КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

Кафедра інтелектуальних та інформаційних систем

Лабораторна робота № 2

з дисципліни

“Нейромережні технології та їх застосування”

Виконав студент
групи КН-31
Пашковський Павло Володимирович

Київ-2021

Контрольні питання

1. Коротка історична довідка про розвиток теорії нейронних мереж.

Історично першою роботою, яка заклала теоретичний фундамент для створення штучних моделей нейронів нейронних мереж, прийнято вважати опубліковану в 1943 р статтю Уоррена Мак-Каллока і Уолтера Піттса «Логічне числення ідей, що відносяться до нервової активності».

2. Будова біологічного нейрона.

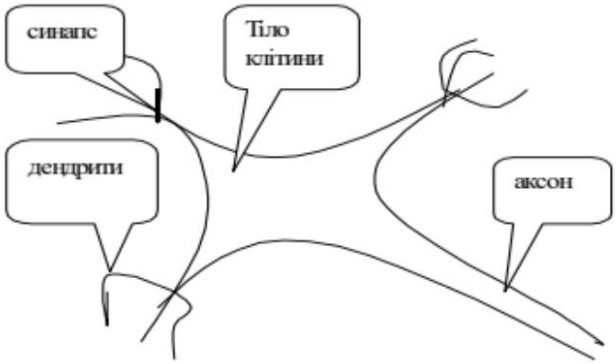


Рис. 1.1. Будова біологічного нейрона

Сигнали між нейронами передаються вздовж аксонів. Сигнали, які підсилюються або послаблюються синапсами, нейрон одержує через дендріти, які підсилюються, або послаблюються синапсами.

3. Штучний нейрон та активаційні функції.

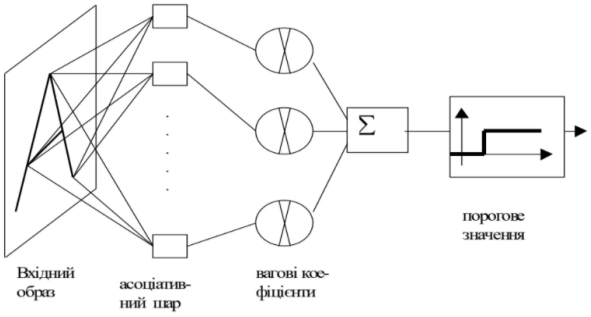


Рисунок 1 Бінарний перцептрон

1. Подати на вхід образ X .

2. Кожна компонента X множиться на відповідну компоненту вектора вагових коефіцієнтів W Знаходимо суму цих добутків

$$a = XW = \sum_{i=1}^{n} x_i w_i$$

3. Якщо a перевищує порогове значення $a > \theta$ то вихід нейрона Yдорівнює одиниці, в іншому випадку він – нуль. Якщо значення Y вірне, то нічого не змінюється і переходимо до кроку 1 у випадку, якщо на вході є інші образи. Якщо ні, то кінець. Якщо значення Y не вірне, то підраховуємо різницю між обчисленим значенням і правильним значенням

$$\delta = T - Y$$

4 . Обчислюємо

$$\Delta_i = \mu \delta x_i$$

Де μ – коефіцієнт швидкості навчання, який знаходиться, в загальному

випадку, в проміжку (0;0,1).

5. Знаходимо

$$w_i(n+1) = w_i(n) + \Delta_i, i = \overline{1,n}$$

Де $w_i(n+1)$ – значення і -го вагового коефіцієнта після корекції, $w_i(n)$- до корекції.

6. Якщо на вході є інші образи, то переходимо до кроку 1, якщо ні – то кінець.

4. Будова та алгоритм функціонування перцептрона.

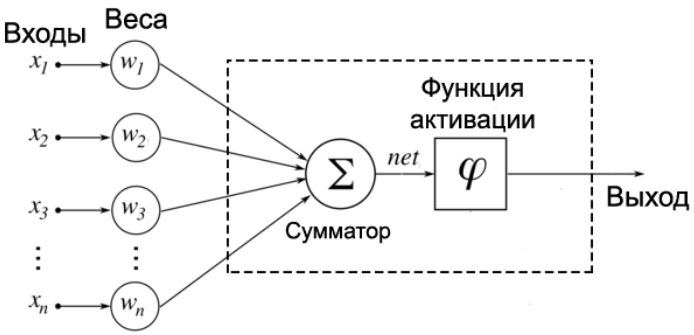


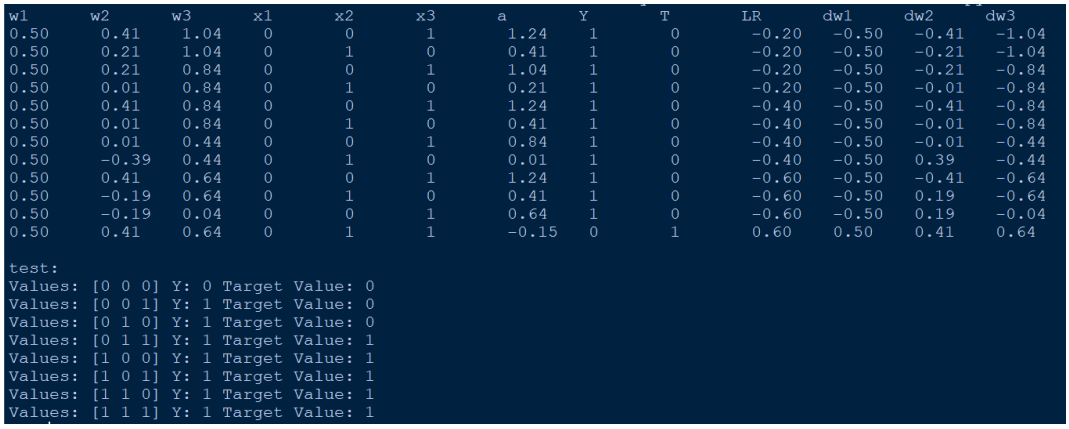
Рис. 1.3. Будова перцептрона

5. Проблема лінійного поділу досліджуваної області.

При розв’язанні реальних задач є обмеження кількості зразків, на підставі яких і проводиться побудова класифікатора. При цьому не можна провести таке попереднє оброблення даних, при якому буде досягнута лінійна роздільність зразків.

Індивідуальне завдання:

№ варіанта	Логічна функція	Можливі значення коефіцієнта η		
1	$x_1 \vee x_2 \wedge x_3$	0.2	0.4	0.6



```
w1      w2      w3      x1      x2      x3      a       Y       T       LR      dw1     dw2     dw3
0.50    0.41    1.04    0       0       1       1.24    1       0       -0.20   -0.50   -0.41   -1.04
0.50    0.21    1.04    0       1       0       0.41    1       0       -0.20   -0.50   -0.21   -1.04
0.50    0.21    0.84    0       0       1       1.04    1       0       -0.20   -0.50   -0.21   -0.84
0.50    0.01    0.84    0       1       0       0.21    1       0       -0.20   -0.50   -0.01   -0.84
0.50    0.41    0.84    0       0       1       1.24    1       0       -0.40   -0.50   -0.41   -0.84
0.50    0.01    0.84    0       1       0       0.41    1       0       -0.40   -0.50   -0.01   -0.84
0.50    0.01    0.44    0       0       1       0.84    1       0       -0.40   -0.50   -0.01   -0.44
0.50    -0.39   0.44    0       1       0       0.01    1       0       -0.40   -0.50   0.39    -0.44
0.50    0.41    0.64    0       0       1       1.24    1       0       -0.60   -0.50   -0.41   -0.64
0.50    -0.19   0.64    0       1       0       0.41    1       0       -0.60   -0.50   0.19    -0.64
0.50    -0.19   0.04    0       0       1       0.64    1       0       -0.60   -0.50   0.19    -0.04
0.50    0.41    0.64    0       1       1       -0.15   0       1       0.60    0.50    0.41    0.64

test:
Values: [0 0 0] Y: 0 Target Value: 0
Values: [0 0 1] Y: 1 Target Value: 0
Values: [0 1 0] Y: 1 Target Value: 0
Values: [0 1 1] Y: 1 Target Value: 1
Values: [1 0 0] Y: 1 Target Value: 1
Values: [1 0 1] Y: 1 Target Value: 1
Values: [1 1 0] Y: 1 Target Value: 1
Values: [1 1 1] Y: 1 Target Value: 1
```

Рис. 2.1. Результат роботи програми

Задача класифікації вхідного вектора u, тобто віднесення його до класу L1, якщо вихідний сигнал приймає значення 1, і до класу L2, якщо вихідний сигнал приймає значення 0. Важливе обмеження – лінійна роздільність векторів

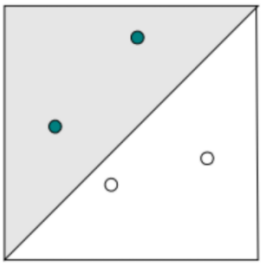


Рис. 2.2 Лінійно-роздільні

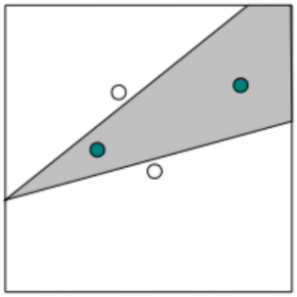


Рис. 2.3. Лінійно-нероздільні

Перцептрон може класифікувати лише ті вектори, які можна розбити на класи єдиною гіперплощиною

Оскільки задано лінійно-нероздільні вектори, дану задачу неможливо вирішити використовуючи один нейрон.

Висновок:

Було створено програму для розпізнавання логічної функції, що подається на вході, за допомогою бінарного перцептрона.

Для подолання проблеми лінійного поділу досліджуваної області необхідно додати до програми один додатковий нейрон та шар.

Отримані навички щодо функціонування і навчання перцептрона знайдуть майбутнє практичне застосування у розпізнаванні індентичних даних, аналізі та класифіковані даних за заданими параметрами.

Код програми:

```
import numpy as np


def train(learning_rate, Weights, epochs=2):
    W = Weights.copy()
    for epoch in range(epochs):
        for i,xi in enumerate(X):
            a = 0
            Y = 0
            for x,w in zip(xi,W):
```

```
                a += x * w
            if a > 0:
                Y = 1
            else:
                Y = 0
            if T[i] == Y:
                continue
            else:
                d = T[i] - Y
                delta = np.zeros(3)
                for index in range(len(delta)):
                    delta[index] = learning_rate * d * xi[index]

                for w_index in range(len(W)):
                    W[w_index] += delta[w_index]

                print('%.2f' % W[0], '\t', '%.2f' % W[1], '\t', '%.2f' % W[2], '\t', str(xi[0]),
                  '\t', str(xi[1]), '\t', str(xi[2]), '\t', '%.2f' % a, '\t', str(Y), '\t', str(T[i]), '\t', '%.2f' %
(learning_rate*d),
                  '\t', '%.2f' % (W[0]*d), '\t', '%.2f' % (W[1]*d), '\t', '%.2f' % (W[2]*d))

        return W


    def test(X, trained_W, T):
        print("\ntest:")
        for i,xi in enumerate(X):
            a = 0
            Y = 0
            for x,w in zip(xi,trained_W):
                a += x * w
            if a > 0:
                Y = 1
            else:
                Y = 0
            print(f"Values: {xi} Y: {Y} Target Value: {T[i]}")


    def inputx(trained_W):
        print("\ninput x:")
        xtest=input()
        xi=xtest.split()
        xi=list(xi)
        a = 0
        Y = 0
        for x,w in zip(xi,trained_W):
            a += int(x) * w
        if a > 0:
            Y = 1
        else:
            Y = 0
        print(f"Y: {Y}")


    X = np.array([[0,0,0],
        [0,0,1],
        [0,1,0],
        [0,1,1],
        [1,0,0],
        [1,0,1],
        [1,1,0],
        [1,1,1]])
```

```
T = [0,0,0,1,1,1,1,1]

print('w1\tw2\tw3\tx1\tx2\tx3\ta\tY\tT\tLR\tdw1\tdw2\tdw3')

learning_rates = [0.2,0.4,0.6]
Weights = np.random.randn(3)
for learning_rate in learning_rates:
    trained_W = train(learning_rate,Weights)

test(X, trained_W, T)

while(1):
    inputx(trained_W)
```