Лабораторна робота 1

Тема: «Нейронна реалізація логічних функцій AND, OR, XOR».

Мета: Дослідити математичну модель нейрона.

Час виконання: 2 години.

Навчальні питання:

1). Нейрони для реалізації функцій AND, OR;

2). Проблема XOR. Нейрон для реалізації функції XOR;

Завдання №1:

Реалізувати обчислювальний алгоритм для функції xor(x1, x2) через функції or(x1, x2) і and(x1, x2) в програмному середовищі (C++, Python, та ін.).

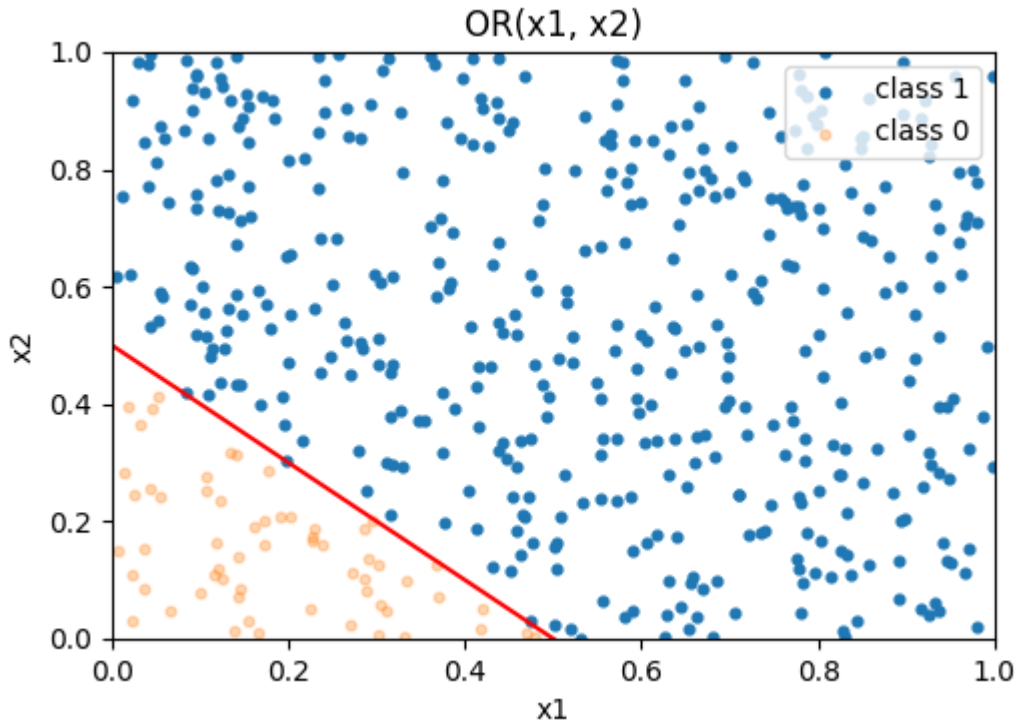

Рис. 1. Побудова лінійного класифікатора функції OR

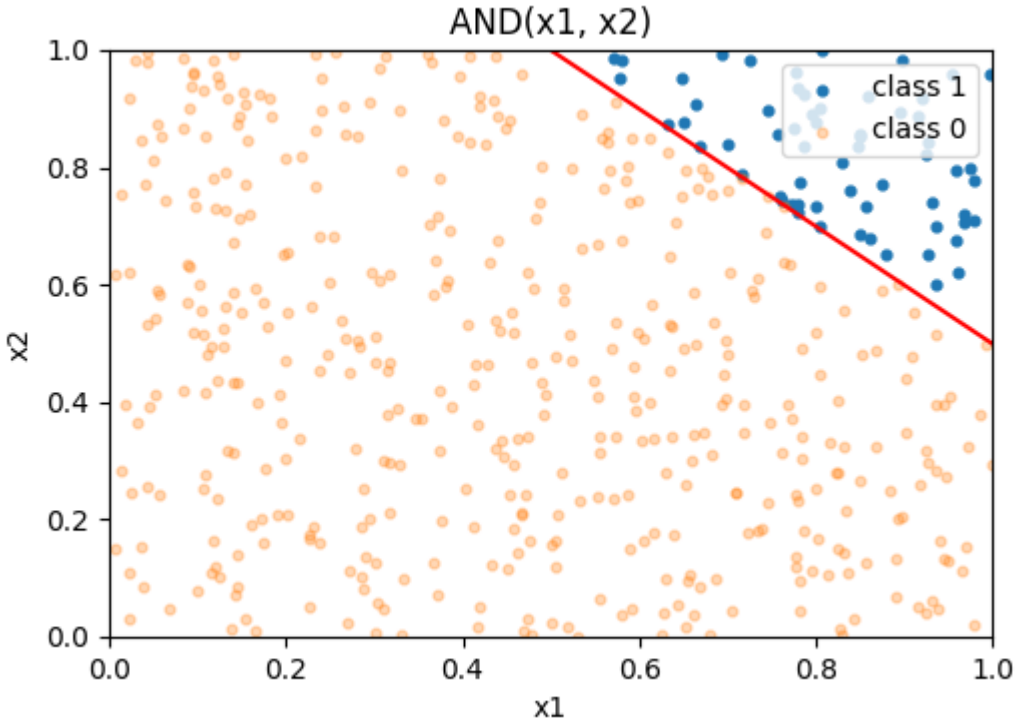

Рис. 2. Побудова лінійного класифікатора функції AND

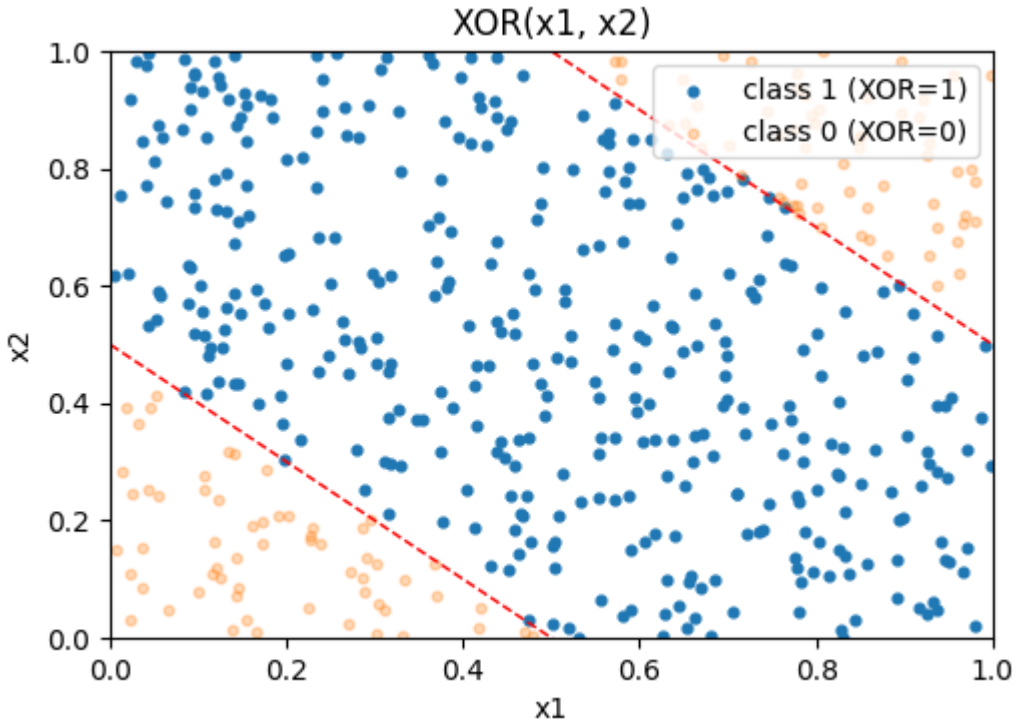

Рис. 3. Побудова лінійного класифікатора функції XOR

Завдання №2:

Зобразити двохслойний персептрон для функції xor(x1, x2) та скласти відповідне рівняння розділяючої прямої, використовуючи теоретичний матеріал даної лабораторної роботи.

Функція XOR(x_1 x_2) не є лінійно відокремлюваною у вхідному просторі (x_1 x_2), тобто однією прямою її коректно розділити неможливо. Тому застосуємо двошаровий персептрон: на першому шарі — нейрони для OR та AND, на другому — один лінійний нейрон, що об'єднує їхні виходи.

Використаємо порогову (східчасту) активацію $f(\nu) = 1[\ \nu > 0]$ і суматор $g(x)=W_1x1+W_2x_2+W_0$

OR: розділяюча пряма $x_1 + x_2 = \frac{1}{2}$

AND: розділяюча пряма $x_1 + x_2 = \frac{3}{2}$

x_1	x_2	y_1	y_2	XOR	Клас
0	0	0	0	0	Ω_1
0	1	1	0	1	Ω_0
1	0	1	0	1	Ω_0
1	1	1	1	0	Ω_1

У просторі (y1,y2) потрібно знайти розділяючу пряму, яка відокремлює клас Ω_0(XOR = 1) від Ω_1 (XOR = 0).

Точки:

$(0,0) \rightarrow \Omega_1$

$(0,1) \rightarrow \Omega_0$

$(1,0) \rightarrow \Omega_0$

$(1,1) \rightarrow \Omega_1$

У просторі (y1,y2) класи для XOR вже лінійно відокремлювані.

Побудуємо дискримінант:

$g(y)= W_1^{II}y_1+W_2^{II}y_2+W_0^{II}$

$$\begin{cases} 0W_1^{II} + 0W_2^{II} + W_0^{II} = 0 \\ 1W_1^{II} + 0W_2^{II} + W_0^{II} = 1 \\ 1W_1^{II} + 1W_2^{II} + W_0^{II} = 0 \end{cases}$$

Підібравши ваги, отримуємо $W_1^{II} = 1, W_2^{II} = -1, W_0^{II} = -0{,}5$

Східчаста функція:

$$f(x) = \begin{cases} 1, & \text{при } x \geq 0 \\ 0, & \text{при } x < 0 \end{cases}$$

Дискрімінантна функція

$g(y)= y_1 - y_2 - 0.5$

Така константа −0.5 усуває неоднозначність на граничних випадках і дає коректну істинність XOR для всіх чотирьох бінарних комбінацій

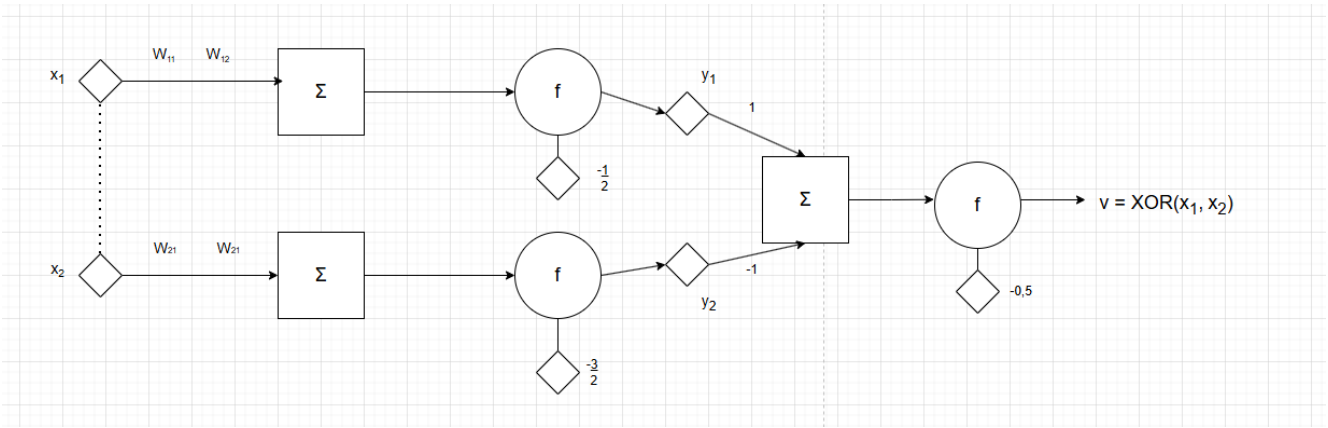

Рис. 4. Схема двошарового персептрона для функції XOR

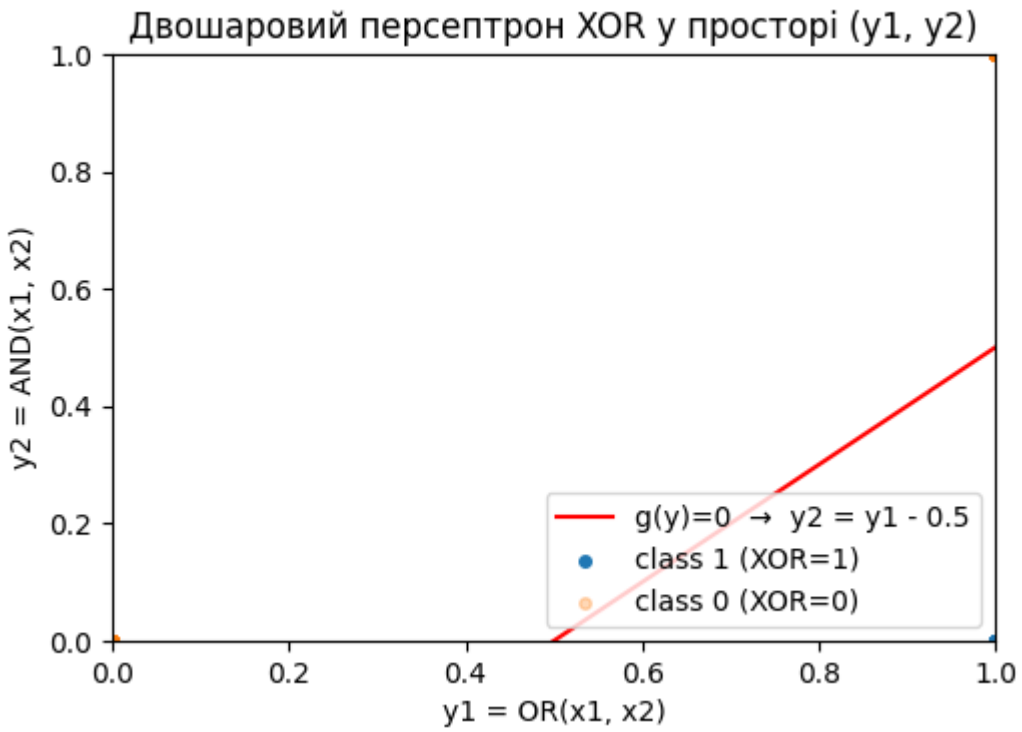

Рис. 5. Реалізація лінійного класифікатора функції XOR через рівняння розділяючої прямої

Лістинг програми:

```
import numpy as np
import matplotlib.pyplot as plt


# Нейрони першого шару: OR, AND
def neuron_or(x1: np.ndarray, x2: np.ndarray) -> np.ndarray:
    """y1 = OR(x1, x2) як лінійний класифікатор."""
    return (x1 + x2 > 0.5).astype(float)

def neuron_and(x1: np.ndarray, x2: np.ndarray) -> np.ndarray:
    """y2 = AND(x1, x2) як лінійний класифікатор."""
    return (x1 + x2 > 1.5).astype(float)


# Завд.1: XOR через OR та AND:  XOR = OR AND NOT AND
def xor_via_or_and(x1: np.ndarray, x2: np.ndarray) -> np.ndarray:
    y1 = neuron_or(x1, x2)
    y2 = neuron_and(x1, x2)
    return (y1 * (1.0 - y2)).astype(float)


# Завд.2: Другий шар. Дискримінант і вихід XOR
# g(y) = 1*y1 - 1*y2 - 0.5;  XOR = 1[g>0]
def decision_g(y1: np.ndarray, y2: np.ndarray) -> np.ndarray:
    return (1.0 * y1 - 1.0 * y2 - 0.5)

def xor_two_layer(x1: np.ndarray, x2: np.ndarray):
    """Повертає (xor_out, y1, y2, g)."""
    y1 = neuron_or(x1, x2)
    y2 = neuron_and(x1, x2)
    g = decision_g(y1, y2)
    xor_out = (g > 0).astype(float)
    return xor_out, y1, y2, g


# Демонстрація/плот
if __name__ == "__main__":
    # Випадкові дані в [0,1]×[0,1]
    RNG = np.random.default_rng(42)
    n_points = 500
    x1 = RNG.uniform(0.0, 1.0, n_points)
    x2 = RNG.uniform(0.0, 1.0, n_points)
    x_lin = np.linspace(0, 1, 201)

    # Обчислення
    xor_bool = xor_via_or_and(x1, x2)
    xor_out, y1v, y2v, g = xor_two_layer(x1, x2)

    # Перевірка еквівалентності двох реалізацій XOR
```

```
    assert np.array_equal(xor_bool, xor_out), "Реалізації XOR мають збігатися"

    # Таблиця для кутів (0/1)
    corners = np.array([[0.,0.],[0.,1.],[1.,0.],[1.,1.]])
    xor_c, y1_c, y2_c, g_c = xor_two_layer(corners[:,0], corners[:,1])
    print("x1 x2 | y1=OR y2=AND | g(y)=y1-y2-0.5 | XOR")
    for (a,b), y1i, y2i, gi, xo in zip(corners, y1_c, y2_c, g_c, xor_c):
        print(f"{int(a)}  {int(b)}  |   {int(y1i)}      {int(y2i)}   |     {gi:>5.1f}
       |  {int(xo)}")

    # Візуалізація
    fig = plt.figure(figsize=(12, 9))
    fig.subplots_adjust(left=0.08, right=0.98, top=0.93, bottom=0.10,
                        wspace=0.45, hspace=0.55)

    ax1 = fig.add_subplot(2, 2, 1)
    ax2 = fig.add_subplot(2, 2, 2)
    ax3 = fig.add_subplot(2, 2, 3)
    ax4 = fig.add_subplot(2, 2, 4)

    #  OR у (x1, x2)
    ax1.set_title("OR(x1, x2)")
    ax1.plot(x_lin, 0.5 - x_lin, 'r-')  # x1 + x2 = 0.5
    mask_or = neuron_or(x1, x2) == 1.0
    ax1.scatter(x1[mask_or], x2[mask_or], s=12, label='class 1')
    ax1.scatter(x1[~mask_or], x2[~mask_or], s=12, alpha=0.3, label='class 0')
    ax1.set_xlim(0, 1); ax1.set_ylim(0, 1)
    ax1.set_xlabel("x1"); ax1.set_ylabel("x2"); ax1.legend(loc='upper right')

    # AND у (x1, x2)
    ax2.set_title("AND(x1, x2)")
    ax2.plot(x_lin, 1.5 - x_lin, 'r-')  # x1 + x2 = 1.5
    mask_and = neuron_and(x1, x2) == 1.0
    ax2.scatter(x1[mask_and], x2[mask_and], s=12, label='class 1')
    ax2.scatter(x1[~mask_and], x2[~mask_and], s=12, alpha=0.3, label='class 0')
    ax2.set_xlim(0, 1); ax2.set_ylim(0, 1)
    ax2.set_xlabel("x1"); ax2.set_ylabel("x2"); ax2.legend(loc='upper right')

    # XOR у (x1, x2) як OR AND NOT AND (нелінійний класифікатор)
    ax3.set_title("XOR(x1, x2)")
    ax3.plot(x_lin, 0.5 - x_lin, 'r--', linewidth=1)
    ax3.plot(x_lin, 1.5 - x_lin, 'r--', linewidth=1)
    mask_xor = xor_out == 1.0
    ax3.scatter(x1[mask_xor], x2[mask_xor], s=12, label='class 1 (XOR=1)')
    ax3.scatter(x1[~mask_xor], x2[~mask_xor], s=12, alpha=0.3, label='class 0
(XOR=0)')
    ax3.set_xlim(0, 1); ax3.set_ylim(0, 1)
    ax3.set_xlabel("x1"); ax3.set_ylabel("x2"); ax3.legend(loc='upper right')

    #  Другий шар у (y1, y2) з розділяючою прямою g(y)=0
```

```
    ax4.set_title("Двошаровий персептрон XOR у просторі (y1, y2)")
    ax4.plot(x_lin, x_lin - 0.5, 'r-', label='g(y)=0  →  y2 = y1 - 0.5')
    ax4.scatter(y1v[mask_xor], y2v[mask_xor], s=16, label='class 1 (XOR=1)')
    ax4.scatter(y1v[~mask_xor], y2v[~mask_xor], s=16, alpha=0.3, label='class 0
(XOR=0)')
    ax4.set_xlim(0, 1); ax4.set_ylim(0, 1)
    ax4.set_xlabel("y1 = OR(x1, x2)"); ax4.set_ylabel("y2 = AND(x1, x2)")
    ax4.legend(loc='lower right')

    plt.show()

    print("\nРівняння розділяючої прямої другого шару:")
    print("g(y) = 1*y1 - 1*y2 - 0.5 = 0   ⇔   y2 = y1 - 0.5")
```

Висновок: В ході виконання лабораторної роботи було досліджено математичну модель нейрона.