感知機實做

A. 程式執行說明

1. 點開執行檔後會產生一個cmd要等候一下GUI介面才會出來

2. 介面中可以輸入學習率、訓練次數(訓練資料個數)

3. 選擇檔案

4. 按下開始執行(必須先按選擇檔案過後才能按不然跑不出東西來)

5. 產生訓練結果

B.程式碼簡介

def select\_file():

此函數是button("選擇檔案")的命令函數

當按下此button就會去選擇檔案，並讀取檔案然後將所讀的一行一行的資 料存入lists這個二微陣列，然後將資料隨機分成2/3.1/3分，分別在 tranlist與lists中

def gogo():

先將weight設初始為三個-1到1的隨機浮點數

擷取gui輸入框內的資料

while times>0:

        for i in range(0,trainnumber):

            sum = -1 \* weight[0]

            for j in range(0,2):

                sum += weight[j+1]\*float(trainlist[i][j])

            if (sum <= 0 and float(trainlist[i][2])%2 == 0 and float(trainlist[i][2]) != 4):

                weight[0] += -1 \* learnrate

                for k in range(0,2):

                    weight[k+1] += learnrate \* float(trainlist[i][k])

            elif (sum >= 0 and float(trainlist[i][2])%2 == 1):

                weight[0] -= -1 \* learnrate

                for k in range(0,2):

                    weight[k+1] -= learnrate \* float(trainlist[i][k])

#######################################################################

            for j in range(0,trainnumber):

                sum = -1 \* weight[0]

                for k in range(0,2):

                    sum += weight[k+1]\*float(trainlist[j][k])

                if(sum < 0 and float(trainlist[j][2])%2 == 1):

                    tcorrectnumber += 1

                elif (sum > 0 and float(trainlist[j][2])%2 == 0 and float(trainlist[i][2]) != 4):

                    tcorrectnumber += 1

            if( tcorrectrate < tcorrectnumber / trainnumber):

                tcorrectrate = tcorrectnumber / trainnumber

                for  p in range(0,3):

                    ans[p] = weight[p]

            tcorrectnumber = 0

            times -= 1

            if(times<=0):

                break

在#字號上方的程式碼簡單來說就是計算W0\*X0+………Wi\*Xi=sum的值然後去根源來的分類去本較

Wi(n+1) = Wi(n) -/+學習率 \* 輸入向量

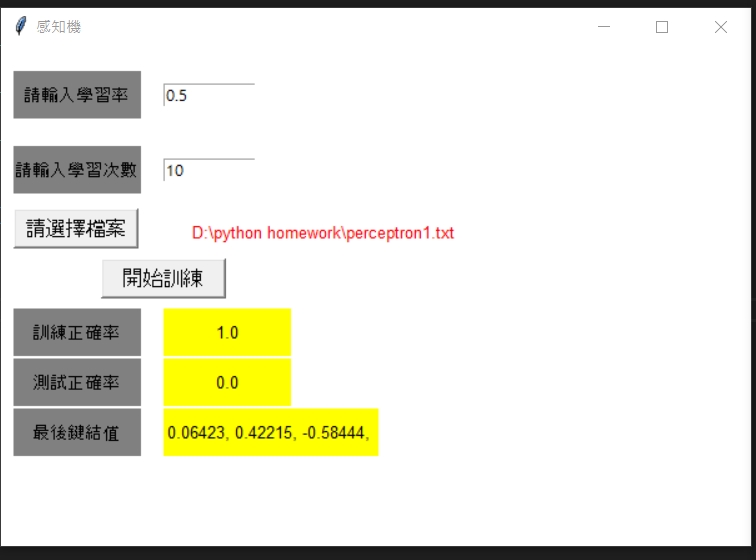
測資中有分類1 and 2我就用mod的方式去處理。

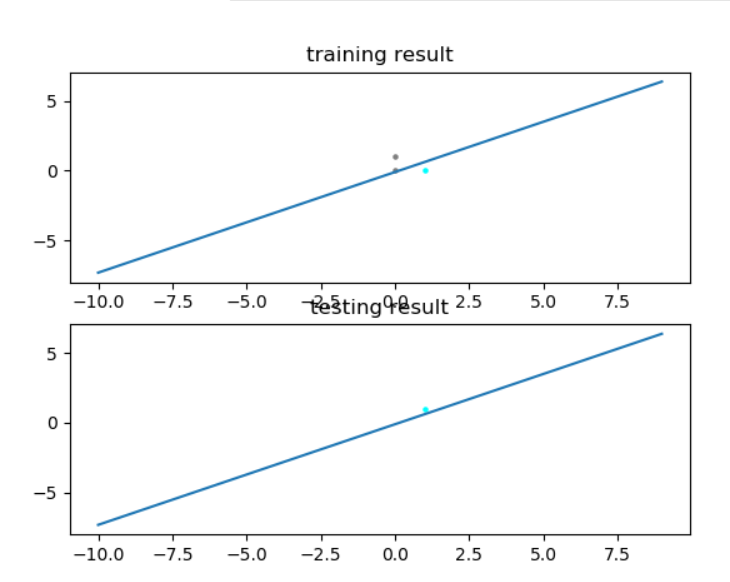
#井字號下方就是計算出一次改變weight之後所有訓練測資的正確率。

最後則是算出訓練正確率和測試正確率並印出兩個的圖片。

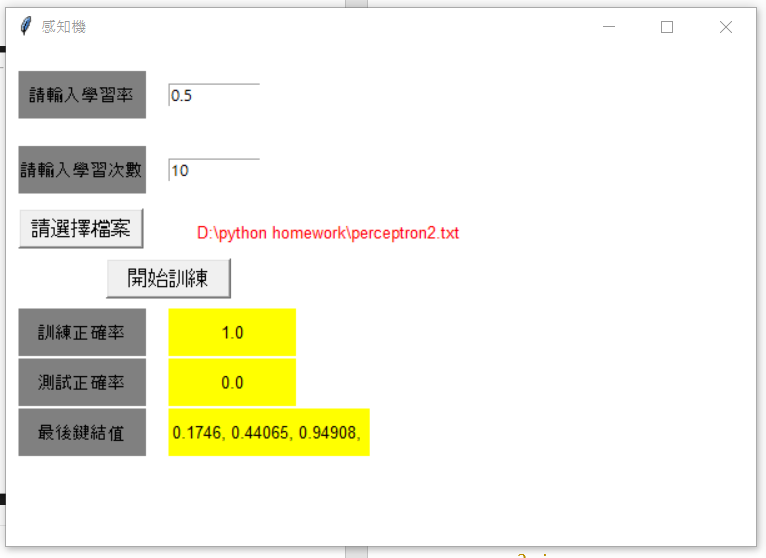
C.實驗結果

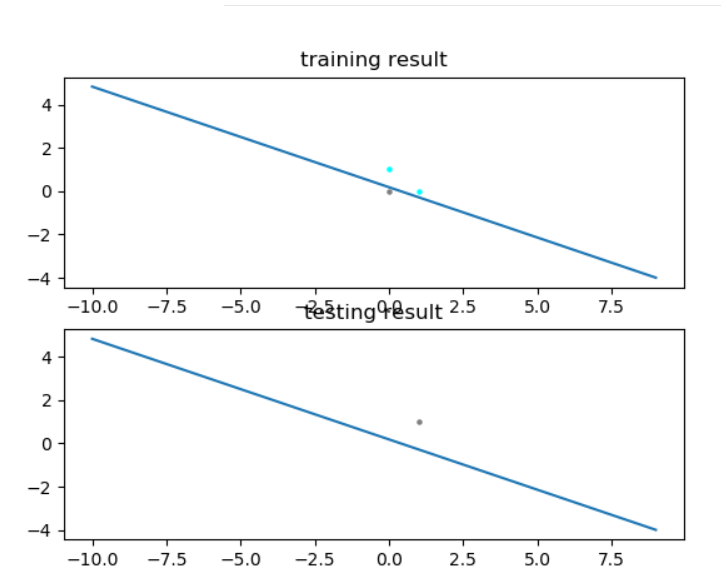
Perceptron1



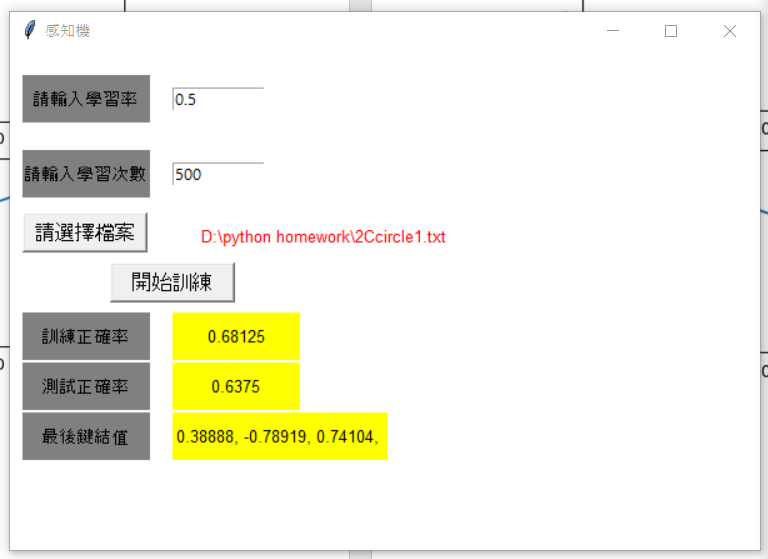


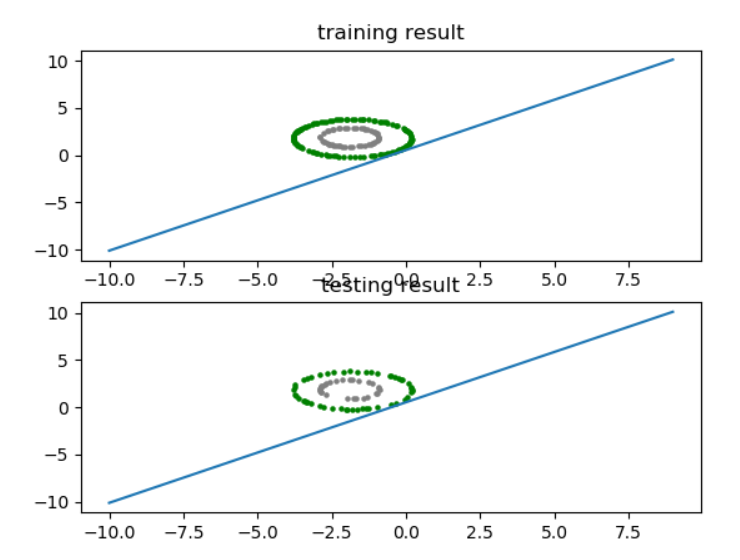
Perceptron2



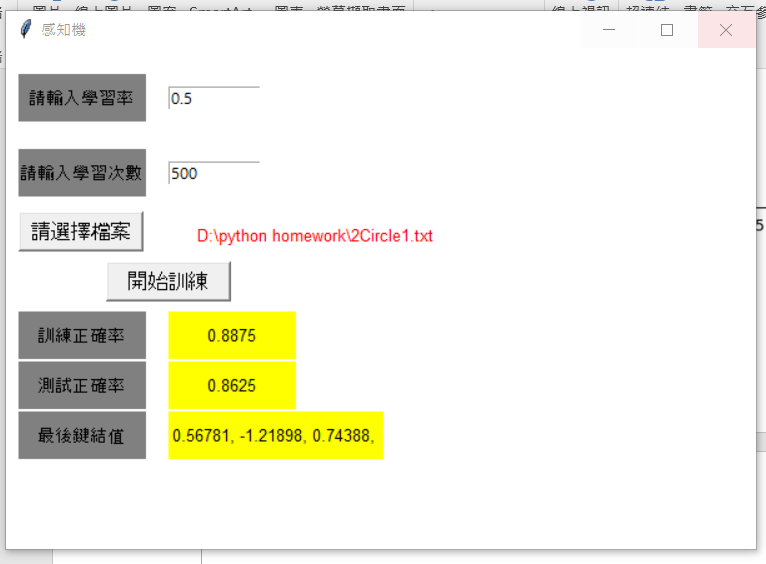


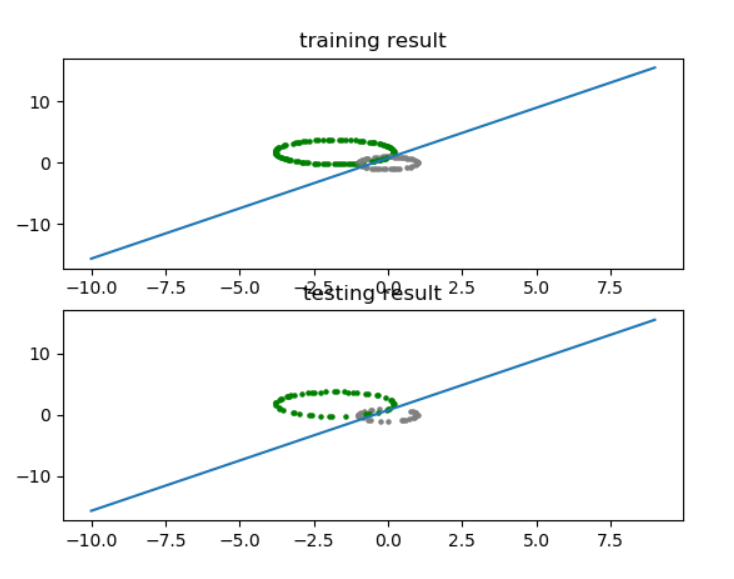
2Ccircle1



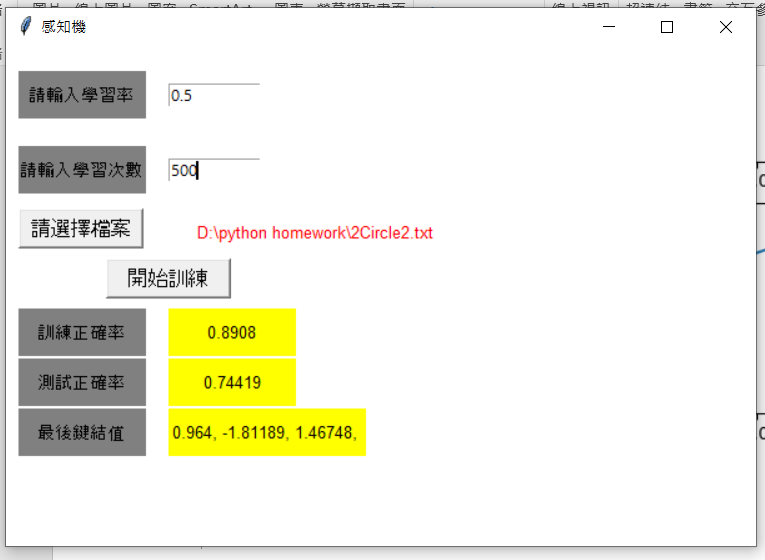


2Circle1



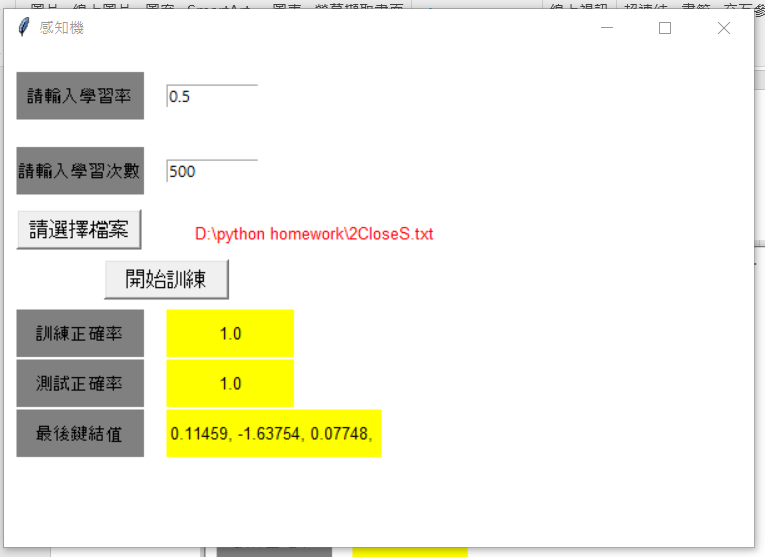


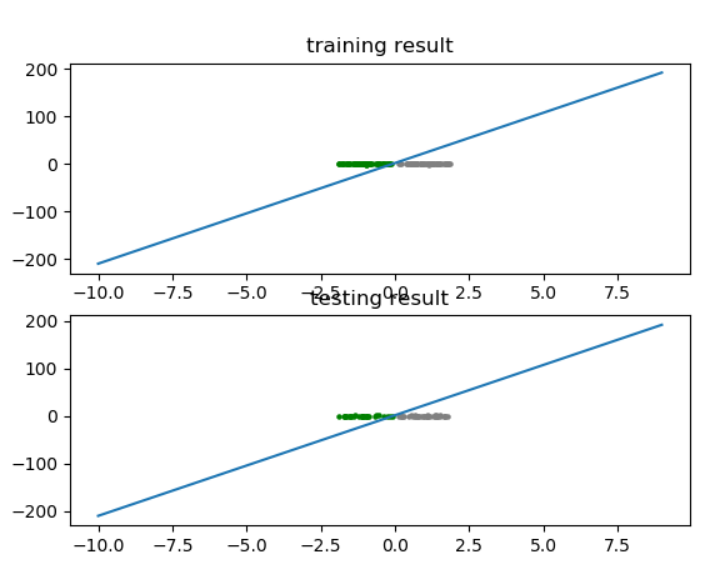
2Circle2



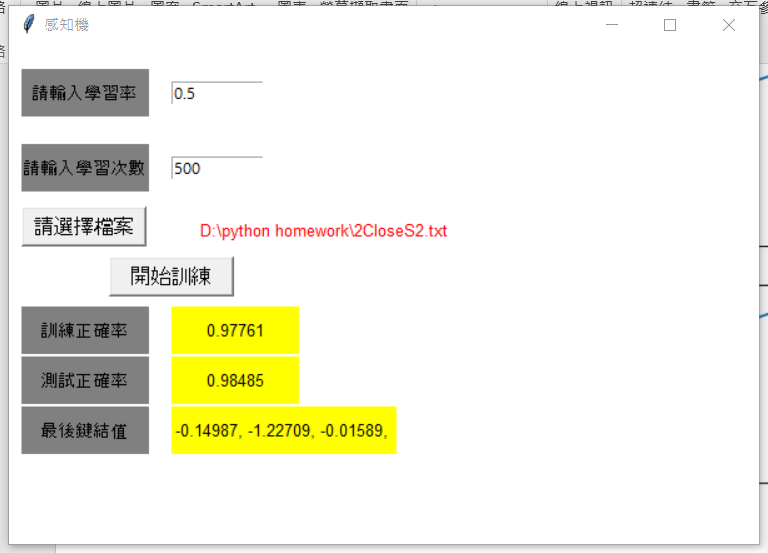


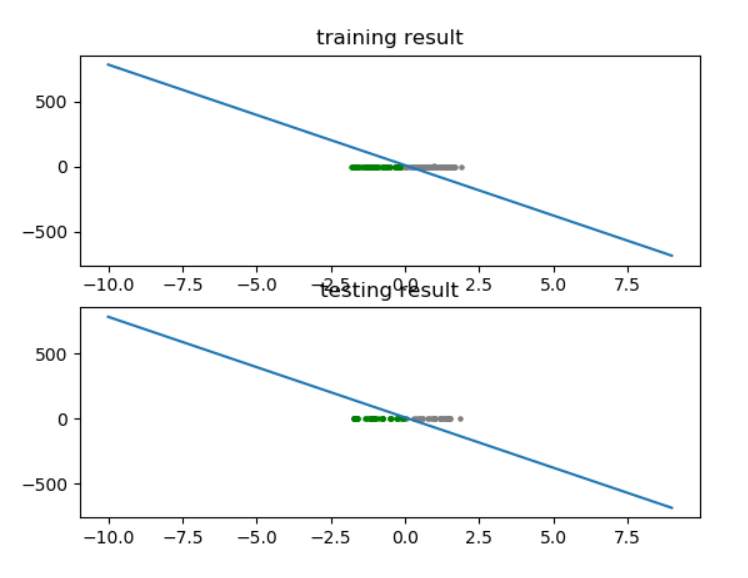
2CloseS



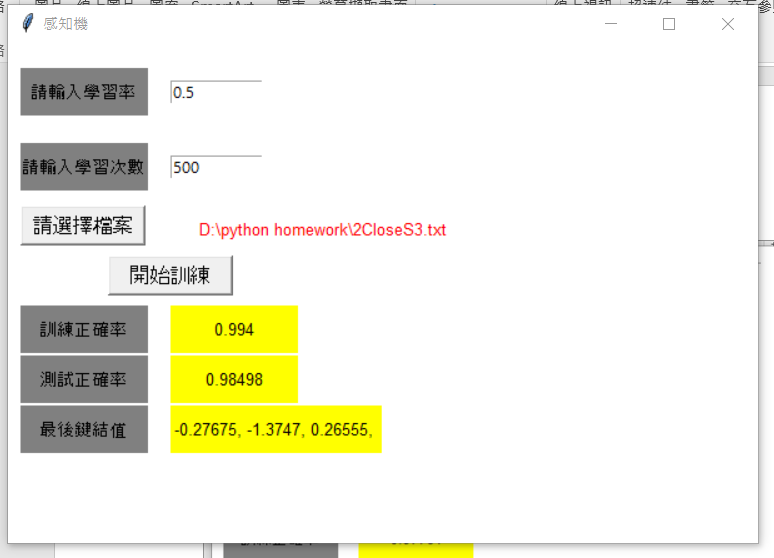


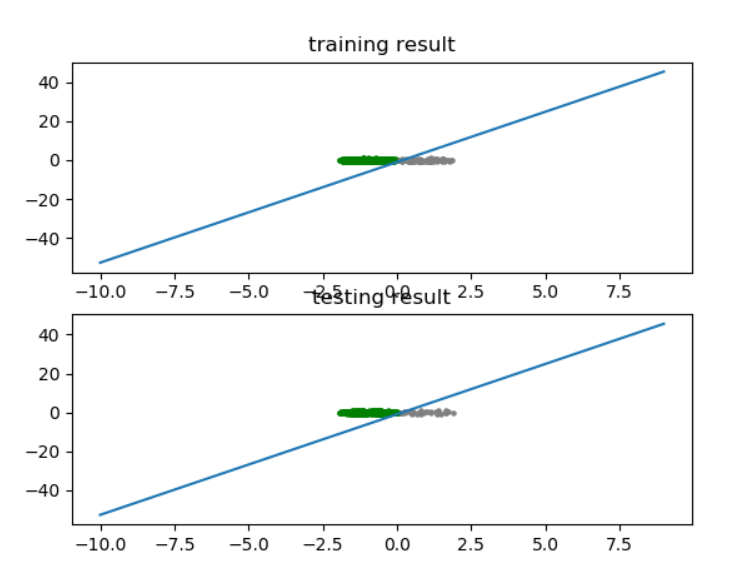
2CloseS2



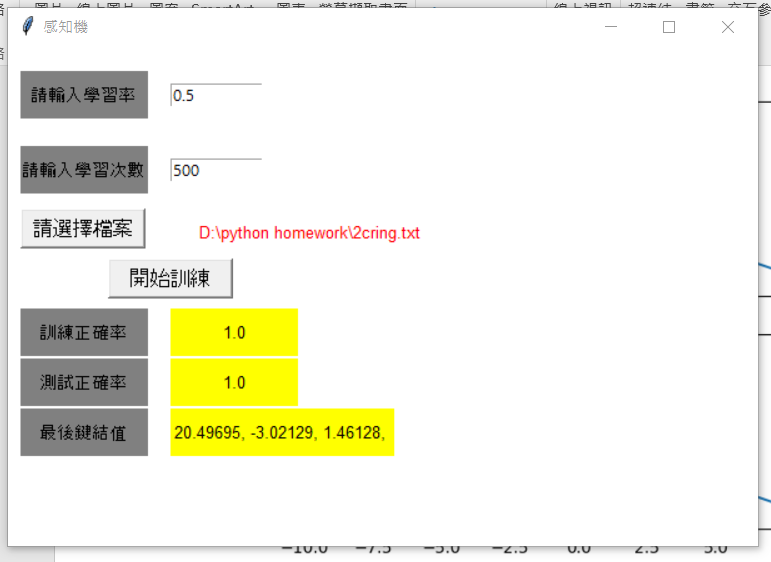


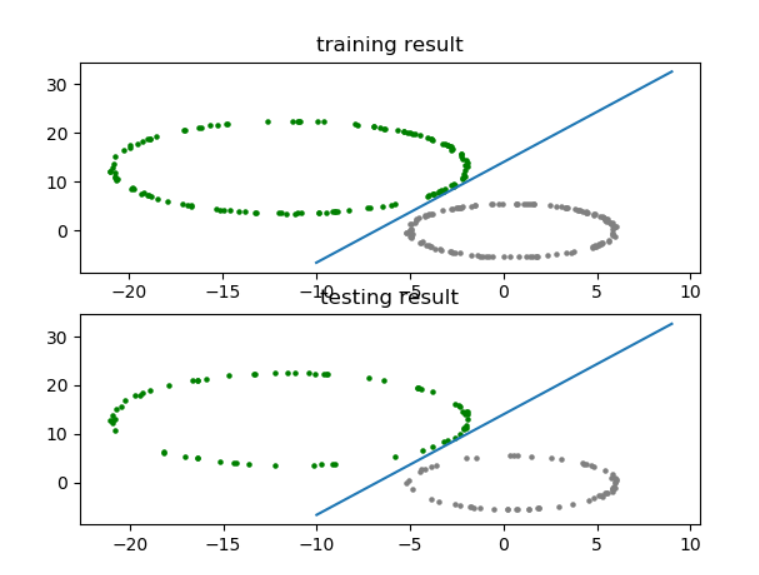
2CloseS3



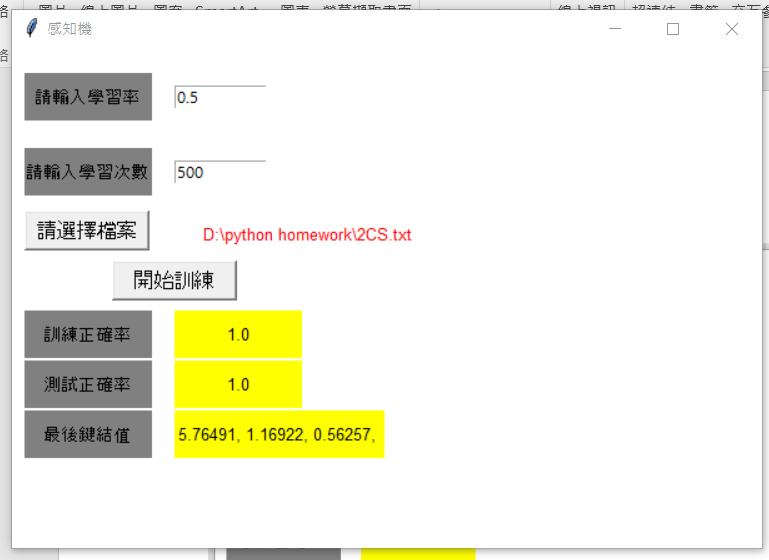


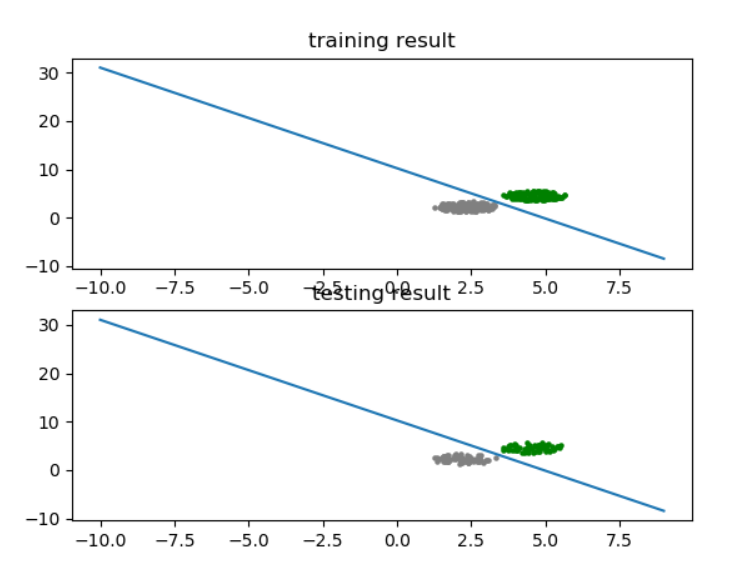
2cring



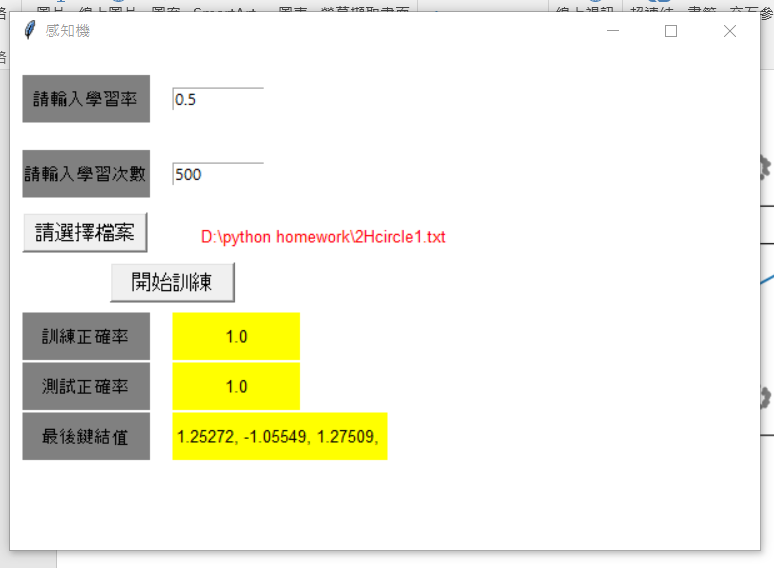


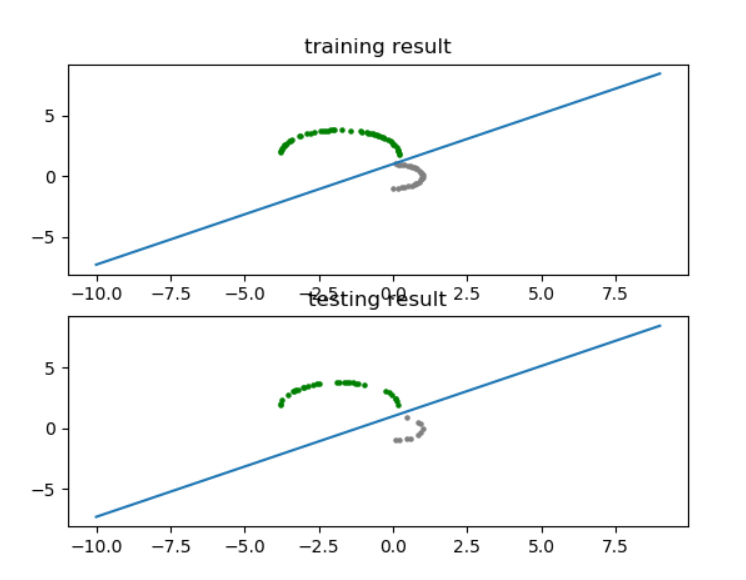
2CS



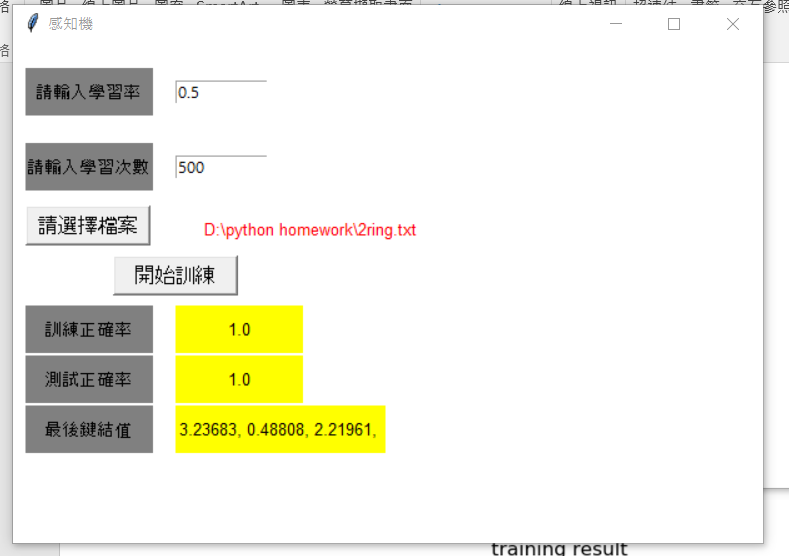


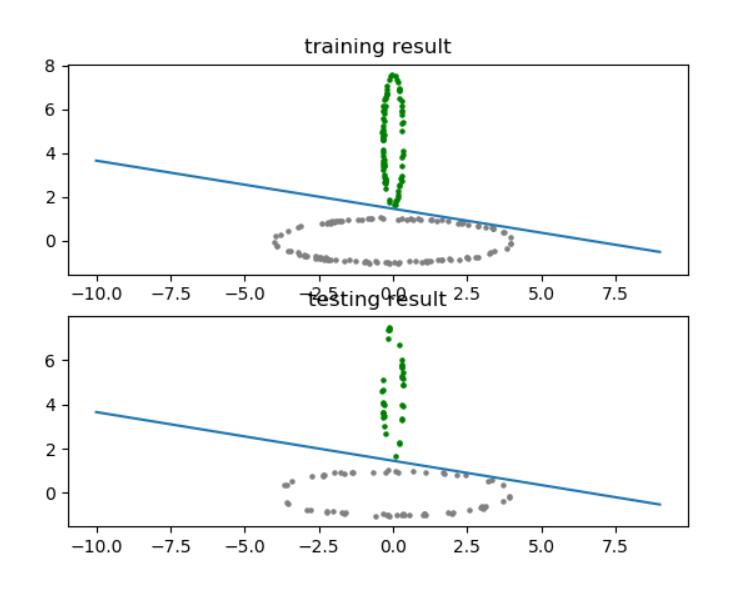
2Hcircle





2ring





D.實踐結果分析及討論

幾乎資料所有把訓練次數加大的話都會測試正確率趨近於1

只有Perceptron2與2Circle2例外

Perceptron2正確率探討

再圖上看來，這四個點相鄰不互為同類，這是XOR的問題需要兩條線來分類

任取三個點分辨率能到100%，但第四個店正確率必為0。

2Circle2正確率探討

因為此測資有3類，我沒有寫出2類以上的解法，第三類一定是分類錯誤，所以正確率會有一定的上限。

F.額外內容

我可以判斷多維兩類輸入，以及產生正確的訓練正確率、測試正確率以及權重，然而印出的圖是投影在XY平面上的圖。

但是兩類以上的分類我只能在平面上分別表示出不同顏色的點，還不能分類。

以C3D為例:

