НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Факультет прикладної математики
Кафедра прикладної математики

Лабораторна робота №1
«Розробка програмного забезпечення для реалізації двошарового персептрону з сигмоїдальною функцією активації»
з кредитного модуля «Методи глибинного навчання»

Виконав:
студент групи КМ-93
Костенко Олександр Андрійович
Викладач:
Терейковська Л. А.

Київ — 2022

Зміст

Завдання 1

Постановка задачі

Розробити програмне забезпечення для реалізації класичного нейрону (мову програмування студент обирає самостійно). Передбачити режим навчання класичного нейрону на одному навчальнім прикладі та режим розпізнавання.

- Провести навчання елементарного класичного нейрону.
  Початкові умови (i=0):
  - Навчальний приклад

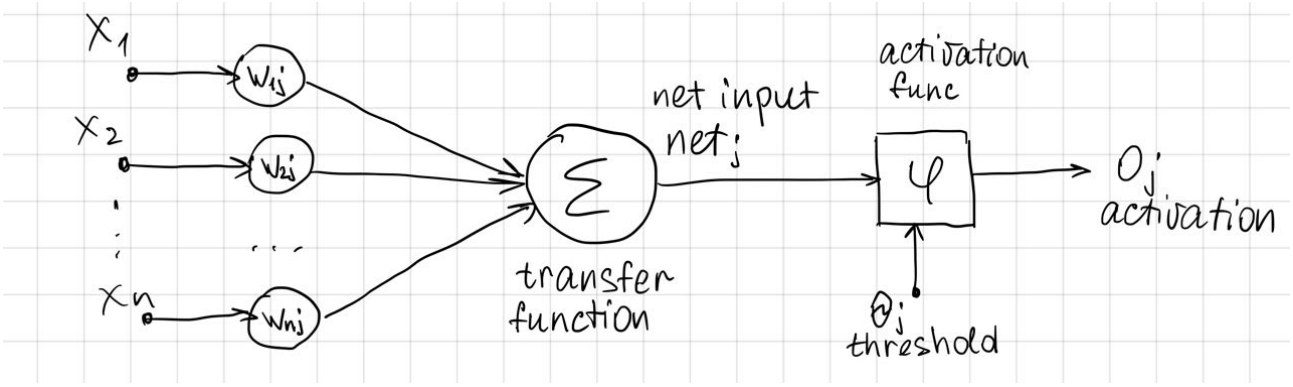
x_0	x_1	x_2	x_3	y_r
1	3	5	7	0,3

  - Вагові коефіцієнти ініціалізовані випадково
  - Допустима помилка *dd*=0.1
- Передбачити режим розпізнаванням нейроном вхідного образу.

Теоретичні відомості

Штучний нейрон — вузол штучної нейронної мережі, що є спрощеною моделлю природного нейрона.

Структура класичного нейрону:



Математична модель класичного нейрону:

$$NET = \sum_{i=1}^{k} x_i w_i$$

$$OUT = F(NET)$$

Функція активації:

$$F(NET) = \frac{1}{1 + e^{-\alpha * NET}}$$

Мета навчання – визначення вагових коефіцієнтів синаптичних зв'язків.

Мета розпізнавання – класифікація/кластеризація невідомого вхідного образу.

$$NN(X) = Y$$

Результат розпізнавання: число та/або назва класу/кластеру.

Критерії оцінки ефективності нейромережевої моделі:

- Точність розпізнавання:
  - Навчальні дані
  - Тестові дані
- Показник обчислюваної потужності:
  - Навчальна вибірка
  - Тестова вибірка
- Термін навчання

Алгоритм навчання:

i=0
навчальний приклад X_0, X_1, X_2, X_3->Yr допустима похибка - dd
вагові коефіцієнти – w0, w1, w2, w3

i=1
A:
$x_S(i) = \sum_{n=0}^{N}(x_n w_n(i-1))$

y(i)=1/(1+exp(-$x_S(i)$))

dn(i)=Abs((Yr- y(i))/Yr)

if dn(i)<=dd then {echo (w0, w1, w2, w3)) stop} else

q(i)= y(i)*(1- y(i))*(Yr- y(i))

$dw_0(i) = x_0 * q(i)$

$w_0(i) = w_0(i-1) + dw_0(i)$
….
=i+1 goto A

Результати роботи програми
Навчання елементарного класичного нейрону:

За навчальних даних:

x_0	x_1	x_2	x_3	y_r
1	3	5	7	0,3

dd=0.1,

Було отримано такий результат:

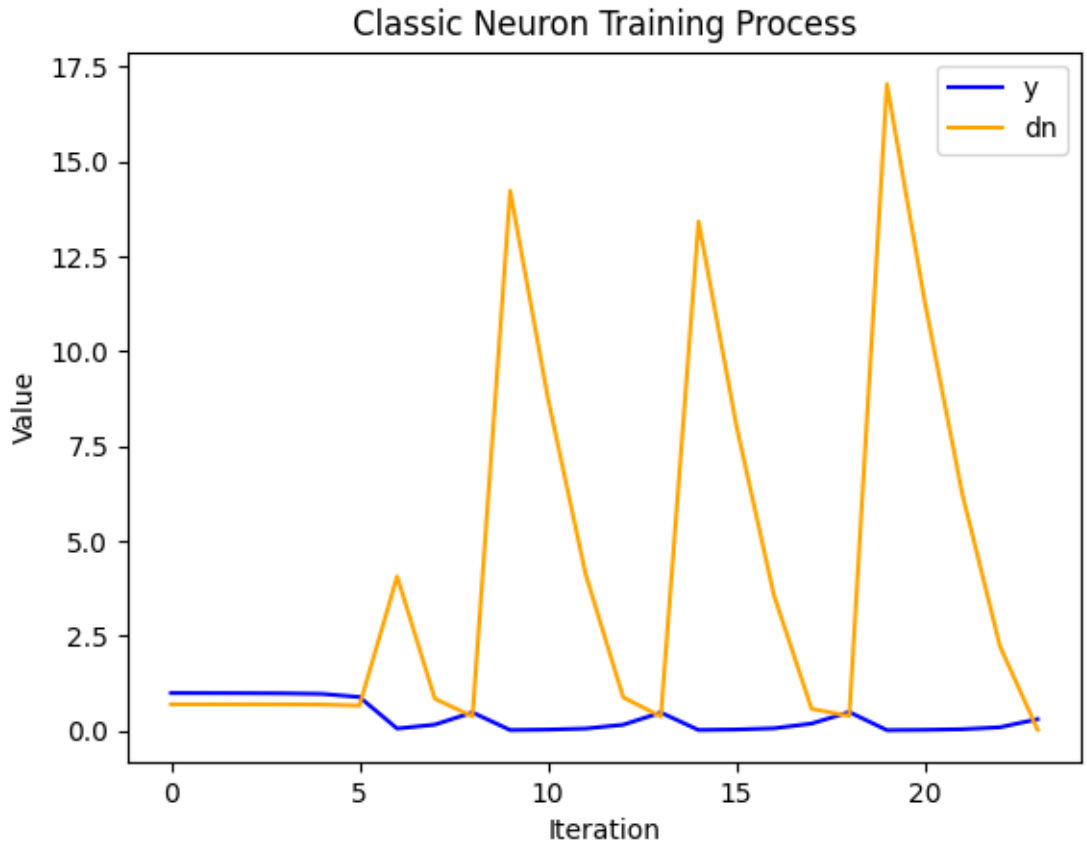


Рисунок 1 Візуалізація змани y та похибки на кожній ітерації

```
iteration 0
y = 0.9954934529057352
dn = 0.6986419156004159
iteration 5
y = 0.8898791964639787
dn = 0.6628755889652448
iteration 10
y = 0.030689815428180327
dn = 8.775229854413878
iteration 15
y = 0.03311429324201388
dn = 8.059532021639946
iteration 20
y = 0.024342557116633863
dn = 11.324095556707256
Conclusive y: 0.3075362824676602
Conclusive w: [ 0.41907525 -0.15436581  0.43557448 -0.42078757]
```

Кінцеві вагові коефіцієнти: [0.41907525, -0.15436581, 0.43557448, -0.42078757]

Тестування елементарного класичного нейрону:

За введених тестувальних даних:

x_0	x_1	x_2	x_3
1,1	3,2	4,9	7,3

dd=0.1,

було отримано результат:

Yr= 0.27479637131226053

Код програми

```
#Kostenko KM-93 lab 1
#part 1: Classic Neuron Training

import numpy as np
from matplotlib import pyplot as plt


#additional list for process visualisation
y_list=[]
dn_list=[]

#training function
def training(x, yr, w, dd):
    iter=0

    #set initial dn > than dd
    dn=0.5

    #training untill condition is met
    while dn>dd:
        xsum=0
        #calculate xs
        for i in range(len(x)):
            xsum+=x[i]*w[i]
        #calculate y
        y=1/(1+np.exp(-1*xsum))
        y_list.append(y)
        #calculate dn
        dn=np.abs((yr-y)/y)
        dn_list.append(dn)
        #print every 5th iteration
        if iter%5==0:
            print(f"iteration {iter}")
            print(f"y = {y}")
            print(f"dn = {dn}")

        if dn<=dd:
            break
        else:
            #adjust weights
```

```
            q=y*(1-y)*(yr-y)
            for i in range(len(w)):
                dw=x[i]*q
                w[i]=w[i]+dw
            iter+=1
    return w, y_list, dn_list

#testing function
def testing(x, w):
    xsum=0
    for i in range(len(x)):
        xsum+=x[i]*w[i]
    y=1/(1+np.exp(-1*xsum))
    return y


def classic_neuron():
    #input values
    x=[1, 3, 5, 7]
    yr=.3
    dd=.1
    w=np.random.rand(len(x))
    #training neuron
    w, y_list, dn_list=training(x, yr, w, dd)

    print(f"Conclusive y: {y_list[-1]}")
    print(f"Conclusive w: {w}")

    #visualizing training process
    iteration=[]
    for i in range(len(y_list)):
        iteration.append(i)

    plt.plot(iteration, y_list, color="blue", label="y")
    plt.plot(iteration, dn_list, color="orange", label="dn")

    plt.title("Classic Neuron Training Process")
    plt.xlabel("Iteration")
    plt.ylabel("Value")
    plt.legend()
    plt.show()

    #testing
    x=input("Input 4 x devided by 'space': ")
    x=x.split(" ")
    for i in range(len(x)):
```

```
        x[i]=float(x[i])
    print(x)
    print(testing(x, w))

classic_neuron()
```

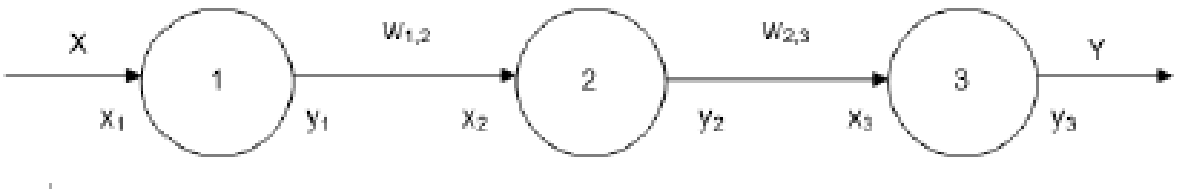
Завдання 2

Постановка задачі

Розробити програмне забезпечення для реалізації елементарного двошарового персептрону зі структурою 1-1-1. Передбачити режим навчання на одному навчальному прикладі та режим розпізнавання.

Теоретичні відомості

Мережі з прямим розповсюдженням сигналу – мережі, в яких існують тільки прямі зв'язки:



Нейронний шар – група нейронів з однаковими зв'язками.

Загальна схема алгоритму зворотнього розповсюдження помилки:

1. Визначення допустимої похибки навчального прикладу та ініціалізація вагових коефіцієнтів.
2. Розрахунок вихідного сигналу кожного з нейронів.
3. Розрахунок похибки навчання
4. Порівняння похибки з допустимою:
    - Якщо обчислена похибка більша за допустиму, то навчання закінчується
    - Якщо навпаки – перехід на наступний пункт
5. Розрахунок корегуючих параметрів
6. Корекція вагових коефіцієнтів
7. Перехід на пункт 2.

Переваги багатошарового персептрону:

- Можливість навчання на зашумлених даних
- Можливість навчання в процесі використання БШП
- Інтелектуальні можливості БШП вважаються найбільш високими серед класичних нейронних мереж
- Достатня швидкість прийняття рішень

Недоліки БШП:

- Занадто велика ресурсоємність
- Не враховує топологію зв'язки вхідних параметрів
- Складність моделювання динамічних рядів даних
- Складність формування навчальної вибірки

Схема навчання елементарного двошарового персептрону зі структурою 1-1-1:

i=0
навчальний приклад X->Yr допустима похибка - dd
вагові коефіцієнти – $w_{1,2}(i)$ $w_{2,3}(i)$

i=1

A:

$x_{2,s}(i)=w_{1,2}(i-1)*x_2$ $y_2(i)=1/(1+\exp(-x_{2,s}(i)))$
$x_{3,s}(i)=w_{2,3}(i-1)*x_3$ $y_3(i)=1/(1+\exp(-x_{3,s}(i)))$

$dn(i)=Abs((Yr- y_3(i))/Yr)$

if $dn(i)<=dd$ then {echo($w_{1,2}(i-1)$, $w_{2,3}(i-1)$) stop} else

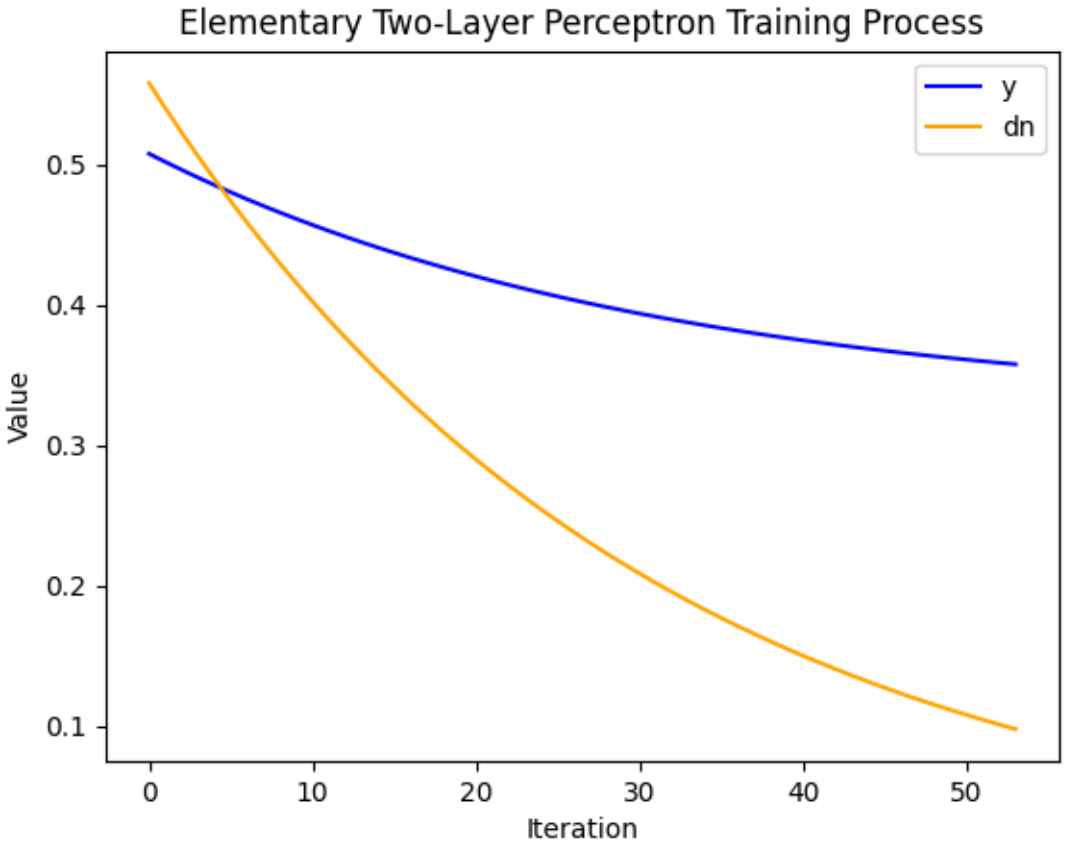
$q_3(i)= y_3(i)*(1- y_3(i))*(Yr- y_3(i))$ $\Delta w_{2,3}(i)= q_3(i)*y_2(i)$
$q_2(i)= y_2(i)*(1- y_2(i))*(q_3(i)* w_{2,3}(i-1))$ $\Delta w_{1,2}(i)= q_2(i)*X$

$w_{2,3}(i)= w_{2,3}(i-1)+ \Delta w_{2,3}(i)$ $w_{1,2}(i)= w_{1,2}(i-1)+ \Delta w_{1,2}(i)$
i=i+1 goto A

Результати роботи програми

Навчання елементарного двошарового персептрону зі структурою 1-1-1

За згерованих навчальних даних: X= 0.9609993018593804, Yr= 0.32583918778900334, при допустимій похибці dd=0.1, було отримано такі результати:



```
x=0.9609993018593804
y=0.32583918778900334
iteration 0
y = 0.5076483105778881
dn = 0.5579719370851579
iteration 10
y = 0.4569163874538838
dn = 0.40227573777782297
iteration 20
y = 0.42029604592504627
dn = 0.28988796214778295
iteration 30
y = 0.3938438761691575
dn = 0.20870629110514016
iteration 40
y = 0.3747737230825948
dn = 0.150180018633237 46
iteration 50
y = 0.36105223634841 69
dn = 0.10806879552564959
Conclusive y: 0.3577441814355142
Conclusive w: [ 1.14068669 -0.780696  ]
```

Кінцеві вагові коефіцієнти: [1.14068669, -0.780696]

Тестування елементарного двошарового персептрону зі структурою 1-1-1

За згенерованих тестувальних даних:

X= 1.6463554969075793

dd=0.1,

було отримано результат:

Yr= 0.3368957053025228

Код програми

```
#Kostenko KM-93 lab 1
#part 2: Elementary Two-Layer Perceptron with 1-1-1 Structure Training

import numpy as np
from matplotlib import pyplot as plt

#training function
def training(x, yr, dd=0.1):
    y_list=[]
```

```
    dn_list=[]
    #set initial dn
    dn=0.5
    #initialize wieghts
    w=np.random.rand(2)
    iter=0
    #training until condition is met
    while dn>dd:
        #calculate y2 and y3
        y2=1/(1+np.exp(-1*w[0]*x))
        y3=1/(1+np.exp(-1*w[1]*(1/(1+np.exp(-1*w[0]*x)))))
        y_list.append(y3)
        #calculate dn
        dn=np.abs((yr-y3)/yr)
        dn_list.append(dn)
        #print y and dn
        print(f"iteration {iter}")
        print(f"y = {y3}")
        print(f"dn = {dn}")

        if dn<=dd:
            break
        else:
            #adjust weights
            q3=y3*(1-y3)*(yr-y3)
            q2=y2*(1-y3)*(q3*w[1])
            dw2=q3*y2
            dw1=q2*x
            w[1]=w[1]+dw2
            w[0]=w[0]+dw1

            iter+=1
    return w, y_list, dn_list


#testing function
def testing(x, w):
    return(1/(1+np.exp(-1*w[1]*(1/(1+np.exp(-1*w[0]*x))))))

def elementary_perceptron_111():
    #input initial data
    x=np.random.rand(1)[0]
    yr=np.random.rand(1)[0]

    #training
    w, y_list, dn_list=training(x, yr)
```

```
    print(f"Conclusive y: {y_list[-1]}")
    print(f"Conclusive w: {w}")

    #visualizing training process
    iteration=[]
    for i in range(len(y_list)):
        iteration.append(i)

    plt.plot(iteration, y_list, color="blue", label="y")
    plt.plot(iteration, dn_list, color="orange", label="dn")

    plt.title("Elementary Two-Layer Perceptron Training Process")
    plt.xlabel("Iteration")
    plt.ylabel("Value")
    plt.legend()
    plt.show()

    #testing
    x+=np.random.rand(1)[0]
    print(f"x={x}")
    print(f"y={testing(x, w)}")

elementary_perceptron_111()
```

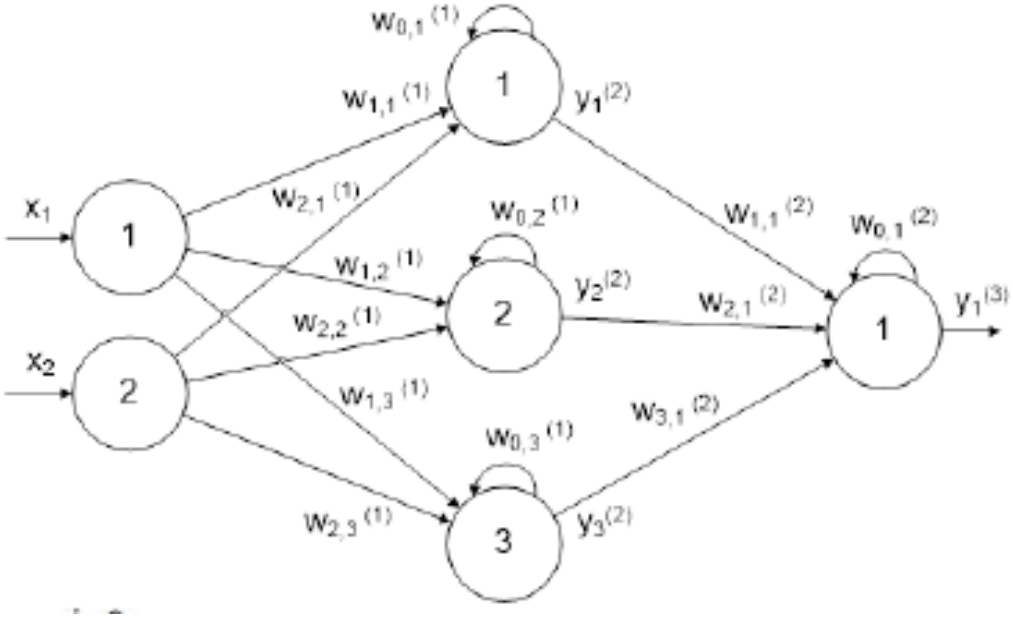
Завдання 3

Постановка задачі

Розробити програмне забезпечення для реалізації елементарного двошарового персептрону зі структурою 2-3-1. Передбачити режим навчання “ON-LINE” та режим розпізнавання.

Теоретичні відомості

Структура елементарного двошарового персептрону зі структурою 2-3-1:



Алгоритм розробки БШП для практичної задачі:

1. Визначити нуменклатуру та допустимі величини вхідних параметрів
2. Підготувати тестову, контрольну та навчальну вибірку
3. Визначення мінімальної та максимальної меж кількості схованих нейронів
4. Вибрати загальну кількість схованих нейронів
5. Вибрати кількість схованих шарів нейронів та кількість нейронів на кожному шарі
6. Вибрати вид та параметри функцій активації для всіх типів нейронів
7. Навчання
8. Тестування
9. Зміна параметрів
10. Повторення пунктів 4-9

Схема навчання елементарного двошарового персептрону зі структурою 2-3-1:

i=0

навчальний приклад - (x_1,x_2)->Yr допустима похибка - dd

вагові коефіцієнти – $w_{0,1}^{(1)}(i), w_{1,1}^{(1)}(i), w_{2,1}^{(1)}(i), \ldots$

$w_{0,1}^{(2)}(i), w_{1,1}^{(2)}(i), w_{2,1}^{(2)}(i), w_{3,1}^{(2)}(i)$

i=1

A:

$x_{1,s}^{(1)}(i)=w_{0,1}^{(1)}(i-1)*x_0 + w_{1,1}^{(1)}(i-1)*x_1 + w_{2,1}^{(1)}(i-1)*x_2 \quad y_1^{(2)}(i)=1/(1+\exp(-x_{1,s}^{(1)}(i)))$

$x_{2,s}^{(1)}(i)=w_{0,2}^{(1)}(i-1)*x_0 + w_{1,2}^{(1)}(i-1)*x_1 + w_{2,2}^{(1)}(i-1)*x_2 \quad y_2^{(2)}(i)=1/(1+\exp(-x_{2,s}^{(1)}(i)))$

$x_{3,s}^{(1)}(i)=w_{0,3}^{(1)}(i-1)*x_0 + w_{1,3}^{(1)}(i-1)*x_1 + w_{2,3}^{(1)}(i-1)*x_2 \quad y_3^{(2)}(i)=1/(1+\exp(-x_{3,s}^{(1)}(i)))$

$x_{1,s}^{(2)}(i)=w_{0,1}^{(2)}(i-1)*x_0 + w_{1,1}^{(2)}(i-1)*y_1^{(2)}(i) + w_{2,1}^{(2)}(i-1)*y_2^{(2)}(i) + w_{3,1}^{(2)}(i-1)*y_3^{(2)}(i)$

$y_1^{(3)}(i)=1/(1+\exp(-x_{1,s}^{(2)}(i)))$

$dn(i)=Abs((Yr- y_1^{(3)}(i))/Yr)$

if $dn(i)<=dd$ then { echo($w_{0,1}^{(1)}(i), \ldots w_{3,1}^{(2)}(i)$) stop } else

$q_1^{(3)}(i)= y_1^{(3)}(i)*(1- y_1^{(3)}(i))*(Yr - y_1^{(3)}(i))$

$b_{0,1}^{(2)}(i)= q_1^{(3)}(i)*x_0 \qquad b_{1,1}^{(2)}(i)= q_1^{(3)}(i)* y_1^{(2)}(i)$

$b_{2,1}^{(2)}(i)= q_1^{(3)}(i)* y_2^{(2)}(i) \qquad b_{3,1}^{(2)}(i)= q_1^{(3)}(i)* y_3^{(2)}(i)$

$w_{0,1}^{(2)}(i)= w_{0,1}^{(2)}(i-1) + b_{0,1}^{(2)}(i) \qquad w_{1,1}^{(2)}(i)= w_{1,1}^{(2)}(i-1) + b_{1,1}^{(2)}(i)$

$w_{2,1}^{(2)}(i)= w_{2,1}^{(2)}(i-1) + b_{2,1}^{(2)}(i) \qquad w_{3,1}^{(2)}(i)= w_{3,1}^{(2)}(i-1) + b_{3,1}^{(2)}(i)$

//розрахунок помилки 1-го нейрону СШН, корегування ваг 1-го нейрону СШН

$q_1^{(2)}(i)= y_1^{(2)}(i)*(1- y_1^{(2)}(i))*(q_1^{(3)}(i)*w_{1,1}^{(2)}(i-1))$

$b_{0,1}^{(1)}(i)= q_1^{(2)}(i)*x_0 \qquad b_{1,1}^{(1)}(i)= q_1^{(2)}(i)*x_1 \qquad b_{2,1}^{(1)}(i)= q_1^{(2)}(i)*x_2$

$w_{0,1}^{(1)}(i)= w_{0,1}^{(1)}(i-1) + b_{0,1}^{(1)}(i) \qquad w_{1,1}^{(1)}(i)= w_{1,1}^{(1)}(i-1) + b_{1,1}^{(1)}(i) \qquad w_{2,1}^{(1)}(i)= w_{2,1}^{(1)}(i-1) + b_{2,1}^{(1)}(i)$

……..

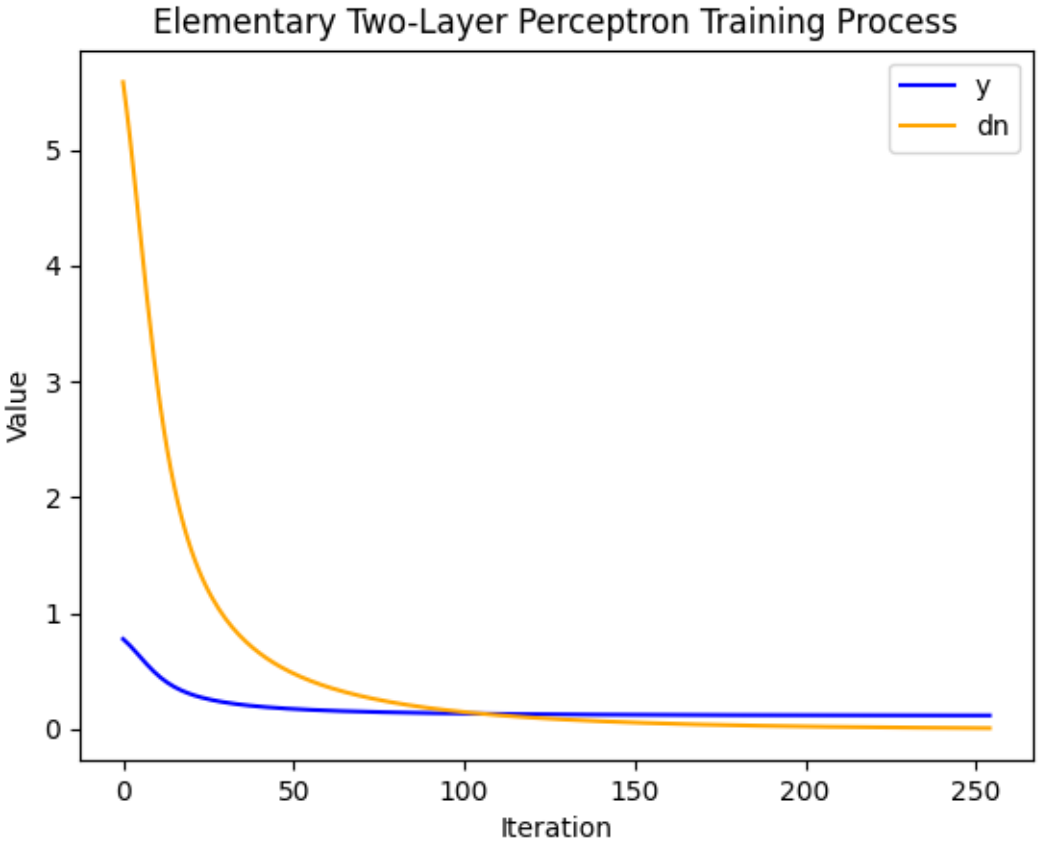
i=i+1 goto A

Результати роботи програми

Навчання елементарного двошарового персептрону зі структурою 2-3-1

За згерованих навчальних даних: X= [0.6849195, -0.56657596], Yr= X1+X2=0.11834354, при допустимій похибці dd=0.1, було отримано такі

результати:


Elementary Two-Layer Perceptron Training Process
y
dn
Value
Iteration
0
1
2
3
4
5
0
50
100
150
200
250

```
iteration 0
y = 0.7789649821400229
dn = 5.582234955853237
iteration 10
y = 0.47143423770687554
dn = 2.9836077230265334
iteration 20
y = 0.30128178144957485
dn = 1.54582364918511
iteration 30
y = 0.2313860145512462
dn = 0.9552061365975563
iteration 40
y = 0.1960741178689222
dn = 0.6568214774292829
iteration 50
y = 0.17519517227240253
dn = 0.4803949002432162
iteration 60
y = 0.16156381342752804
dn = 0.3652102529975668
iteration 70
y = 0.1520655487137201
dn = 0.2849501495877634
iteration 80
y = 0.14514257461841845
dn = 0.22645118861602384
iteration 90
y = 0.13993037622682616
dn = 0.182408240297952
iteration 100
y = 0.13591039140598168
dn = 0.14843946735379732
y = 0.12176250025380407
dn = 0.028890134805327778
iteration 200
y = 0.12122701633775017
dn = 0.024365308874303323
iteration 210
y = 0.12077796167143222
dn = 0.02057080797953049
iteration 220
y = 0.12040065437850778
dn = 0.017382570626725852
iteration 230
y = 0.12008311439150943
dn = 0.014699365540202451
iteration 240
y = 0.11981550831856551
dn = 0.012438100800302446
iteration 250
y = 0.11958972402175724
dn = 0.010530229040969676
X0: [0.93883293 0.21696893 0.68070459 0.74564339]
Conclusive y: 0.11950957443670716
```

Кінцеві вагові коефіцієнти:

$w_{1,i}^{(1)}$:[0.19778623, 0.66741481, 0.86165125],

$w_{2,i}^{(1)}$:[0.37121486, 0.04075722, 0.61057352]]),

$w_{0,i}^{(1)}$:[0.97298031, 0.63316507, 0.96867181]),

$w_{1,i}^{(2)}$[-1.11984531, -1.17655278, -1.67040752]),

$w_{0,1}^{(2)}$[-1.75025221]

Тестування елементарного двошарового персептрону зі структурою 2-3-1

За згерованих навчальних даних: X= [1.22959146, 0.37430511], при допустимій похибці dd=0.1, було отримано такі результати: Yr = 1.57802582325825086

Код програми

```
#Kostenko KM-93 lab 1
#part 3: Elementary Two-Layer Perceptron with 2-3-1 Structure Training

import numpy as np
from matplotlib import pyplot as plt

#training function
def training(x, dd=0.01):
    yr=np.sum(x)
    print(yr)
    y_list=[]
    dn_list=[]
    x0=np.random.rand(4)
    #set initial dn
    dn=0.5
    #initialize wieghts
    w=[np.random.rand(2,3), np.random.rand(3), np.random.rand(3),
np.random.rand(1)]
    iter=0
    #train until condition is met
    while dn>dd:
        xsums=[]
        ys=[]
        #calculate xs
        for i in range(len(w[1])):
            xsum=w[1][i]*x0[i]+w[0][0][i]*x[0]+w[0][1][i]*x[1]
            xsums.append(xsum)
            ys.append(1/(1+np.exp(-1*xsum)))
        xsum=w[1][0]*x0[3]+w[2][0]*ys[0]+w[2][1]*ys[1]+w[2][2]*ys[2]
```

```
        #calculate output value
        y=1/(1+np.exp(-1*xsum))
        y_list.append(y)
        #calculate dn
        dn=np.abs((yr-y)/yr)
        dn_list.append(dn)
        print(f"iteration {iter}")
        print(f"y = {y}")
        print(f"dn = {dn}")

        #check if codition is met
        if dn<=dd:
            break

        else:
            #adjust weights
            q=y*(1-y)*(yr-y)
            b=[q*x0[3], q*ys[0], q*ys[1], q*ys[2]]
            print(w[3][0])
            w[3][0]=w[3][0]+b[0]
            w[2][0]=w[2][0]+b[1]
            w[2][1]=w[2][1]+b[2]
            w[2][2]=w[2][2]+b[3]

            for i in range(len(ys)):
                q1=ys[i]*(1-ys[i])*(q*w[2][0])
                b=[q1*x0[i], q1*x[0], q1*x[1]]
                w[1][i]=w[1][i]+b[0]
                w[0][0][i]=w[0][0][i]+b[1]
                w[0][1][i]=w[0][1][i]+b[2]

            iter+=1

    return w, x0, y_list, dn_list

#testing function
def testing(x, x0, w):
    xsums=[]
    ys=[]

    for i in range(len(w[1])):
        xsum=w[1][i]*x0[i]+w[0][0][i]*x[0]+w[0][1][i]*x[1]
        xsums.append(xsum)
        ys.append(1/(1+np.exp(-1*xsum)))
```

```
    xsum=w[1][0]*x0[3]+w[2][0]*ys[0]+w[2][1]*ys[1]+w[2][2]*ys[2]
    y=1/(1+np.exp(-1*xsum))
    return y

def elementary_perceptron_231():
    #input initial data
    x=np.random.randn(2)
    #training
    w, x0, y_list, dn_list=training(x)
    print(f"Conclusive y: {y_list[-1]}")
    print(f"Conclusive w: {w}")

    #visualizing training process
    iteration=[]
    for i in range(len(y_list)):
        iteration.append(i)

    plt.plot(iteration, y_list, color="blue", label="y")
    plt.plot(iteration, dn_list, color="orange", label="dn")

    plt.title("Elementary Two-Layer Perceptron Training Process")
    plt.xlabel("Iteration")
    plt.ylabel("Value")
    plt.legend()
    plt.show()

    #testing
    x+=np.random.rand(2)
    print(f"x={x}")
    print(f"y={testing(x, x0, w)}")

elementary_perceptron_231()
```