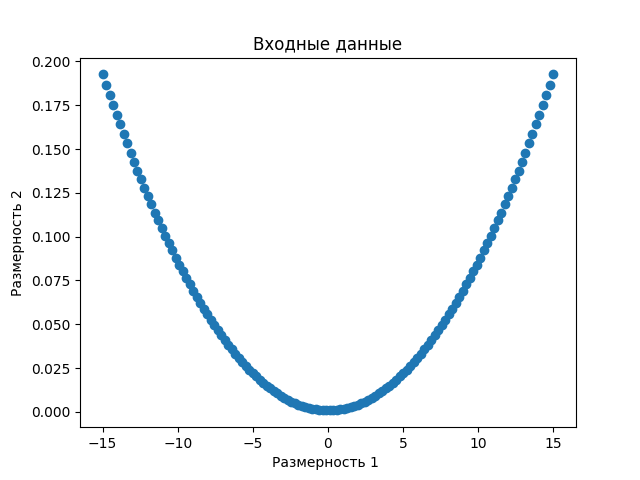
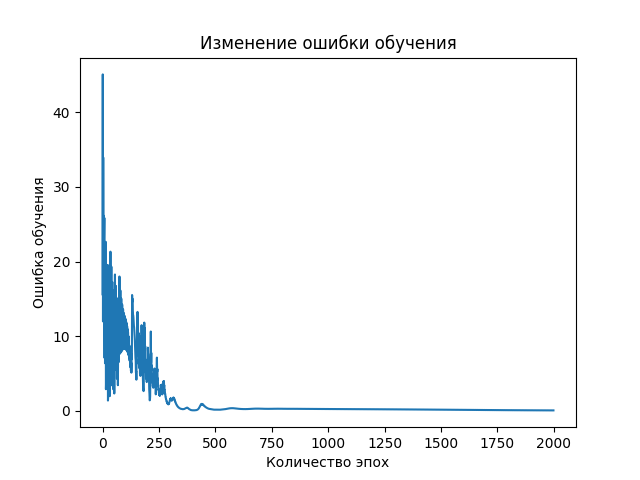


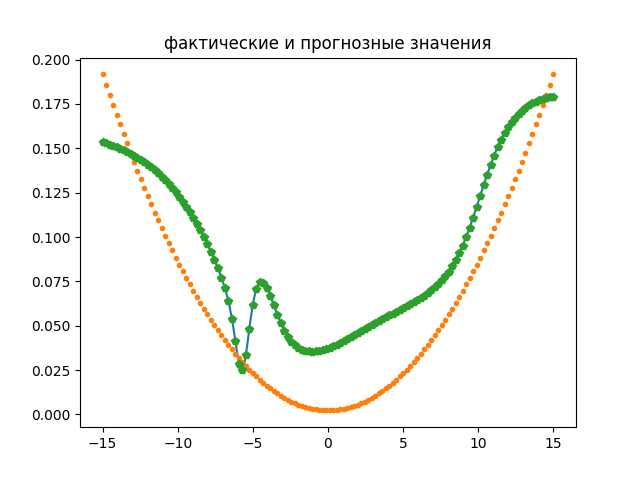
**Лістинг програми:**

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) + 8  
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]],[5,5,1])  
  
# 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()

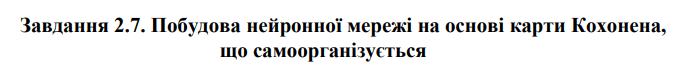
**Результат виконання:**

****

**** 

****

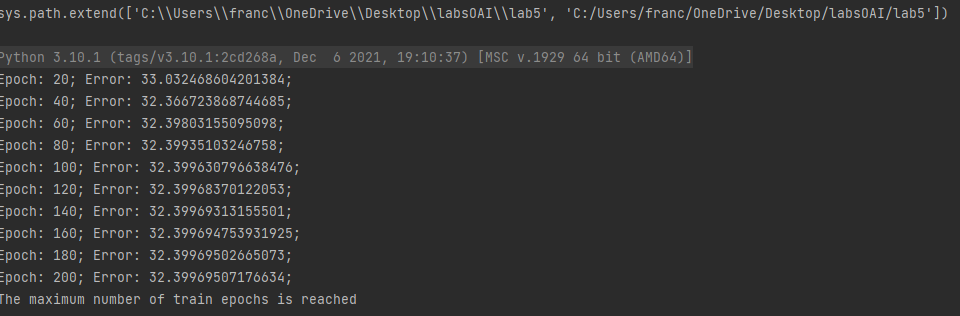
**Висновок:** В терміналі було показано навчання мережі(номер епохи та значення її помилки),навчання відбулось протягом 2000 епох, найкраща досягнута помилка становила приблизно 0.070(при початковій – 12.62).

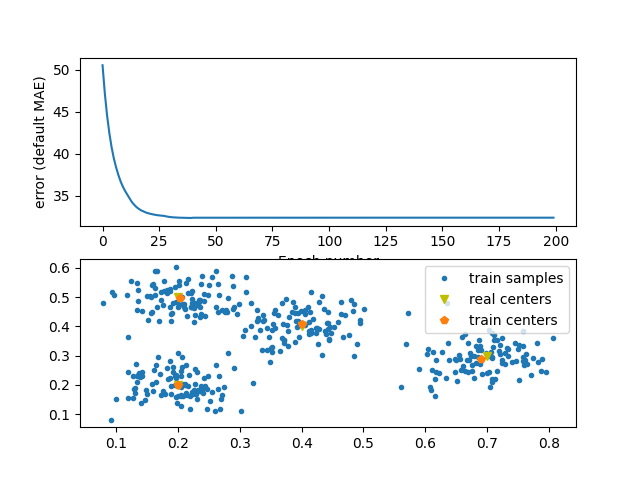


**Лістинг програми:**

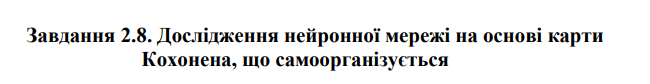
import numpy as np  
import neurolab as nl  
import numpy.random as rand  
import pylab as pl  
skv = 0.05  
centr = np.array([[0.2, 0.2], [0.4, 0.4], [0.7, 0.3], [0.2, 0.5]])  
rand\_norm = skv \* rand.randn(100, 4, 2)  
inp = np.array([centr + r for r in rand\_norm])  
inp.shape = (100 \* 4, 2)  
rand.shuffle(inp)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

****

****

**Висновок:** Ми побудували нейрону мережу на основі карти Кохонена,модель не справилась з завданням(не покращувала свою точність),значення через 200 епох зменшилось на 0.633.Помилка MAE – вимірює наскільки середньоквадратична похибка змінюється під час навчання нейромережі.





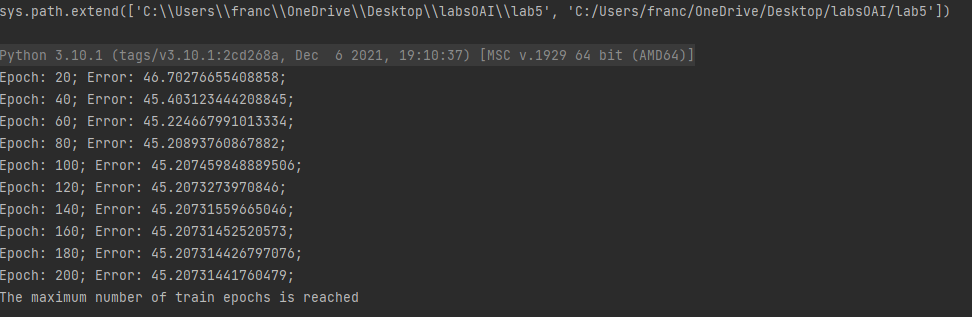


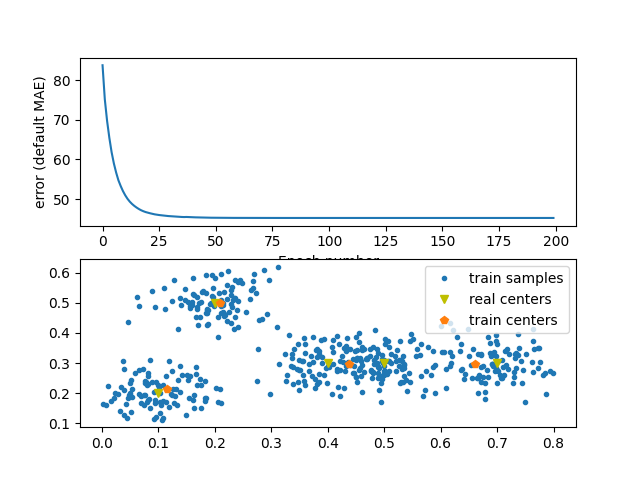


**Лістинг програми:**

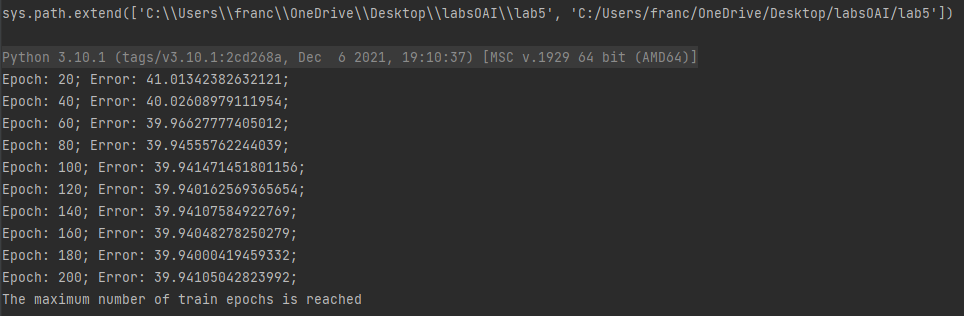
import numpy as np  
import neurolab as nl  
import numpy.random as rand  
import pylab as pl  
skv = 0.05  
centr = np.array([[0.1, 0.2], [0.4, 0.3], [0.7, 0.3], [0.2, 0.5],[0.5, 0.3]])  
rand\_norm = skv \* rand.randn(100, 5, 2)  
inp = np.array([centr + r for r in rand\_norm])  
inp.shape = (100 \* 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]], 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()

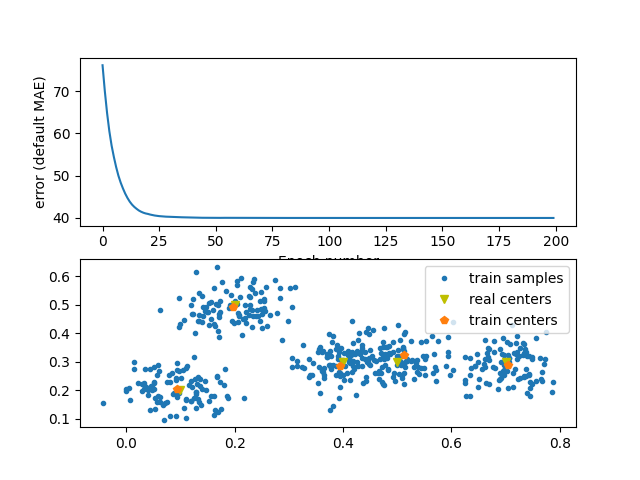
**Результат виконання:**

****

****

****

**Результат виконання:  
**

****

**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 – 39.94, при 5 кластерах і 4 нейронах – 45,20. Якщо порівнювать з попереднім завданням,то в цьому нейрона мережа відпрацювала краще,т.к., значення помилки значно зменшилось.