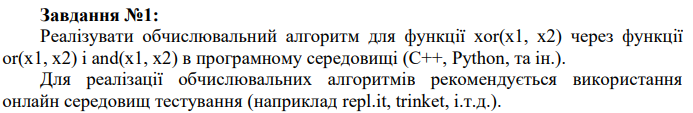
**Лабораторна робота** **№ XOR**

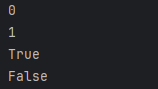
Тема: «Нейронна реалізація логічних функцій AND, OR, XOR».

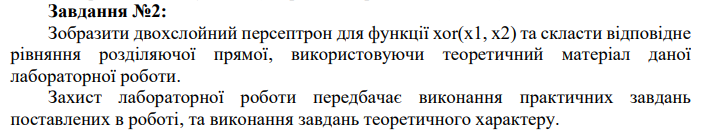
Мета: Дослідити математичну модель нейрона



# Реалізація функцій AND, OR та NOT  
def AND(x1, x2):  
 return x1 and x2  
def OR(x1, x2):  
 return x1 or x2  
def NOT(x):  
 return not x  
# Реалізація функції XOR через OR і AND  
def XOR(x1, x2):  
 return OR(AND(x1, NOT(x2)), AND(NOT(x1), x2))  
  
# Тестування функції XOR  
print(XOR(0, 0))  
print(XOR(0, 1))  
print(XOR(1, 0))  
print(XOR(1, 1))

Результат правильний:





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:

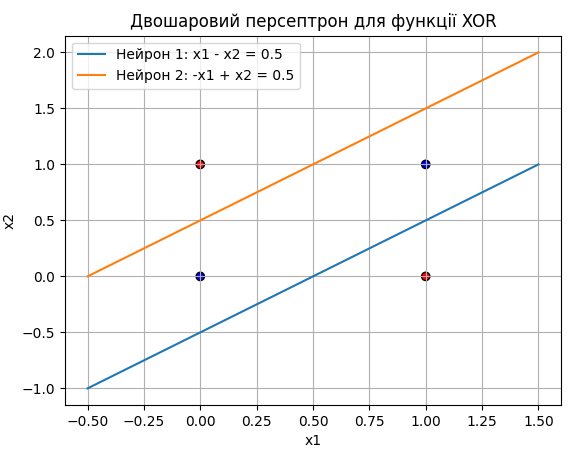
Нейрон 3: реалізує OR(Нейрон 1, Нейрон 2)

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

import numpy as np  
  
  
# Функція активації  
def step\_function(x):  
 return np.where(x >= 0, 1, 0)  
  
  
# Вхідні дані для XOR  
inputs = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])  
  
# Встановлюємо ваги та зміщення для прихованого шару  
weights\_hidden = np.array([[1, -1], [-1, 1]])  
bias\_hidden = np.array([-0.5, -0.5])  
  
# Встановлюємо ваги та зміщення для вихідного шару  
weights\_output = np.array([1, 1])  
bias\_output = -0.5  
  
  
# Обчислення значень нейронів у прихованому шарі та вихідному шарі  
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  
  
# Візуалізація рівнянь розділяючих прямих  
def plot\_decision\_boundary():  
 import matplotlib.pyplot as plt  
  
 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()



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

Ссилка на GitHub: <https://github.com/UshakowIllia/lab3Ushakow.git>

***Висновок:*** Під час виконання лабораторної роботи, я дослідив математичну модель нейрона