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.
Мельникова К.О.

Перевірив:
Шимкович В. М.

Київ – 2023

Мета роботи: промоделювати та дослідити нейронну мережу Хебба

Індивідуальні завдання

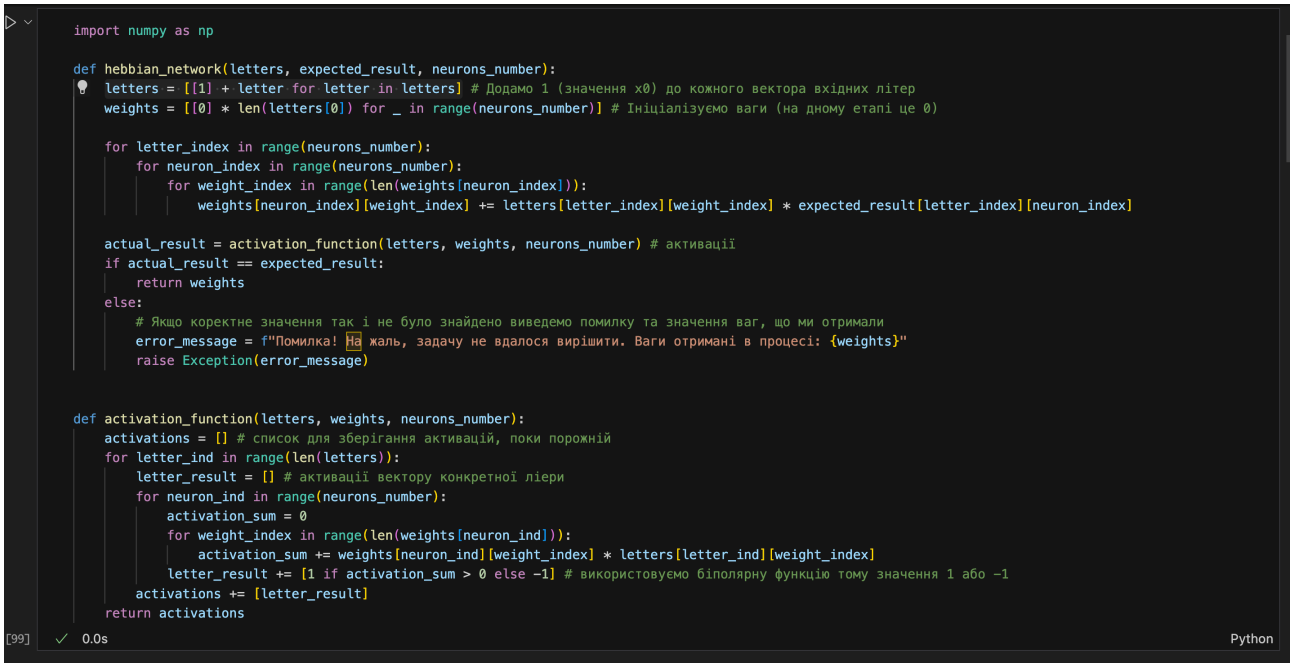
1) Розробіть структуру мережі Жебба яка здатна розпізнавати чотири різні літери вашого імені або прізвища.
2) Розробіть алгоритм і програму, що моделює мережу Хебба. При цьому в алгоритмі обов'язково передбачте можливість виникнення ситуацій з нерозв'язними проблемами адаптації ваг зв'язків нейромережі.
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.
5) Оформіть звіт по лабораторній роботі.

Хід роботи

1) Для початку визначимо літери що розпізнаватиме нейронна мережа. Нехай для мого імені і прізвища Melnykova Kateryna це будуть літери: "K" "T" "A" та "O". Для кожної літери в структурі мережі використовуватимемо по одному нейрону. Тобто, всього знадобиться 4 нейрони.

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

2) Тепер за допомогою мови програмування Python розробимо алгоритм, що моделює мережу Хебба. Для випадків з виникненням нерозв'язних проблем адаптації ваг зв’язків нейромережі передбачимо повернення помилки та виведення повідомлення про вихід:



```python
import numpy as np

def hebbian_network(letters, expected_result, neurons_number):
    letters = [[1] + letter for letter in letters] # Додамо 1 (значення x0) до кожного вектора вхідних літер
    weights = [[0] * len(letters[0]) for _ in range(neurons_number)] # Ініціалізуємо ваги (на дному етапі це 0)

    for letter_index in range(neurons_number):
        for neuron_index in range(neurons_number):
            for weight_index in range(len(weights[neuron_index])):
                weights[neuron_index][weight_index] += letters[letter_index][weight_index] * expected_result[letter_index][neuron_index]

    actual_result = activation_function(letters, weights, neurons_number) # активації
    if actual_result == expected_result:
        return weights
    else:
        # Якщо коректне значення так і не було знайдено виведемо помилку та значення ваг, що ми отримали
        error_message = f"Помилка! На жаль, задачу не вдалося вирішити. Ваги отримані в процесі: {weights}"
        raise Exception(error_message)


def activation_function(letters, weights, neurons_number):
    activations = [] # список для зберігання активацій, поки порожній
    for letter_ind in range(len(letters)):
        letter_result = [] # активації вектору конкретної ліери
        for neuron_ind in range(neurons_number):
            activation_sum = 0
            for weight_index in range(len(weights[neuron_ind])):
                activation_sum += weights[neuron_ind][weight_index] * letters[letter_ind][weight_index]
            letter_result += [1 if activation_sum > 0 else -1] # використовуємо біполярну функцію тому значення 1 або -1
        activations += [letter_result]
    return activations
```

[99] ✓ 0.0s Python

3-4) Навчимо мережу та випробуємо її на коректних даних та даних з помилками:

Спочатку створимо матриці для очікуваних результатів та списки для літер:

```python
K = [1, -1, 1,
     1, 1, -1,
     1, -1, 1]
T = [1, 1, 1,
     -1, 1, -1,
     -1, 1, -1]
A = [-1, 1, -1,
     1, 1, 1,
     1, -1, 1]
O = [1, 1, 1,
     1, -1, 1,
     1, 1, 1]

expected_result = [ # у випадку з коректними даними очікуваний результат – одинична матриця
[1, -1, -1, -1], # K
[-1, 1, -1, -1], # T
[-1, -1, 1, -1], # A
[-1, -1, -1, 1] # O
]
```

✓ 0.0s

Натренуємо модель та перевіримо її на коректних даних:

```
train_letters = [K, T, A, O]
number_of_neurons = len(train_letters)
final_weights = hebbian_network(train_letters, expected_result, number_of_neurons)
print('Вагові коефіцієнти натренованої моделі:')
for weight in final_weights:
    print(weight)

def print_results(expected, recognized):
    letters = ['K', 'T', 'A', 'O']
    expected_letter = letters[np.argmax(expected)]
    recognized_letter = letters[np.argmax(recognized)]
    print(f"Очікувано: {expected_letter} | Розпізнано: {recognized_letter} " f"| {True if expected_letter == recognized_letter else False}")

correct_letters = [K, T, A, O]
correct_letters = [[1] + letter for letter in correct_letters]
actual_result = activation_function(correct_letters, final_weights, number_of_neurons)

print('\nМатриця для коректних даних (літери "K", "T", "A" та "O"):')
for res in actual_result:
    print(res)

print('\nРезультати для коректних даних:')
for i, res in enumerate(actual_result):
    print_results(expected_result[i], res)
```

[105] ✓ 0.0s

Результати:

```
Вагові коефіцієнти натренованої моделі:
[-2, 0, -4, 0, 0, 0, -2, 0, -2, 0]
[-2, 0, 0, 0, -4, 0, -2, -4, 2, -4]
[-2, -4, 0, -4, 0, 0, 2, 0, -2, 0]
[-2, 0, 0, 0, 0, -4, 2, 0, 2, 0]

Матриця для коректних даних (літери "K", "T", "A" та "O"):
[1, -1, -1, -1]
[-1, 1, -1, -1]
[-1, -1, 1, -1]
[-1, -1, -1, 1]

Результати для коректних даних:
Очікувано: K | Розпізнано: K | True
Очікувано: T | Розпізнано: T | True
Очікувано: A | Розпізнано: A | True
Очікувано: O | Розпізнано: O | True
```

Тепер перевіримо модель на даних з помилками:

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

```
K_error = [1, -1, -1,
           1, 1, -1,
           1, -1, 1]
T_error = [1, 1, 1,
           -1, 1, -1,
           -1, 1, 1]
A_error = [-1, 1, -1,
           1, -1, 1,
           1, -1, 1]
O_error = [1, -1, 1,
           1, -1, 1,
           1, 1, 1]

error_letters = [K_error, T_error, A_error, O_error]
error_letters = [[1] + letter for letter in error_letters]
actual_result_with_mistakes = activation_function(error_letters, final_weights, number_of_neurons)

print('\nМатриця для даних з помилками (літери "K", "T", "A" та "O"):')
for res in actual_result_with_mistakes:
    print(res)

print('\nРезультати для даних з помилками:')
for i, res in enumerate(actual_result_with_mistakes):
    print_results(expected_result[i], res)
```

[106] ✓ 0.0s

Результат:

```
Матриця для даних з помилками (літери "K", "T", "A" та "O"):
[1, -1, -1, -1]
[-1, 1, -1, -1]
[-1, -1, 1, 1]
[-1, -1, -1, 1]

Результати для даних з помилками:
Очікувано: K | Розпізнано: K | True
Очікувано: T | Розпізнано: T | True
Очікувано: A | Розпізнано: A | True
Очікувано: O | Розпізнано: O | True
```

Висновок:

Отже, під час виконання даної лабораторної роботи я дослідила та створила нейронну мережу Хебба з чотирма нейронами, що відповідають кількості вхідних зображень. За допомогою цієї нейронної мережі я розпізнала зображення з літерами свого імені та прізвища K T A та O. Результати розпізнавання були досить вдалими, створена нейронна мережа впоралась навіть із зображеннями, що містили певні помилки. Тобто, з цього можна зробити висновок, що нейронна мережа Хебба є доволі ефективним інструментом для вирішення задач такого роду.