Кушер Дмитро Євгенович ЗІПЗк-22-1
Лабораторна №5

Встановив бібліотеку Neurolab

```
C:\Users\Onibi>pip3 install neurolab
Collecting neurolab
  Downloading neurolab-0.3.5.tar.gz (645 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 645.3/645.3 kB 3.1 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Building wheels for collected packages: neurolab
  Building wheel for neurolab (pyproject.toml) ... done
  Created wheel for neurolab: filename=neurolab-0.3.5-py3-none-any.whl size=22200 sha256=fe98553dad594aab87976f52cf0938e
ca64271b046f6b6bfe858f8ce63a20401
  Stored in directory: c:\users\onibi\appdata\local\pip\cache\wheels\5e\ee\92\6e99c58786234fd536e400ac1f98af9cf9b43ee4ac
8fec4204
Successfully built neurolab
Installing collected packages: neurolab
Successfully installed neurolab-0.3.5

C:\Users\Onibi>
```

Завдання 2.1

Створити простий нейрон

```python
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)
weights = np.array([0, 1])
bias = 4  # b = 4
n = Neuron(weights, bias)
x = np.array([2, 3])
print(n.feedforward(x))
```

```
C:\Users\Onibi\AppDat
0.9990889488055994
```

Завдання 2.2

Створити просту нейронну мережу для передбачення статі людини

```python
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)
weights = np.array([0, 1])
bias = 4  # b = 4
n = Neuron(weights, bias)
x = np.array([2, 3])
print(n.feedforward(x))
class PolonevychNeuralNetwork:
    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 = PolonevychNeuralNetwork()
x = np.array([2, 3])
print(network.feedforward(x))  # 0.7216325609518421
```

```
C:\Users\Onibi\AppData
0.9990889488055994
0.7216325609518421
```

V2

```
import numpy as np
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
def deriv_sigmoid(x):
    fx = sigmoid(x)
    return fx * (1 - fx)
def mse_loss(y_true, y_pred):
    return ((y_true - y_pred) ** 2).mean()
class PolonevychNeuralNetwork:
    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)
                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)
                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)
                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)
                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
                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
                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))
data = np.array([
    [-2, -1],
    [25, 6],
```

```
    [17, 4],
    [-15, -6],
])
all_y_trues = np.array([
    1,
    0,
    0,
    1,
])
network = PolonevychNeuralNetwork()
network.train(data, all_y_trues)
emily = np.array([-7, -3])
frank = np.array([20, 2])
print("Emily: %.3f" % network.feedforward(emily))
print("Frank: %.3f" % network.feedforward(frank))
```

```
C:\Users\Onibi\AppData\Loc
Epoch 0 loss: 0.493
Epoch 10 loss: 0.482
Epoch 20 loss: 0.457
Epoch 30 loss: 0.434
Epoch 40 loss: 0.389
Epoch 50 loss: 0.297
Epoch 60 loss: 0.177
Epoch 70 loss: 0.109
Epoch 80 loss: 0.077
Epoch 90 loss: 0.058
Epoch 100 loss: 0.046
Epoch 110 loss: 0.038
Epoch 120 loss: 0.032
Epoch 130 loss: 0.027
Epoch 140 loss: 0.024
Epoch 150 loss: 0.021
Epoch 160 loss: 0.019
Epoch 170 loss: 0.017
Epoch 180 loss: 0.015
Epoch 190 loss: 0.014
Epoch 200 loss: 0.013
Epoch 210 loss: 0.012
Epoch 220 loss: 0.011
Epoch 230 loss: 0.010
Epoch 240 loss: 0.010
Epoch 250 loss: 0.009
Epoch 260 loss: 0.009
Epoch 270 loss: 0.008
Epoch 280 loss: 0.008
Epoch 290 loss: 0.007
Epoch 300 loss: 0.007
Epoch 310 loss: 0.007
Epoch 320 loss: 0.007
Epoch 330 loss: 0.006
Epoch 340 loss: 0.006
Epoch 350 loss: 0.006
Epoch 360 loss: 0.006
Epoch 370 loss: 0.005
```

```
Epoch 380 loss: 0.005
Epoch 390 loss: 0.005
Epoch 400 loss: 0.005
Epoch 410 loss: 0.005
Epoch 420 loss: 0.005
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: 0.004
Epoch 490 loss: 0.004
Epoch 500 loss: 0.004
Epoch 510 loss: 0.004
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: 0.003
Epoch 650 loss: 0.003
Epoch 660 loss: 0.003
Epoch 670 loss: 0.003
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
```

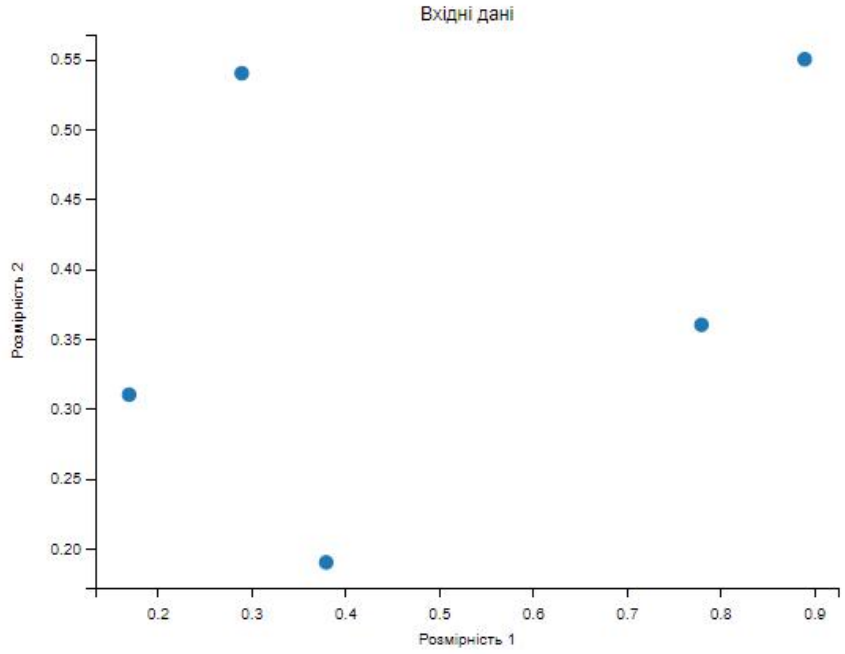
```
Epoch 770 loss: 0.002
Epoch 780 loss: 0.002
Epoch 790 loss: 0.002
Epoch 800 loss: 0.002
Epoch 810 loss: 0.002
Epoch 820 loss: 0.002
Epoch 830 loss: 0.002
Epoch 840 loss: 0.002
Epoch 850 loss: 0.002
Epoch 860 loss: 0.002
Epoch 870 loss: 0.002
Epoch 880 loss: 0.002
Epoch 890 loss: 0.002
Epoch 900 loss: 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: 0.965
Frank: 0.040
```

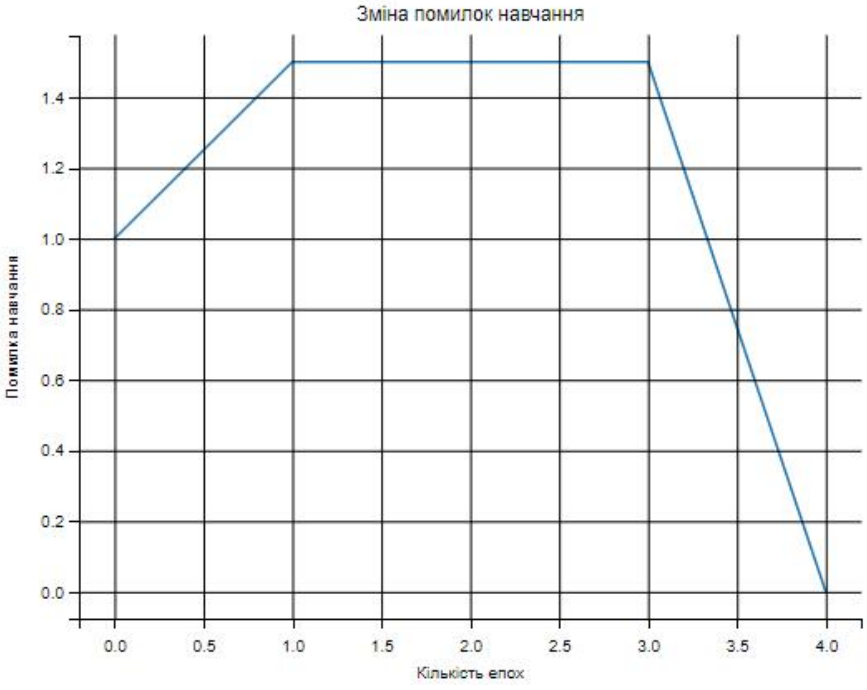
Завдання 2.3

Класифікатор на основі перцептрону з використанням бібліотеки NeuroLab

```
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')
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 помилок навчання')
plt.grid()
plt.show()
```





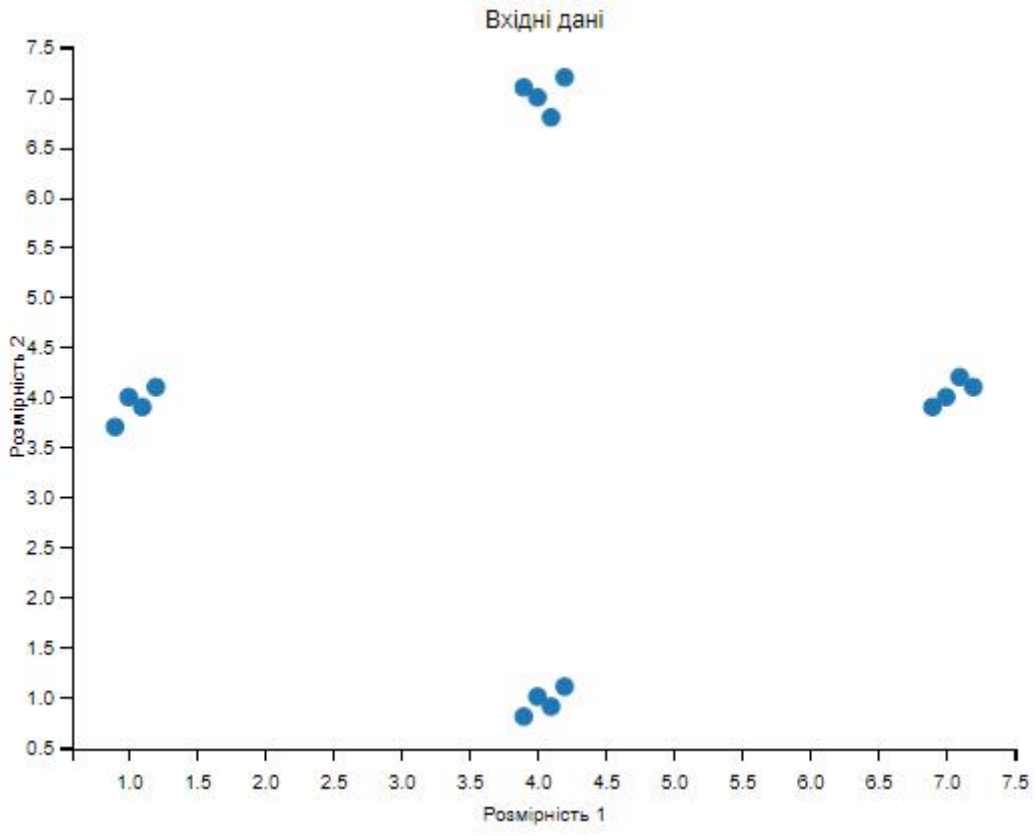
Завдання 2.4

Побудова одношарової нейронної мережі

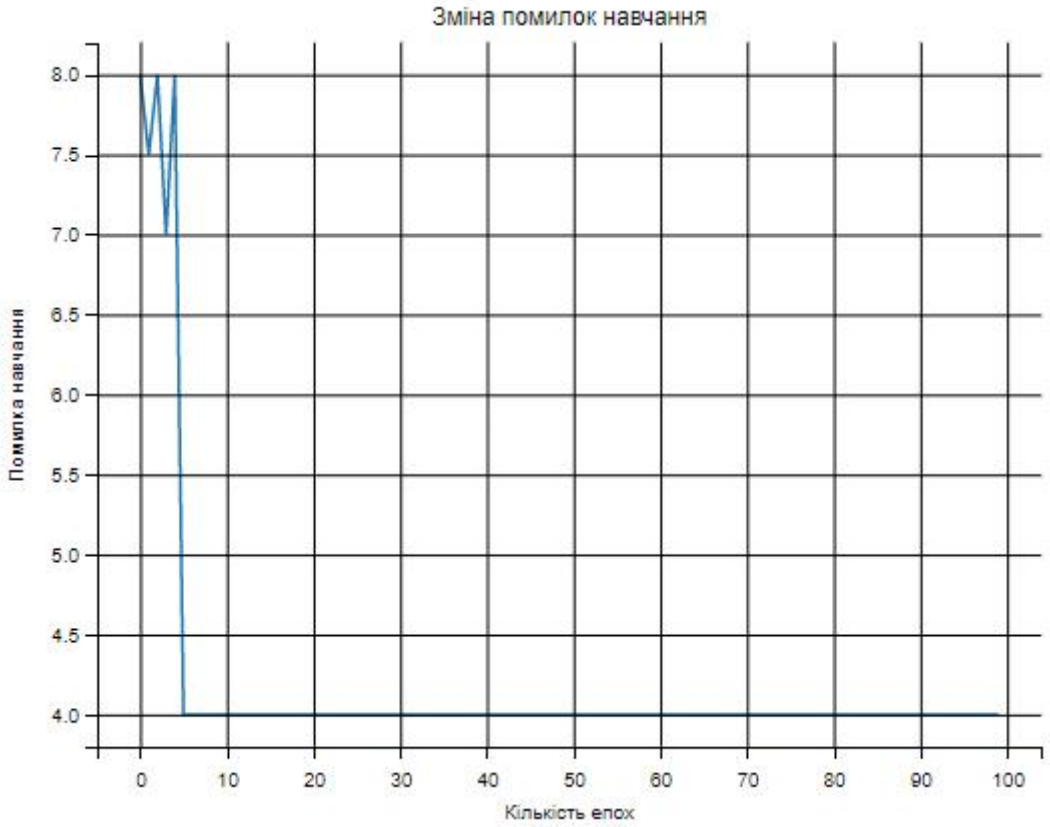
```
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl
text = np.loadtxt('data_simple_nn.txt')
data = text[:, 0:2]
```

```
labels = text[:, 2:]
plt.figure()
plt.scatter(data[:, 0], data[:, 1])
plt.xlabel('Розмірність 1')
plt.ylabel('Розмірність 2')
plt.title('Вхідні дані')
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)
plt.figure()
plt.plot(error_progress)
plt.xlabel('Кількість епох')
plt.ylabel('Помилка навчання')
plt.title('Зміна помилок 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])
```

Графік вхідних даних



Графік просування процесу навчання



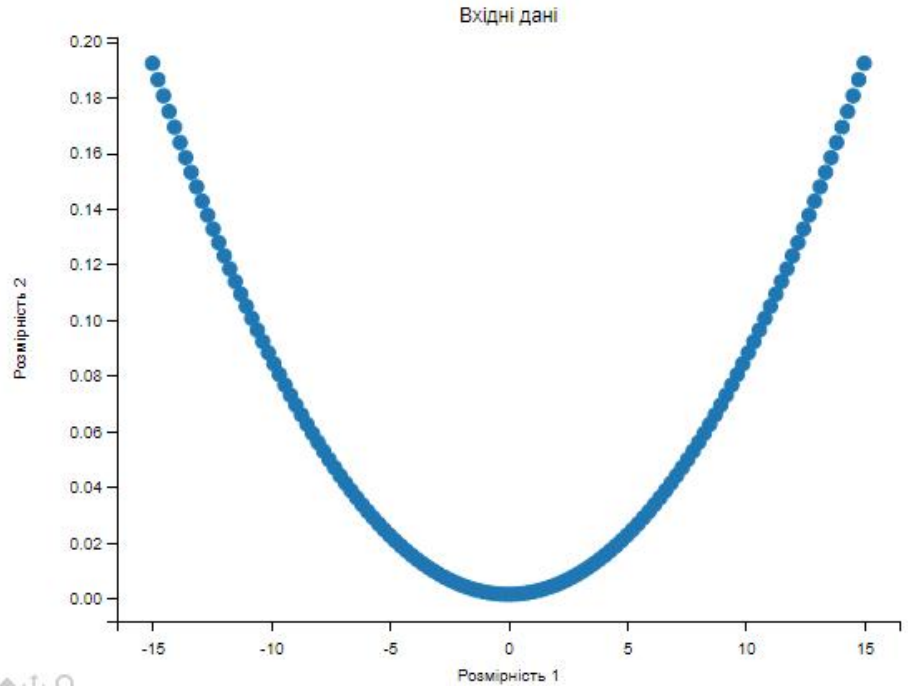

```
C:\Users\Onibi\AppData\Local\Programs\Python\Py
Epoch: 20; Error: 4.0;
Epoch: 40; Error: 4.0;
Epoch: 60; Error: 4.0;
Epoch: 80; Error: 4.0;
Epoch: 100; Error: 4.0;
The maximum number of train epochs is reached

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

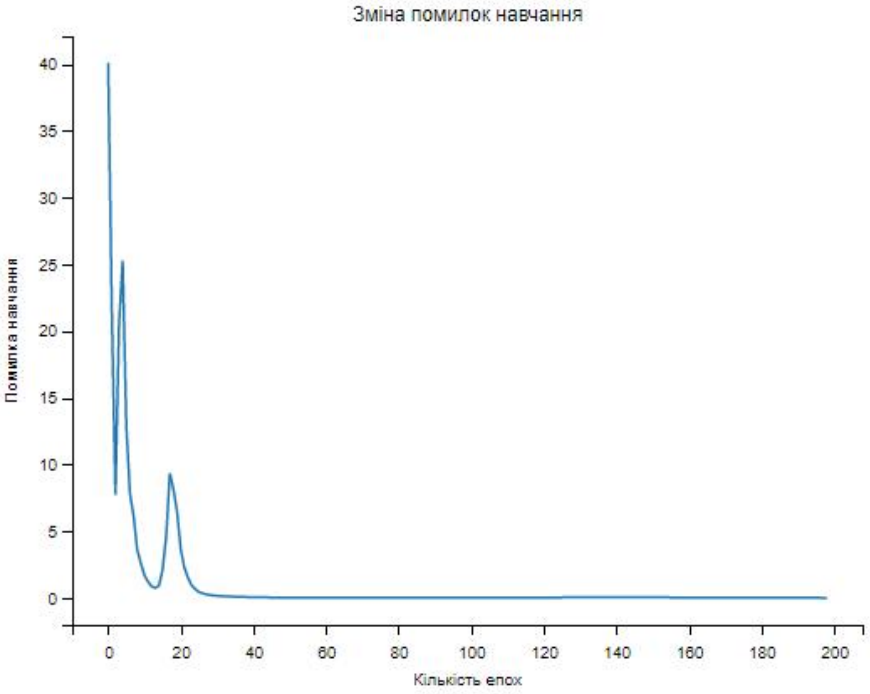
Завдання 2.5

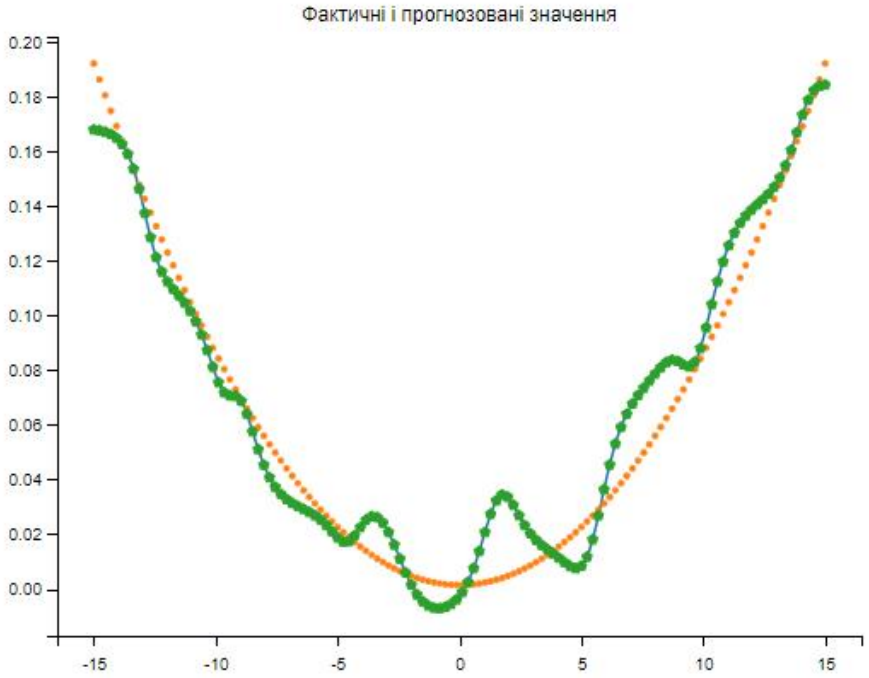
Побудова багатошарової нейронної мережі

```
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) + 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('Зміна помилок навчання')
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.show()
```








```
C:\Users\Onibi\AppData\Local\Programs\Pyt
Epoch: 100; Error: 0.017030507415998833;
The goal of learning is reached
```