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:

```python
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:

```
Вивід Перцептрону для XOR:

XOR [0 0] --> [0]

XOR [0 1] --> [1]

XOR [1 0] --> [1]

XOR [1 1] --> [0]
```

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.