Національний технічний університет України «КПІ ім. Ігоря Сікорського»
Факультет Інформатики та Обчислювальної Техніки

Кафедра інформаційних систем та технологій

Лабораторна робота №5

з дисципліни «Методи та технології штучного інтелекту»

на тему

«Моделювання нейронної мережі Хебба»

Виконала:
студентка групи ІС-12
Павлова Софія

Викладач:
Шимкович В. М.

Київ – 2023

1. Постановка задачі

Мета: Промоделювати та дослідити нейронну мережу Хебба.

Завдання:

1. Розробіть структуру мережі Хебба, яка здатна розпізнавати чотири різні літери вашого імені або прізвища.

2. Розробіть алгоритм і програму, що моделює мережу Хебба. При цьому в алгоритмі обов'язково передбачте можливість виникнення ситуацій з нерозв'язними проблемами адаптації ваг зв'язків нейромережі.

3. Навчіть нейронну мережу Хебба розпізнаванню чотирьох заданих букв вашого імені або прізвища.

4. Продемонструйте працездатність мережі при пред'явленні навчальних зображень і зображень, що містять помилки.

2. Виконання

Структура мережі:

Створимо мережу Хебба з наступною структурою:

- **1 нейрон для кожного зображення**.

Зображень буде 4, тому **загальна кількість нейронів – 4**.

Алгоритм:

Для моделювання мережі Хебба наступний **алгоритм**:

Крок 1 – **Підготовка даних**:

Кожен вхідний шаблон (літера) розширюється додаванням одиничного значення (1) попереду. Це для врахування зсуву (*bias*) в зв'язках.

Крок 2 – **Ініціалізація ваг**:

Створюється матриця ваг *weights*, де *neurons_number* – кількість нейронів, і *len(letters[0])* – кількість входів (з урахуванням доданого одиничного значення).

Крок 3 – **Навчання зв'язків**:

Внутрішній цикл розглядає кожну літеру та кожен нейрон.

Здійснюється ітерація по вагах і ваги збільшуються на добуток відповідних елементів літери та очікуваного результату.

Крок 4 – **Розрахунок вихідного результату**:

Використовується функція *calculate_output*, яка розраховує вихідні значення для нейромережі з використанням отриманих ваг та вхідних літер.

Крок 5 – **Перевірка завершення**:

Перевіряється, чи отриманий результат (*actual_result*) співпадає з очікуваним результатом (*expected_result*).

Крок 6 – **Повернення ваг**:

Якщо умова завершення виконана, повертаються отримані вагові коефіцієнти.

Лістинг:

```
def hebbian_network(letters, expected_result, neurons_number):
    for index in range(neurons_number):
        letters[index] = [1] + letters[index]
    weights = [[0] * len(letters[0]) for _ in range(neurons_number)]
    # Розрахунок вагових коефіцієнтів
    for index_of_letter in range(neurons_number):
        for index_of_neuron in range(neurons_number):
            for index_of_weight in range(len(weights[index_of_neuron])):
                weights[index_of_neuron][index_of_weight] +=
letters[index_of_letter][index_of_weight] *
expected_result[index_of_letter][index_of_neuron]
    # Розрахунок результату розпізнавання
    actual_result = calculate_output(letters, weights, neurons_number)
    # Перевірка виконання правила для завершення
    if actual_result == expected_result:
        return weights
```

Нерозв'язна проблема адаптації ваг:

Для ситуацій з виникненням нерозв'язних проблем адаптації ваг зв'язків нейромережі передбачимо максимальну кількість ітерацій.

Якщо ж проблема не буде усунута – попереджувальне повідомлення та вихід з програми.

Лістинг:

```
def hebbian_network(letters, expected_result, neurons_number):
    [...]
    # Якщо нерозв'язна проблема адаптації ваг
    raise Exception('Виникла нерозв\'язна проблема адаптації ваг зв\'язків нейромережі.
Ваги: ' + str(weights))
```

Навчання нейромережі:

Навчимо нейромережу розпізнавати 4 літери з імені «Sofiia Pavlova». Нехай це будуть літери **«*I*», «*P*», «*L*», «*O*».** Навчальні зображення літер являють собою список формату *9*1* і виглядають наступним чином:

-1	1	-1
-1	1	-1
-1	1	-1

Літера **«*I*»**

1	1	1
1	1	1
1	-1	-1

Літера **«*P*»**

1	-1	-1
1	-1	-1
1	1	1

Літера **«*L*»**

1	1	1
1	-1	1
1	1	1

Літера **«*O*»**

У разі успішного навчання моделі, результат розпізнавання має бути таким:

	«*I*»	**«*P*»**	**«*L*»**	**«*O*»**
«*I*»	1	-1	-1	-1
«*P*»	-1	1	-1	-1
«*L*»	-1	-1	1	-1
«*O*»	-1	-1	-1	1

Рисунок 1 – Умова успішного навчання моделі

«1» означають успішне розпізнавання літери.

Лістинг:

```
def calculate_output(letters, weights, neurons_number):
    actual_result = []
    for index_of_letter in range(len(letters)):
```

```
        letter_result = []
        for index_of_neuron in range(neurons_number):
            s = 0
            for index_of_weight in range(len(weights[index_of_neuron])):
                s += weights[index_of_neuron][index_of_weight] *
letters[index_of_letter][index_of_weight]
            if s > 0:
                letter_result += [1]
            else:
                letter_result += [-1]
        actual_result += [letter_result]
    return actual_result

# Правильні літери
# Sofiia Pavlova
I_letter = [-1,  1, -1,
            -1,  1, -1,
            -1,  1, -1]
P_letter = [ 1,  1,  1,
             1,  1,  1,
             1, -1, -1]
L_letter = [ 1, -1, -1,
             1, -1, -1,
             1,  1,  1]
O_letter = [ 1,  1,  1,
             1, -1,  1,
             1,  1,  1]

# Очікуваний результат
expected_result = [[ 1, -1, -1, -1],
                   [-1,  1, -1, -1],
                   [-1, -1,  1, -1],
                   [-1, -1, -1,  1]]

# Навчання моделі
train_letters = [I_letter, P_letter, L_letter, O_letter]
number_of_neurons = len(train_letters)
final_weights = hebbian_network(train_letters, expected_result, number_of_neurons)
print('Натреновані вагові коефіцієнти:')
for weights in final_weights:
    print(weights)
```

Результат:

```
Натреновані вагові коефіцієнти:
[-2, -4, 0, -2, -4, 2, -2, -4, 0, -2]
[-2, 0, 0, 2, 0, 2, 2, 0, -4, -2]
[-2, 0, -4, -2, 0, -2, -2, 0, 0, 2]
[-2, 0, 0, 2, 0, -2, 2, 0, 0, 2]
```

Рисунок 2 – Вагові коефіцієнти натренованої моделі

У результаті отримуємо вищезазначені вагові коефіцієнти моделі.

Розпізнавання:

Розпізнаємо для початку **тренувальні зображення літер**.

Лістинг:

```
# Правильні дані
letters_no_mistakes = [I_letter, P_letter, L_letter, O_letter]
for index in range(len(letters_no_mistakes)):
    letters_no_mistakes[index] = [1] + letters_no_mistakes[index]
actual_result = calculate_output(letters_no_mistakes, final_weights, number_of_neurons)
print('\nРезультат для "I", "P", "L", "O":')
for res in actual_result:
    print(res)
```

Результат:

```
Результат для "I", "P", "L", "O":
[1, -1, -1, -1]
[-1, 1, -1, -1]
[-1, -1, 1, -1]
[-1, -1, -1, 1]
```

Рисунок 3 – Результат розпізнавання тренувальних літер

Бачимо, що результат **співпадає з очікуваним** (рисунок 1), а отже нейромережа успішно розпізнає тренувальні літери.

Тепер перевіримо працездатність моделі на **літерах з помилками**. Візьмемо літери **«*P*»** та **«*O*»**.

1	1	1
1	-1	-1
1	-1	-1

Літера **«*P*»** з помилкою

-1	1	1
1	-1	1
1	1	1

Літера **«*O*»** з помилкою

У разі успішного тестування моделі, результат має бути наступним:

	«*I*»	«*P*»	«*L*»	«*O*»
«$\overline{P}$»	-1	1	-1	-1
«$\overline{O}$»	-1	-1	-1	1

Рисунок 4 – Умова успішного тестування моделі

Лістинг:

```
# Дані з помилками
letters_with_mistakes = [P_mistake, O_mistake]
for index in range(len(letters_with_mistakes)):
    letters_with_mistakes[index] = [1] + letters_with_mistakes[index]
actual_result = calculate_output(letters_with_mistakes, final_weights, number_of_neurons)
print('\nРезультат для "P" та "O" з помилками:')
for res in actual_result:
    print(res)
```

Результат:

```
Результат для "P" та "O" з помилками:
[-1, 1, -1, -1]
[-1, -1, -1, 1]
```

Рисунок 5 – Результат розпізнавання літер з помилками

Бачимо, що результат **співпадає з очікуваним** для літер з помилками (рисунок 4), а отже нейромережа працює коректно.

Висновок:

У даній лабораторній роботі я дослідила структуру та розробила алгоритм для моделювання нейронної мережі Хебба з 4 нейронами.

Створила мережу Хебба за допомогою засобів мови програмування Python. Навчила мережу розпізнавати літери **«*I*»**, **«*P*»**, **«*L*»**, **«*O*»**. Довела працездатність мережі шляхом тестування на наборі літер з помилками ***«$\overline{P}$»***, ***«$\overline{O}$»***. Мережа розпізнала ці зображення коректно.