Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота № 1

з дисципліни «Проектування та реалізація програмних систем з нейронними мережами» на тему

«Парцептрон»

ВИКОНАЛА:

Студентка групи ІМ-21

Рабійчук Дар’я Олександрівна

ПЕРЕВІРИВ:

Шимкович В. М.

Київ 2025

Завдання: Написати програму, що реалізує нейронну мережу

Парцептрон та навчити її виконувати функцію XOR для 4х змінних.

Хід роботи

Підготовка вхідних даних:

Було реалізовано функцію create_xor_dataset(), яка автоматично генерує всі можливі комбінації з 4 біт (від 0000 до 1111). Для кожного вхідного вектора обчислюється значення XOR як сума 1 у вхідному векторі за модулем 2. Також було створено два набори даних inputs (матриця 16×4 із двійковими комбінаціями вхідних значень) та outputs (результати XOR для кожного вхідного вектора).

```
def create_xor_dataset():
    data = np.array([[int(bit) for bit in f"{num:04b}"] for num in range(16)])
    labels = np.array([[np.count_nonzero(sample) % 2] for sample in data])
    return data, labels

inputs, outputs = create_xor_dataset()
```

Створення та архітектура моделі:

Використано багатошаровий перцептрон, який складається з вхідного шару з 4 нейронами та трьох прихованих шарів із 16, 12 та 8 нейронами, активованих функцією ReLU. Для бінарної класифікації було додано вихідний шар з 1 нейроном та функцією **sigmoid**.

```
model = Sequential()
model.add(Dense(16, activation='relu', input_dim=4))
model.add(Dropout(0.2))
model.add(Dense(12, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

Модель тренувалася на всіх 16 можливих комбінаціях вхідних даних на 2000 епох із розміром пакета (batch_size) 4 для покращення збіжності. Також було використано оптимізатор Adam, щоб забезпечити швидке та стабільне навчання моделі.

```
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

model.fit(inputs, outputs, epochs=2000, batch_size=4, verbose=1)

predicted_values = model.predict(inputs)
predicted_labels = (predicted_values > 0.5).astype(int)

print("\nРезультати роботи нейронної мережі:")
for inp, pred in zip(inputs, predicted_labels):
    print(f"Вхідні дані: {inp} -> Передбачене значення: {pred[0]}")
```

```
Результати роботи нейронної мережі:
Вхідні дані: [0 0 0 0] -> Передбачене значення: 0
Вхідні дані: [0 0 0 1] -> Передбачене значення: 1
Вхідні дані: [0 0 1 0] -> Передбачене значення: 1
Вхідні дані: [0 0 1 1] -> Передбачене значення: 0
Вхідні дані: [0 1 0 0] -> Передбачене значення: 1
Вхідні дані: [0 1 0 1] -> Передбачене значення: 0
Вхідні дані: [0 1 1 0] -> Передбачене значення: 0
Вхідні дані: [0 1 1 1] -> Передбачене значення: 1
Вхідні дані: [1 0 0 0] -> Передбачене значення: 1
Вхідні дані: [1 0 0 1] -> Передбачене значення: 0
Вхідні дані: [1 0 1 0] -> Передбачене значення: 0
Вхідні дані: [1 0 1 1] -> Передбачене значення: 1
Вхідні дані: [1 1 0 0] -> Передбачене значення: 0
Вхідні дані: [1 1 0 1] -> Передбачене значення: 1
Вхідні дані: [1 1 1 0] -> Передбачене значення: 1
Вхідні дані: [1 1 1 1] -> Передбачене значення: 0
```

Точність моделі:

```
Epoch 1/2000
accuracy: 0.5750 - loss: 0.7181

Epoch 200/2000
accuracy: 0.7750 - loss: 0.5418


Epoch 400/2000

accuracy: 0.9750 - loss: 0.2357
```

```
Epoch 600/2000
accuracy: 1.0000 - loss: 0.2024

Epoch 800/2000
accuracy: 0.8833 - loss: 0.2153

Epoch 1000/2000
accuracy: 0.9750 - loss: 0.1123

Epoch 1200/2000
accuracy: 1.0000 - loss: 0.0109

Epoch 1400/2000
accuracy: 0.9333 - loss: 0.1041

Epoch 1600/2000
accuracy: 1.0000 - loss: 0. 0.0467

Epoch 1800/2000
accuracy: 1.0000 - loss: 0.0419

Epoch 2000/2000
accuracy: 1.0000 - loss: 0.0025
```

Висновок

В ході роботи було реалізовано нейронну мережу типу багатошаровий перцептрон (MLP) для розв’язання задачі обчислення функції XOR для 4 вхідних змінних. Мережа була побудована з трьома прихованими шарами (16, 12, 8 нейронів) з активацією ReLU та вихідним шаром з активацією sigmoid. Використано оптимізатор Adam та функцію втрат binary_crossentropy.

Модель тренувалася протягом 2000 епох з batch_size=4. В процесі навчання спостерігалися коливання функції втрат і точності, однак до завершення навчання точність досягла 100%, а функція втрат зменшилася до 0.0025. Це свідчить про повне засвоєння функції XOR для заданих вхідних даних.