

					ДУ «Житомирська політехніка».20.121.12.				
Змн.	Арк.	№ докум.	Підпис	Дата					
Розроб.		Анкудевич Д.Р				Літ.	Арк.	Аркушів	
Перевір.		Філіпов В.О.						1	20
Керівник						ФІКТ Гр. ІПЗк-20-1			
Н. контр.									
Зав. каф.									

Завдання 2.2. Створити просту нейронну мережу для передбачення статі людини

```

1  import numpy as np
2  def sigmoid(x):
3      return 1 / (1 + np.exp(-x))
4  class Neuron:
5      def __init__(self, weights, bias):
6          self.weights = weights
7          self.bias = bias
8      def feedforward(self, inputs):
9          total = np.dot(self.weights, inputs) + self.bias
10         return sigmoid(total)
11 weights = np.array([0, 1]) # w1 = 0, w2 = 1
12 bias = 4 # b = 4
13 n = Neuron(weights, bias)
14 x = np.array([2, 3]) # x1 = 2, x2 = 3
15 class AnkudevichNeuralNetwork:
16     def __init__(self):
17         weights = np.array([0, 1])
18         bias = 0
19         # Клас Neuron із попереднього завдання
20         self.h1 = Neuron(weights, bias)
21         self.h2 = Neuron(weights, bias)
22         self.o1 = Neuron(weights, bias)
23     def feedforward(self, x):
24         out_h1 = self.h1.feedforward(x)
25         out_h2 = self.h2.feedforward(x)
26         # Входи для o1 є виходами h1 и h2
27         out_o1 = self.o1.feedforward(np.array([out_h1, out_h2]))
28         return out_o1
29 network = AnkudevichNeuralNetwork()
30 x = np.array([2, 3])
31 print(network.feedforward(x))

```

Рис 5.3 Код файлу LR_5_task_2_1.py

```

C:\Users\ankud\AppData\Local\Programs\Python\Python311\python.exe C:\Users\ankud\Desktop\myAi\lab5\LR_5_task_2_1.py
0.7216325609518421

```

Рис 5.4 Результат файлу LR_5_task_2_1.py

		Анкудевич Д.Р			ДУ «Житомирська політехніка».20.121.12 – Лр5	Арк.
		Філіпов В.О.				2
Змн.	Арк.	№ докум.	Підпис	Дата		

```

1 import numpy as np
2 def sigmoid(x):
3     # Функція активації sigmoid:  $f(x) = 1 / (1 + e^{(-x)})$ 
4     return 1 / (1 + np.exp(-x))
5 def deriv_sigmoid(x):
6     # Похідна від sigmoid:  $f'(x) = f(x) * (1 - f(x))$ 
7     fx = sigmoid(x)
8     return fx * (1 - fx)
9 def mse_loss(y_true, y_pred):
10    # y_true и y_pred є масивами numpy з однаковою довжиною
11    return ((y_true - y_pred) ** 2).mean()
12 class AnkudevichNeuralNetwork:
13     def __init__(self):
14         #Bara
15         self.w1 = np.random.normal()
16         self.w2 = np.random.normal()
17         self.w3 = np.random.normal()
18         self.w4 = np.random.normal()
19         self.w5 = np.random.normal()
20         self.w6 = np.random.normal()
21         #Зміщення
22         self.b1 = np.random.normal()
23         self.b2 = np.random.normal()
24         self.b3 = np.random.normal()
25     def feedforward(self, x):
26         # x є масивом numpy з двома елементами
27         h1 = sigmoid(self.w1 * x[0] + self.w2 * x[1] + self.b1)
28         h2 = sigmoid(self.w3 * x[0] + self.w4 * x[1] + self.b2)
29         o1 = sigmoid(self.w5 * h1 + self.w6 * h2 + self.b3)
30         return o1
31     def train(self, data, all_y_trues):
32         learn_rate = 0.1
33         epochs = 1000
34         for epoch in range(epochs):
35             for x, y_true in zip(data, all_y_trues):
36                 # --- Виконуємо зворотній зв'язок (ці значання нам потрібні в подальшому )
37                 sum_h1 = self.w1 * x[0] + self.w2 * x[1] + self.b1
38                 h1 = sigmoid(sum_h1)
39                 sum_h2 = self.w3 * x[0] + self.w4 * x[1] + self.b2

```

```

for x, y_true in zip(data, all_y_trues):
    # --- Виконуємо зворотній зв'язок (ці значання нам потрібні в подальшому )
    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
    # --- Найменування: d_L_d_w1 означає "частково L / частково w1"
    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
    # Підрахунок середньої помилки в кожній епоху

```

		Анкудевич. Д.Р.			ДУ «Житомирська політехніка».20.121.12 – Лр5	Арк.
		Філіпов В.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

```

5         if epoch % 10 == 0:
6             y_preds = np.apply_along_axis(self.feedforward, 1, data)
7             loss = mse_loss(all_y_trues, y_preds)
8             print("Epoch %d loss: %.3f" % (epoch, loss))
9             # Задання набору даних
10            data = np.array([
11                [-2, -1], # Alice
12                [25, 6], # Bob
13                [17, 4], # Charlie
14                [-15, -6], # Diana
15            ])
16
17            all_y_trues = np.array([
18                1, # Alice
19                0, # Bob
20                0, # Charlie
21                1, # Diana
22            ])
23
24            # Тренуємо вашу нейронну мережу!
25            network = AnkudevichNeuralNetwork()
26            network.train(data, all_y_trues)
27
28            # Робимо передбачення
29            emily = np.array([-7, -3]) # 128 фунтов, 63 дюйма
30            frank = np.array([20, 2]) # 155 фунтов, 68 дюймов
31            print("Emily: %.3f" % network.feedforward(emily)) # 0.951 - F
32            print("Frank: %.3f" % network.feedforward(frank)) # 0.039 - F

```

Рис 5.5 Код файлу LR_5_task_2_2.py

		Анкудевич. Д.Р			ДУ «Житомирська політехніка».20.121.12 – Лр5	Арк.
		Філіпов В.О.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

```
C:\Users\ankud\AppData\Local\Programs\Python\Python311\python.exe C:\Users\ankud\Desktop\myAi\lab5\LR_5_task_2_2.py
```

```
Epoch 0 loss: 0.351
Epoch 10 loss: 0.228
Epoch 20 loss: 0.148
Epoch 30 loss: 0.102
Epoch 40 loss: 0.074
Epoch 50 loss: 0.057
Epoch 60 loss: 0.045
Epoch 70 loss: 0.037
Epoch 80 loss: 0.031
Epoch 90 loss: 0.027
Epoch 100 loss: 0.024
Epoch 110 loss: 0.021
Epoch 120 loss: 0.019
Epoch 130 loss: 0.017
Epoch 140 loss: 0.015
Epoch 150 loss: 0.014
Epoch 160 loss: 0.013
Epoch 170 loss: 0.012
Epoch 180 loss: 0.011
Epoch 190 loss: 0.010
Epoch 200 loss: 0.010
Epoch 210 loss: 0.009
Epoch 220 loss: 0.009
Epoch 230 loss: 0.008
Epoch 240 loss: 0.008
Epoch 250 loss: 0.007
Epoch 260 loss: 0.007
Epoch 270 loss: 0.007
Epoch 280 loss: 0.007
Epoch 290 loss: 0.006
Epoch 300 loss: 0.006
Epoch 310 loss: 0.006
Epoch 320 loss: 0.006
Epoch 330 loss: 0.005
Epoch 340 loss: 0.005
Epoch 350 loss: 0.005
Epoch 360 loss: 0.005
Epoch 370 loss: 0.005
Epoch 380 loss: 0.005
Epoch 390 loss: 0.004
Epoch 390 loss: 0.005
Epoch 390 loss: 0.004
Epoch 400 loss: 0.004
Epoch 410 loss: 0.004
Epoch 420 loss: 0.004
Epoch 430 loss: 0.004
Epoch 440 loss: 0.004
Epoch 450 loss: 0.004
Epoch 460 loss: 0.004
Epoch 470 loss: 0.004
Epoch 480 loss: 0.003
Epoch 490 loss: 0.003
Epoch 500 loss: 0.003
Epoch 510 loss: 0.003
Epoch 520 loss: 0.003
Epoch 530 loss: 0.003
Epoch 540 loss: 0.003
Epoch 550 loss: 0.003
Epoch 560 loss: 0.003
Epoch 570 loss: 0.003
Epoch 580 loss: 0.003
Epoch 590 loss: 0.003
Epoch 600 loss: 0.003
Epoch 610 loss: 0.003
Epoch 620 loss: 0.003
Epoch 630 loss: 0.003
Epoch 640 loss: 0.002
Epoch 650 loss: 0.002
Epoch 660 loss: 0.002
Epoch 670 loss: 0.002
Epoch 680 loss: 0.002
Epoch 690 loss: 0.002
Epoch 700 loss: 0.002
Epoch 710 loss: 0.002
Epoch 720 loss: 0.002
Epoch 730 loss: 0.002
Epoch 740 loss: 0.002
Epoch 750 loss: 0.002
Epoch 760 loss: 0.002
Epoch 770 loss: 0.002
Epoch 780 loss: 0.002
```

		Анкудевич. Д.Р			ДУ «Житомирська політехніка».20.121.12 – Лр5	Арк.
		Філіпов В.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

```

Epoch 700 loss: 0.002
Epoch 710 loss: 0.002
Epoch 720 loss: 0.002
Epoch 730 loss: 0.002
Epoch 740 loss: 0.002
Epoch 750 loss: 0.002
Epoch 760 loss: 0.002
Epoch 770 loss: 0.002
Epoch 780 loss: 0.002
Epoch 790 loss: 0.002
Epoch 800 loss: 0.002
Epoch 810 loss: 0.002
Epoch 820 loss: 0.002
Epoch 830 loss: 0.002
Epoch 840 loss: 0.002
Epoch 850 loss: 0.002
Epoch 860 loss: 0.002
Epoch 870 loss: 0.002
Epoch 880 loss: 0.002
Epoch 890 loss: 0.002
Epoch 900 loss: 0.002
Epoch 910 loss: 0.002
Epoch 920 loss: 0.002
Epoch 930 loss: 0.002
Epoch 940 loss: 0.002
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: 0.966
Frank: 0.039

```

Рис 5.6 Результат файлу LR_5_task_2_2.py

		Анкудевич. Д.Р			ДУ «Житомирська політехніка».20.121.12 – Лр5	Арк.
		Філіпов В.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

Висновок: Функція активації, або передавальна функція штучного нейрона — залежність вихідного сигналу штучного нейрона від вхідного. Більшість видів нейронних мереж для функції активації використовують сигмоїди.

Можливості нейронних мереж прямого поширення полягають в тому, що сигнали поширюються в одному напрямку, починаючи від вхідного шару нейронів, через приховані шари до вихідного шару і на вихідних нейронах отримується результат опрацювання сигналу. В мережах такого виду немає зворотніх зв'язків.

Нейронні мережі прямого поширення знаходять своє застосування в задачах комп'ютерного бачення та розпізнаванні мовлення, де класифікація цільових класів ускладнюється. Такі типи нейронних мереж добре справляються із зашумленими даними.

Завдання 2.3. Класифікатор на основі перцептрону з використанням бібліотеки NeuroLab

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 import neurolab as nl
4 # Завантаження вхідних даних
5 text = np.loadtxt('data_perceptron.txt')
6 # Поділ точок даних та міток
7 data = text[:, :2]
8 labels = text[:, 2].reshape((text.shape[0], 1))
9 # Побудова графіка вхідних даних
10 plt.figure()
11 plt.scatter(data[:, 0], data[:, 1])
12 plt.xlabel('Розмірність 1')
13 plt.ylabel('Розмірність 2')
14 plt.title('Вхідні дані')
15 # Визначення максимального та мінімального значень для кожного виміру
16 dim1_min, dim1_max, dim2_min, dim2_max = 0, 1, 0, 1
17 # Кількість нейронів у вихідному шарі
18 num_output = labels.shape[1]
19 # Визначення перцептрону з двома вхідними нейронами (оскільки
20 # Вхідні дані - двовимірні)
21 dim1 = [dim1_min, dim1_max]
22 dim2 = [dim2_min, dim2_max]
23 perceptron = nl.net.newp([dim1, dim2], num_output)
24 # Тренування перцептрону з використанням наших даних
25 error_progress = perceptron.train(data, labels, epochs = 100, show = 20, lr = 0.03)
26 # Побудова графіка процесу навчання
27 plt.figure()
28 plt.plot(error_progress)
29 plt.xlabel('Кількість епох')
30 plt.ylabel('Помилка навчання')
31 plt.title('Зміна помилок навчання')
32 plt.grid()
33 plt.show()

```

Рис 5.7 Код файлу LR_5_task_3.py

		Анкудавич. Д.Р			ДУ «Житомирська політехніка».20.121.12 – Лр5	Арк.
		Філіпов В.О.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

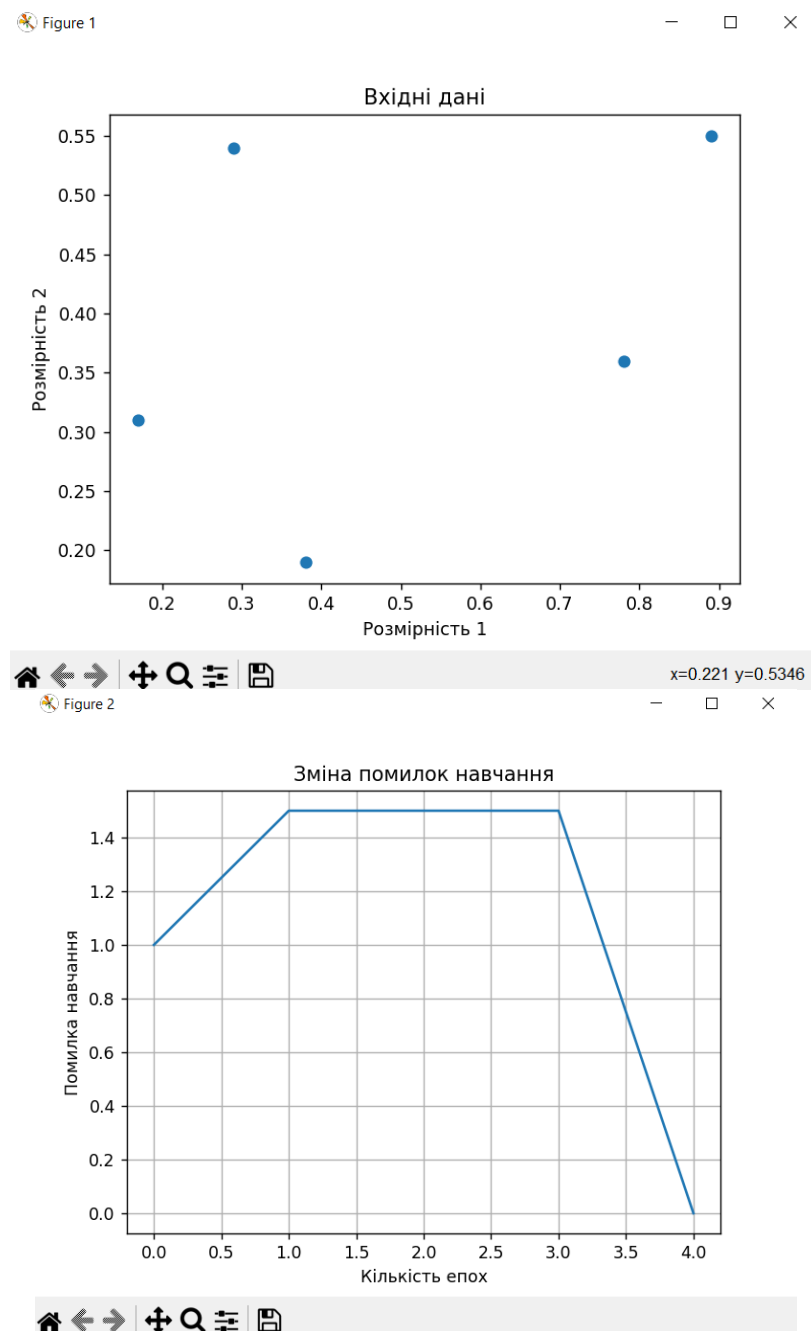


Рис 5.8 Результат файлу LR_5_task_3.py

Завдання 2.4. Побудова одношарової нейронної мережі

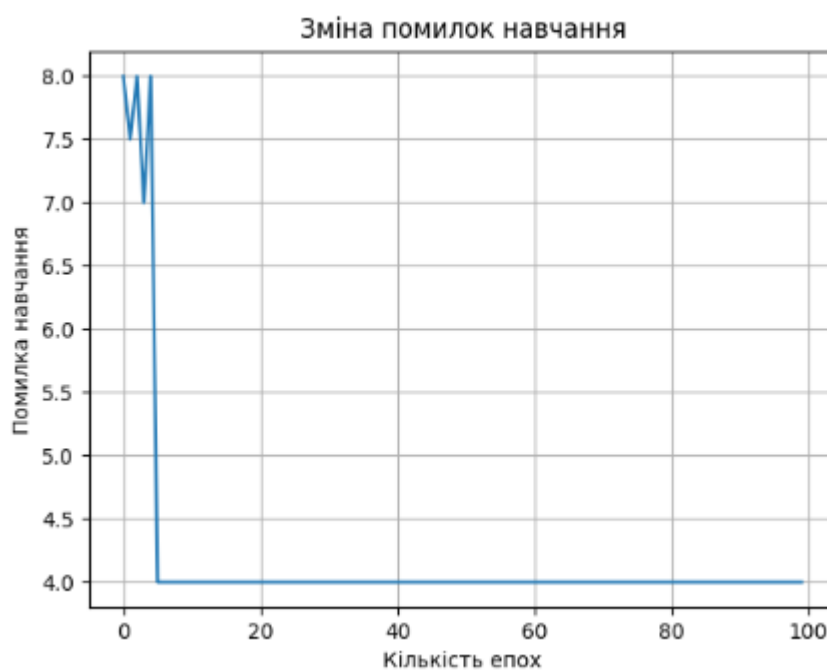
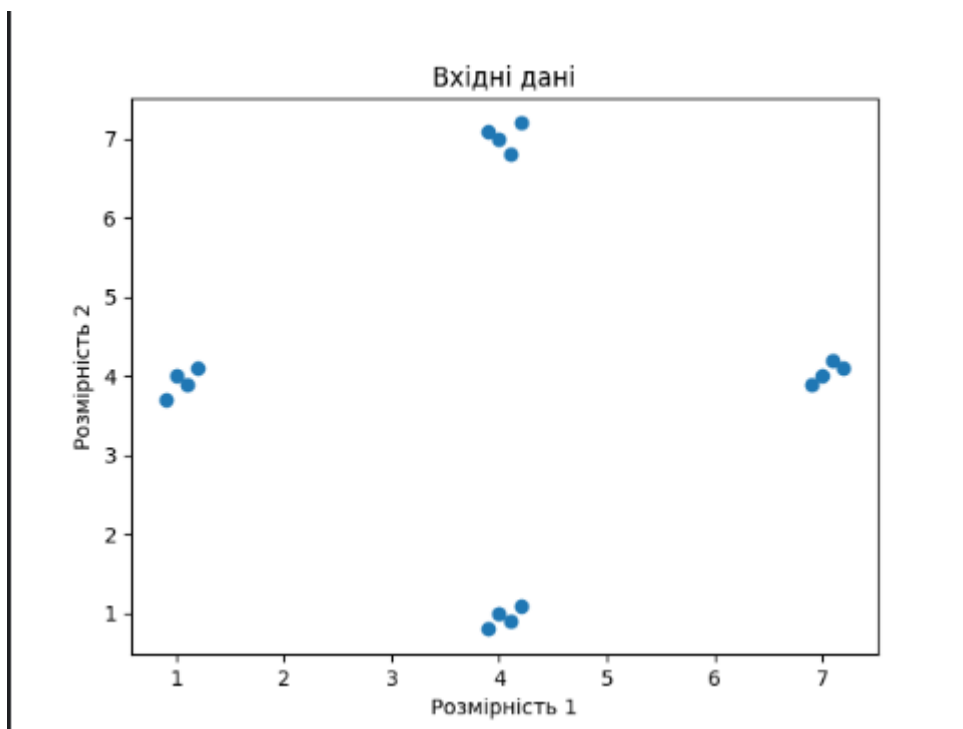
```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 import neurolab as nl
4 text = np.loadtxt('data_simple_nn.txt')
5 data = text[:, 0:2]
6 labels = text[:, 2:]
7 plt.figure()
8 plt.scatter(data[:, 0], data[:, 1])
9 plt.xlabel('Розмірність 1')
10 plt.ylabel('Розмірність 2')
11 plt.title('Вхідні дані')
12 dim1_min, dim1_max = data[:, 0].min(), data[:, 0].max()
13 dim2_min, dim2_max = data[:, 1].min(), data[:, 1].max()
14 num_output = labels.shape[1]
15 dim1 = [dim1_min, dim1_max]
16 dim2 = [dim2_min, dim2_max]
17 nn = nl.net.newp([dim1, dim2], num_output)
18 error_progress = nn.train(data, labels, epochs = 100, show = 20, lr = 0.03)
19 plt.figure()
20 plt.plot(error_progress)
21 plt.xlabel('Кількість епох')
22 plt.ylabel('Помилка навчання')
23 plt.title('Зміна помилок навчання')
24 plt.grid()
25 plt.show()
26 print('\nTest results:')
27 data_test = [[0.4, 4.3], [4.4, 0.6], [4.7, 8.1]]
28 for item in data_test:
29     print(item, '-->', nn.sim([item])[0])

```

Рис 5.9 Код файлу LR_5_task_4.py

		Анкудевич. Д.Р			ДУ «Житомирська політехніка».20.121.12 – Лр5	Арк.
		Філіпов В.О.				10
Змн.	Арк.	№ докум.	Підпис	Дата		



```

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.]

Process finished with exit code 0

```

Рис 5.10 Результат файлу LR_5_task_4.py

Висновок: На рис. 20 зображено процес навчання мережі. На 20 епосі відбулось 4 помилки, аналогічно на 40, 60, 80 та 100. Потім вивелось повідомлення, що ми досягли максимальної кількості епох для тренування. Ми вирішили визначити вибірккові тестові точки даних та запустили для них нейронну мережу. І це його результат.

		Анкудевич. Д.Р			ДУ «Житомирська політехніка».20.121.12 – Лр5	Арк.
		Філіпов В.О.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

Завдання 2.5. Побудова багат шарової нейронної мережі

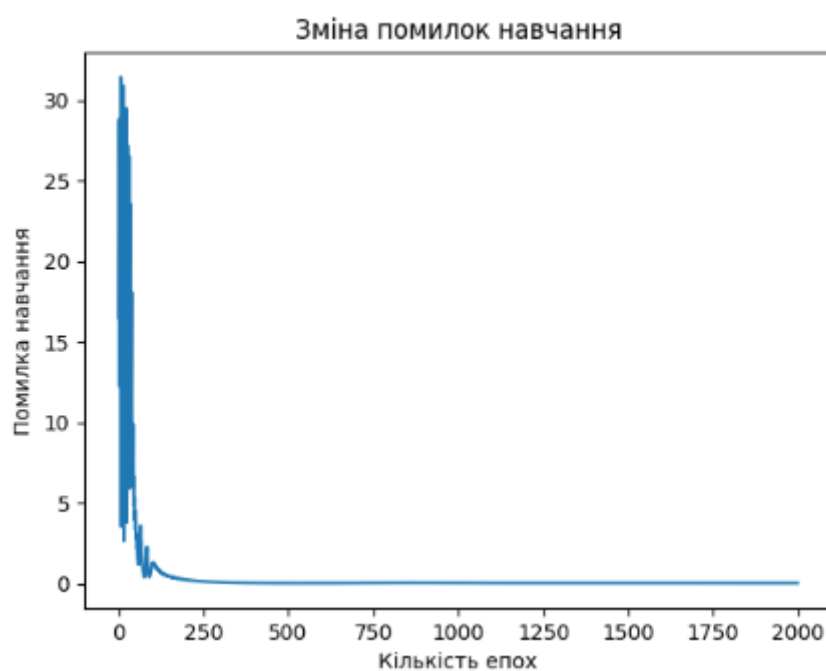
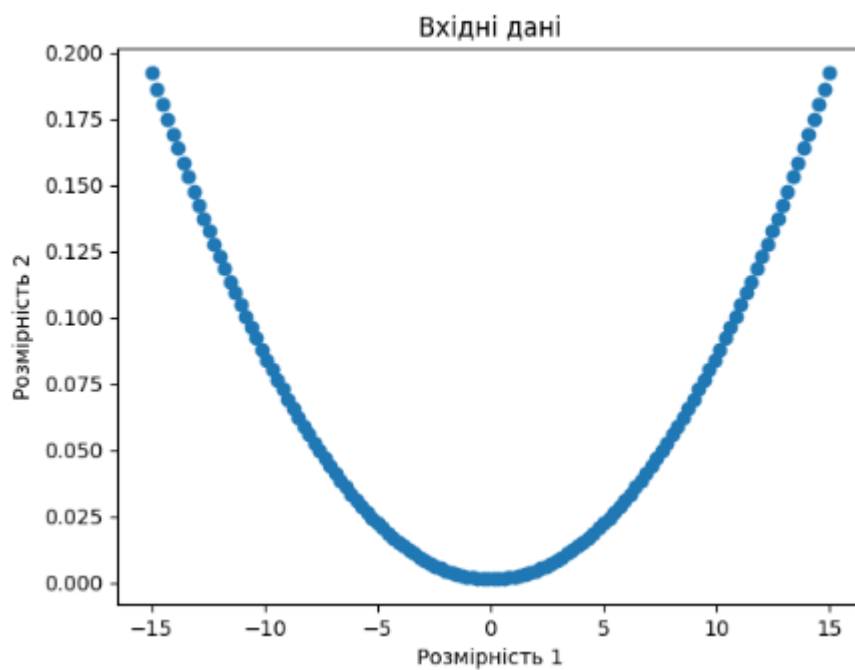
```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 import neurolab as nl
4 min_val = -15
5 max_val = 15
6 num_points = 130
7 x = np.linspace(min_val, max_val, num_points)
8 y = 3 * np.square(x) + 5
9 y /= np.linalg.norm(y)
10 data = x.reshape(num_points, 1)
11 labels = y.reshape(num_points, 1)
12 plt.figure()
13 plt.scatter(data, labels)
14 plt.xlabel('Розмірність 1')
15 plt.ylabel('Розмірність 2')
16 plt.title('Вхідні дані')
17 nn = nl.net.newff([[min_val, max_val]], [10, 6, 1])
18 nn.trainf = nl.train.train_gd
19 error_progress = nn.train(data, labels, epochs=2000, show = 100, goal = 0.01)
20 output = nn.sim(data)
21 y_pred = output.reshape(num_points)
22 plt.figure()
23 plt.plot(error_progress)
24 plt.xlabel('Кількість епох')
25 plt.ylabel('Помилка навчання')
26 plt.title('Зміна помилок навчання')
27 x_dense = np.linspace(min_val, max_val, num_points * 2)
28 y_dense_pred = nn.sim(x_dense.reshape(x_dense.size, 1)).reshape(x_dense.size)
29 plt.figure()
30 plt.plot(x_dense, y_dense_pred, '-', x, y, '.', x, y_pred, 'p')
31 plt.title('Фактичні і прогнозовані значення')
32 plt.show()

```

Рис 5.11 Код файлу LR_5_task_5.py

		Анкудевич. Д.Р			ДУ «Житомирська політехніка».20.121.12 – Лр5	Арк.
		Філіпов В.О.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

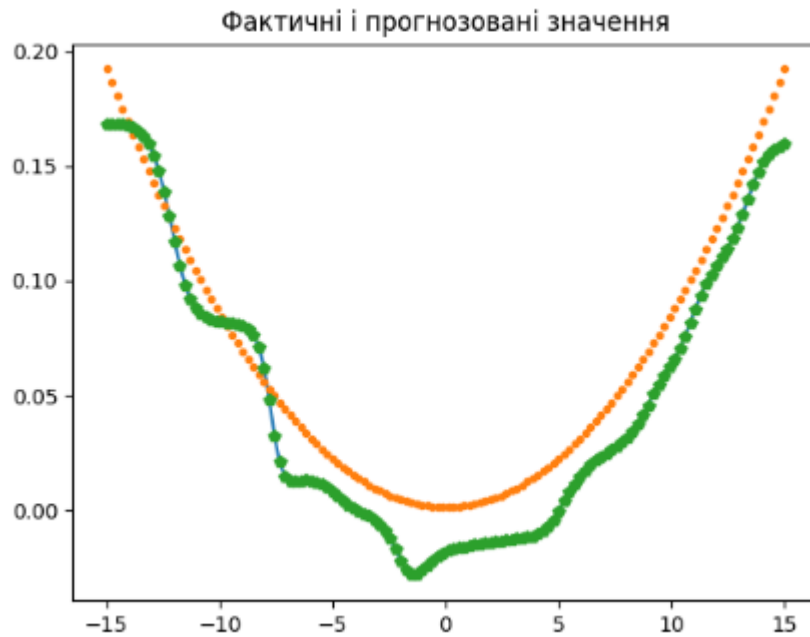


		Анкудевич. Д.Р		
		Філіпов В.О.		
Змн.	Арк.	№ докум.	Підпис	Дата

ДУ «Житомирська політехніка».20.121.12 – Лр5

Арк.

14



```
C:\Users\ankud\AppData\Local\Programs\Python\Python311\python.exe C:\Users\ankud\Desktop\myAI\Lab5
Epoch: 100; Error: 1.138713012257457;
Epoch: 200; Error: 0.22890211316269424;
Epoch: 300; Error: 0.07298904656459555;
Epoch: 400; Error: 0.03153138743658987;
Epoch: 500; Error: 0.02085697044230702;
Epoch: 600; Error: 0.020750850928761437;
Epoch: 700; Error: 0.027451640430873385;
Epoch: 800; Error: 0.03629963649343601;
Epoch: 900; Error: 0.03748428003816216;
Epoch: 1000; Error: 0.03235993852915879;
Epoch: 1100; Error: 0.027742895870706823;
Epoch: 1200; Error: 0.025681209696639663;
Epoch: 1300; Error: 0.025611243974995646;
Epoch: 1400; Error: 0.02630749494902149;
Epoch: 1500; Error: 0.026606035985796493;
Epoch: 1600; Error: 0.026082990747648013;
Epoch: 1700; Error: 0.02510764351267697;
Epoch: 1800; Error: 0.02421838317736591;
Epoch: 1900; Error: 0.023678702975998986;
Epoch: 2000; Error: 0.023454064657700807;
The maximum number of train epochs is reached
```

Рис 5.12 Результат файлу LR_5_task_5.py

Висновок: На рис. 12 зображено процес навчання мережі. Відносно кожної епосі відбувались помилки. На 100 0.83 помилки. На 1200 0.06. Потім вивелось повідомлення, що ми досягли цілі навчання.

		Анкудевич Д.Р			ДУ «Житомирська політехніка».20.121.12 – Лр5	Арк.
		Філіпов В.О.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

Завдання 2.6. Побудова багатошарової нейронної мережі для свого варіанту

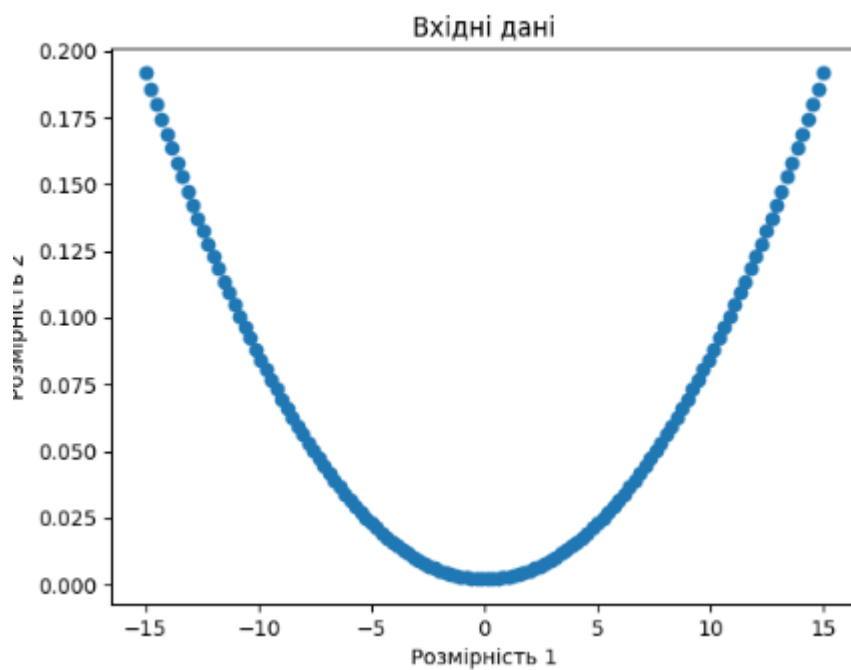
Варіант 1	$y = 2x^2 + 5$	Варіант 16	$y = 5x^2 + 7$
	1	2	3-1

```

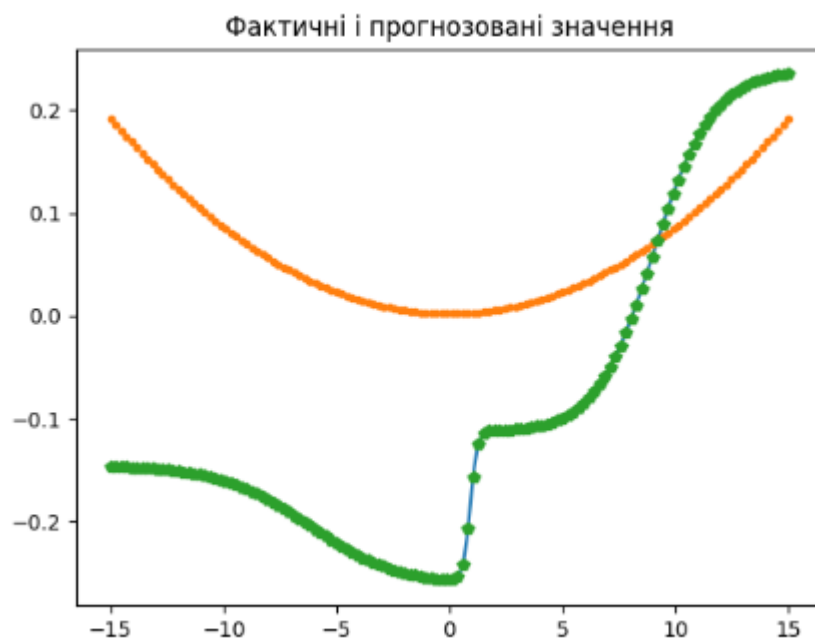
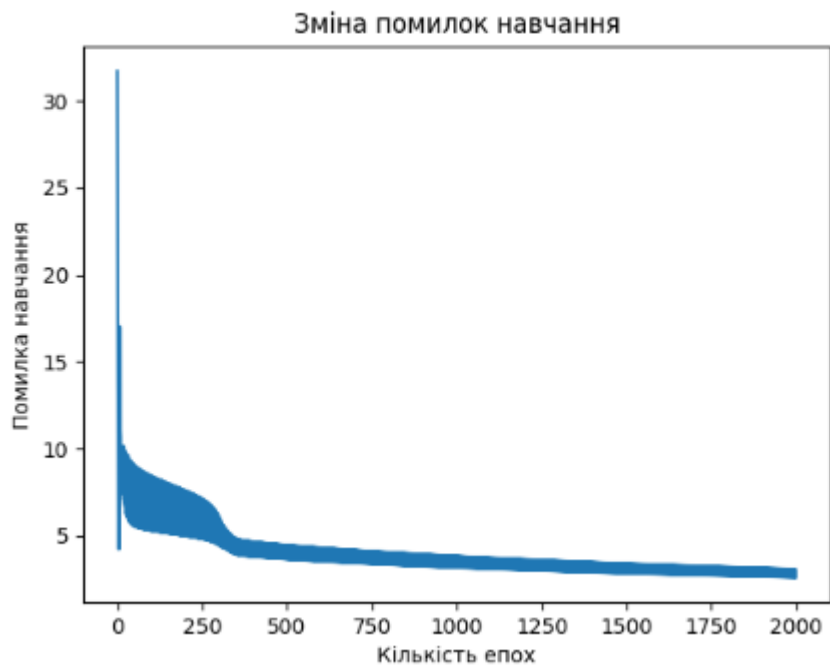
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import neurolab as nl
4 min_val = -15
5 max_val = 15
6 num_points = 130
7 x = np.linspace(min_val, max_val, num_points)
8 y = 5 * x * x + 3
9 y /= np.linalg.norm(y)
10 data = x.reshape(num_points, 1)
11 labels = y.reshape(num_points, 1)
12 plt.figure()
13 plt.scatter(data, labels)
14 plt.xlabel('Розмірність 1')
15 plt.ylabel('Розмірність 2')
16 plt.title('Вхідні дані')
17 nn = nl.net.newff([[min_val, max_val]], [3, 4, 1])
18 nn.trainf = nl.train.train_gd
19 error_progress = nn.train(data, labels, epochs=2000, show=100, goal=0.01)
20 output = nn.sim(data)
21 y_pred = output.reshape(num_points)
22 plt.figure()
23 plt.plot(error_progress)
24 plt.xlabel('Кількість епох')
25 plt.ylabel('Помилка навчання')
26 plt.title('Зміна помилок навчання')
27 x_dense = np.linspace(min_val, max_val, num_points * 2)
28 y_dense_pred = nn.sim(x_dense.reshape(x_dense.size, 1)).reshape(x_dense.size)
29 plt.figure()
30 plt.plot(x_dense, y_dense_pred, '-', x, y, '.', x, y_pred, 'p')
31 plt.title('Фактичні і прогнозовані значення')

```

Рис 5.13 Код файлу LR_5_task_6.py



		Анкудевич. Д.Р			ДУ «Житомирська політехніка».20.121.12 – Лр5	Арк.
		Філіпов В.О.				17
Змн.	Арк.	№ докум.	Підпис	Дата		



		Анкудевич. Д.Р		
		Філіпов В.О.		
Змн.	Арк.	№ докум.	Підпис	Дата

```
C:\Users\ankud\AppData\Local\Programs\Python\Python311\python.exe C:\Users\ankud\Desktop\myAi\lab5\LR_5_task_6.py
Epoch: 100; Error: 0.5785067796460632;
Epoch: 200; Error: 32.296050000611714;
Epoch: 300; Error: 10.403601196027466;
Epoch: 400; Error: 10.672026309603872;
Epoch: 500; Error: 29.667397258941065;
Epoch: 600; Error: 0.543219376895684;
Epoch: 700; Error: 31.690818991288282;
Epoch: 800; Error: 7.174695231303939;
Epoch: 900; Error: 4.696149548340339;
Epoch: 1000; Error: 12.220285120064156;
Epoch: 1100; Error: 0.07366823720878896;
Epoch: 1200; Error: 0.23184551755960314;
Epoch: 1300; Error: 0.23618396781068313;
Epoch: 1400; Error: 0.2129780575130564;
Epoch: 1500; Error: 0.19347879416365665;
Epoch: 1600; Error: 0.17918185146695553;
Epoch: 1700; Error: 0.16931290106974972;
Epoch: 1800; Error: 0.16283768951551988;
Epoch: 1900; Error: 0.15885970791688658;
Epoch: 2000; Error: 0.15667054558201376;
The maximum number of train epochs is reached
```

Рис 5.14 Результат файлу LR_5_task_6.py

На рис. 14 зображено процес навчання мережі. На 100 епосі відбулось 0.75 помилки, на 300 епосі відбулось 0.21 помилки, і так далі, на 2000 епосі відбулось 0.03 помилки,. Потім вивелось повідомлення, що ми досягли максимальної кількості епох для тренування.

		Анкудевич. Д.Р			ДУ «Житомирська політехніка».20.121.12 – Лр5	Арк.
		Філіпов В.О.				19
Змн.	Арк.	№ докум.	Підпис	Дата		

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

```

1 import numpy as np
2 import neurolab as nl
3 import numpy.random as rand
4 skv = 0.05
5 centr = np.array([[0.2, 0.2], [0.4, 0.4], [0.7, 0.3], [0.2, 0.5]])
6 rand_norm = skv * rand.randn(100, 4, 2)
7 inp = np.array([centr + r for r in rand_norm])
8 inp.shape = (100 * 4, 2)
9 rand.shuffle(inp)
10 # Create net with 2 inputs and 4 neurons
11 net = nl.net.newc([[0.0, 1.0], [0.0, 1.0]], 4)
12 # train with rule: Conscience Winner Take All algorithm (CWTA)
13 error = net.train(inp, epochs=200, show=100)
14 # Plot results:
15 import pylab as pl
16 pl.title('Classification Problem')
17 pl.subplot(211)
18 pl.plot(error)
19 pl.xlabel('Epoch number')
20 pl.ylabel('error (default MAE)')
21 w = net.layers[0].np['w']
22 pl.subplot(212)
23 pl.plot(inp[:, 0], inp[:, 1], '.', '\
24         centr[:, 0], centr[:, 1], 'yv', '\
25         w[:, 0], w[:, 1], 'p')
26 pl.legend(['train samples', 'real centers', 'train centers'])
27 pl.show()

```

Рис 5.15 Код файлу LR_5_task_7.py

		Анкудевич. Д.Р			ДУ «Житомирська політехніка».20.121.12 – Лр5	Арк.
		Філіпов В.О.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

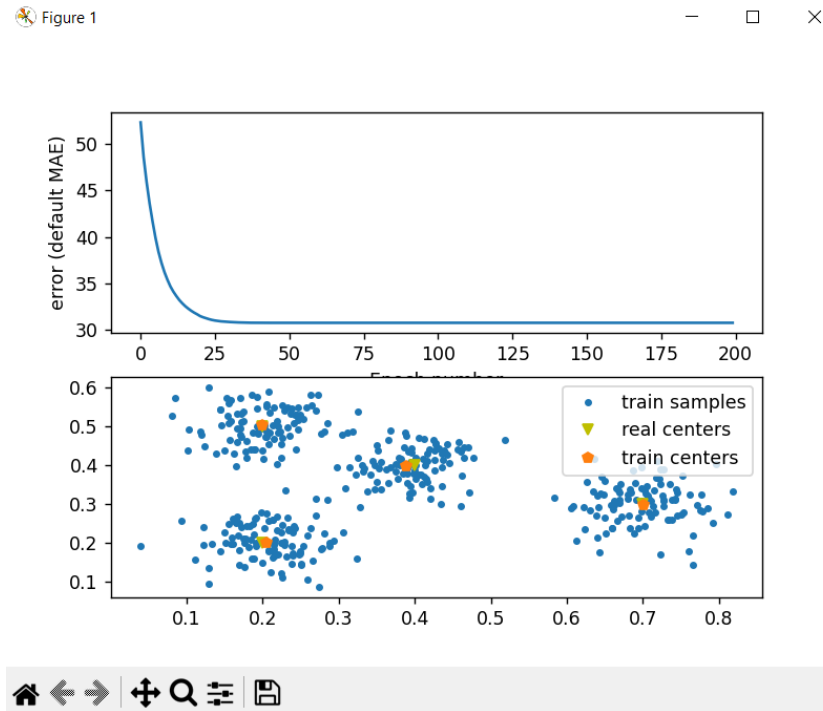


Рис 5.16 Результат файлу LR_5_task_7.py

Помилка MAE - Середня абсолютна помилка (Mean Absolute Error). Середньою абсолютною похибкою називають середнє арифметичне з абсолютних похибок усіх вимірювань.

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

Варіант 1	[0.2, 0.2], [0.4, 0.4], [0.7, 0.3], [0.2, 0.5], [0.5, 0.5]	0,03
Варіант 12	[0.3, 0.3], [0.5, 0.4], [0.7, 0.3], [0.2, 0.4], [0.6, 0.5]	0,05

```

1 import numpy as np
2 import neurolab as nl
3 import numpy.random as rand
4 skv = 0.05
5 centr = np.array([[0.2, 0.2], [0.4, 0.4], [0.7, 0.3], [0.2, 0.5]])
6 rand_norm = skv * rand.randn(100, 4, 2)
7 inp = np.array([centr + r for r in rand_norm])
8 inp.shape = (100 * 4, 2)
9 rand.shuffle(inp)
10 # Create net with 2 inputs and 4 neurons
11 net = nl.net.newc([[0.0, 1.0], [0.0, 1.0]], 4)
12 # train with rule: Conscience Winner Take All algorithm (CWTA)
13 error = net.train(inp, epochs=200, show=100)
14 # Plot results:
15 import pylab as pl
16 pl.title('Classification Problem')
17 pl.subplot(211)
18 pl.plot(error)
19 pl.xlabel('Epoch number')
20 pl.ylabel('error (default MAE)')
21 w = net.layers[0].np['w']
22 pl.subplot(212)
23 pl.plot(inp[:,0], inp[:,1], '.', \
24         centr[:,0], centr[:,1], 'yv', \
25         w[:,0], w[:,1], 'p')
26 pl.legend(['train samples', 'real centers', 'train centers'])
27 pl.show()

```

Рис 5.17 Код файлу LR_5_task_8.py

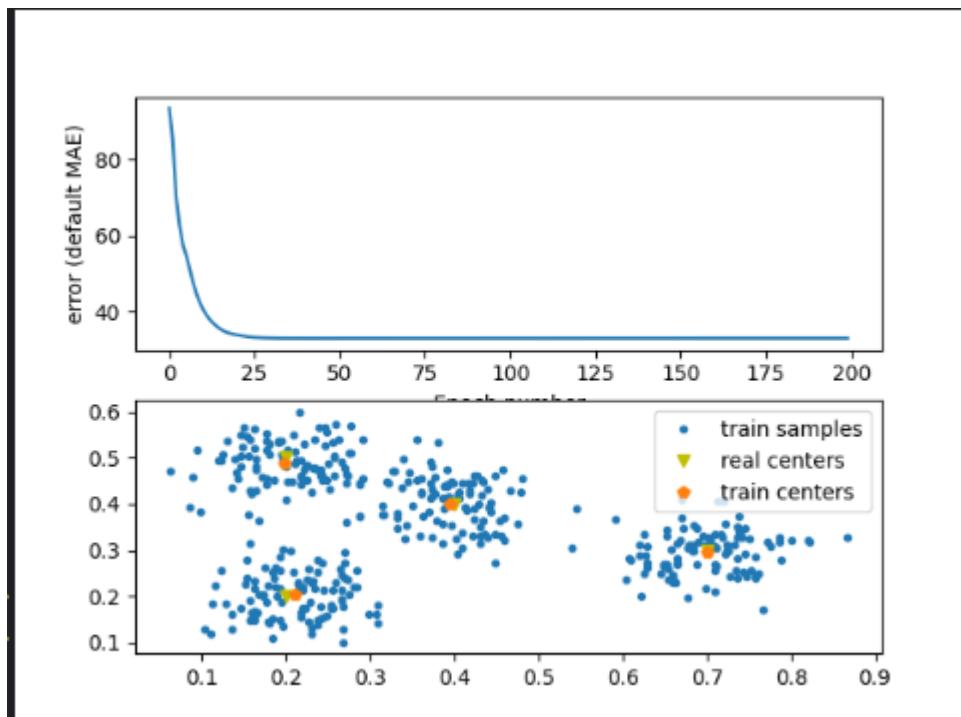


Рис 5.18 Результат файлу LR_5_task_8.py

На рис. 18 зображено процес навчання мережі. На 20 епосі відбулось 44.36 помилки, помилки і так далі, на 200 епосі відбулось 39.01 помилки,. Потім вивелось повідомлення, що ми досягли максимальної кількості епох для тренування.

Рис 5.19 Результат файлу LR_5_task_8.py

Якщо порівнювати нейронну мережу Кохонена з 4 нейронами та 5 нейронами, можна зробити такі висновки. При 4 нейронах Помилка MAE повільніше зменшується, ніж з 5 нейронами, також з 5 нейронами ця помилка нижча. З 5 нейронами обоє центрів збігаються майже в одні точці. Число нейронів в шарі Кохонена має відповідати числу класів вхідних сигналів. Тобто в нашому випадку нам давалось 5 вхідних сигналів, значить у нас має бути 5 нейронів, а не 4.Отже, невірний вибір кількості нейронів числу кластерів впливає на величину помилки ускладнюючи навчання мережі і швидкості, тому на рис. 18 набагато гірші результати, ніж на рис. 19.

Висновок: на лабораторній роботі я навчився використовувати спеціалізовані бібліотеки та мову програмування Python, навчився створювати та застосовувати прості нейронні мережі.

Посилання на гітхаб: <https://github.com/AnkuNoName/ai>

		Анкудевич. Д.Р			ДУ «Житомирська політехніка».20.121.12 – Лр5	Арк.
		Філіпов В.О.				24
Змн.	Арк.	№ докум.	Підпис	Дата		