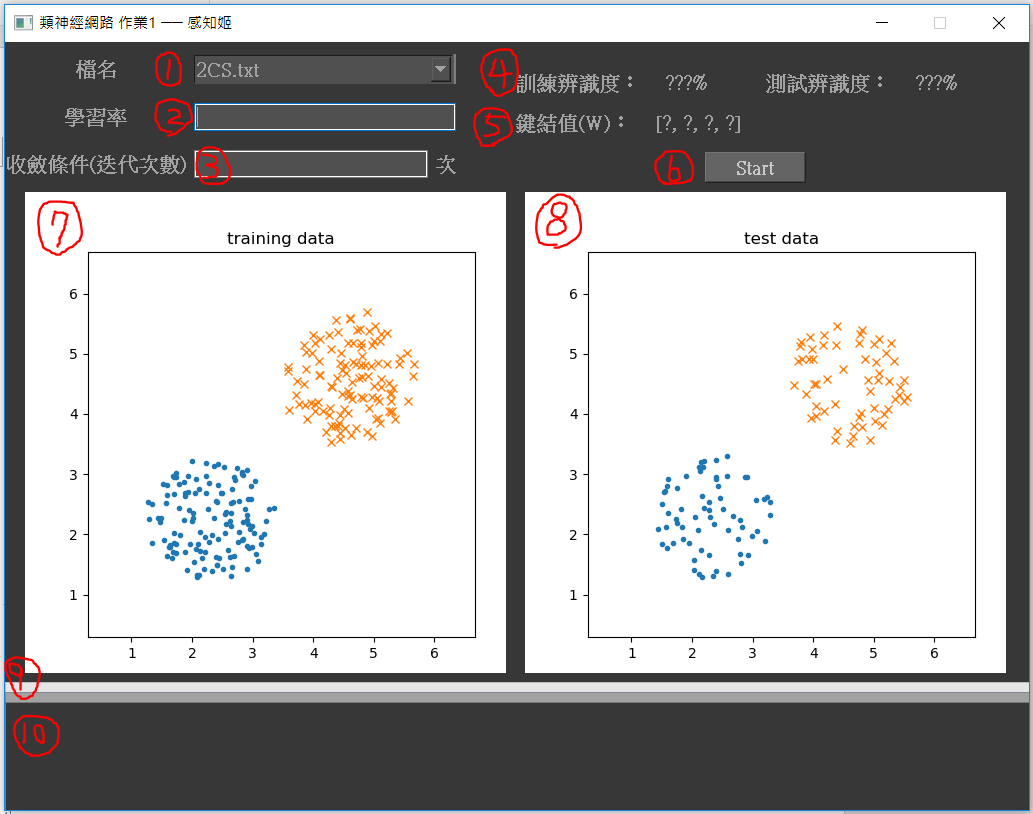
類神經網路HW1─書面報告

104502518 資工4A 劉冠聲

**程式介面說明**



1. 下拉式選單：供使用者選擇DataSet中的檔案(根據電腦不同可能會有些跑位，但不影響使用)
2. 輸入欄位：供使用者輸入學習率
3. 輸入欄位：供使用者輸入收斂次數(以迭代次數計)
4. 顯示計算過程及完成後的訓練資料辨識度和測試資料辨識度
5. 顯示計算過程及完成後的鍵結值([w0, w1, w2])
6. 按鈕：點擊以開始計算或結束計算
7. 訓練顯示圖：顯示訓練過程及結果之鍵結值變化和分群結果
8. 測試顯示圖：顯示以訓練資料計算出的鍵結值變化和套用在測試資料之分群結果
9. 進度條：顯示目前計算執行進度
10. 訊息欄：顯示文字提示使用者進行操作

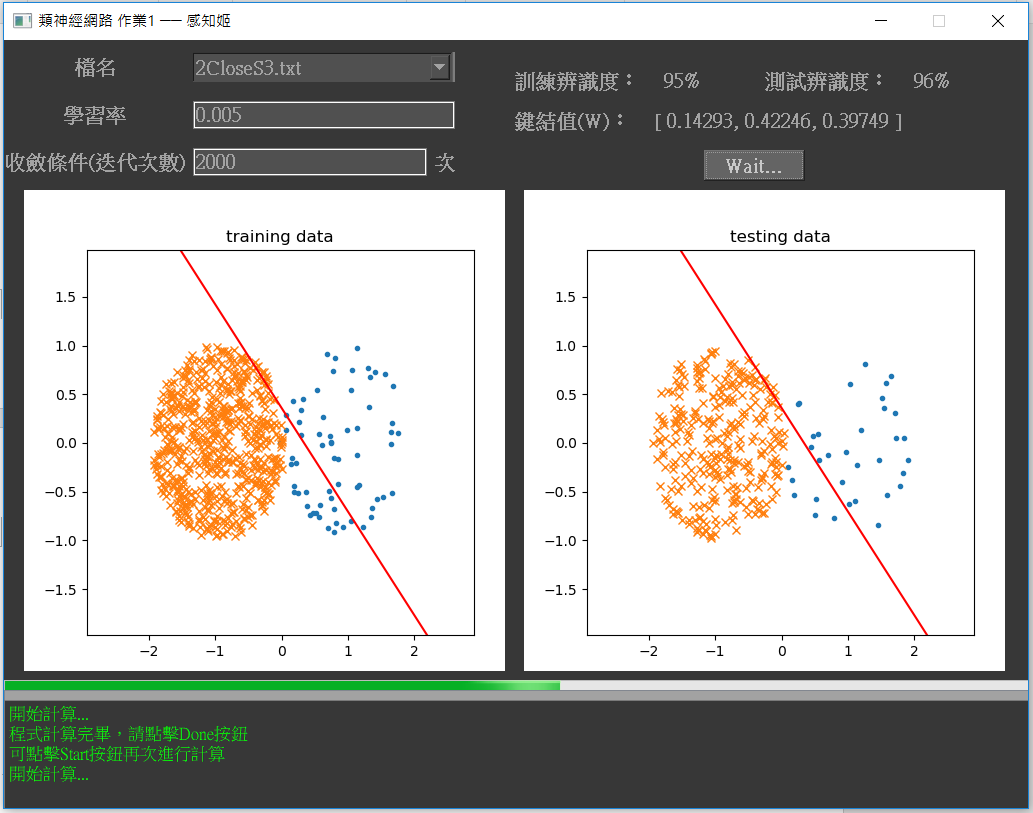
**程式操作說明**

1. 將測試資料放入DataSet資料夾中

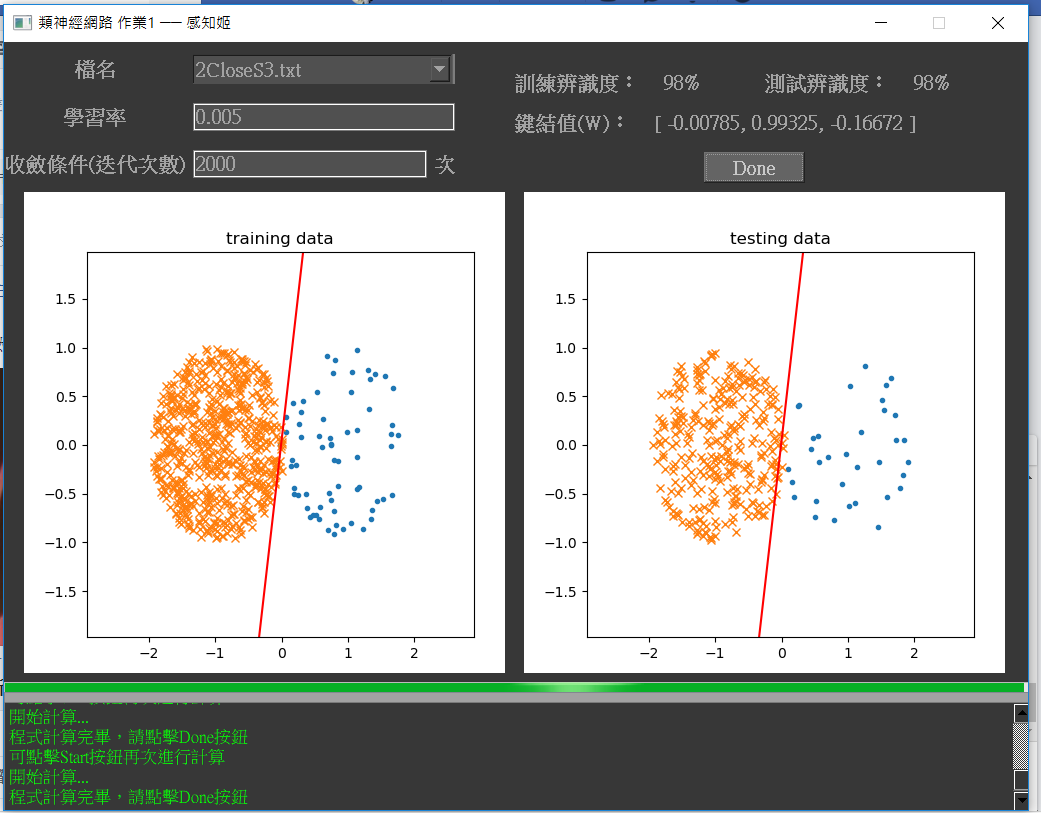
(由於只有完成2維資料的感知機，建議不要放入3維以上的資料)

1. 執行程式 104502518\_劉冠聲\_作業一.exe
2. 等待小黑窗執行一段時間即會顯示GUI視窗
3. 選擇欲計算之檔案
4. 輸入學習率和收斂條件(迭代次數)
5. 點擊Start按鈕開始對訓練資料進行計算
6. 待計算完畢後，顯示之訓練、測試辨識度和圖中之分群狀況即為最終結果
7. 點擊Done按鈕即完成一次感知機操作
8. 可再從第2步開始重複操作

**程式執行示意圖**

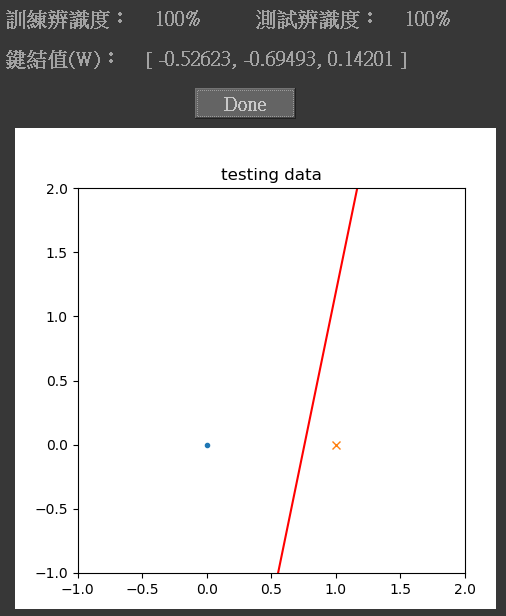
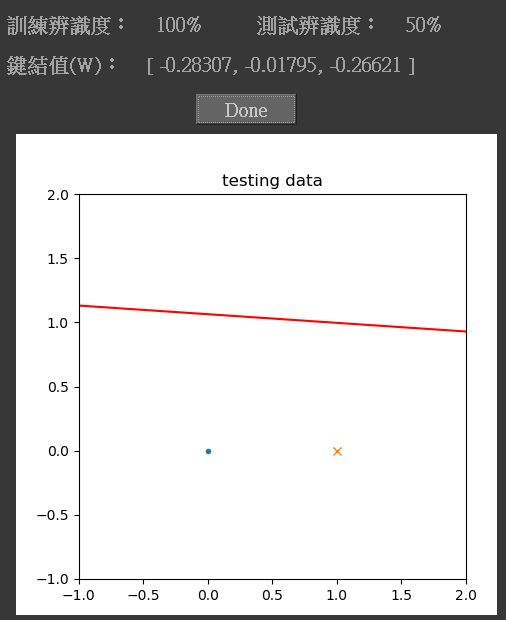


**程式完成示意圖**

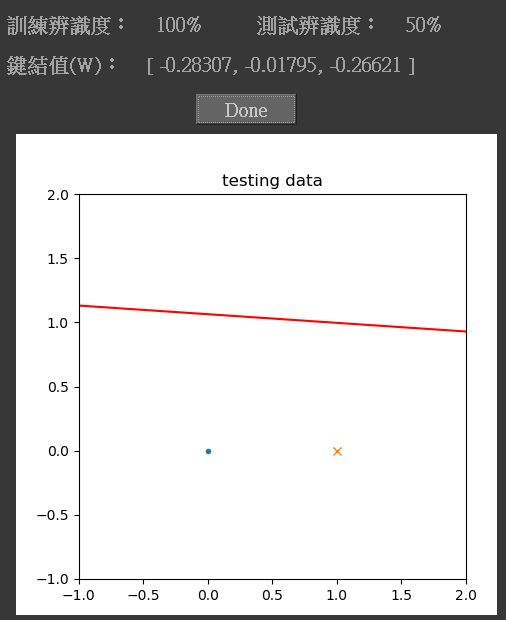
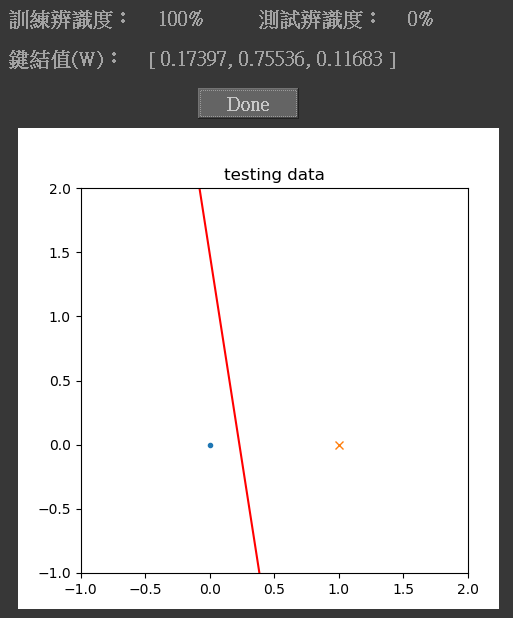


**實驗結果**

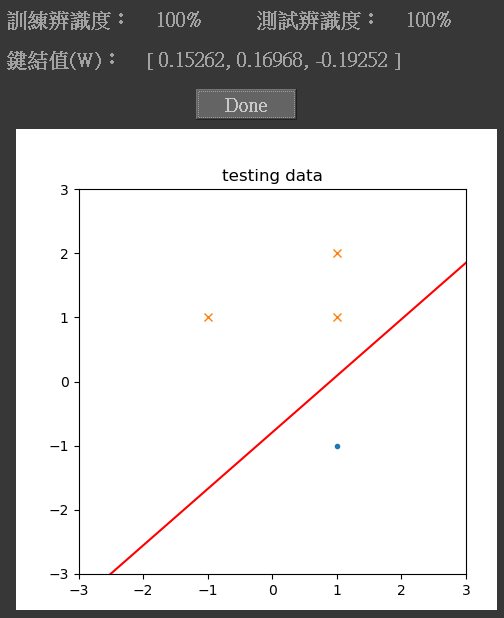
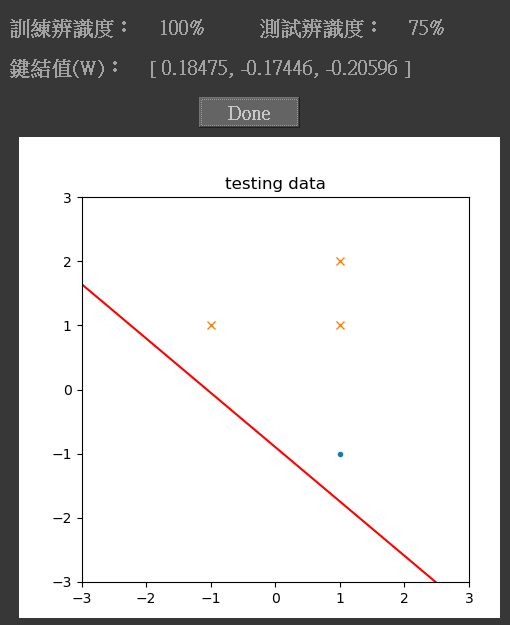
* **perceptron1.txt**：由於資料集只有4點，再分成training data後會過少，導致雖然訓練辨識度一定是100%，但測試辨識度可能是100%或50%

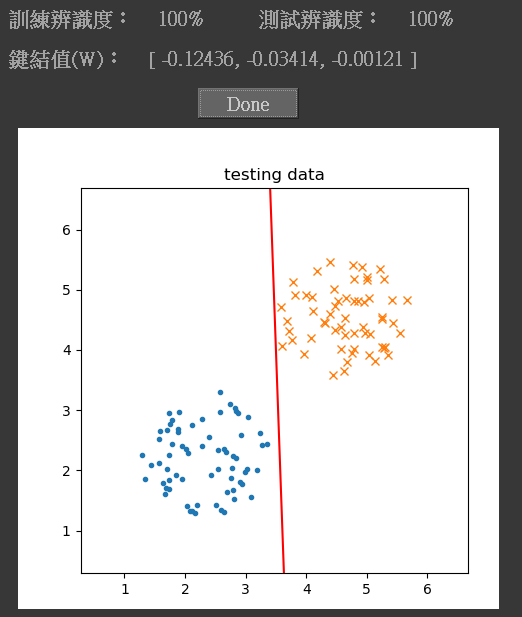
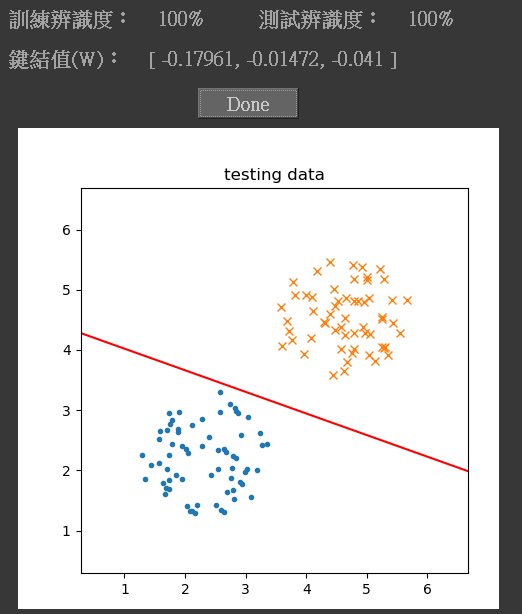
* **perceptron2.txt**：資料點過少同上，但由於此資料集之分布為線性不可分割，所以雖然訓練辨識度一樣是100%，測試辨識度卻是50%或0%

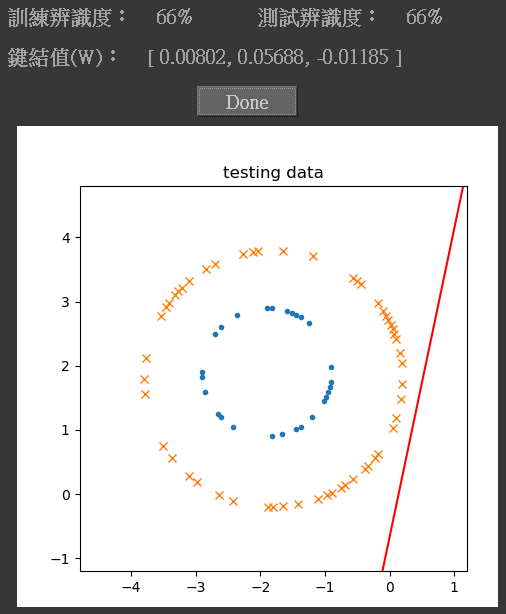
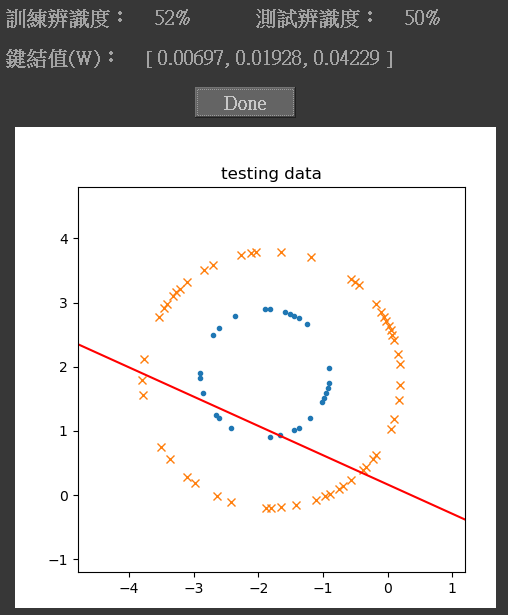
* **perceptron4.txt**：雖然資料比上兩個多一些，但仍舊不足，訓練辨識度依舊一定是100%，但測試資料可能未完全分群成功即停止繼續收斂

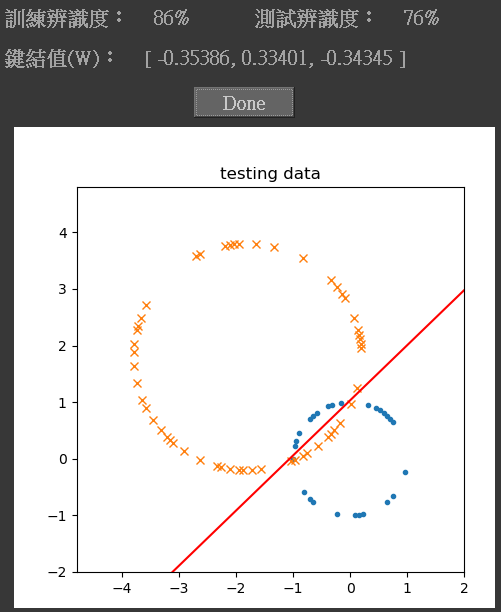
* **xor.txt**：同perceptron2.txt
* **2CS.txt**：斜率能成功收斂至以下兩鍵結值間，通常達到100%辨識率

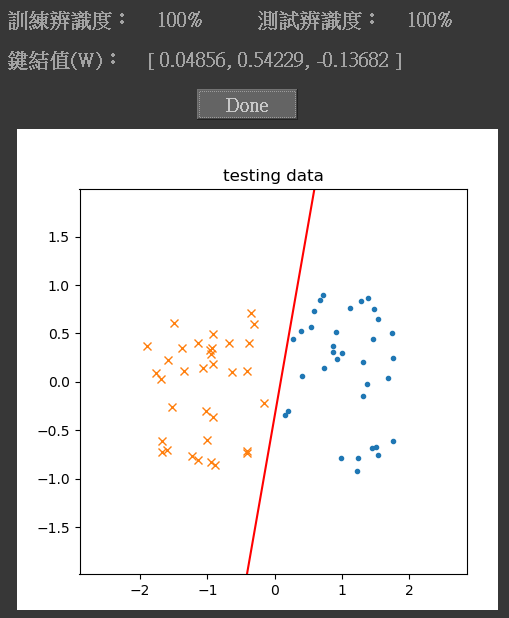
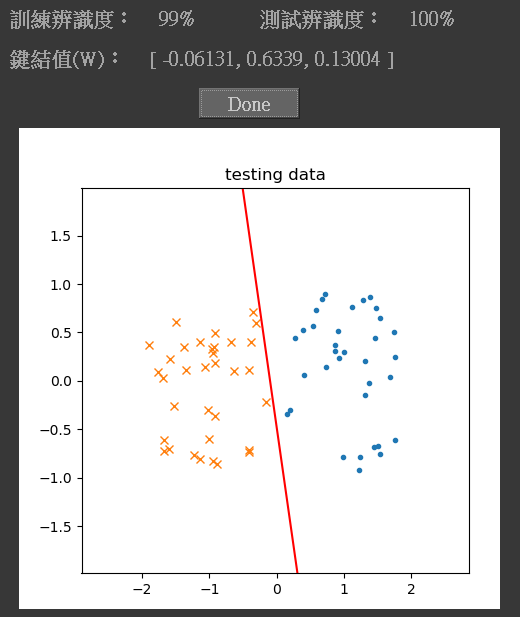
* **2Ccircle1.txt**：由於並非線性可分割，無法達成100%辨識率，如下圖，通常收斂至圈外，或切在圓上某一地方，無參考性

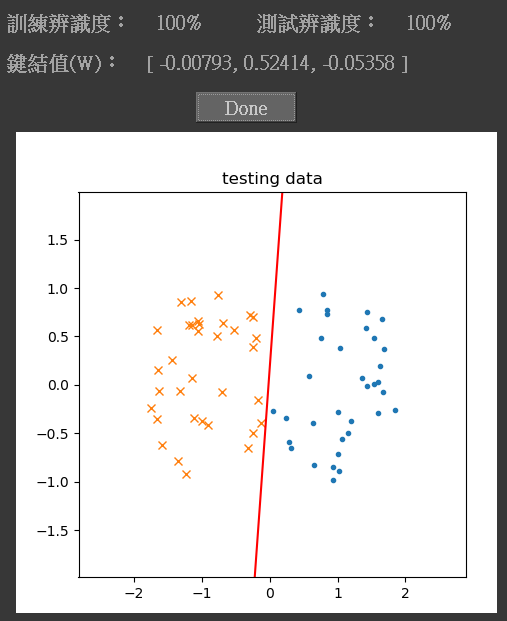
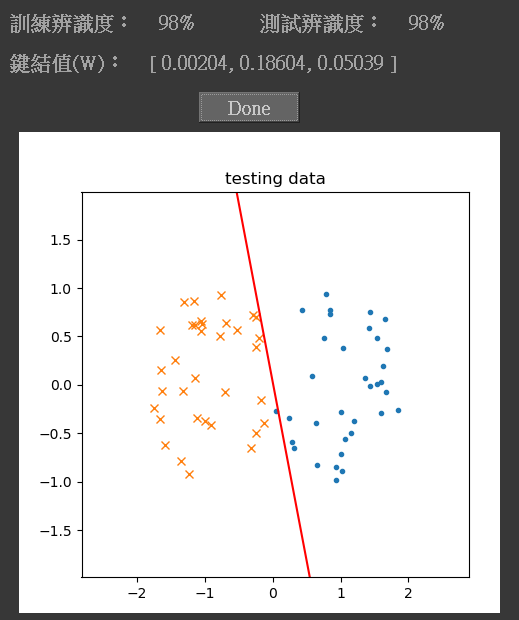
* **2Circle1.txt**：也是非線性可分割，不過兩圓交集只有一部份，通常能收斂至兩圓相交點達到高辨識率



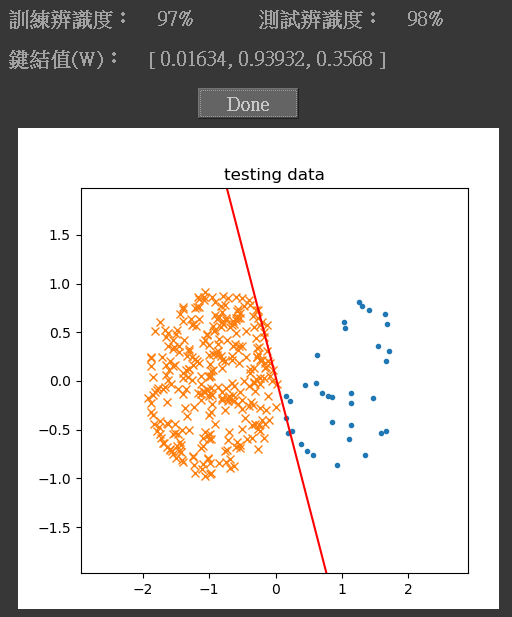
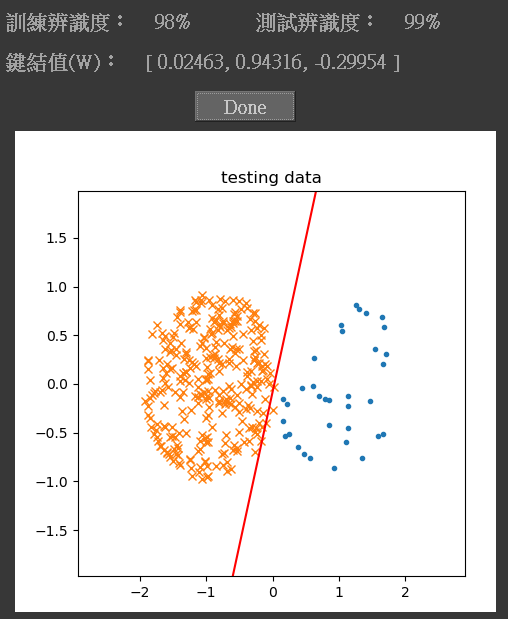
* **2Circle2.txt**：同上，只是資料集更多
* **2CloseS.txt**：幾乎為線性可分割，不過兩群集相聚頗近，有時收斂次數不夠或資料分布剛好交錯，辨識率只能達到近100%，收斂至以下兩斜率之間

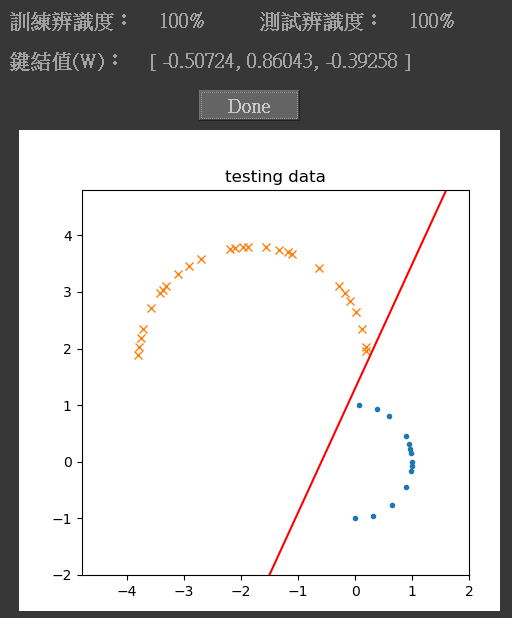
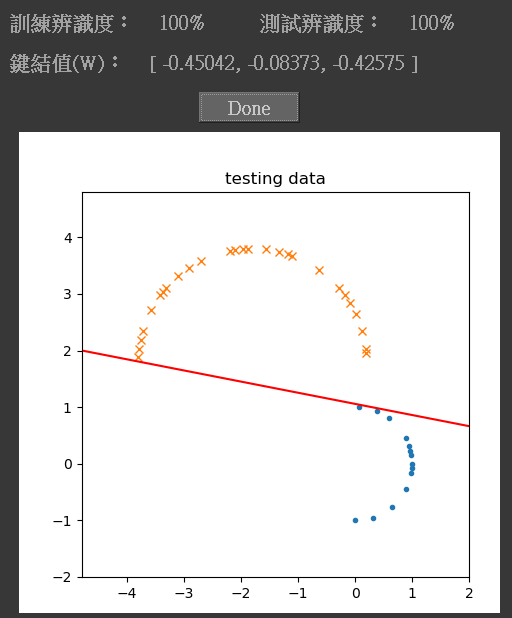
* **2CloseS2.txt**：同上，只是資料集更多

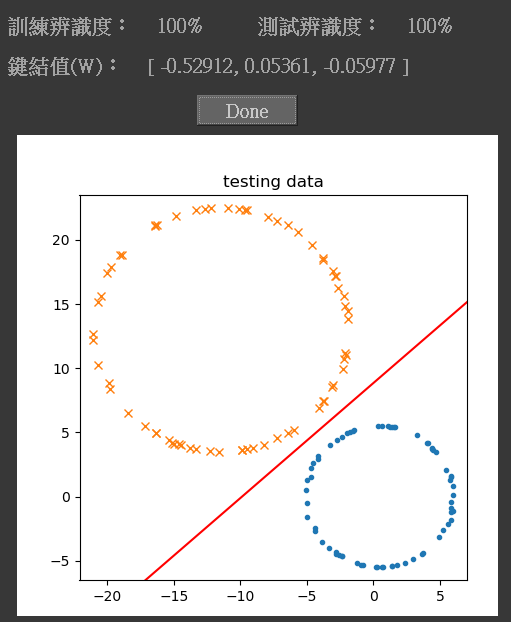
* **2CloseS3.txt**：同上上，只是資料集更多多多多，更難完全分割

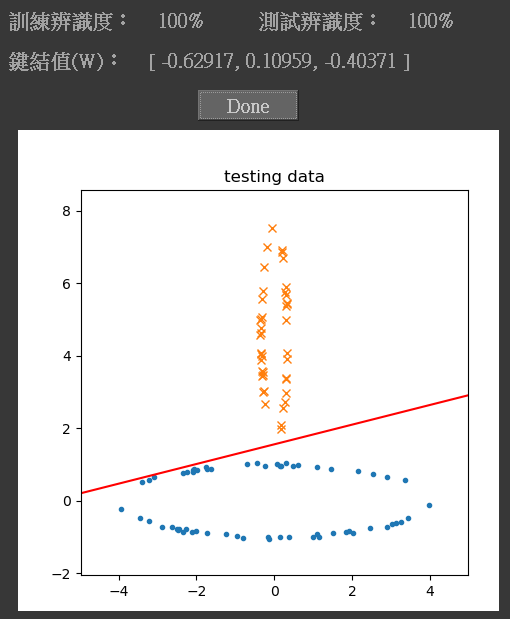
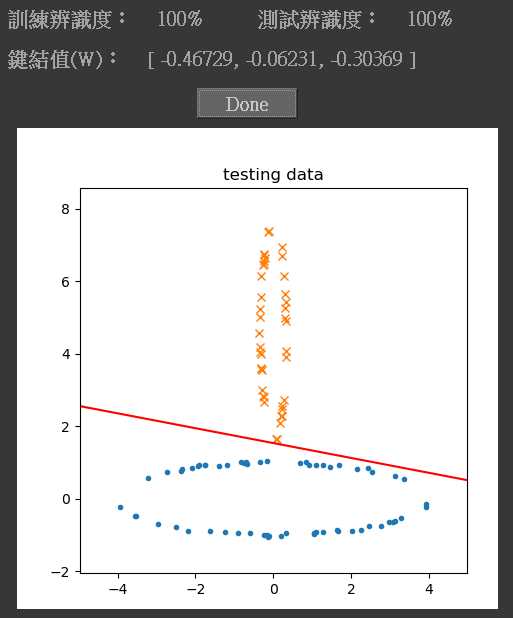
* **2Hcircle1.txt**：為線性可分割，通常都能收斂至以下兩鍵結值之間，通常辨識率達到100%

* **2cring.txt**：為線性可分割，能成功收斂至兩圓之間分群，通常辨識率100%



* **2ring.txt**：線性可分割，能成功收斂至以下兩鍵結值之間，通常辨識率100%

**分析及討論**

我的作法是先用Qt Designer刻好GUI介面後，再開一個類別去繼承ui檔產生的python code中的Qt\_MainWindow，作為跑主程式的類別(Main)，之後在Main中加入其他類別的instance(FileManager、Calculator、PlotCanvas)，並為這些instance開QThread去跑他們的方法完成這次感知機的作業。

一開始會先由FileManager讀取檔案的每一行後進行shuffle打亂順序達成隨機的效果，分成2種不同的result後將前1/3的資料為test data，後2/3的資料作為train data，並讓PlotCanvas存取進行繪圖，再來透過Calculator存取train data進行計算，更新Weights後傳給PlotCanvas繪製來視覺化分群效果，由於各thread跑的速度各自時快時慢，計算過程中train圖和test圖不會同步更新，但不影響最終結果的呈現，最後透過Signal和Slot關閉QThread，等待下一次感知機計算。

對於2維資料集的分析，為了測試方便快速，通常學習率都會設定比較大一點次數少一點，但有時無法保證辨識率達到100%或是盡可能高，但學習率過小可能會導致在某一區域來回震動無法優化，可能真的要透過一開始設定學習率大，藉由每次迭代慢慢減少學習率的方式來改善，迭代次數也不能太少，是這次作業實作還能再改善的部分。