МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО»

ЗВІТ

з лабораторної роботи №1

з навчальної дисципліни «Проектування та реалізація програмних систем з

нейронними мережами»

|  |  |
| --- | --- |
| Виконав:  студент групи ІП-15  Мєшков Андрій Ігорович | Перевірив:  Шимкович В.М. |

Київ 2024

**ЛАБОРАТОРНА РОБОТА №1**

**Тема:** Парцептрон

**Індивідуальні завдання**

**Завдання** – Написати програму, що реалізує нейронну мережу Парцептрон

та навчити її виконувати функцію XOR.

**Хід роботи та короткі теоретичні відомості:**

Перцептрон - це проста форма штучного нейрону, основна ідея якого була запропонована Френком Розенблаттом у 1957 році. Це основний будівельний блок для багатьох нейронних мереж і є одним із перших видів штучних нейронних мереж.

***Основні відомості про перцептрон включають наступне:***

*Структура перцептрону:*

- Вхідні дані (Input): Кожен перцептрон приймає вхідні сигнали, які мають числові значення. Це можуть бути дані з сенсорів, значення функцій або будь-які інші числові дані.

- Ваги (Weights): Для кожного вхідного сигналу є вага, яка вказує, наскільки важливий є цей вхід для виходу перцептрону. Ваги показують силу зв'язку між вхідними сигналами та виходом перцептрону.

- Суматор (Summation): Вхідні сигнали помножуються на відповідні ваги і підсумовуються, щоб отримати загальний сигнал.

- Функція активації (Activation Function): Це функція, яка визначає вихідний сигнал перцептрону на основі його вхідних даних і ваг. Це може бути ступенева (як у випадку XOR), лінійна, сигмоїдна, ReLU або інша функція.

- Вихід (Output): Результат роботи перцептрону, який передається наступним шаром нейронів або є вихідним значенням для задачі класифікації або регресії.

*Робота перцептрону:*

1. Вхідні дані та ваги: Вхідні дані перемножуються на відповідні ваги.

2. Сума вагованих вхідних сигналів: Отримані значення помножені на ваги підсумовуються, утворюючи ваговану суму.

3. Функція активації: Вагована сума передається через функцію активації, яка визначає вихідний сигнал перцептрону.

4. Вихід: Отриманий вихід подається наступному шару нейронів або є кінцевим результатом від перцептрону.

*Тренування перцептрону:*

- Спадковий градієнтний спуск (Gradient Descent): У процесі навчання перцептрону, ваги поступово коригуються для зменшення помилки передбачення. Це може включати в себе розрахунок похідних функції втрати по відношенню до ваг і оновлення ваг у напрямку, який мінімізує помилку.

- Зворотнє поширення помилки (Backpropagation): Техніка, що використовується для покращення точності передбачень штучного нейрону. Після кожного випадку вхідних даних, помилка порівнюється з очікуваним результатом, і ваги перцептрону оновлюються у відповідності з цією помилкою.

**Лістинг програми:**

*import* numpy *as* np

class Perceptron:

*# Ініціалізація ваг для вхідного та вихідного шару*

def \_\_init\_\_(self, input\_size, hidden\_size, output\_size):

*self*.input\_layer = np.random.rand(input\_size, hidden\_size)

*self*.output\_layer = np.random.rand(hidden\_size, output\_size)

*# Функція активації, яка визначає виходи нейронів*

def activation\_fn(self, x):

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

*# Передбачення вихідного значення для вхідних даних x*

def predict(self, x):

*# Обчислюємо вихід прихованого шару*

hidden\_output = *self*.activation\_fn(np.dot(x, *self*.input\_layer))

*# Обчислюємо вихідний результат за допомогою вихідного шару*

output = *self*.activation\_fn(np.dot(hidden\_output, *self*.output\_layer))

*return* output

def train(self, input\_data, output\_data, epochs, learning\_rate):

*# Проходимо через кожну епохи навчання*

*for* \_ *in* range(epochs):

*# Прямий прохід*

hidden\_output = *self*.activation\_fn(np.dot(input\_data, *self*.input\_layer))

output = *self*.activation\_fn(np.dot(hidden\_output, *self*.output\_layer))

*# Зворотнє поширення помилки*

error = output\_data - output

output\_delta = error \* 1

hidden\_error = np.dot(output\_delta, *self*.output\_layer.T)

hidden\_delta = hidden\_error \* 1

*# Оновлення ваг*

*self*.output\_layer += learning\_rate \* np.dot(hidden\_output.T, output\_delta)

*self*.input\_layer += learning\_rate \* np.dot(input\_data.T, hidden\_delta)

def main():

input\_data = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])

output\_data = np.array([[0], [1], [1], [0]])

epochs = 150

learning\_rate = 0.1

input\_size = 2

hidden\_size = 2

output\_size = 1

perceptron = Perceptron(input\_size, hidden\_size, output\_size)

perceptron.train(input\_data, output\_data, epochs, learning\_rate)

*# Перевірка результатів*

print("Вивід Перцептрону для XOR:\n")

*for* i *in* input\_data:

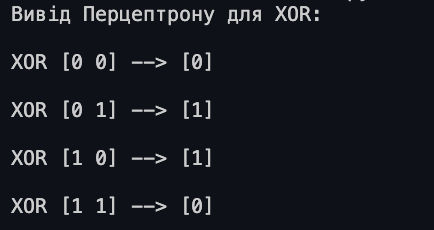
y = perceptron.predict(i)

print("XOR", i, "-->", y, '\n')

*if* \_\_name\_\_ == "\_\_main\_\_":

main()

**Результати:**



**Висновок:**

Задля виконання лабораторної роботи було розроблено нейронну мережу на базі перцептрону, якого було навчено виконувати функцію XOR. Отримані результати підтверджують правильність алгоритму організації навчання нейронної мережі.