多層感知機實作

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、鍵結值、均方根誤差

B.程式碼簡介(多層感知機、一層隱藏層兩個神經元)

 for i in range(len(lists)):

        if(int(lists[i][3])<minn):

            minn = int(lists[i][3])

        elif(int(lists[i][3])>maxx):

            maxx = int(lists[i][3])

    for i in range(len(lists)):

        lists[i][3] = str((int(lists[i][3])-minn)/(maxx-minn))

將分類編號正規化到介於0~1之間

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

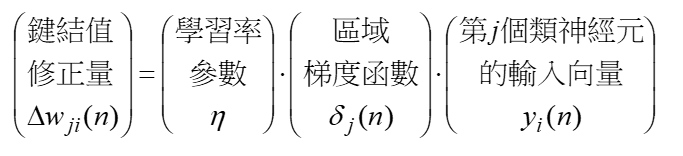
def gogo():

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

計算出每個神經元的輸出存在midd陣列中

再用midd的值去計算每個神經的區域梯度函數並存在Q陣列中



再運用上述公式去調整鍵結值

調整過後的鍵結值去跑所有的測試資料，算出正確率(correctrate)與均方誤差(Eav)，如果正確率比之前大則將正確率替換、均方誤差替換並將答案的鍵結值替換為當下的鍵結值。

while times>0:

        index = cnt % trainnumber

        for i in range(2):

            sum = 0

            for j in range(0,degree+1):

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

            midd[i] = 1/(1+math.exp(-1\*sum))

        sum = -1 \* weight[2][0]

        for j in range(0,degree):

            sum += weight[2][j+1]\*midd[j]

        midd[2] = 1/(1+math.exp(-1\*sum))

        Q[2] = (float(trainlist[index][3]) - midd[2]) \* midd[2] \* (1 - midd[2])

        Q[0] = (midd[0] \* (1 - midd[0]) \* Q[2] \* weight[2][1])

        Q[1] = (midd[1] \* (1 - midd[1]) \* Q[2] \* weight[2][2])

        for i in range(2):

            for j in range(3):

                weight[i][j] += learnrate \* Q[i] \* float(trainlist[index][j])

        for j in range(3):

            if j==0:

                weight[2][j] += learnrate \* Q[2] \* -1

            else:

                weight[2][j] += learnrate \* Q[2] \* midd[j-1]

        correctnumber = 0

        E = 0

        for i in range(trainnumber):

            tempa = 0

            tempb = 0

            tempc = 0

            for l in range(3):

                tempa += weight[0][l] \* float(trainlist[i][l])

            tempa = 1/(1+math.exp(-1\*tempa))

            for l in range(3):

                tempb += weight[1][l] \* float(trainlist[i][l])

            tempb = 1/(1+math.exp(-1\*tempb))

            tempc = -1 \* weight[2][0] + tempa \* weight[2][1] + tempb \* weight[2][2]

            tempc = 1/(1+math.exp(-1\*tempc))

            tempE = 1/2 \* (tempc - float(trainlist[i][3])) \* (tempc - float(trainlist[i][3]))

            E = E + tempE

            if(float(trainlist[i][3]) == 0 and tempc<0.5):

                correctnumber += 1

            elif(float(trainlist[i][3]) == 1 and tempc>0.5):

                correctnumber += 1

        if(correctnumber/trainnumber > correctrate):

            correctrate = correctnumber/trainnumber

            Eav = E/trainnumber

            for i in range(3):

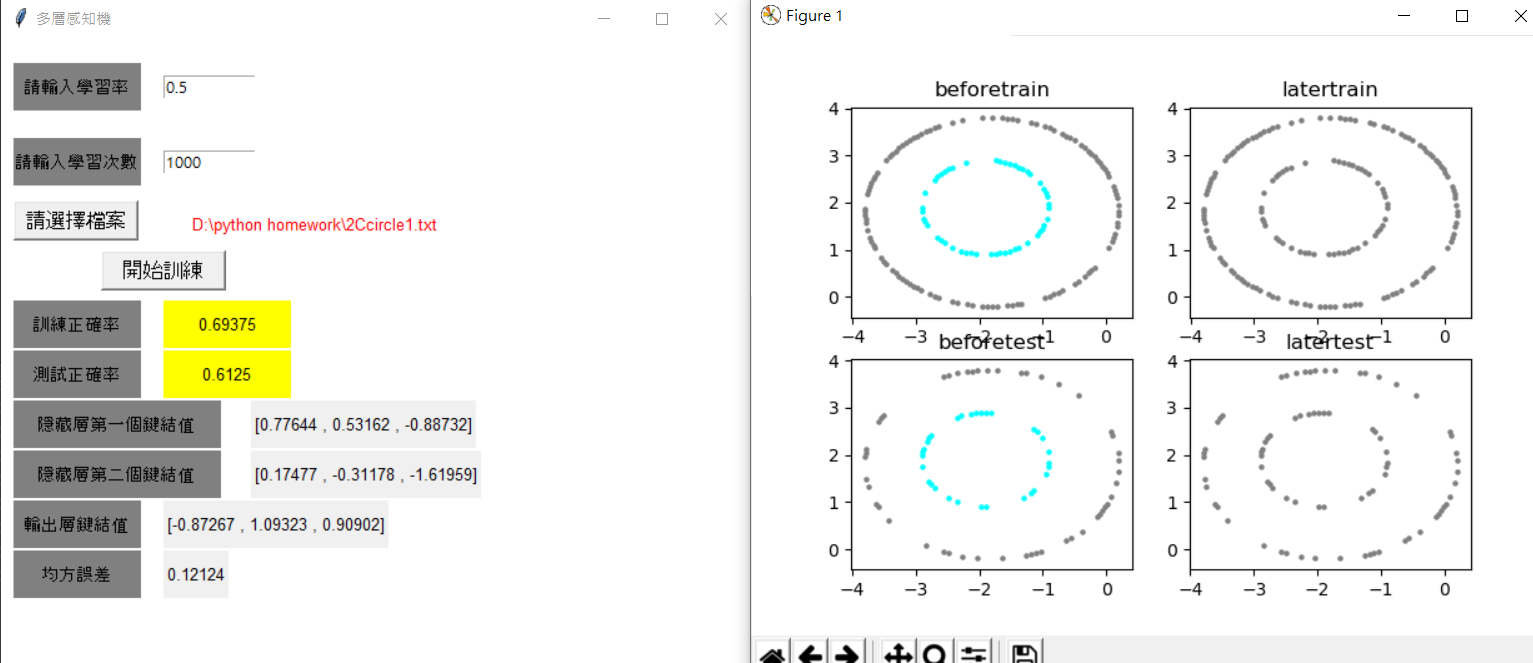
                ans[i] = weight[i]

        times-=1

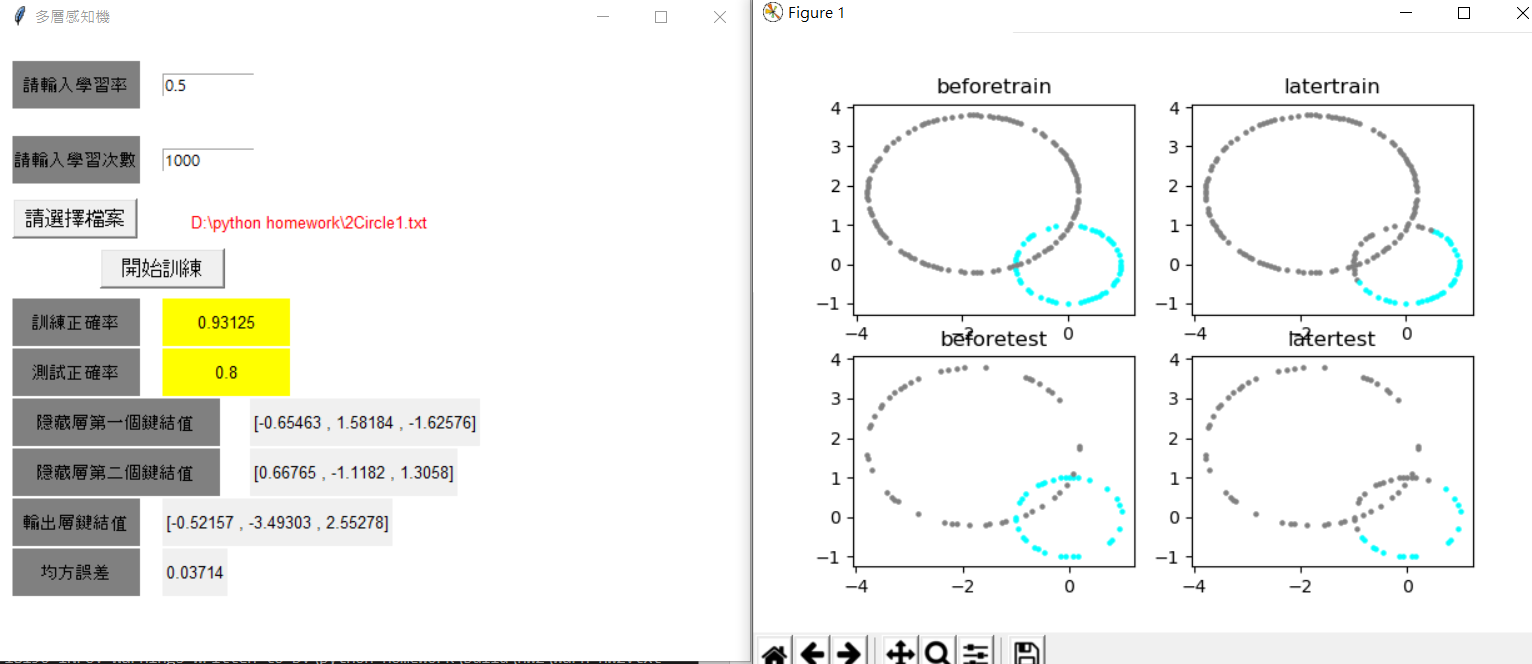
        cnt+=1

C.實驗結果

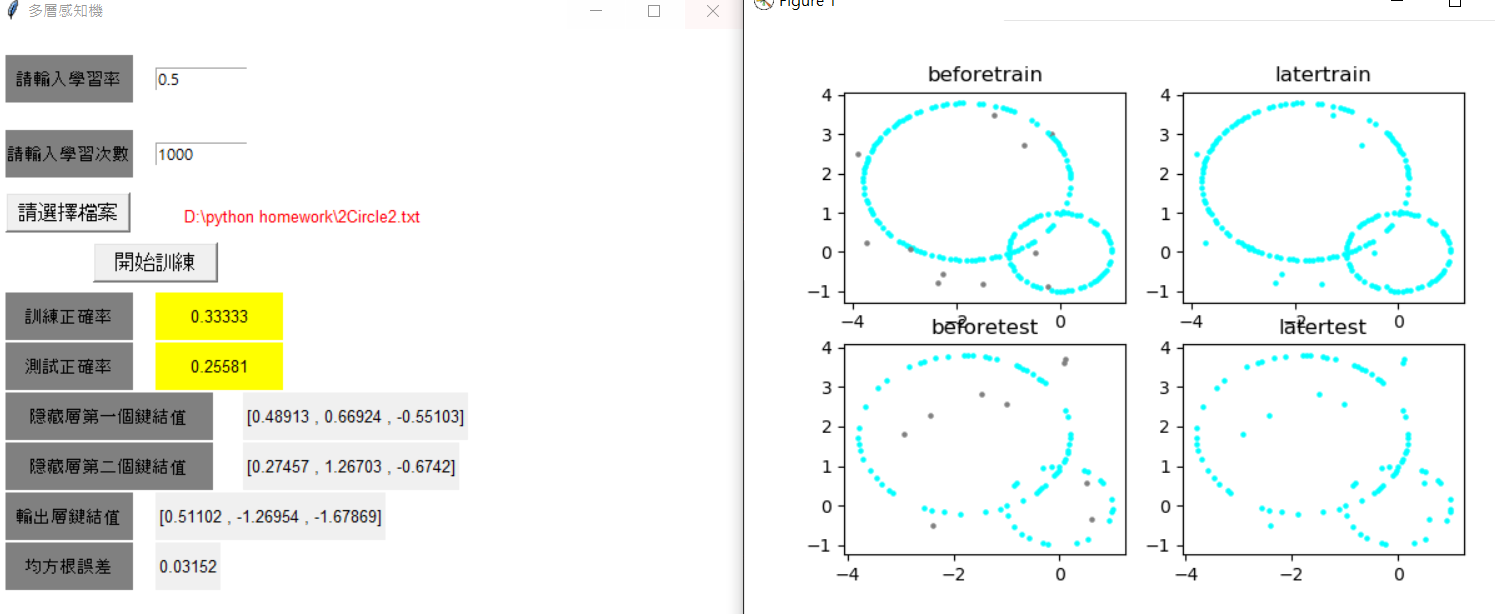
2Ccircle1



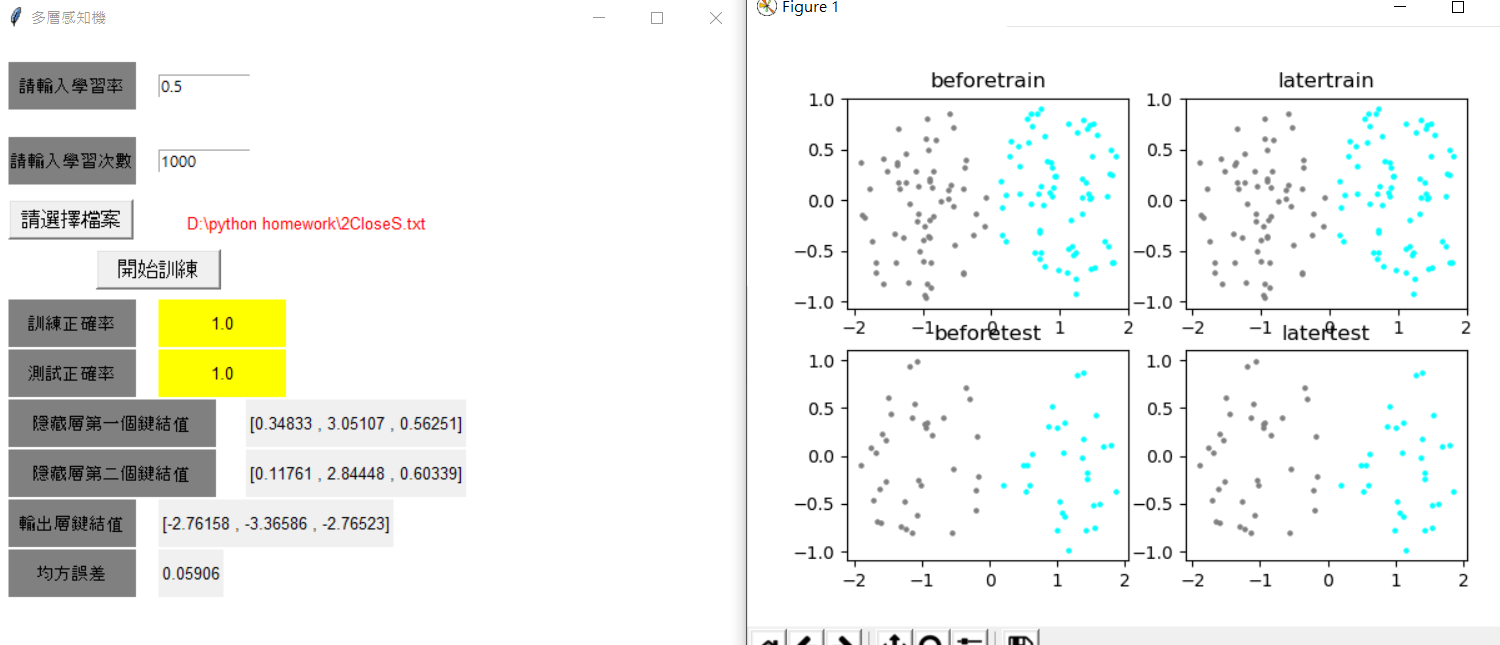
2Circle1



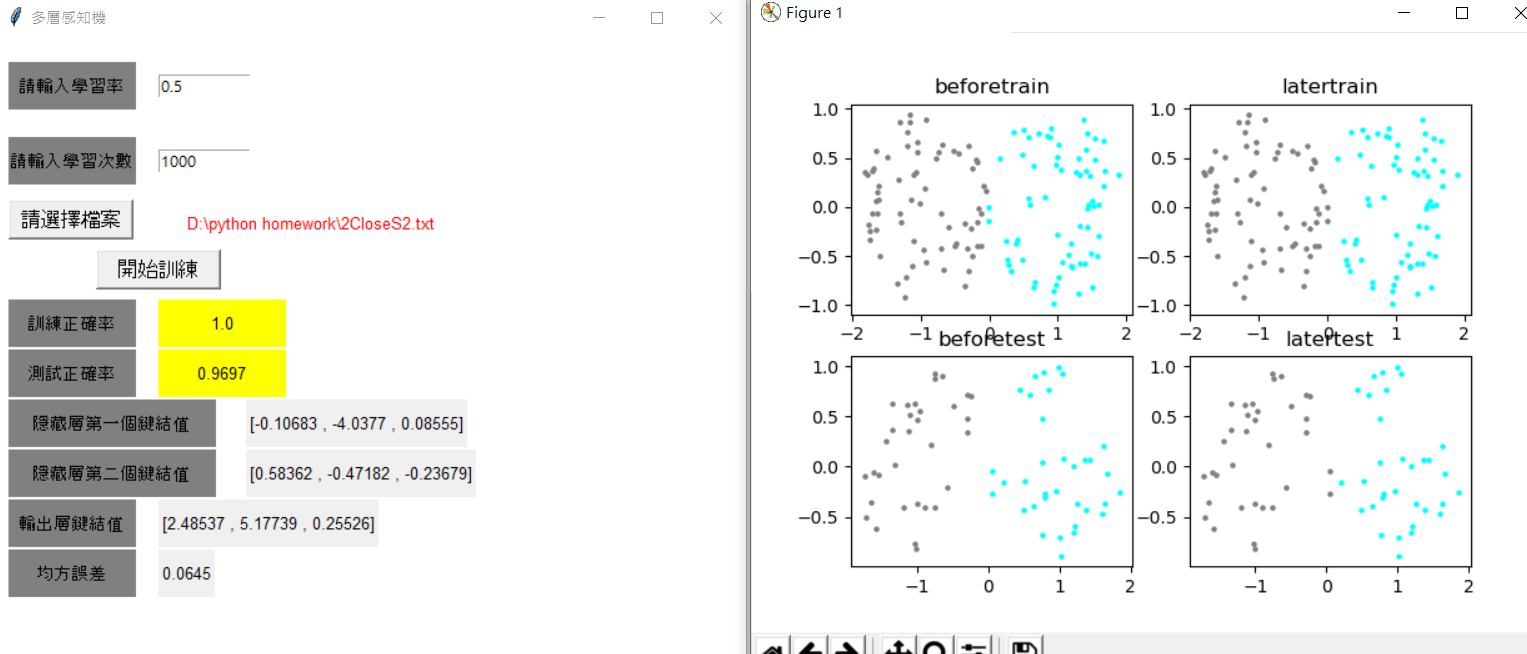
2Circle2(無法分辨大於兩群)



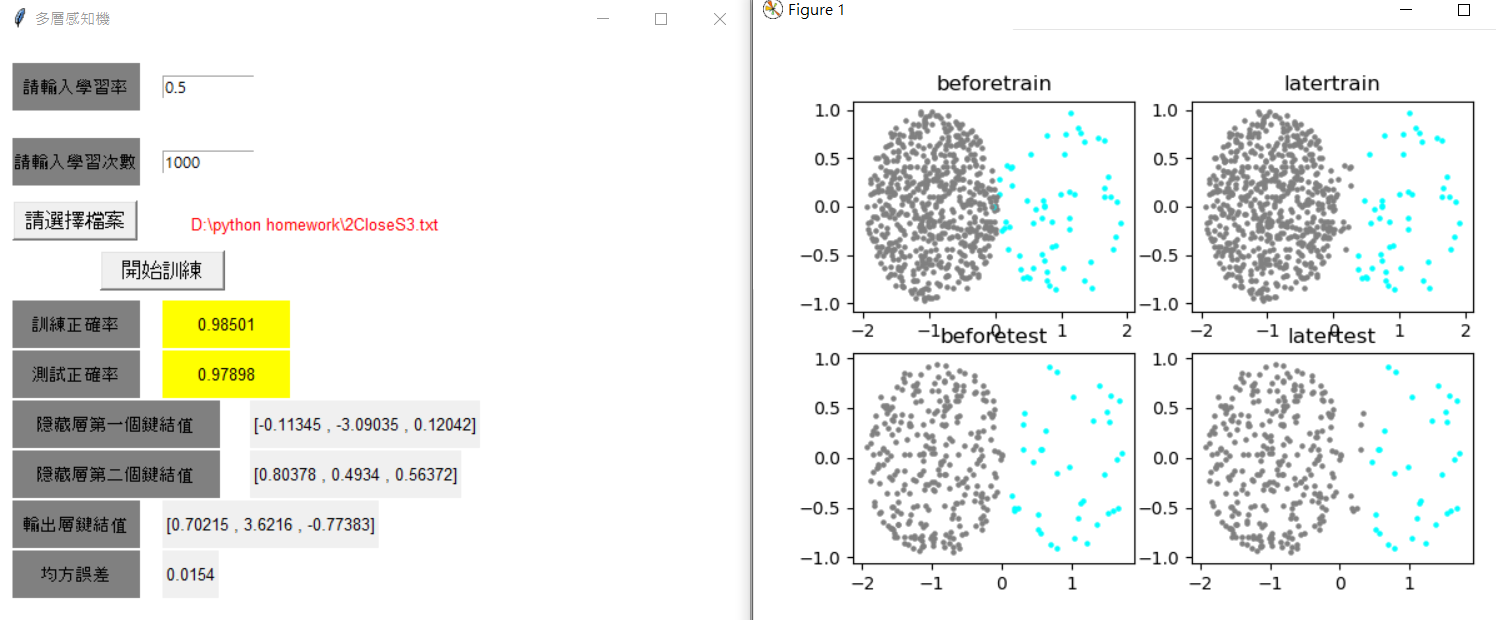
2CloseS



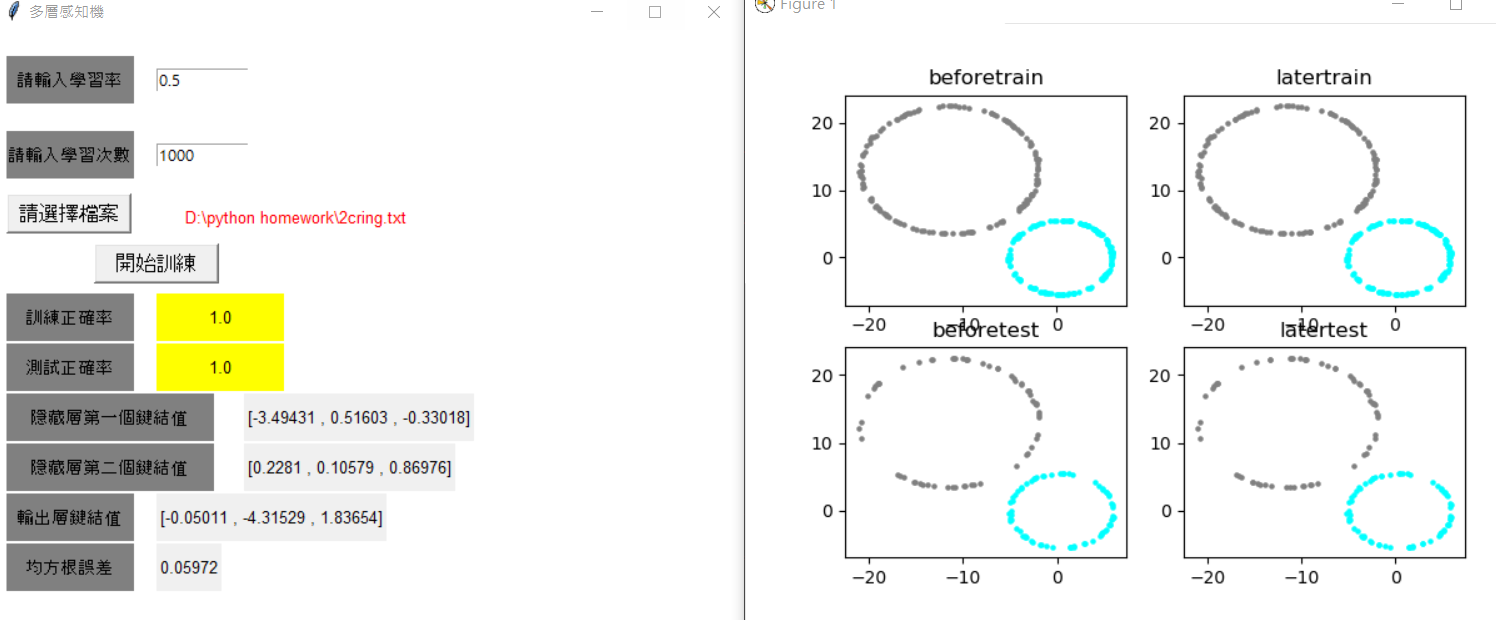
2CloseS2



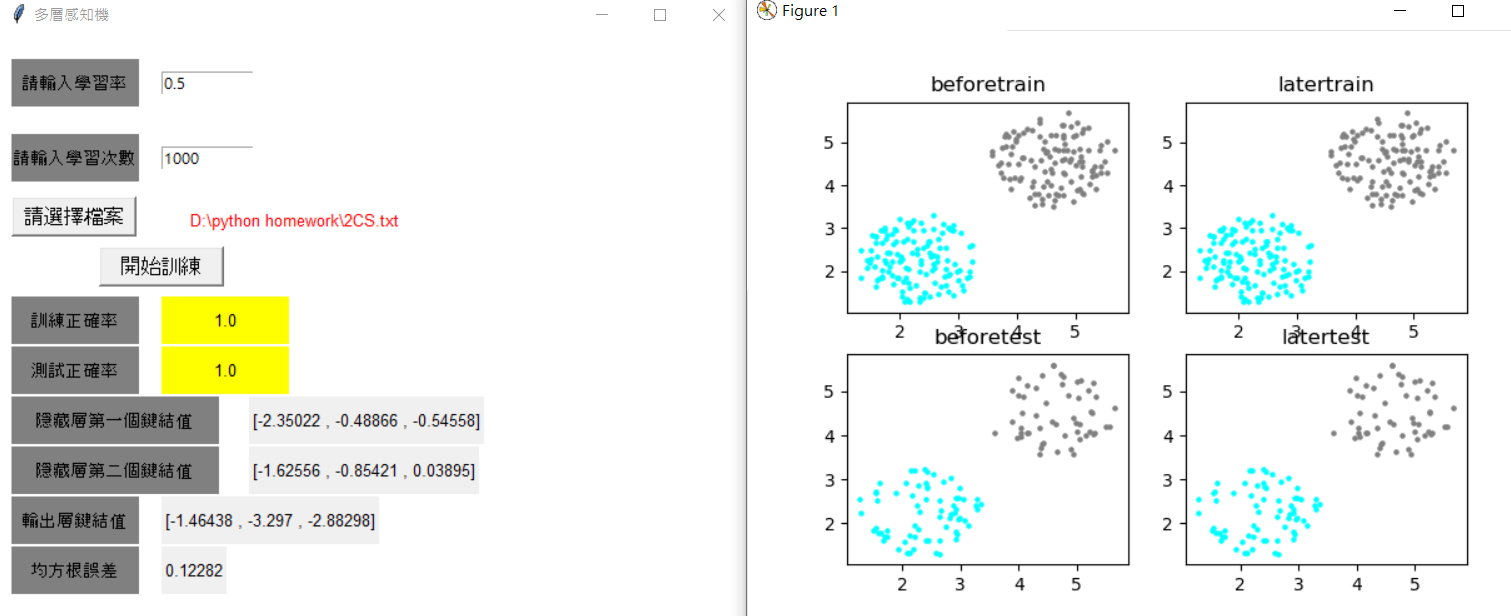
2CloseS3



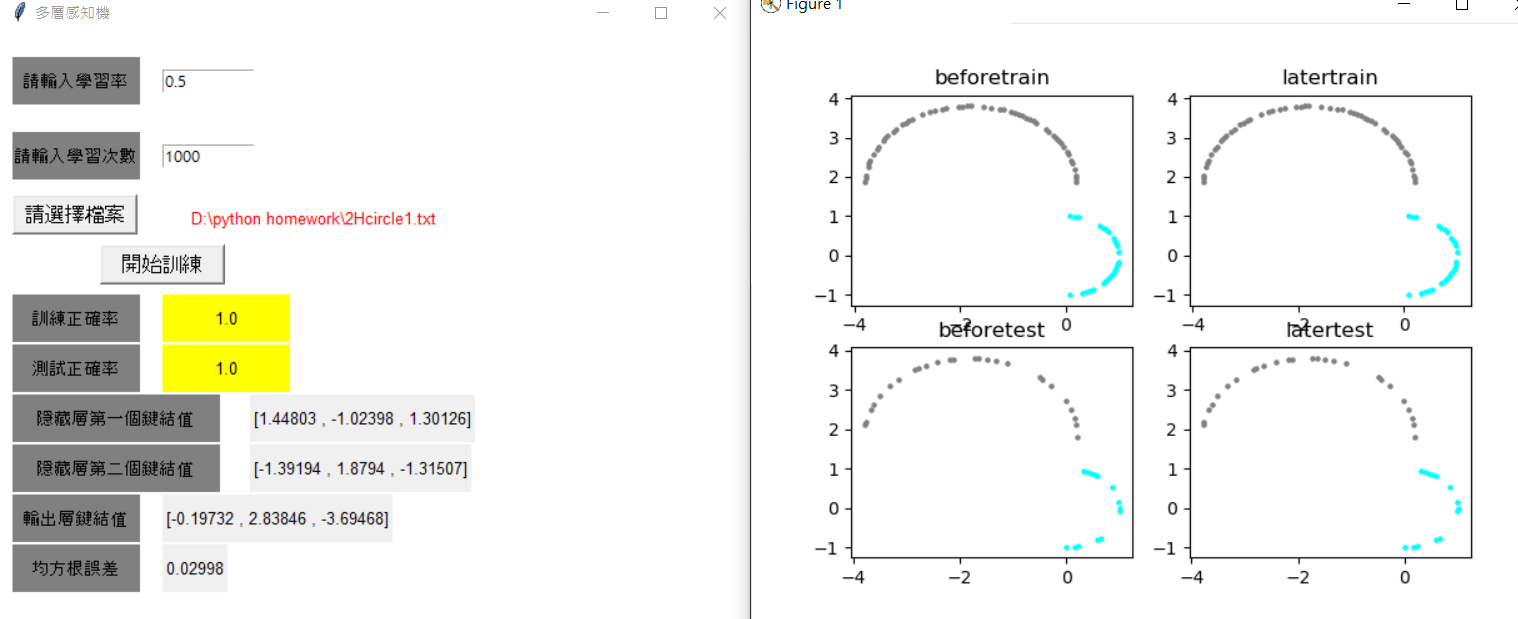
2cring



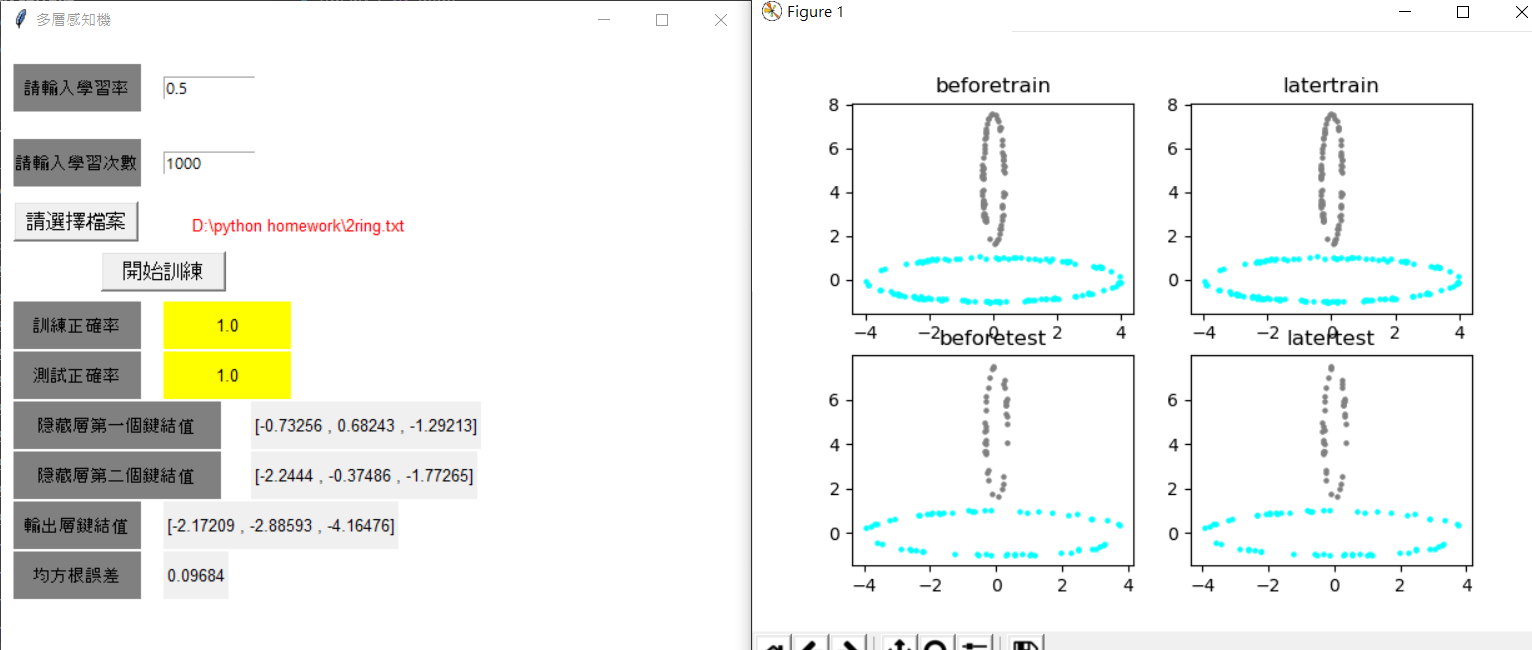
2CS



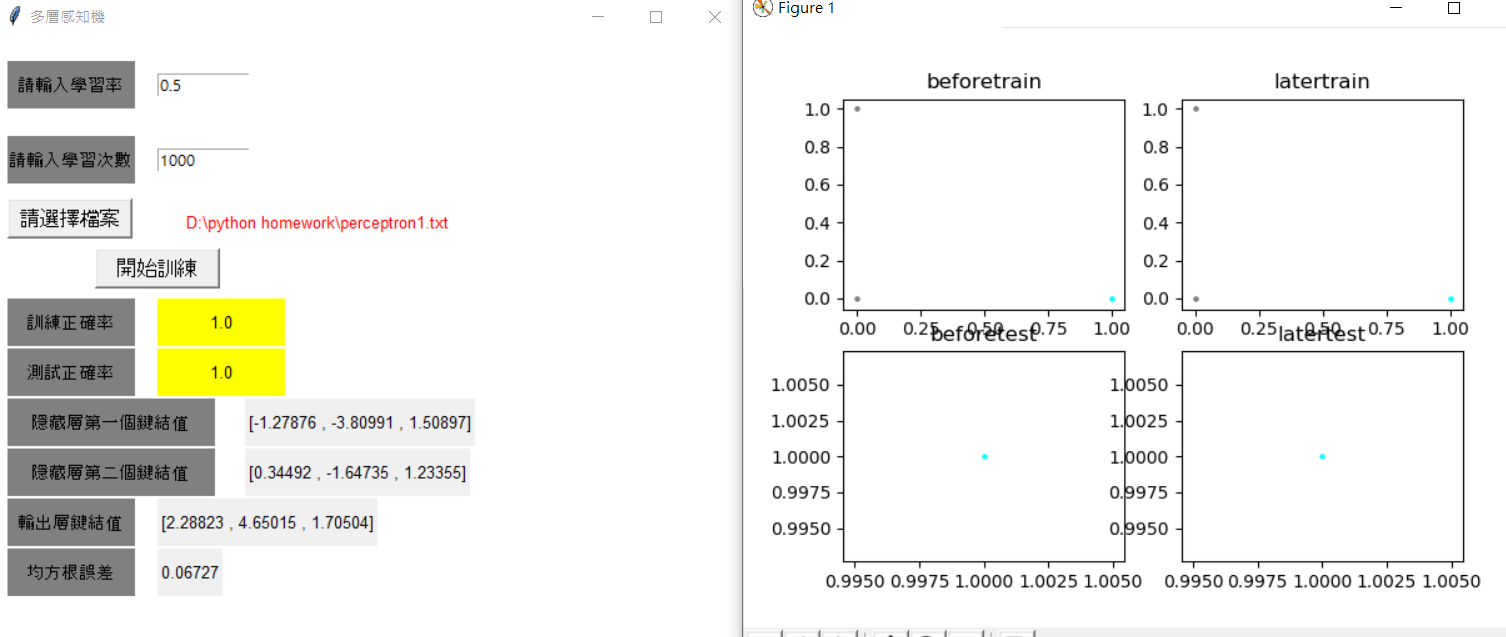
2Hcircle



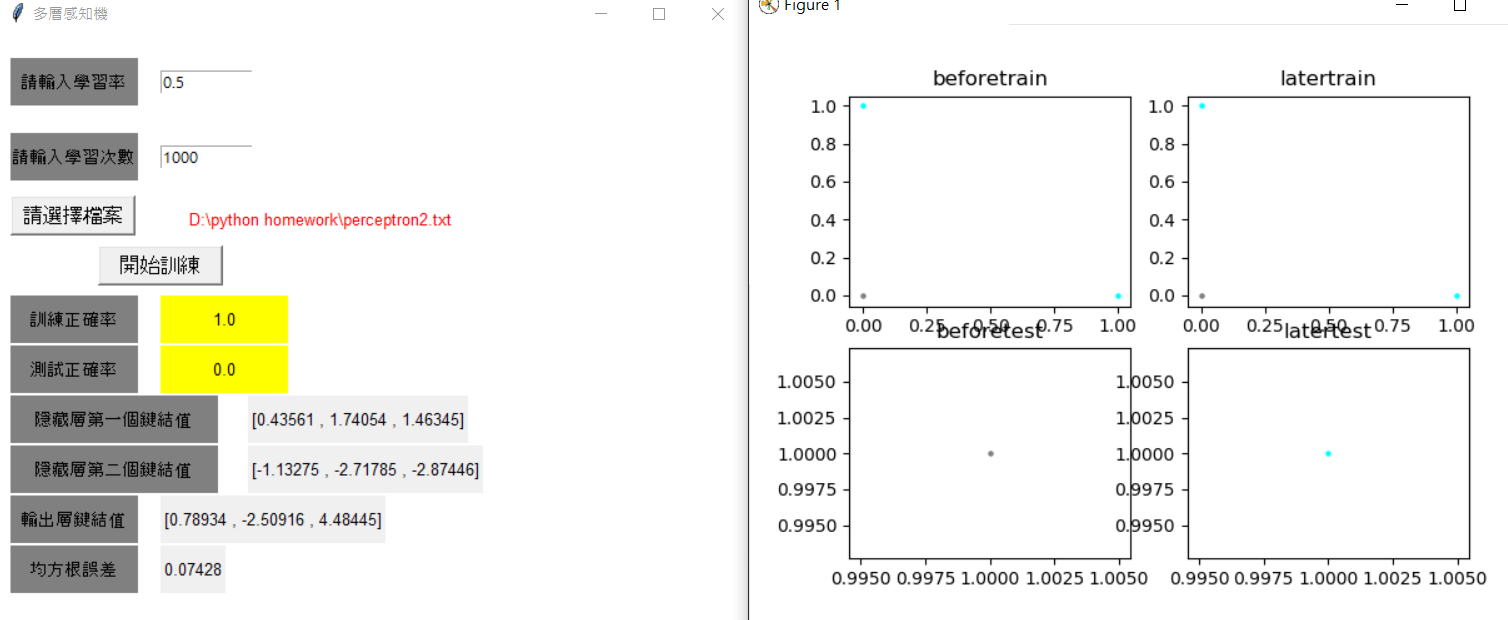
2ring



Perceptron1



Perceptron2



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

如果以訓練的3比測資來看是看不出來最後分類是xor，一定要有4筆資料才有看的出來是xor，所以測試正確率為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.加分題內容

沒有QQ