類神經網路 Neural Networks

作業一

學號 112522101 資工碩一

姓名 吳宥俞

目錄 Outline

1. 基本題

1.1 GUI 功能及程式流程

GUI 部分採用 Tkinter 套件呈現，並使用 PAGE 圖型編輯器完成 UI 外觀。

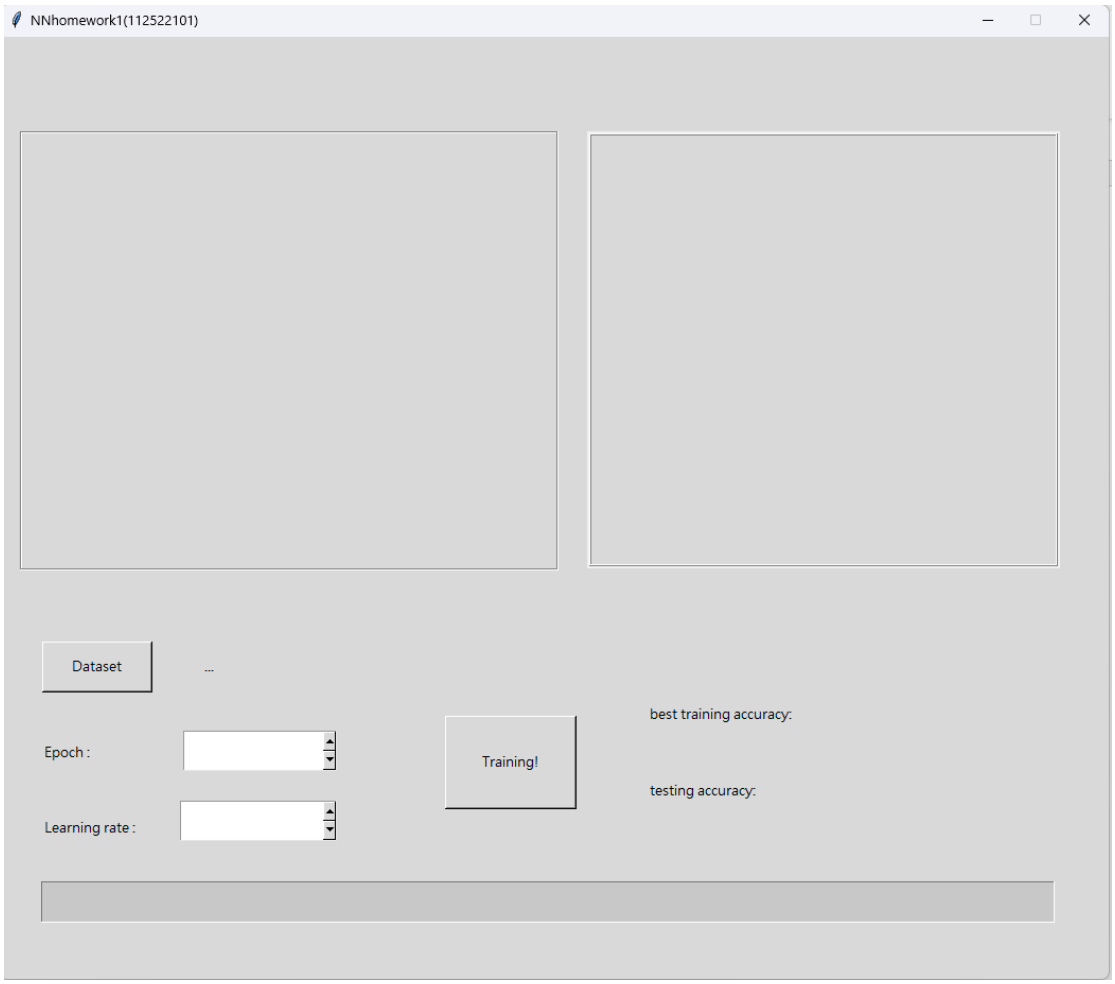


圖 1 初始外觀

使用方法:

i. 點擊 Dataset 按鈕，獲得資料集路徑。

ii. 輸入 Epoch、Learning rate 數值。

iii. 按下 Training! 按鈕，開始訓練，下方有進度條，可得知訓練進度。

核心程式流程:

i. 當按下 Training! 按鈕時，會去取得輸入的路徑、Epoch、學習率數值，並將參數傳進 main function 進行訓練。

ii. 在訓練過程中，會將**每次訓練的結果**更新於**左邊的方框**，當中**顯示分類結果、鍵結值、訓練正確率**。

iii. 訓練完成後，會將**測試資料**的分類結果展示於**右邊的方框**，當中**顯示分類結果、鍵結值、測試正確率**，並在右下方顯示最佳的訓練正確率、測

試正確率。

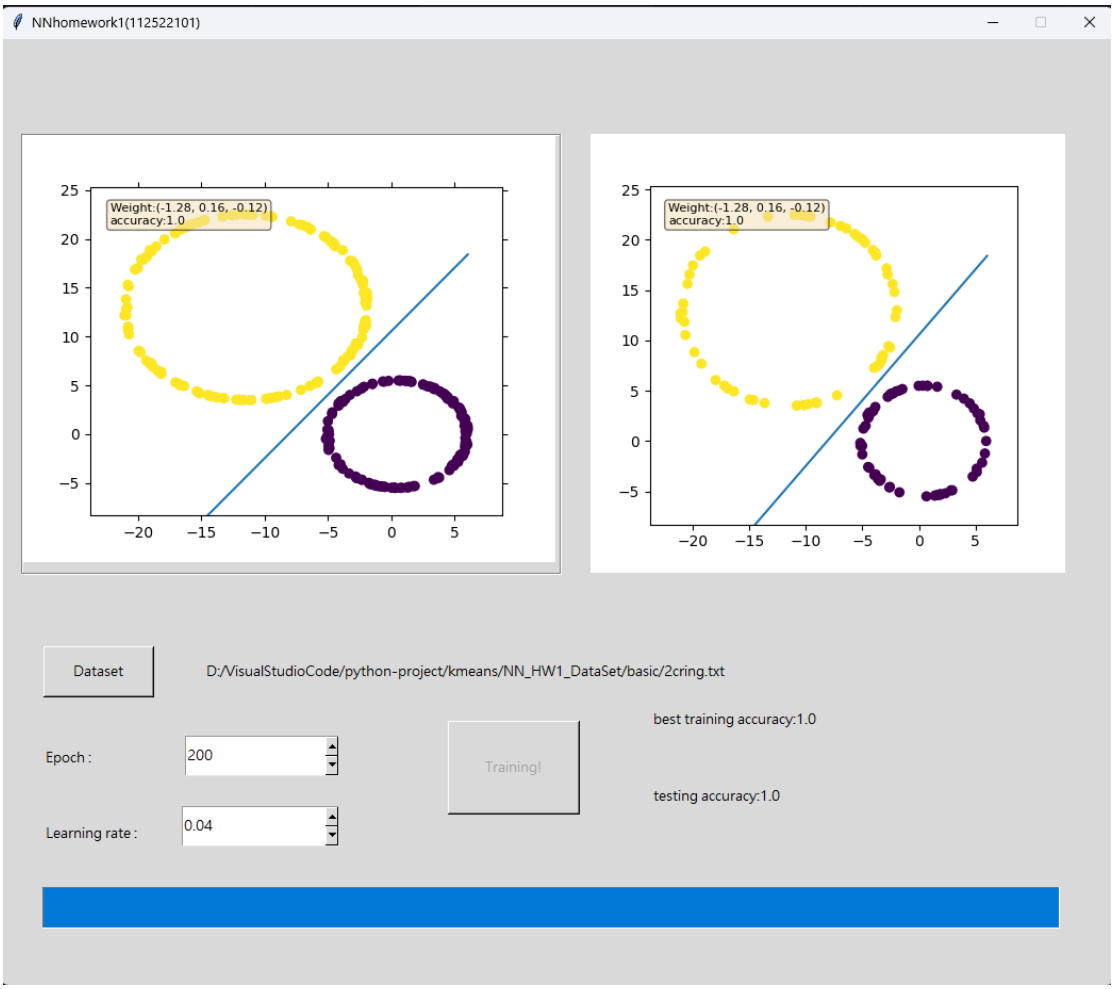


圖 2 流程完成圖(**註:左方分類的線會隨著訓練結果一直更新**)

1.2 主要 function 說明

作品架構主要有 3 個檔案，3 個檔案皆相互 import:

i. UIhw1.py:UI 外觀程式碼(大部分都是 PAGE 自動生成)。

ii. UIhw1_support.py:事件反應程式碼(如按鈕點擊後會做甚麼)

iii. NeuralNetworkHW1.py:**感知機及作圖程式碼**

UIhw1.py、UIhw1_support.py 重點程式碼:

```
def trainbtn(*args):
    if _debug:
        #獲取路徑、epoch、learning rate的數據
        path = str(_w1.l1.get())
        epoch = int(_w1.Spinbox1.get())
        lr = float(_w1.Spinbox2.get())
        #呼叫程式
        NeuralNetworkHW1.trainAndTest(path,epoch,lr)
```

(ii.)中的 Training!按鈕，觸發後將參數傳入(iii.)

NeuralNetworkHW1.py 重點程式碼:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
import tkinter
import UIhw1_support #tkinter的UI檔案
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg #tkinter導入matplotlib
import matplotlib.pyplot as plt
from matplotlib.figure import Figure
```

使用到 pandas、numpy、train_test_split、tkinter、matplotlib 套件

```
def trainAndTest(Path,Epoch,Lr):
    df=pd.read_table(Path,header=None,sep=" ")
    df_data = pd.DataFrame(data=np.c_[df],columns= ['x1','x2','y'])
    df_data.insert(0,'threshold',-1) #神經元之閥值為-1

    X = df_data.drop(labels=['y'],axis=1).values #取二維資料
    Y = df_data['y'].values #取答案
    Yset = list(set(Y))
    X_train, X_test, y_train, y_test = train_test_split(X, Y,
                                test_size=0.3, random_state=40, stratify=Y)#隨機切割資料集(7:3)
```

利用 pandas 讀取輸入資料，並將每筆資料**最左邊加上閥值-1**，並將資料集切分成(訓練 7:測試 3)

```
#開始Training!
lr = Lr
epoch = Epoch
w = np.array([-1,0,1]) #初始w為[0,1] 通過(0,-1)的水平線:0*x1+1*x2+1=0
k=0
trainAccuracy=0
bestTrainAccuracy=0
bestw=list()
```

步驟一:網路初始化。初始鍵結值 w 設為[-1, 0, 1](過(0, -1)的水平線)

```
for i in range(epoch):
    #將UI的進度條加在for迴圈裡面
    UIhw1_support._w1.progressbtn.set(i*(100/epoch))
    #如果正確率==1就停止訓練
    if bestTrainAccuracy==1:
        UIhw1_support._w1.progressbtn.set(100)
        break
    #Epoch大於資料數量時就循環資料
    if k==X_train.shape[0]-1:
        k=0
```

重複步驟二～五，直到達到 epoch 次數或是訓練正確率達 1.0。

```
vj=np.dot(w,X_train[k]) #計算w跟x的內積
yj = Yset[0] if vj>=0 else Yset[1] #激勵函數(sgn二分法)
```

步驟二：計算網路輸出值。$v_j = \left(\sum w_{ji} - x_i\right) - \theta$（$\theta$已經先寫在 x 裡面了）

$y_j = \varphi(v_j)$ 以是否>=0 來分兩類

```
#更新鍵結值
if yj == Yset[0] and y_train[k]==Yset[1]:
    w = w - (lr*X_train[k])
if yj == Yset[1] and y_train[k]==Yset[0]:
    w = w + (lr*X_train[k])
```

步驟三：調整鍵結值向量。

$$\underline{w}(n+1) = \begin{cases} \underline{w}(n) + \eta \underline{x}(n) & \text{如果} \underline{x}(n) \in C_1 \text{和} \underline{w}^T(n)\underline{x}(n) < 0 \\ \underline{w}(n) - \eta \underline{x}(n) & \text{如果} \underline{x}(n) \in C_2 \text{和} \underline{w}^T(n)\underline{x}(n) \geq 0 \end{cases}$$

```
#算訓練時的正確率，並記錄最好的weight
correct=0
for j in range(X_train.shape[0]):
    tmp = np.dot(w,X_train[j])
    tmp = Yset[0] if tmp>=0 else Yset[1]
    if tmp == y_train[j]:
        correct+=1
tmpAccuracy = round(correct/X_train.shape[0],3)
if tmpAccuracy>bestTrainAccuracy:
    bestw=w
    bestTrainAccuracy=tmpAccuracy
```

步驟四：計算該次的正確率。並保留最好的鍵結值，拿來 testing 用

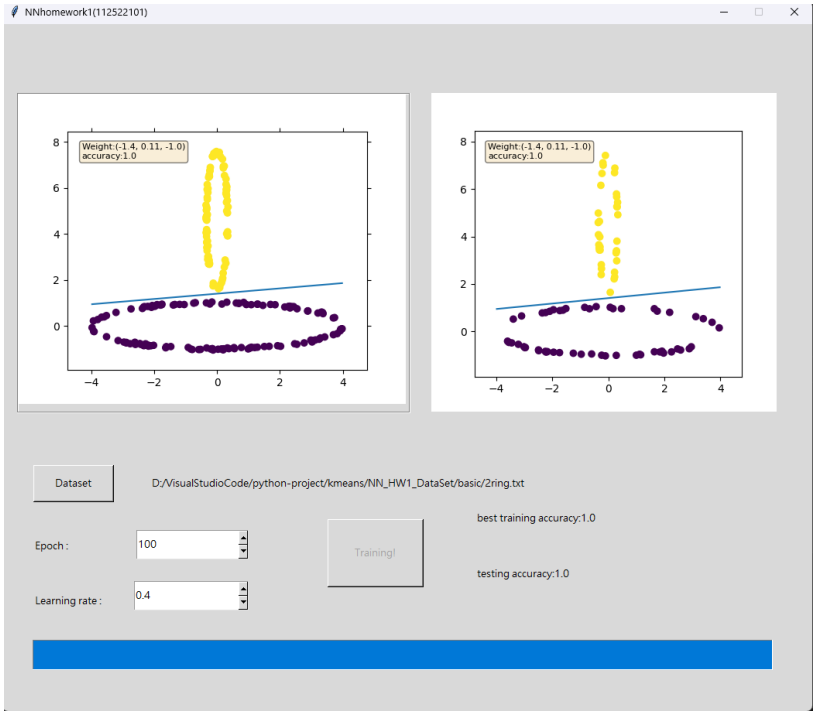
```
#畫圖(每筆資料的散佈圖、分類結果的線)
plt.ion() #讓matplotlib的圖能夠動態更新
ax1.clear() #每次都先把之前的圖清掉，節省記憶體
UIhw1_support.root.update() #畫面刷新
#畫出分類的線
xplot=np.linspace(xmin, xmax, 100)
yplot=(-w[1]*xplot+w[0])/w[2]
#畫出各資料在二維座標上的分布
ax1.scatter(X1,X2, c=y_train)
ax1.set_xlim([xmin*1.1-xmax*0.1,xmax*1.1-xmin*0.1])
ax1.set_ylim([ymin*1.1-ymax*0.1,ymax*1.1-ymin*0.1])
#畫出每次的鍵結值跟正確率
props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)
ax1.text(0.05, 0.95,'Weight:('+str(round(w[0],2))+", "+str(round(w[1],2))
        transform=ax1.transAxes, fontsize=8,verticalalignment='top', bbox
ax1.plot(xplot, yplot)
canvas1 = FigureCanvasTkAgg(fig1, master=UIhw1_support._w1.Frame1)
canvas1.draw()
canvas1.get_tk_widget().place(x=0,y=0)
```

步驟五：將每次結果透過 matplotlib 畫圖。(每次的鍵結值、訓練正確率、分類結果的線)

1.3 實驗結果及分析

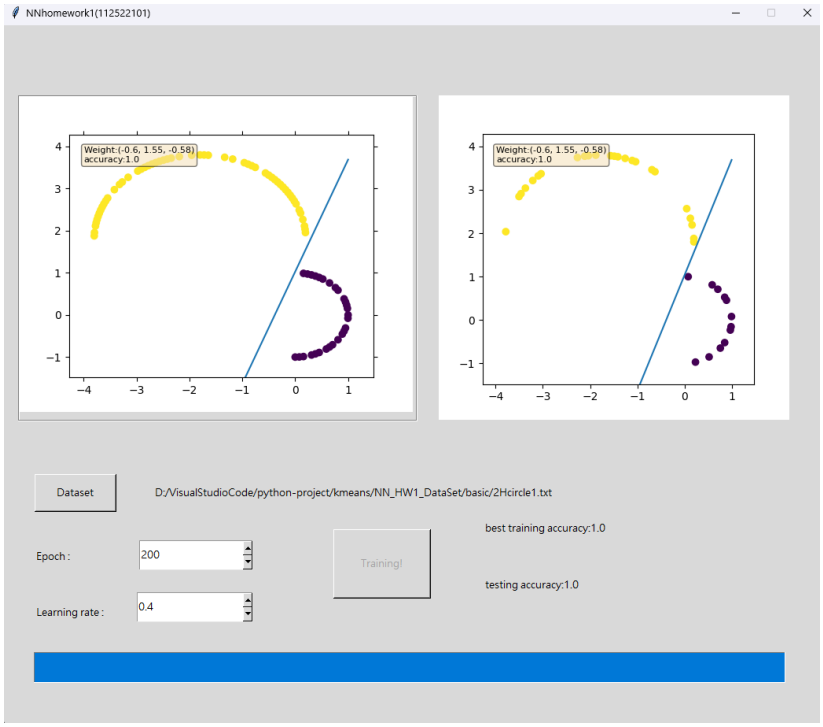
以下將針對每一個資料集進行分析:

2ring.txt(測試準確率:1.0)鍵結值:(-1.4, 0.11, -1.0)



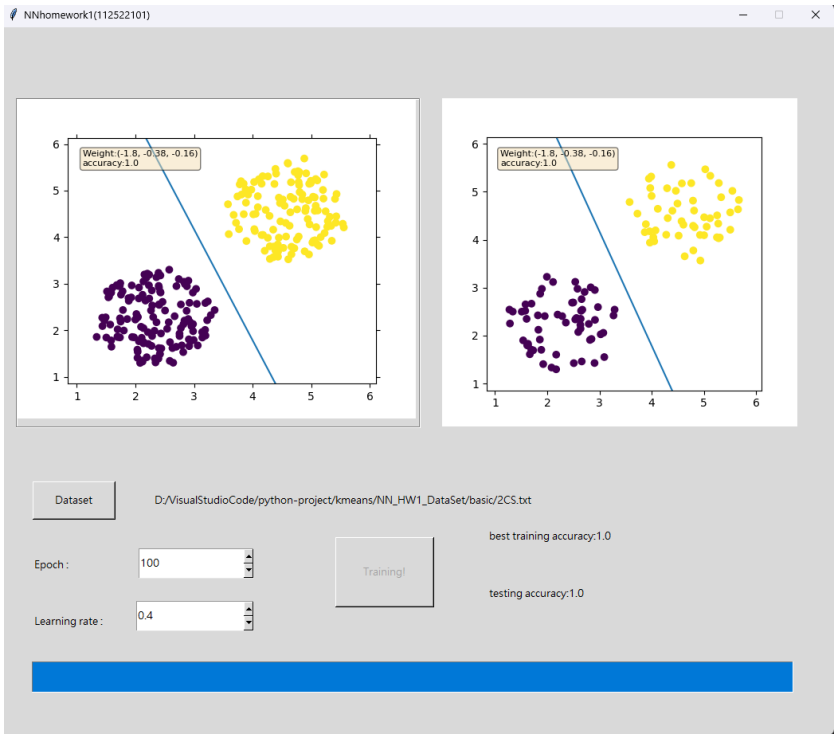
此圖為**線性可分割**，且資料分布算均勻，故迭代次數不用太大、學習率不用太小就可以使訓練跟測試的正確率達到 1.0。

2Hcircle1.txt(測試準確率:1.0) 鍵結值:(-0.6, 1.55, -0.58)



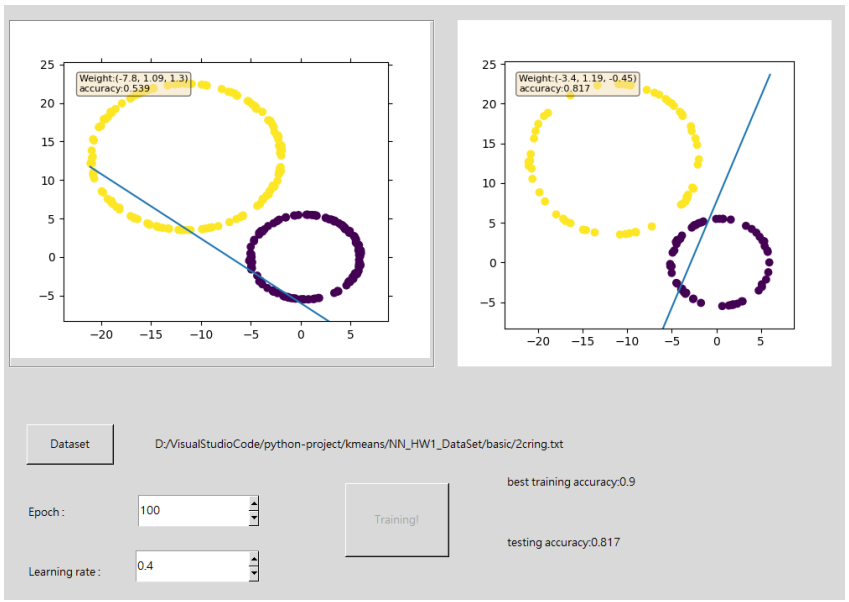
同上，此圖為**線性可分割**，且資料分布算均勻，故迭代次數不用太大、學習率不用太小就可以使訓練跟測試的正確率達到 1.0。

2CS.txt(測試準確率:1.0) 鍵結值:(-1.8, 0.38, -0.16)



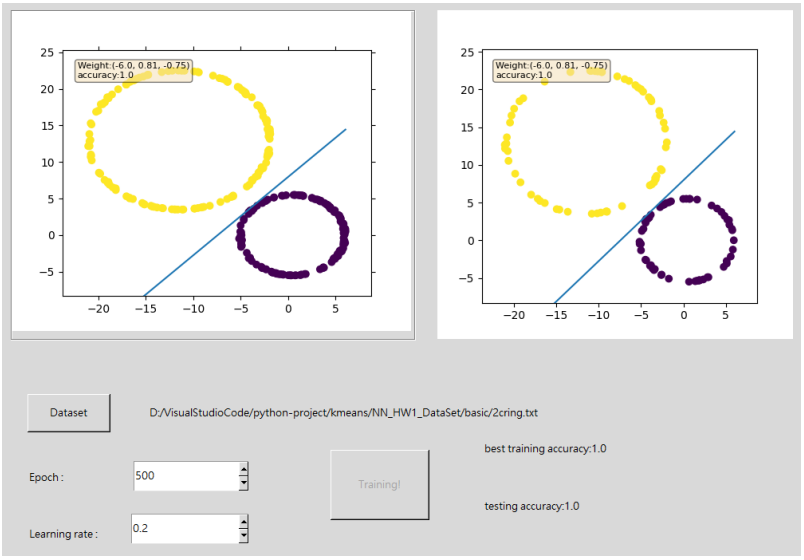
此圖為標準的**線性可分割**，且資料分布非常均勻，兩種類別分的蠻開的，故迭代次數不用太大、學習率不用太小就可以使訓練跟測試的正確率達到 1.0。

2cring.txt(測試準確率:1.0) 鍵結值:(-6.0, 0.81, -0.75)



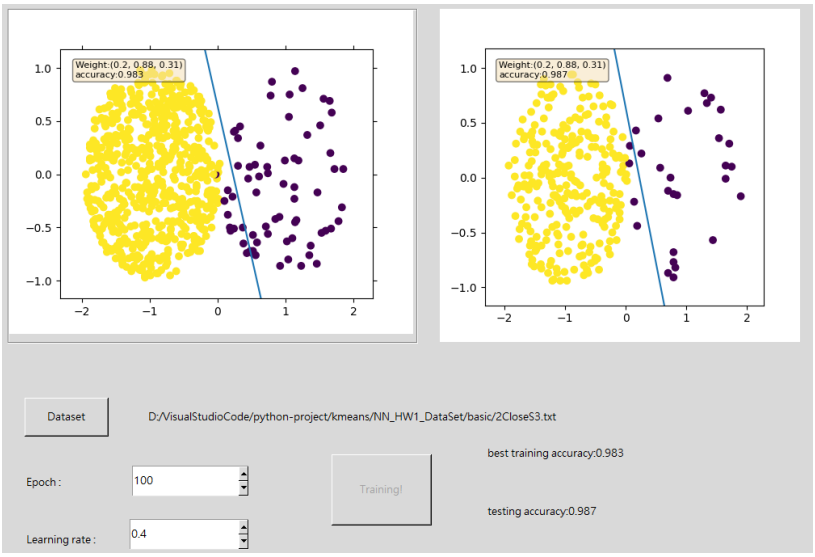
此圖線性可分割，**但兩種類別在座標上較為靠近**，故使用較多的 Epoch、較小

的學習率、慢慢接近看看。

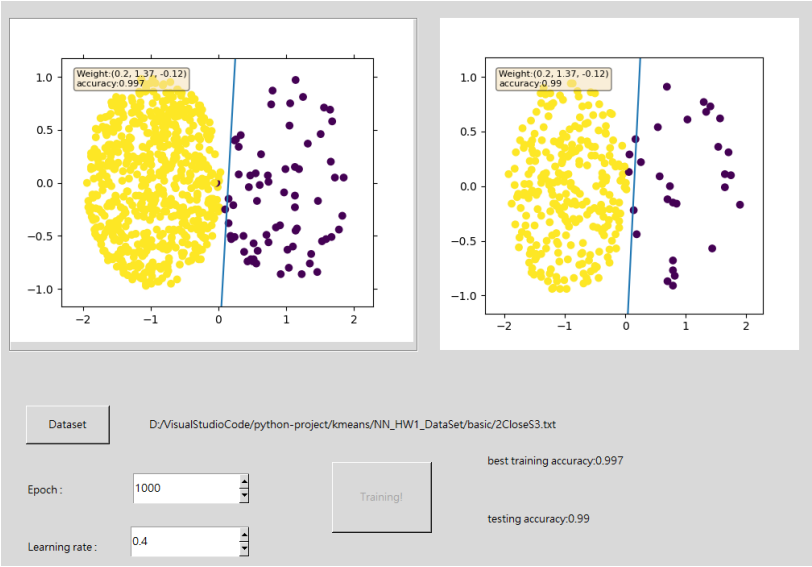


調整完後，效果很好。

2CloseS3.txt(測試準確率:0.99) 鍵結值:(0.2,1.37,-0.12)

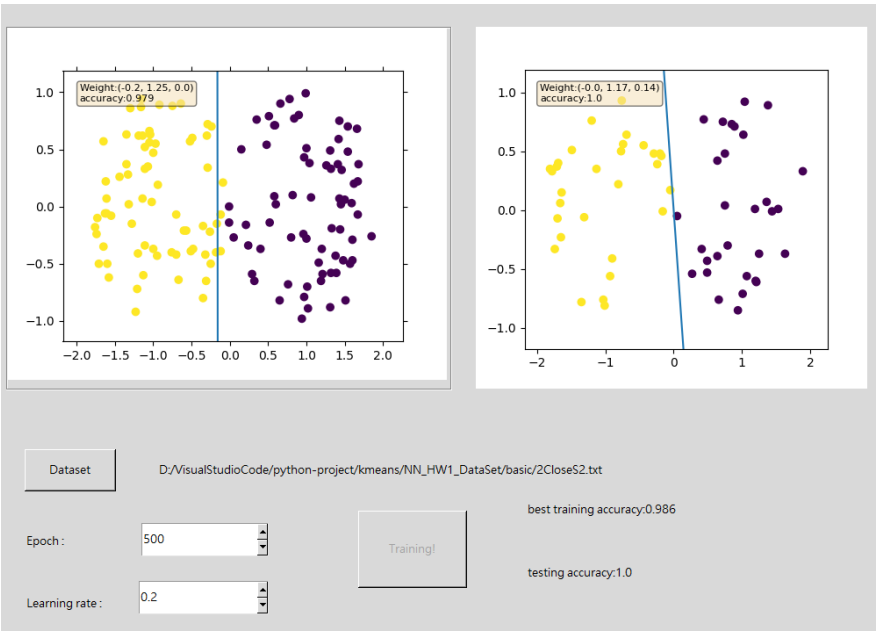


此圖**非線性可分割**，且資料不均勻，雖然說**分類結果不會**像最小均方法那樣會**受影響**，但是在迭代的時候比較常算到黃色的資料，所以就更新的比較慢，**故嘗試增加 Epoch 次數，但不調整學習率**看看。



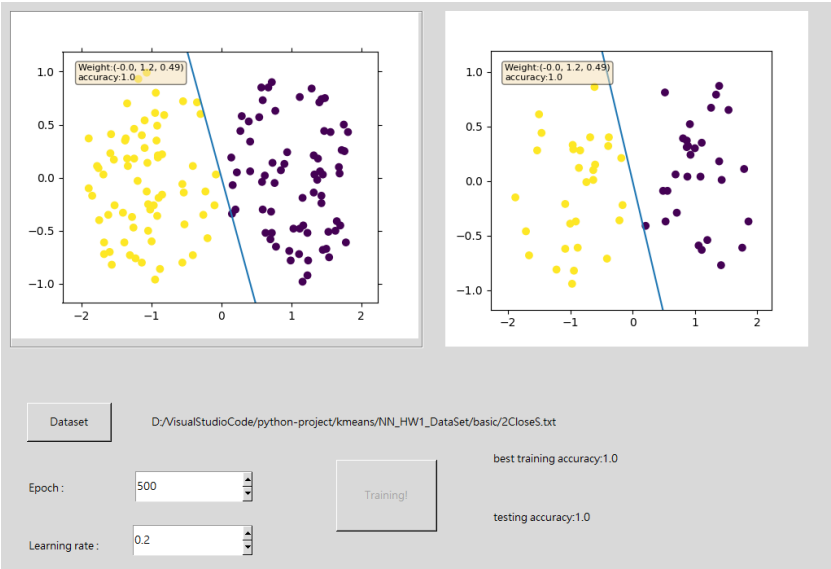
結果好很多，但是畢竟兩個類別有重疊，**非線性可分割**，所以準確率無法透過一條線準確分類。

2CloseS2.txt(測試準確率:1.0) 鍵結值:(0.0,1.17,0.14)



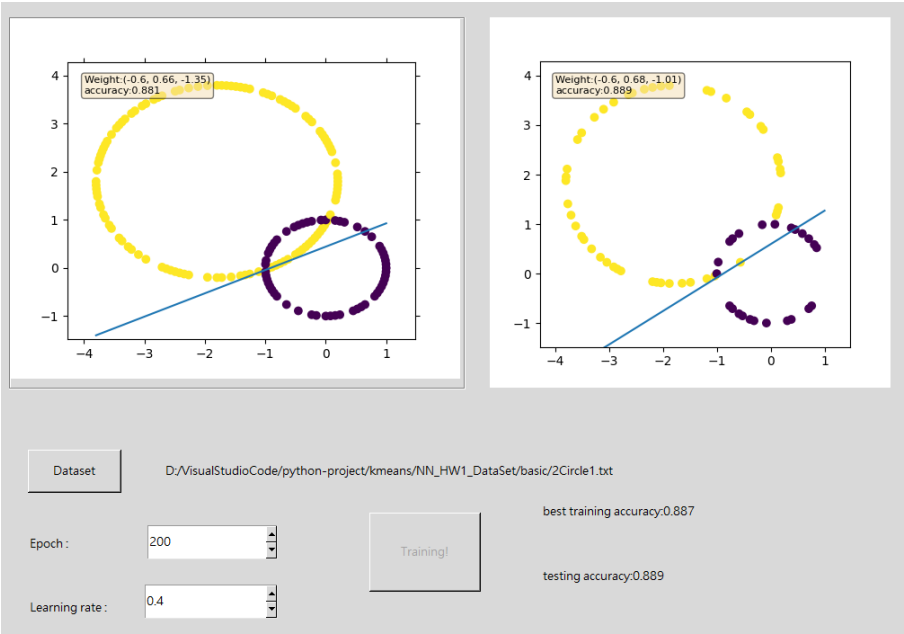
此圖兩種類別剛好相切，為線性可分，但是**因為剛好相切，需要剛剛好切到那條線，所以把學習率降低，並增加訓練次數**，設定到 500 次時，準確率剛好為 1.0

2CloseS.txt(測試準確率:1.0) 鍵結值:(0.0,1.2,0.49)



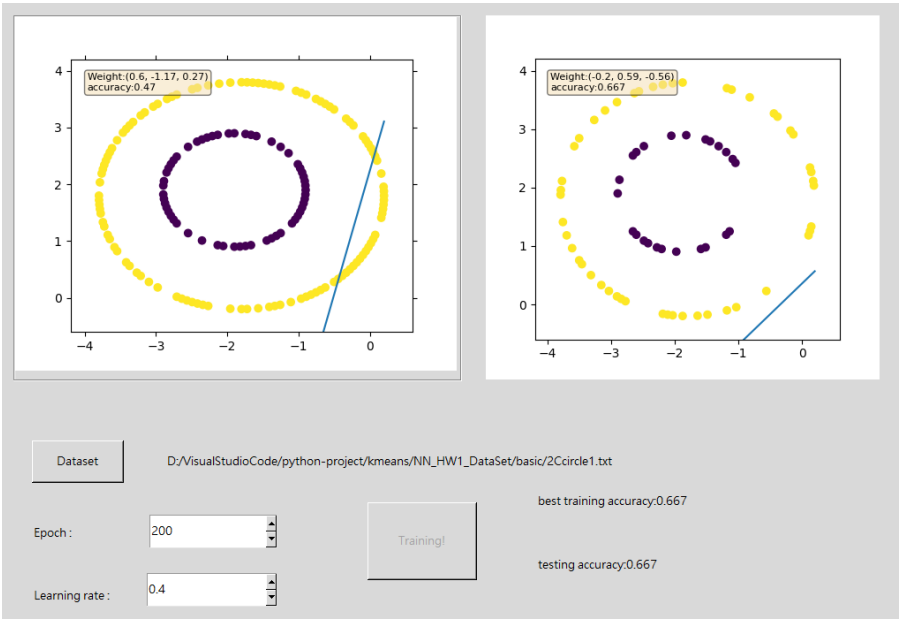
此圖**兩種類別非常接近，為線性可分**，因為兩類別較為靠近，所以一樣把學習率降低，並增加訓練次數，設定到約 450 次時，準確率剛好為 1.0，**(不過也有可能資料其實有重疊到，剛好被訓練跟測試資料分開來了)**

2Circle1.txt(測試準確率:0.889) 鍵結值:(-0.6, 0.68, -1.01)



此圖為**非線性可分割**，無法準確分割，嘗試了增加 Epoch，盡量增加正確率

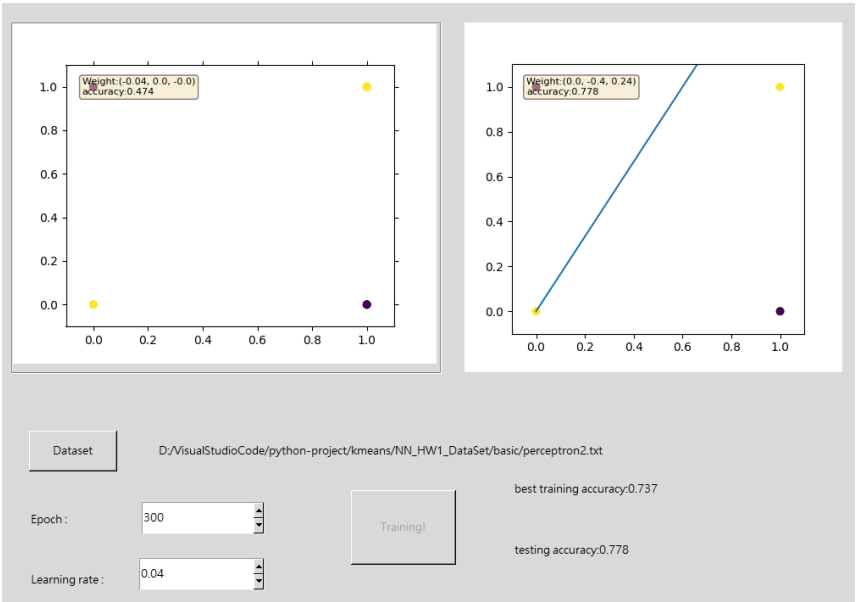
2Ccircle1.txt(測試準確率:0.667) 鍵結值:(-0.2, 0.59, -0.56)



同上為同心圓，是**非線性可分圖**型，有試過幾次都是從中間切一刀，準確率最好為 0.667，**上圖為較特別的狀況**，可能是因為外圈的點多於內圈的點太多，所以直接將外圈的都分在同一類就好，這樣準確率一樣是 0.667。XD

perceptron2.txt(測試準確率:0.778) 鍵結值:(0.0, -0.4, 0.24)

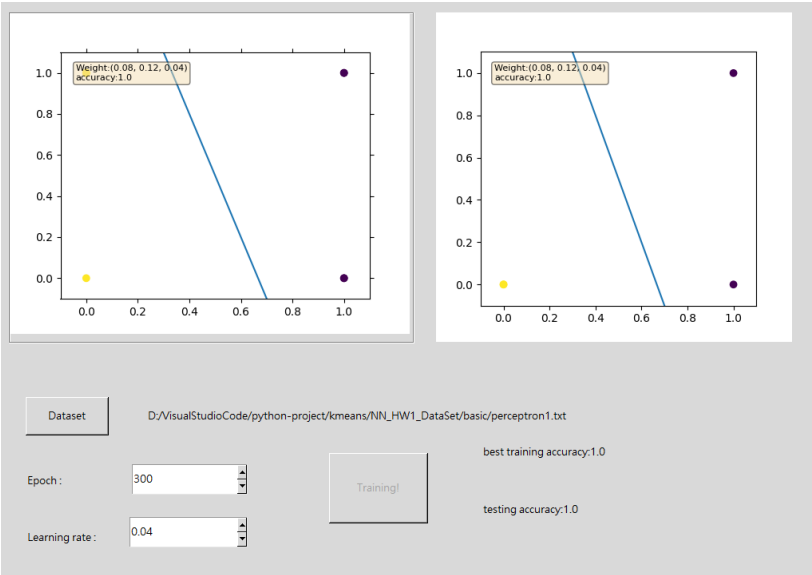
該資料集只有 4 筆資料，不夠切成訓練集及測試集，故加入 3 倍相同的資料進去。



因為資料間距離只有 1，所以**學習率設低一些，讓她不會來回反覆震盪**。而此圖為非線性可分，無法準確分類。(我的圖為了方便看，有將座標固定，所以若分類的線在座標外，就會看不見)

perceptron1.txt(測試準確率:1.0) 鍵結值:(0.08, 0.12, 0.04)

該資料集只有 4 筆資料，不夠切成訓練集及測試集，故加入 3 倍相同的資料進去。



為**線性可分圖**型，但因**距離間只有 1，所以學習率設低一些，不讓線反覆震盪**，而 Epoch 在約 50 次的時候就找到準確率為 1.0 的線了。

心得結論:若非線性可分的資料，靠單層感知機無法準確分類。且若資料間距離較近、需要很精準的線來分類的情況下，學習率需要降低、Epoch 次數需增加，正確率才會提升。

PS:原(.exe)檔案破百 MB，所以有先壓縮了，故打開需要等一陣子。