**Лабораторна робота №3**

**Нейронна реалізація логічних функцій AND, OR, XOR**

***Мета роботи:*** Дослідити математичну модель нейрона.

**Завдання на лабораторну роботу**

**Завдання №1:** Реалізувати обчислювальний алгоритм для функції xor(x1, x2) через функції or(x1, x2) і and(x1, x2) в програмному середовищі (С++, Python, та ін.). Для реалізації обчислювальних алгоритмів рекомендується використання онлайн середовищ тестування (наприклад repl.it, trinket, і.т.д.).

Лістинг програми

def AND(x1, x2):

*return* x1 and x2

def OR(x1, x2):

*return* x1 or x2

def NOT(x):

*return* not x

def XOR(x1, x2):

*return* OR(AND(x1, NOT(x2)), AND(NOT(x1), x2))

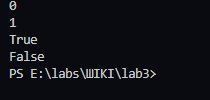
print(XOR(0, 0))

print(XOR(0, 1))

print(XOR(1, 0))

print(XOR(1, 1))

Результат виконання



**Завдання №2**: Зобразити двохслойний персептрон для функції xor(x1, x2) та скласти відповідне рівняння розділяючої прямої, використовуючи теоретичний матеріал даної лабораторної роботи. Захист лабораторної роботи передбачає виконання практичних завдань поставлених в роботі, та виконання завдань теоретичного характеру.

Персептрон - це нейронна мережа, яка є алгоритмом для виконання бінарної класифікації. Він визначає, чи стосується об'єкта певної категорії

**Логіка двошарового персептрона**

Прихований шар:

Нейрон 1: реалізує AND(x1, NOT(x2))

Нейрон 2: реалізує AND(NOT(x1), x2)

Вихідний шар:

Нейрон 3: реалізує OR(Нейрон 1, Нейрон 2)

Ваги визначають важливість кожного вхідного значення для нейрона. Якщо у вас є нейрон з кількома вхідними значеннями ​, то кожне вхідне значення множиться на відповідну вагу, а зміщення додається до зваженої суми вхідних значень перед застосуванням функції активації. Воно дозволяє моделі краще пристосовуватися до даних, зміщуючи активацію нейрона вгору або вниз

Оскільки вихідний нейрон реалізує логічну операцію OR, то загальне рівняння розділяючої прямої можна представити у вигляді: y = x

Літсинг програми

*import* numpy *as* np

*import* matplotlib.pyplot *as* plt

def step\_function(x):

*return* np.where(x >= 0, 1, 0)

inputs = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])

weights\_hidden = np.array([[1, -1], [-1, 1]])

bias\_hidden = np.array([-0.5, -0.5])

weights\_output = np.array([1, 1])

bias\_output = -0.5

def forward\_pass(x):

    hidden\_input = np.dot(x, weights\_hidden) + bias\_hidden

    hidden\_output = step\_function(hidden\_input)

    final\_input = np.dot(hidden\_output, weights\_output) + bias\_output

    final\_output = step\_function(final\_input)

*return* final\_output

def plot\_decision\_boundary():

    x = np.linspace(-0.5, 1.5, 400)

    y1 = (0.5 - 1 \* x) / -1

    y2 = (0.5 - (-1) \* x) / 1

    plt.plot(x, y1, label='Нейрон 1: x1 - x2 = 0.5')

    plt.plot(x, y2, label='Нейрон 2: -x1 + x2 = 0.5')

    plt.scatter(inputs[:, 0], inputs[:, 1], c=[forward\_pass(x) *for* x *in* inputs], cmap='bwr', edgecolor='k')

    plt.xlabel('x1')

    plt.ylabel('x2')

    plt.title('Двошаровий персептрон для функції XOR')

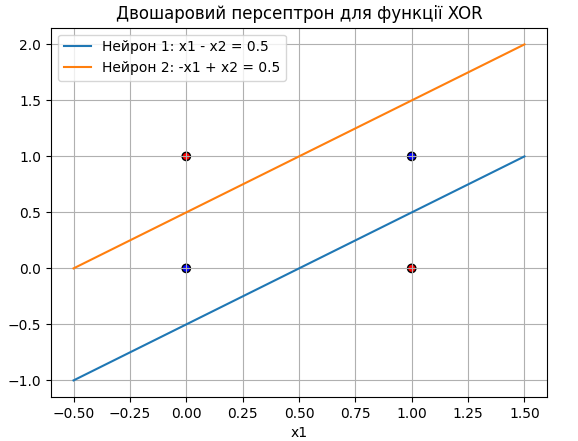
    plt.legend()

    plt.grid(True)

    plt.show()

plot\_decision\_boundary()

Результат виконання



Точки на графіку відображають вхідні дані. Точки з червоним кольором відповідають класу 1, а точки з синім кольором - класу 0. Кожна точка на графіку представляє комбінацію значень x1 та x2. Графік демонструє, як двошаровий персептрон розділяє простір на два класи за допомогою розділяючих прямих і вирішує задачу XOR

Посилання на GitHub - https://github.com/MischenchukMykola/lab3

**Висновок:** виконуючи цю лабораторну роботу я дослідив математичну модель нейрона