110502201 資工三 A 蔡皓宇 hw3-hopfield

一、程式簡介、須包含實作架構 (Hopfield)

Hopfield.py

class-Hopfield 為網路架構，具有

adjust_weight():使用 Kronecker product 來調整鍵結值的改變量

hop_train():將訓練資料的每筆圖形都丟入並用來訓練鍵結值

hop_run():將測試資料的圖形丟入並返回聯想之結果

hw3_main.py

load_data():負責讀取選到的檔案資料，並將圖樣轉成 0/1 bit 的陣列

Hop_Run():會先將選到的{header}(Basic, Bonus, Noise).Training.txt 做剖析

接著將獲得的圖形 0/1 bit 陣列丟入 class-Hopfield 的 hop_train()去訓練

最後再將{header}.Testing.txt 的資料丟入 class-Hopfield 的 hop_run()

搭配 plot_2D_data()在終端機印出回想結果(將 1 轉成*，看起來比較清楚)、

store_result()則把終端機顯示的結果寫入 hw3_{header}_output.txt

(Tkinter 想定時刷新 window 的 label 會出現很多問題，出錯很久後決定寫入

一個 txt 就好，對於檢視結果也比較方便和清楚)

main():負責 Hop_Run()的執行

GUI 介面標籤和按鈕寫在 if __name__ == "__main__":

只要選取下拉選單的 header 後，並按下 Run Hopfield 的按鈕即可執行

Hopfield.py

```python
import numpy as np

unsigned_int = np.uint64

class Hopfield:
    def __init__(self, image_num, n):
        self.image_num, self.n = image_num, n
        self.w = np.zeros([n, n])

    def adjust_weight(self, data_arr, data_arr_mean):
        adjust = np.zeros([self.n, self.n])
        for i in range(self.n):
            # Kronecker product
            adjust[i] = (data_arr - data_arr_mean)[i] * (data_arr - data_arr_mean)

        return adjust / (self.n ** 2) / (data_arr_mean * (1 - data_arr_mean))

    # train hopfield
    def hop_train(self, train_arr):
        for i in range(self.image_num):
            data_arr = train_arr[i]
            data_arr_mean = float(data_arr.sum()) / len(data_arr)
            # adjust hopfield's weight
            self.w = self.w + self.adjust_weight(data_arr, data_arr_mean)
```

```python
            # weight's diagonal set to 0
            idx = range(0, self.n)
            self.w[idx, idx] = 0.0

    # run hopfield to output
    def hop_run(self, data_arr):
        for i in range(self.image_num):
            data_matrix = np.tile(data_arr, (self.n, 1))
            data_matrix = self.w * data_matrix
            ouput_arr = data_matrix.sum(axis = 1)
            # normalization
            ouput_arr = (ouput_arr - float(np.amin(ouput_arr))) / \
                        (float(np.amax(ouput_arr)) - float(np.amin(ouput_arr)))

            ouput_arr[ouput_arr > 0.5] = 1.0
            ouput_arr[ouput_arr <= 0.5] = 0.0

            return ouput_arr
```

hw3_main.py

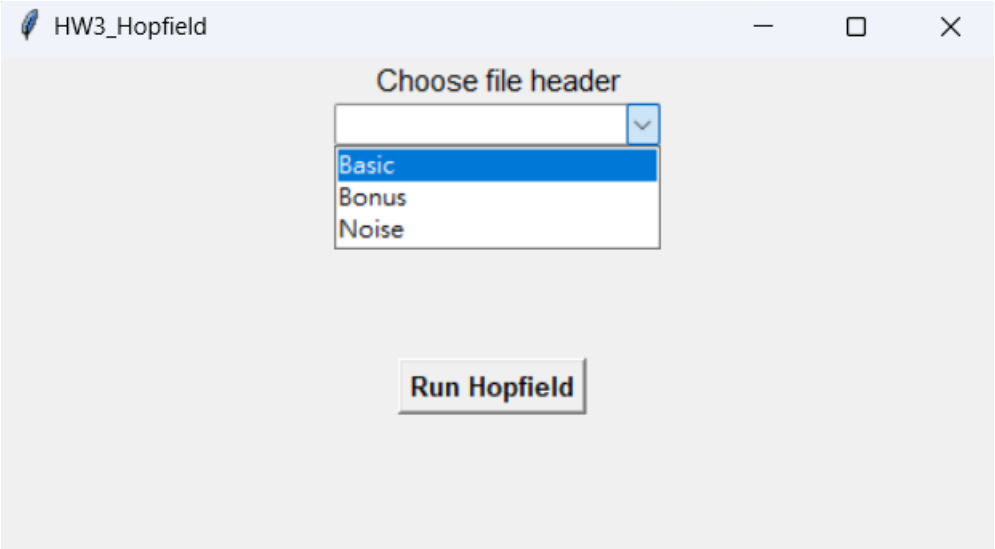
```python
from Hopfield import Hopfield
from tkinter import ttk
from tkinter import *
import numpy as np
import os

def combo_box_on_select(event):
    global selected_header
    selected_header.set(combo_box.get())
    print(f'The chosen header_file is {selected_header.get()}.')

unsigned_int = np.uint64
data_path = os.path.join(os.getcwd(), "Hopfield_dataset")
hw3_dataset_list = ['Basic', 'Bonus', 'Noise']

def plot_2D_data(arr, row, col):
    global figure
    figure = ''
    for i in range(row):
        temp = ''
        for j in range(col):
            idx = i * col + j
            # chage token '1' to '*'(more comfortable to look)
            if arr[idx] == 1.0: temp += '*'
            else: temp += ' '
        figure += (f'{temp}\n')
        print(temp)


def convert_to_bit(arr, row, col):
    temp_image_idx, temp_col_ct = 0, 0
    for token in data_read:
        if token == '\n': continue
        elif token == '1' or token == ' ':
            if token == '1': arr[temp_image_idx][temp_col_ct] = 1
            else: arr[temp_image_idx][temp_col_ct] = 0
            temp_col_ct += 1
        else: pass

        if temp_col_ct == row * col:
            temp_image_idx += 1
            temp_col_ct = 0

    return arr

# load data and convert to 0/1 bit
def load_data(file_name):
    global data_read
    file_name_path = os.path.join(data_path, file_name)
    with open(file_name_path, 'r') as f:
        data_read = f.read()
```

```
    image_num = len(data_read.split('\n\n')) # how many input data(# of image)
    col_num = len(data_read.split('\n')[0])
    row_num = int((len(data_read.split('\n\n')[0]) + 1) / (col_num + 1))

    train_arr = np.zeros((image_num, row_num * col_num), dtype = unsigned_int)
    train_arr = convert_to_bit(train_arr, row_num, col_num)

    return row_num, col_num, train_arr


def store_result(fig_buffer):
    with open(f'hw3_{selected_header.get()}_output.txt', 'w') as f:
        for i in range(len(fig_buffer)):
            if i != 0:
                print('----------------------------', file = f)
            print(f'Training data:\n\n{fig_buffer[i][0]}', file = f)
            print(f'Testing data:\n\n{fig_buffer[i][1]}', file = f)
            print(f'Recall result:\n\n{fig_buffer[i][2]}', file = f)
            # check this image's recall T or F
            print(f'Recall success? : {fig_buffer[i][0] == fig_buffer[i][2]}', file = f)


def Hop_Run():
    header = selected_header.get()
    fig_buffer = []
```

```
    # input training data to train hopfield
    row_num, col_num, train_arr = load_data(f'{header}_Training.txt')
    hop = Hopfield(train_arr.shape[0], train_arr.shape[1]) # image_num, row_num * col_num
    hop.hop_train(train_arr)
    # input testing data
    row_num, col_num, test_arr = load_data(f'{header}_Testing.txt')
    for i in range(hop.image_num):
        Training, Testing = train_arr[i], test_arr[i]
        recall = hop.hop_run(Testing)
        # show original data and the recall data
        print('----------------------------')
        print('Training data:\n')
        plot_2D_data(Training, row_num, col_num)
        temp_train_figure = figure
        print('\nTesting data:\n')
        plot_2D_data(Testing, row_num, col_num)
        temp_test_figure = figure
        print('\nRecall result:\n')
        plot_2D_data(recall, row_num, col_num)
        temp_recall_figure = figure
        # check this image's recall T or F
        print(f'\nRecall success? : {temp_train_figure == temp_recall_figure}')
        fig_buffer.append([temp_train_figure, temp_test_figure, temp_recall_figure])

    store_result(fig_buffer)
```

```python
def main():
    Hop_Run()


# GUI
if __name__ == "__main__":

    window = Tk()
    window.title("HW3_Hopfield")
    window.geometry("500x250")

    combo_box_frame = Frame(window)
    combo_box_label = Label(window, text = "Choose file header", font = ("Arial", 11, "normal"))
    combo_box_label.pack()
    # ttk combobox, contents = hw3_dataset_list
    combo_box = ttk.Combobox(window, values = hw3_dataset_list)
    combo_box.pack(side = TOP)
    # binding event, trigger "on_select" function when the selection changes
    combo_box.bind("<<ComboboxSelected>>", combo_box_on_select)
    selected_header = StringVar() # combobox string buffer, store file_header

    button_run = Button(window, text = 'Run Hopfield', font = ("Arial", 10, "bold"), command = main)
    button_run.place(x = 200, y = 150)

    window.mainloop()
```

二、程式執行說明。(如何操作、使用)



如同前面所述，開啟 GUI 選好 header 後按下 Run Hopfield 做訓練

在終端機即可看到印出的回想結果，分別為

Training data、Testing data、Recall result、Recall success? : (True/False)

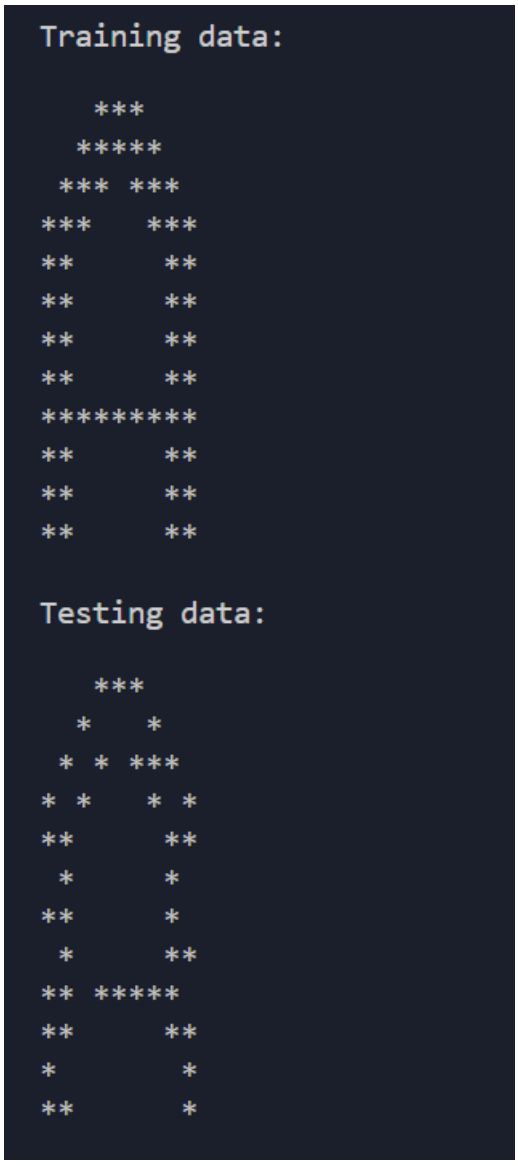
※ Recall success? : 為比對 Training data == Recall result ? 的結果

資料夾也會出現 hw3_{header}_output.txt 可供檢視執行的結果，如下:

hw3_Basic_output.txt	2023/12/6 下午 08:22	文字文件	2 KB
hw3_Bonus_output.txt	2023/12/6 下午 08:22	文字文件	7 KB
hw3_Noise_output.txt	2023/12/8 下午 12:41	文字文件	2 KB

三、實驗結果(所有資料集都須有實驗結果集說明)

1. Basic : Hopfield 聯想輸出與訓練資料 -> 3 個匹配 ; 0 個不匹配



```
Training data:

    *****
   *******
***
***
***
***
***
***
***
***
   *******
    *****

Testing data:

    *****
   ** *
***
* *****
* *   *
* *
* *   *
* *
*
***
   *** *
    **  *
```

```
Recall result:

    *****
   *******
***
***
***
***
***
***
***
***
   *******
    *****

Recall success? : True
```

```
Training data:

***
***
***
***
***
***
***
***
***
***
*********
*********

Testing data:

***
* *
* *
***   ***

***   ***

      ***
***
* ** ** *
* * * **
** ** **
```

```
Recall result:

***
***
***
***
***
***
***
***
***
***
*********
*********

Recall success? : True
```

2. Bonus : Hopfield 聯想輸出與訓練資料 -> 6 個匹配 ; 9 個不匹配

```
Training data:

*****
*****
*****
*****
*****
     *****
     *****
     *****
     *****
     *****

Testing data:

*****
*  **
*****

*****
     ** **
     *****
       ***
     **  *
     * ***
```

```
Recall result:

*****
*****
*****
*****
*****
     *****
     *****
     *****
     *****
     *****

Recall success? : True
```

```
Training data:

*  *  *  *
 *  *  *
  *  *  *
*  *  *  *
 *  *  *
  *  *  *
*  *  *  *
 *  *  *
  *  *  *
*  *  *  *

Testing data:

*  *  *  *

  *  *  *
*        *
 *  *
  *     *
*     *  *
 *     *
  *     *
*  *     *
```

```
Recall result:

*  *  *  *
 *  *  *
  *  *  *
*  *  *  *
 *  *  *
  *  *  *
*  *  *  *
 *  *  *
  *  *  *
*  *  *  *

Recall success? : True
```

```
Training data:

**********
*        *
* ****** *
* *    * *
* * ** * *
* * ** * *
* *    * *
* ****** *
*        *
**********

Testing data:

* ***   *
*       *
* *** ** *
*     * *
* * ** * *
  * ** * *
* *     *
* **  ** *
*       *
* ** * **
```

```
Recall result:

**********
*        *
* *** ** *
* *    * *
* * **   *
*   ** * *
* *    * *
* ** *** *
*        *
**********

Recall success? : False
```

```
Training data:

 * * * * *
* * * * *
 * * * * *
* * * * *
 * * * * *

Testing data:

 *       *
*       *
 *   * * *
* *   * *
 * * * * *
```

```
Recall result:

* * * *
 *
* * *
 * *     *
* * *
 * * * * *
* * * * *
 * * * * *
* * * * *
 * * * * *

Recall success? : False
```

```
Training data:

* * * * *
 * * * * *
* * * * *
 * * * * *
* * * * *
 * * * * *
* * * * *
 * * * * *
* * * * *
 * * * * *

Testing data:

* * * * *
    *   * *
* *   * *
 *     * *
*       *
 *     * *
*  *     *
 * * * * *
* * * * *
 * * * * *
```

```
Recall result:

* * * * *
 *   * * *
* * * * *
 * *   * *
* * *   *
 * * * * *
* * * * *
 * * * * *
* * * * *
 * * * * *

Recall success? : False
```

```
Training data:

*** *    *
 *  *** **
  * * ** *
 ***   *
**  *  ***
 * ***   *
* **  *  *
   * ***
** ***  *
 *  *  ***

Testing data:

**  *    *
 *  *   **
  * * ** *
 * *   *
*   *  ***
 * * *   *
* **  *  *
   * * *
*  * * *
 *  *  * *
```

```
Recall result:

*** *   **
 *  **   *
  * * **
  **   *
**  *   **
 * ***   *
* **  *
   * ***
**  **  *
 * *** ***

Recall success? : False
```

```
Training data:

**  **  **
**  **  **
  **  **
  **  **
**  **  **
**  **  **
  **  **
  **  **
**  **  **
**  **  **

Testing data:

    **  **
    **  **
   *  **
   *  **
**  **   *
**  **   *
   *  **
   *  **
**   *   *
**  **   *
```

```
Recall result:

**  **  **
**  **  **
  **  **
  **  **
**  **  **
**  **  **
  **  **
  **  **
**  **  **
**  **  **

Recall success? : True
```

```
Training data:

*****
*   * ***
*   * ***
*   * ***
*****
     *****
 *** *   *
 *** *   *
 *** *   *
     *****

Testing data:

**  *
    * * *
    * * *
*   * * *
** **
     ** **
 * * *
 * * *
 * * *   *
     **  *
```

```
Recall result:

*****
**  * ***
       **
*  ** * *
*****
     ** **
 * * **  *
   *
 *** *  **
      * **

Recall success? : False
```

```
Training data:

*****
*****
*****
*****
*****
     *****
     *****
     *****
     *****
     *****

Testing data:

** **
*   *
*   *
** **
*****
     *****
     **  *
     *   *
     *  **
     *****
```

```
Recall result:

*****
** **
*****
*****
*****
     *****
     *****
     *** *
     *****
     *****

Recall success? : False
```

```
Training data:

*  ****  *
**  ****
***  ****
****  ****
 ****  ***
  ****  **
*  ****  *
**  ****
***  ****
****  ****

Testing data:

*  ****  *
**  ** *
* *   * *
* **  ** *
 ****  * *
  * **  **
*  * **  *
**  *  *
* *  *  *
* **  *  *
```

```
Recall result:

*  ** *  *
 *  *  *
  *  *  *
*  *  *  *
 *  *  *
  *  *  *
*  *  *  *
 *  *  *
  *  *  *
*  *  * **

Recall success? : False
```

```
Training data:

*  *  *  *
 *  *  *
  *  *  *
*  *  *  *
 *  *  *
  *  *  *
*  *  *  *
 *  *  *
  *  *  *
*  *  *  *

Testing data:

*  *  *  *
       *
     *  *
*       *
 *  *  *
  *  *  *
*     *  *
 *     *
  *     *
*  *  *  *
```

```
Recall result:

*  *  *  *
 *  *  *
  *  *  *
*  *  *  *
 *  *  *
  *  *  *
*  *  *  *
 *  *  *
  *  *  *
*  *  *  *

Recall success? : True
```

```
Training data:

**********
*        *
*        *
*        *
*   **   *
*   **   *
*        *
*        *
*        *
**********

Testing data:

**********
*        *
*        *
*        *
*        *
*        *
*        *
*        *
*        *
**********
```

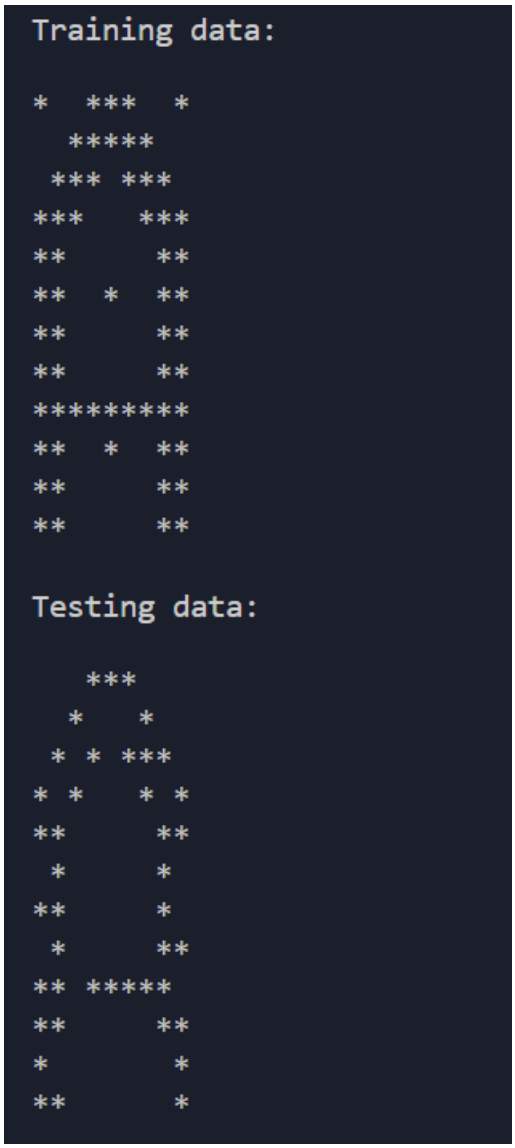
```
Recall result:

**********
*        *
* ****** *
* *    * *
* * ** * *
* * ** * *
* *    * *
* ****** *
*        *
**********

Recall success? : False
```

```
Training data:

**********
*        *
* ****** *
* *    * *
* * ** * *
* * ** * *
* *    * *
* ****** *
*        *
**********

Testing data:

*** ******
*        *
* ** *** *
* *    * *
*      * *
*        *
* *    * *
* ** *** *
*        *
** ***  **
```

```
Recall result:

**********
*        *
* *** ** *
* *    * *
* * ** * *
* * ** * *
* *    * *
* ****** *
*        *
**********

Recall success? : False
```

***Noise : 自行將訓練資料集加入雜訊**

3. Noise : Hopfield 聯想輸出與訓練資料 -> 3 個匹配 ; 0 個不匹配



```
Training data:

    *****
   *******
***
***
***    ***
***    * *
***    * *
***    ***
***
***
   *******
    *****

Testing data:

    *****
   ** *
***
* *****
* *    *
* *
* *    *
* *
*
***
   *** *
    **  *
```

```
Recall result:

    *****
   *******
***
***
***    ***
***    * *
***    * *
***    ***
***
***
   *******
    *****

Recall success? : True
```

```
Training data:

***   ***
***
***
***    *
***
***    *
***
***
***   ***
***
*********
*********

Testing data:

***
* *
* *
***   ***

***   ***

      ***
***
* ** ** *
* * * **
** ** **
```

```
Recall result:

***   ***
***
***
***    *
***
***    *
***
***
***   ***
***
*********
*********

Recall success? : True
```

四、實驗結果分析及討論

* Basic 聯想結果 100%正確，Bonus 則是 40%正確, 60%錯誤

Bonus 推測是因為需要記憶的訓練量從 Basic 的 3 個圖像躍升成 15 個，

加上訓練的 15 個圖像有些差異頗大，才導致 Hopfield 較無法準確去聯想

而 Basic 只需記憶 3 個訓練輸入，因此準確率非常高

* Noise(添加雜訊)的聯想結果也是 100%正確

Noise 的訓練輸入設置是 3 個，因此成功回想在預期之內

我有試著改變測試資料的圖樣，經觀察，發現除非將測試資料點之間大幅離散

化或圖樣改成與訓練圖樣差非常多，否則對於較少訓練輸入的資料集，在原先

的測試資料上增加些微雜訊，基本上聯想率仍然是 100%正確

五、加分項目

1. Bonus 資料集的訓練與測試

2. 自行將訓練資料集加入雜訊(Noise 資料集)，並能夠正確回想

※ 這次作業 zip 有額外放了一個 NN_HW2_DataSet，因為沒有這個 file

作業二的 exe 不確定助教能否執行，所以在這次作業補上