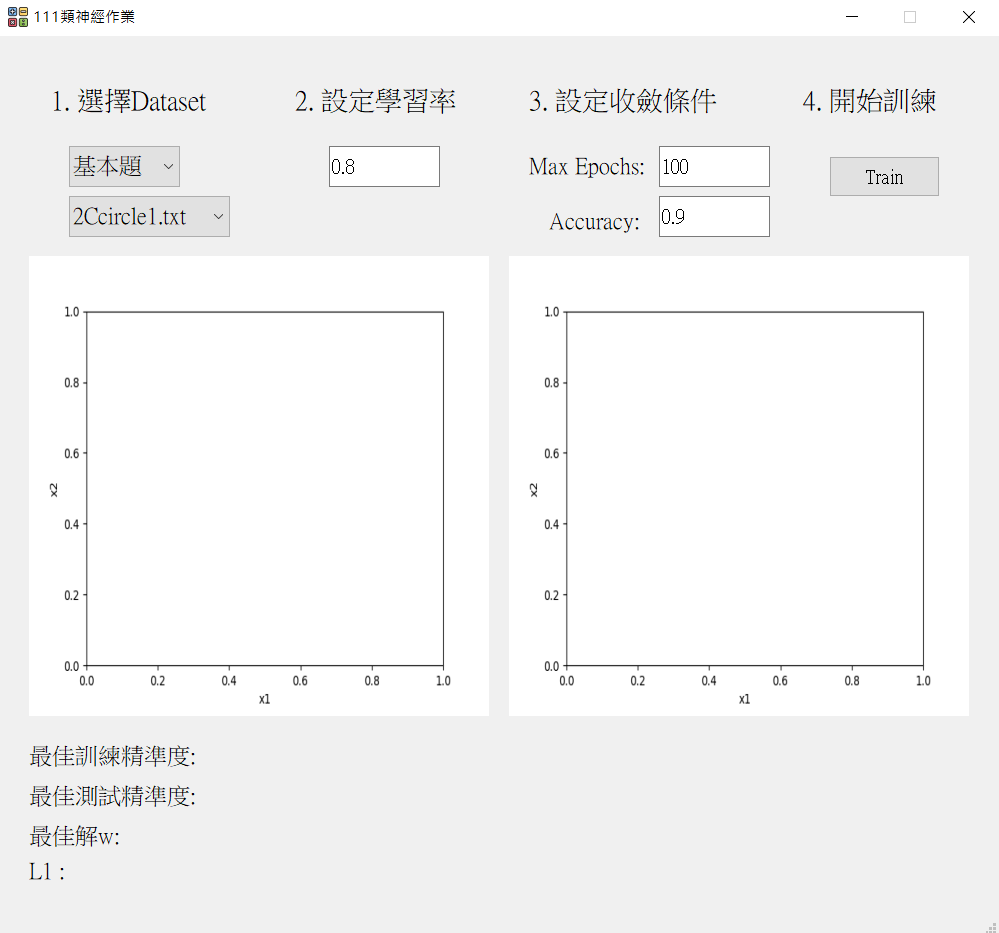
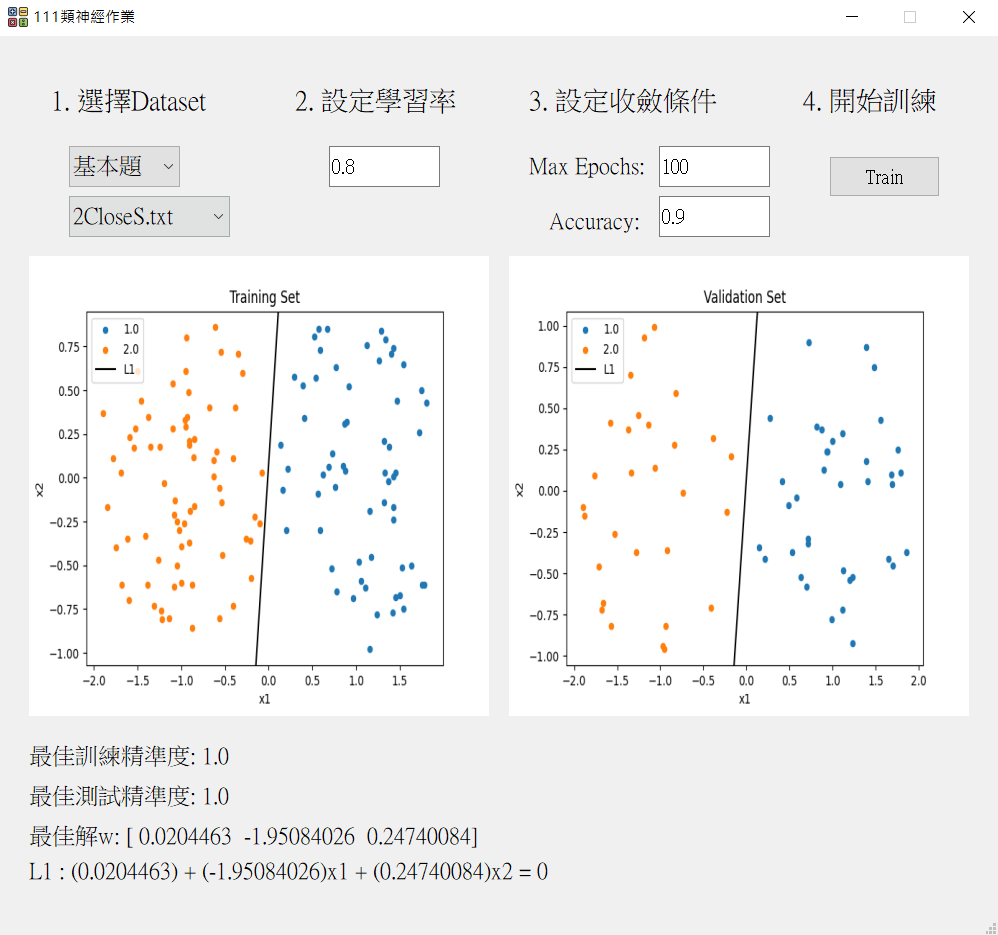
1. 程式執行說明:

**主視窗:** 開始訓練前先設定好以下要求，並在訓練完成後顯示Training set之二維資料圖以及Validation set之二維資料圖，最下方會顯示最佳訓練精準度、最佳測試精準度、最佳解w以及方程式L1。

1. 選擇Dataset: 目前只完成基本題，因此只有基本題的題號可以選擇。
2. 設定學習率: 輸入浮點數學習率。
3. 設定收斂條件: 最大迭代圈數 以及 精準度門檻，其中一個先達標就會停止訓練。
4. 開始訓練: 按鈕點下去後就會開始訓練並把訓練以及測試結果顯示於下方，若學習率、最大迭代圈數、精準度有不符合規格之輸入會在Terminal印出”輸入格式錯誤”之訊息。



圖一、主畫面



圖二、運行”2CloseS.txt”之結果

1. 程式碼簡介

程式碼分為幾個部分:

def main(): 執行視窗 & 視窗關掉後會刪除暫存的圖片。

class My\_UI(QtWidgets.QMainWindow): 自定義之Qt視窗類別，包含一些選擇物件、輸入物件、一顆按鈕、一些文字label、兩個圖片label以及以下function。

def comboBox\_change(self): 切換基本題以及加分題會改變另一個QcomboBox之選項，但目前只完成基本題因此只有基本題可以選。

def raw\_data\_process(self,path\_to\_file): 將檔案內的資料讀取出來，初步處理成nparray並回傳。

def Training\_single\_perceptron(self,max\_epoch,lr,data): 將raw\_data\_process回傳之data進一步處理成可以訓練的X以及Ground Truth Y，再進一步將X Y分割成訓練集以及測試集；訓練模型的方式採用基本PLA演算法，每跑完一個資料點就會更新w；訓練過程如果發現w能完美切割訓練資料點則會立即停止訓練並回傳w，其餘時候則將最大迭代圈數內能找到之最佳解w當作回傳值。

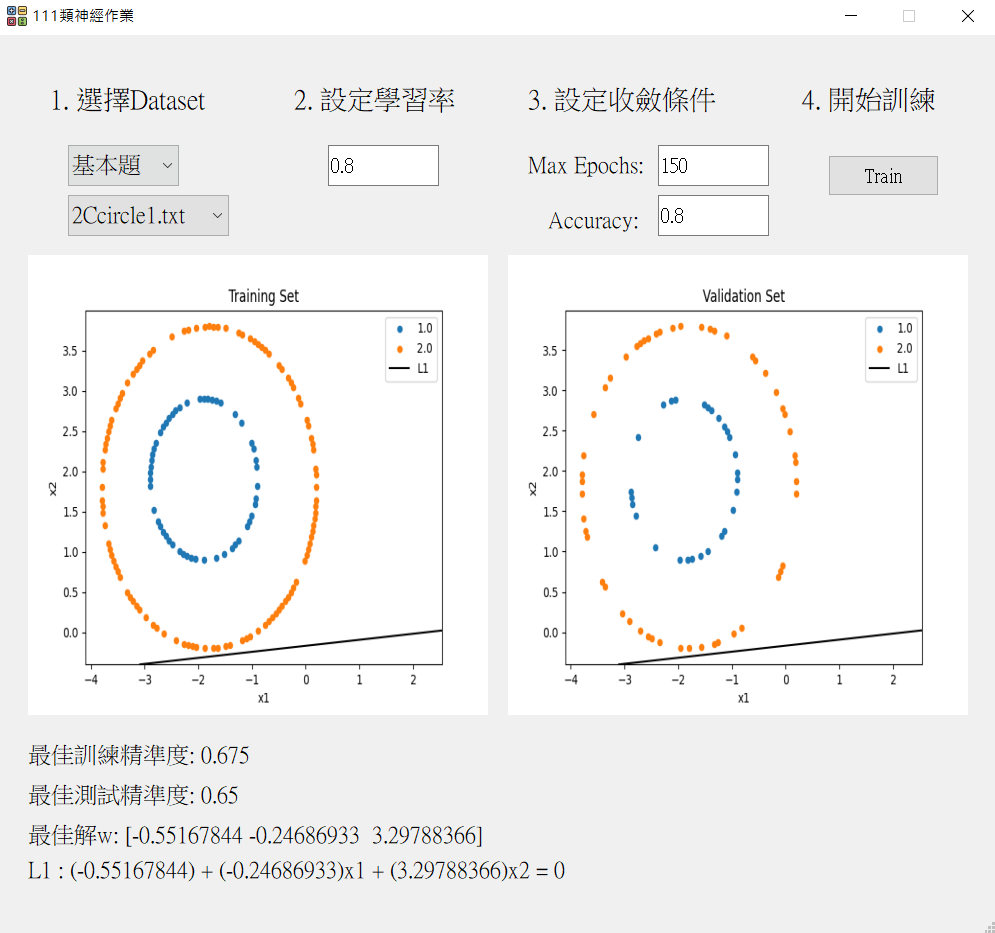
def Training(self): 呼叫function “Training\_single\_perceptron”取得回傳值w，再呼叫function “Plot”進行資料點繪圖。

def Plot(self,w,data,X\_train,X\_valid,Y\_train,Y\_valid): 根據傳入之資料進行繪圖。

def Hard\_limit(self,x,Y): 當x<=0回傳Y[0]，其他時候回傳Y[1]，Y代表當前分類的種類(ex:[1,2]、[0,1])

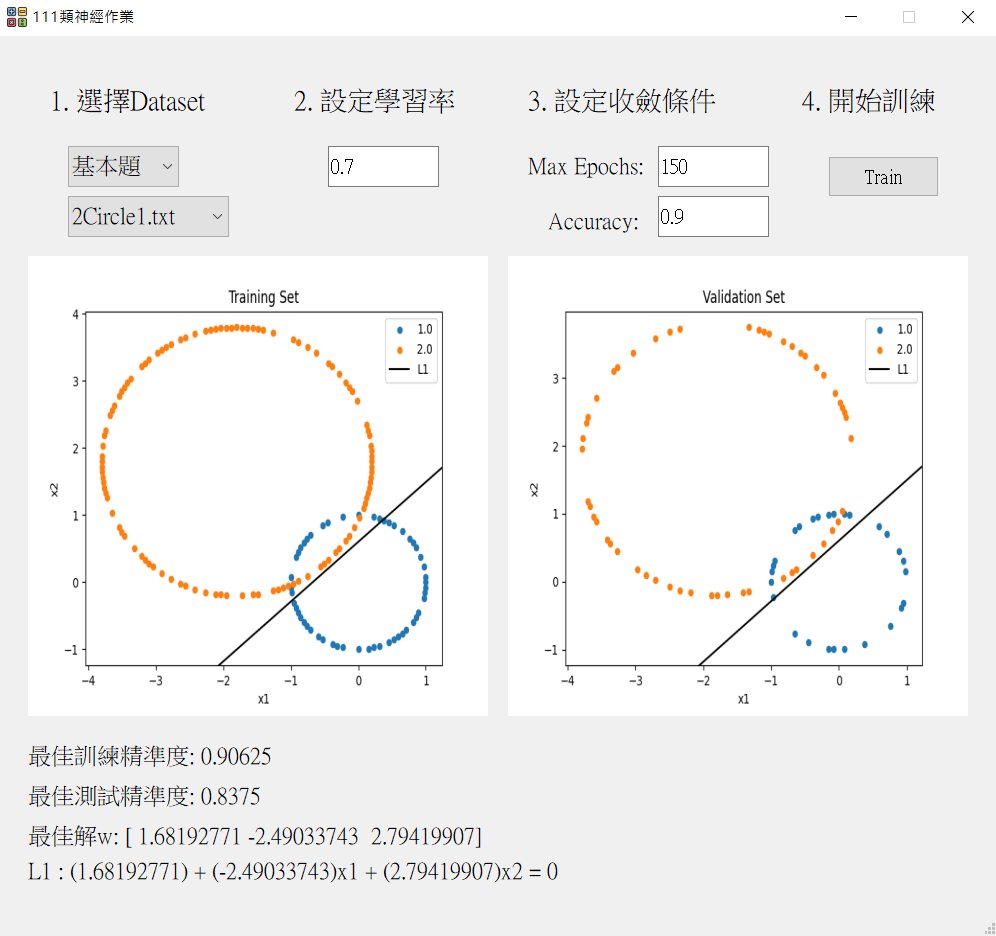
1. 實驗結果

1.2Ccircle1.txt: 這題無法分割資料點，最佳訓練精準度落在0.6~0.7。



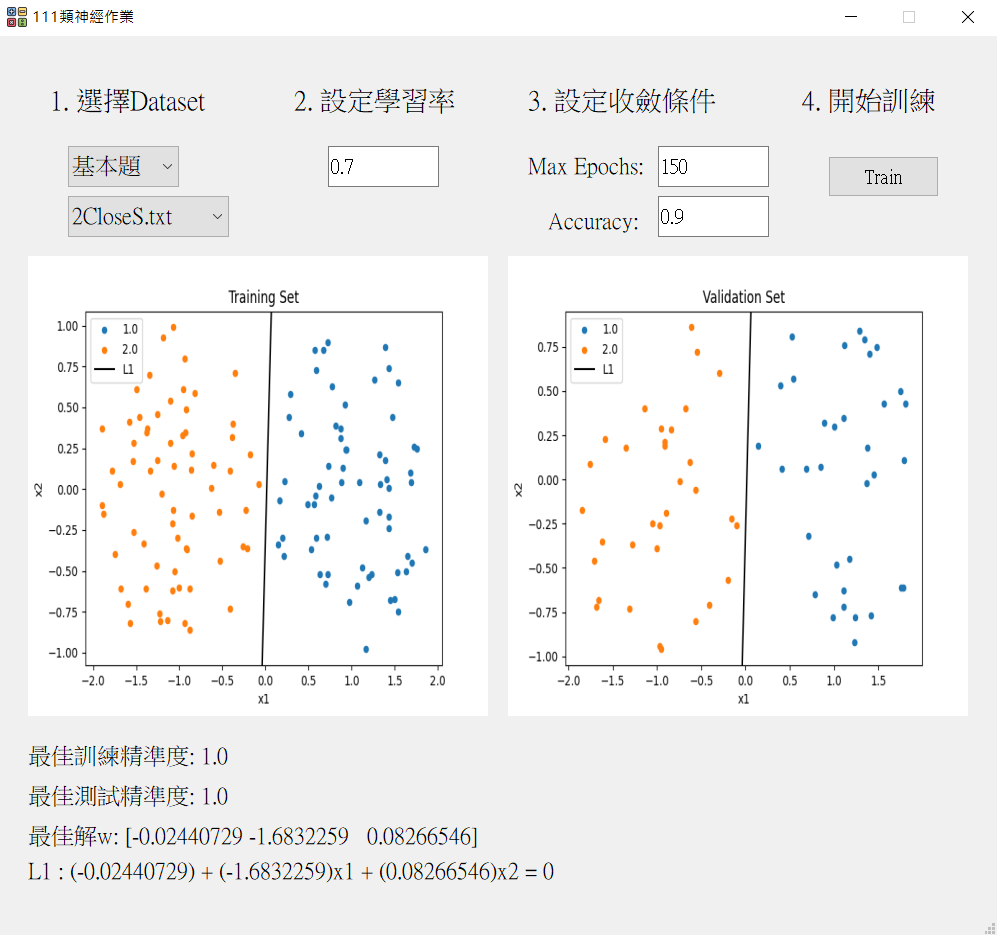
圖三、運行” 2Ccircle1.txt”之結果

2.2Circle1.txt: 這題無法完全分割訓練資料點，最佳訓練精準度落在0.85~0.9。



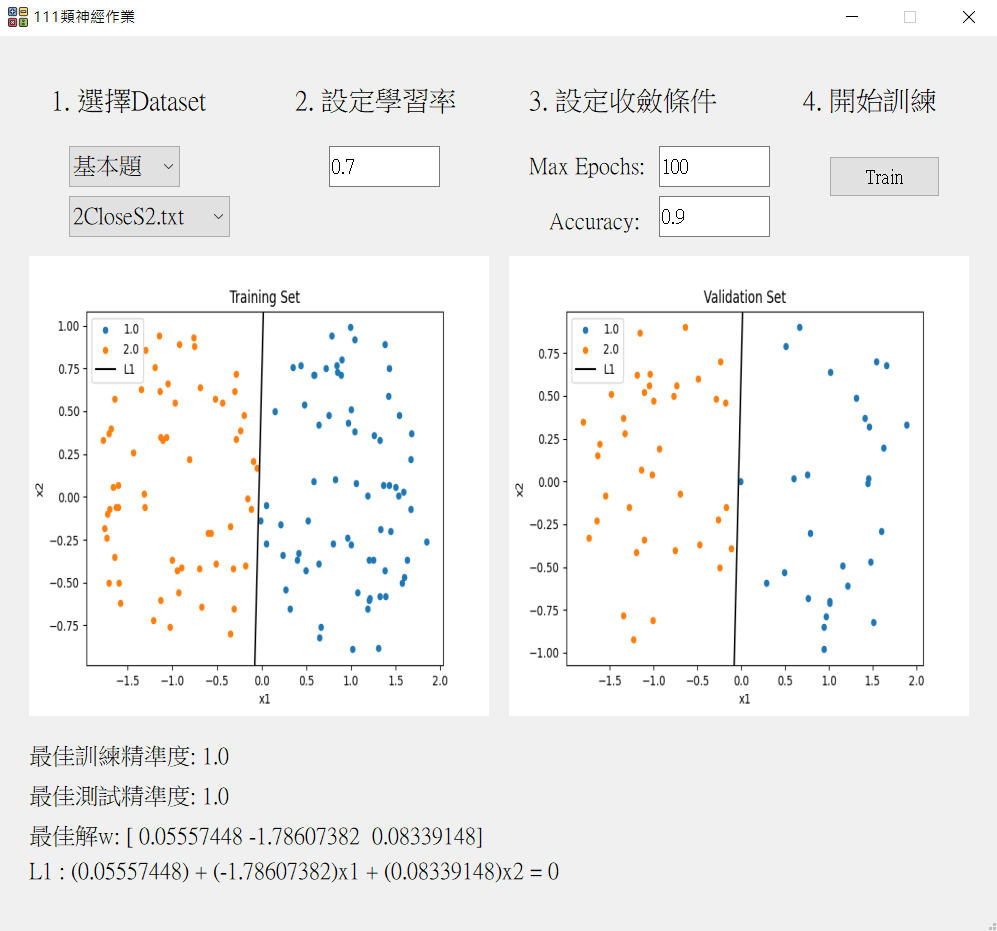
圖四、運行” 2Circle1.txt”之結果

3.2CloseS.txt: 輕鬆分割。



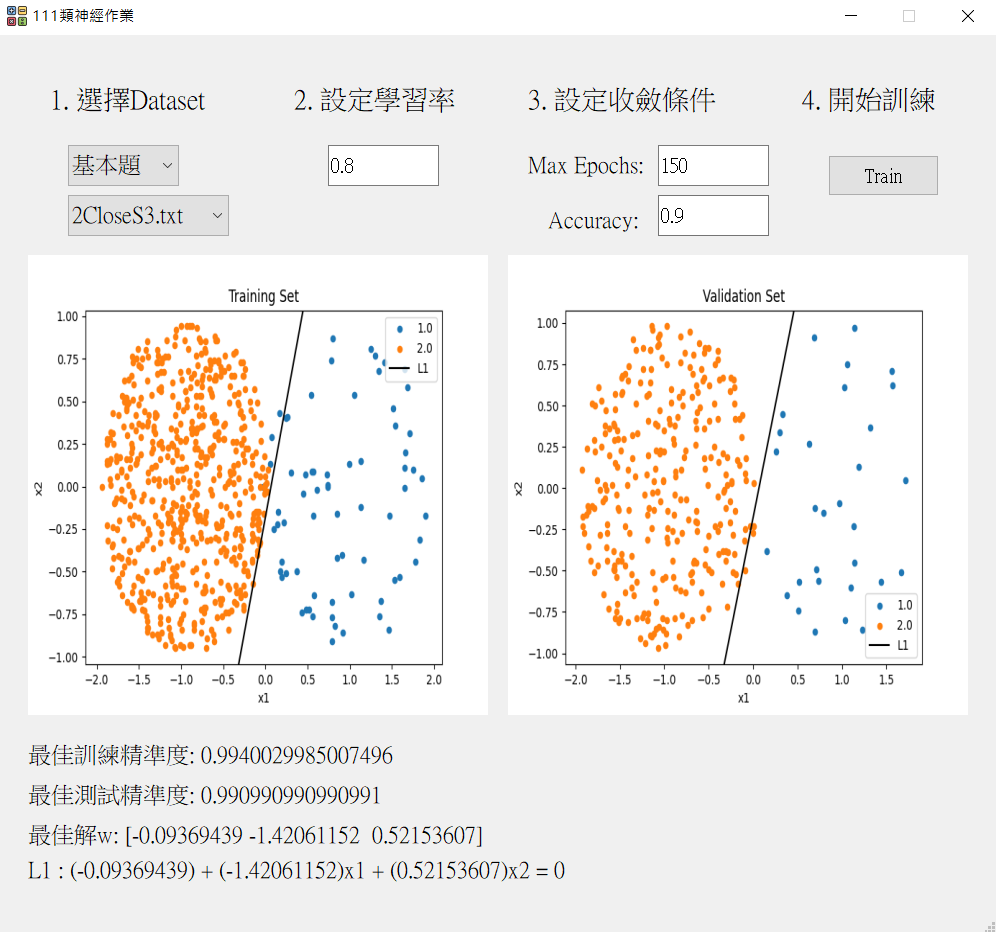
圖五、運行”2CloseS.txt”之結果

4.2CloseS2.txt: 隨機採樣1/3當作測試集，這題運氣好一點可以完美分割。



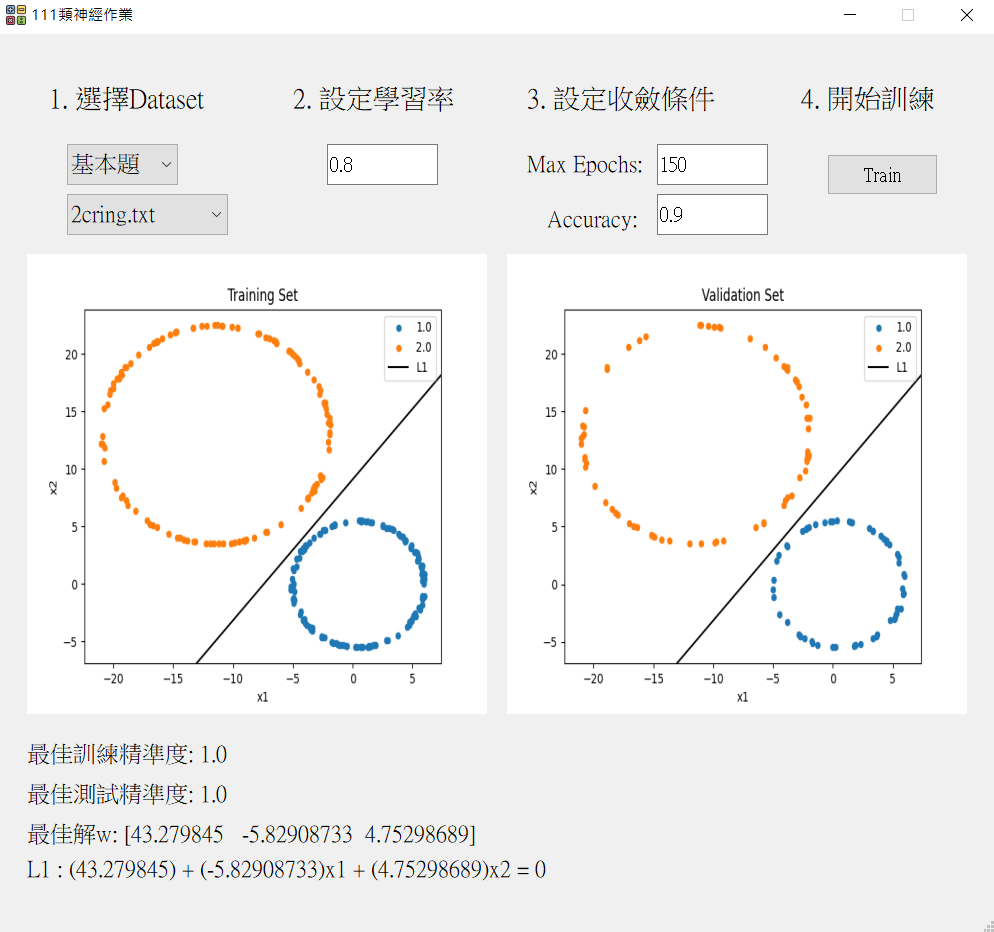
圖六、運行”2CloseS2.txt”之結果

5.2CloseS3.txt: 這題有1~2個藍點無法與橘點做分割，最佳訓練精準度落在0.99。



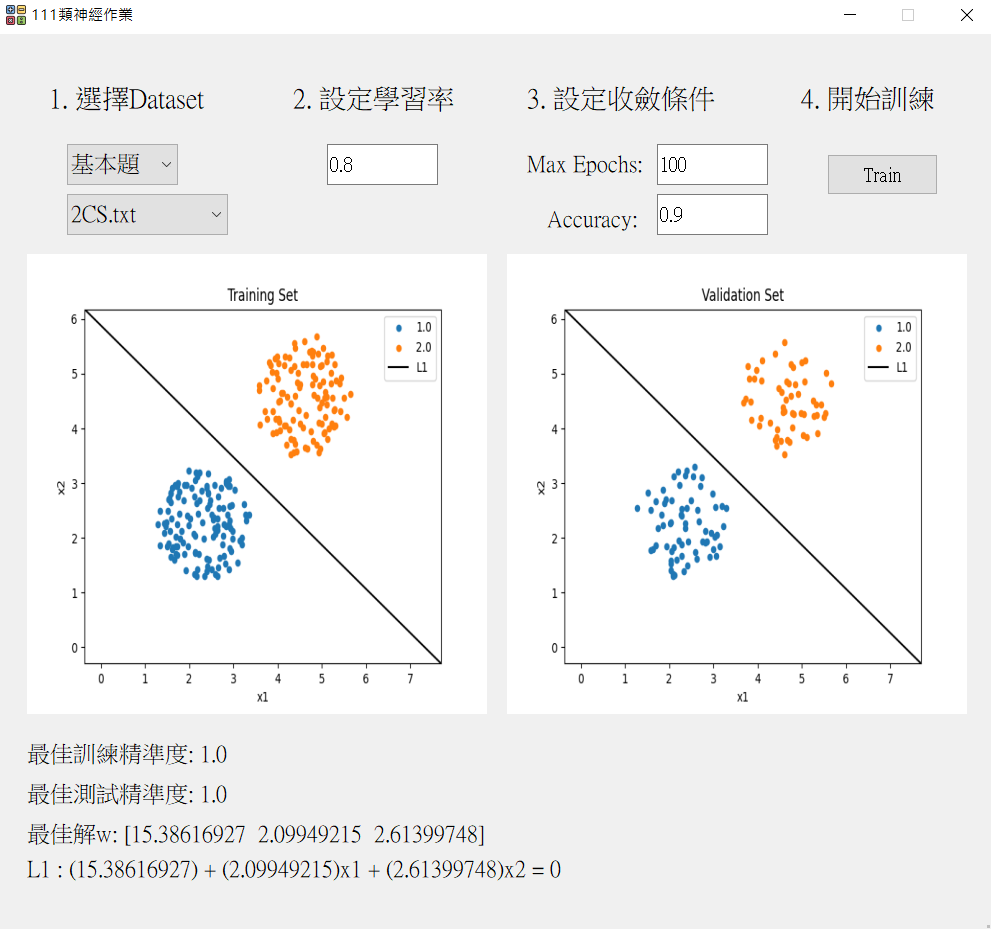
圖七、運行”2CloseS3.txt”之結果

6.2cring.txt: 輕鬆分割。



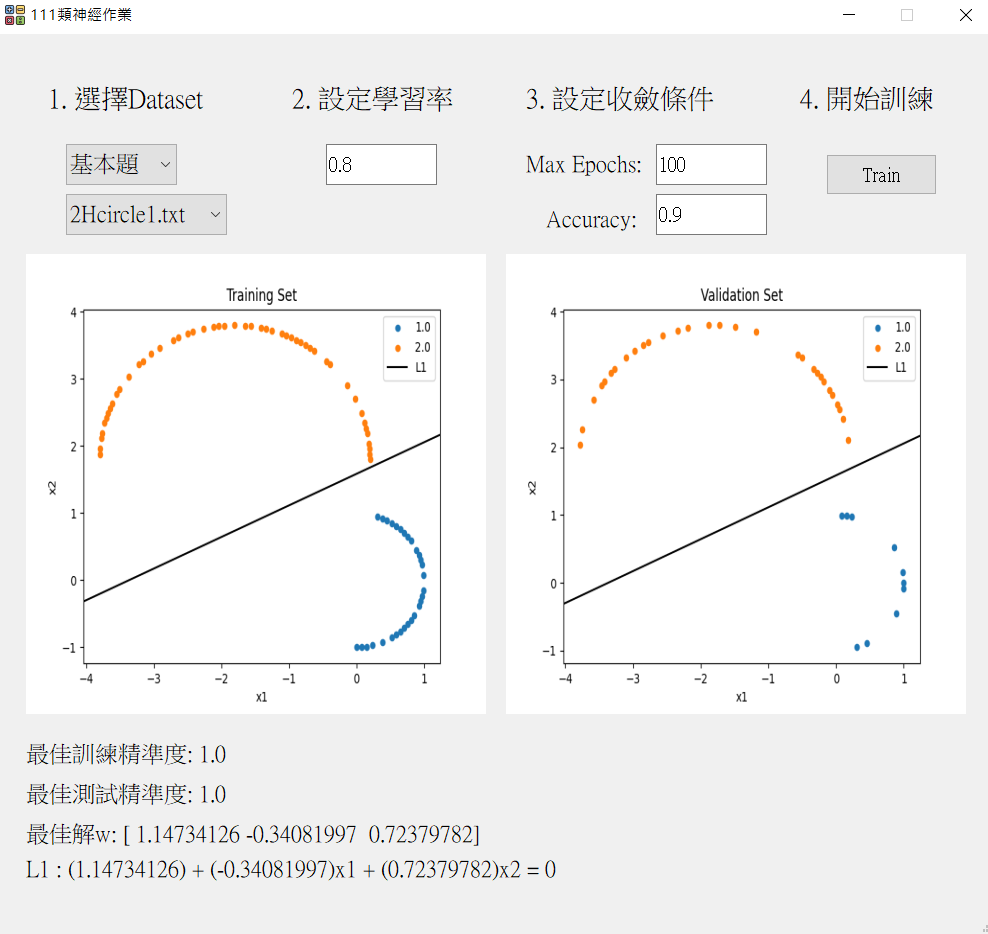
圖八、運行”2cring.txt”之結果

7. 2CS.txt: 輕鬆分割。



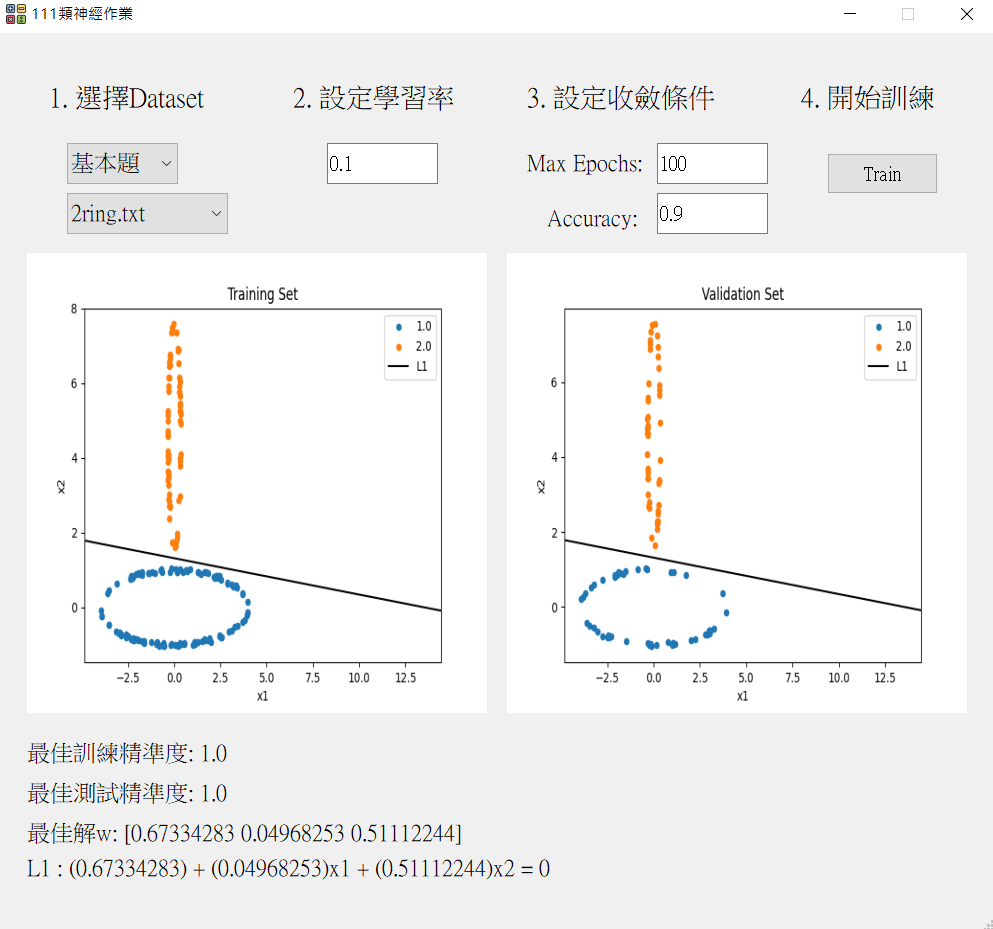
圖九、運行”2CS.txt”之結果

8.2Hcircle.txt: 輕鬆分割。



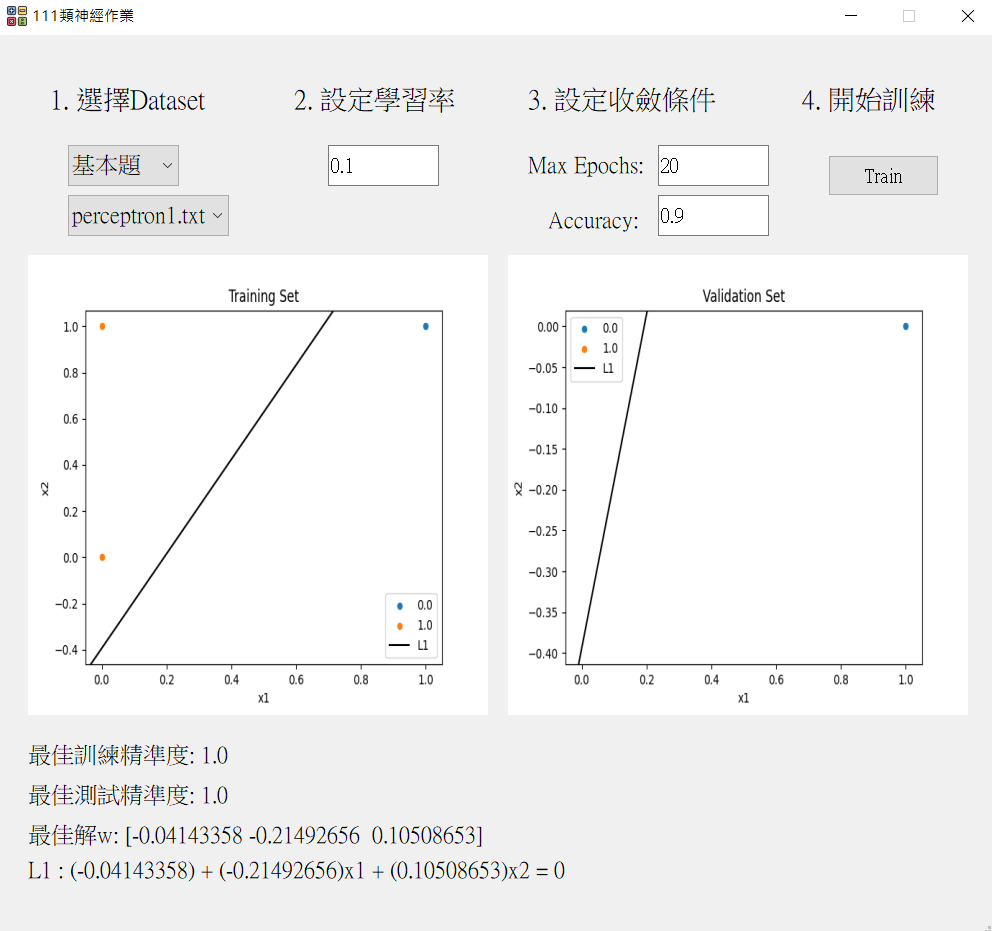
圖十、運行”2Hcircle.txt”之結果

9.2ring.txt: 輕鬆分割。



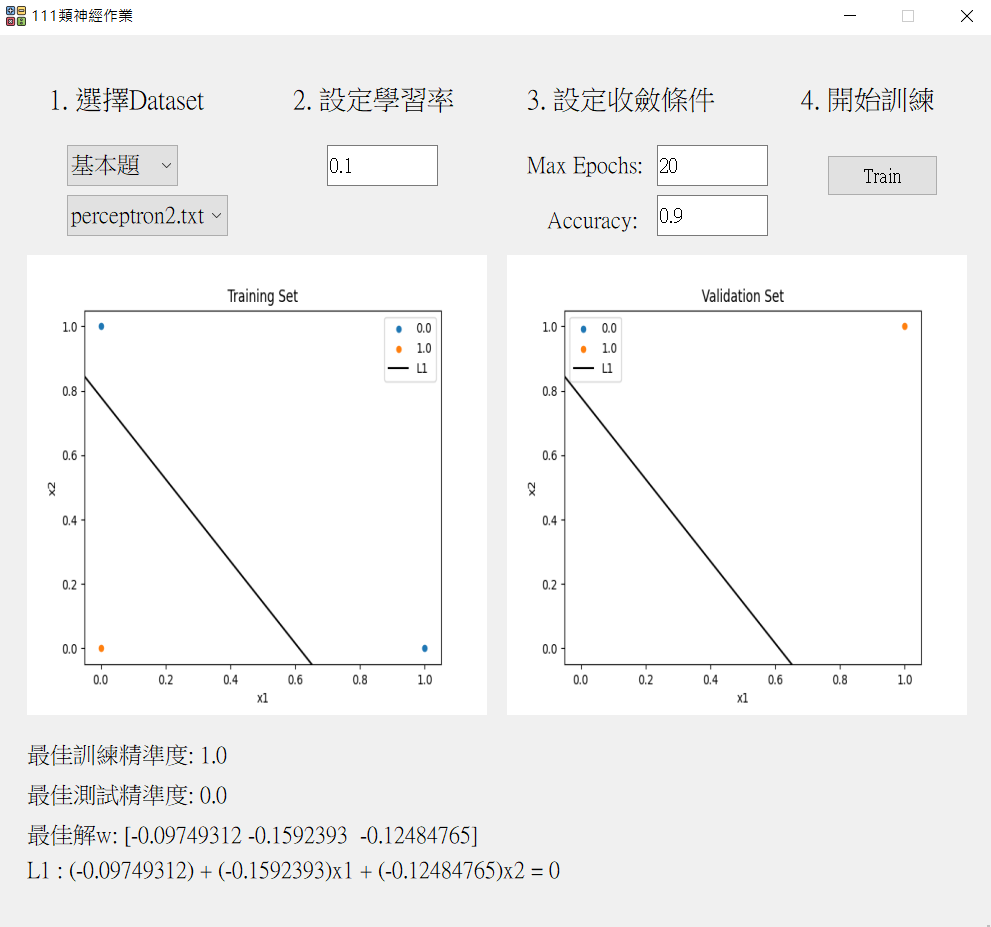
圖十一、運行”2ring.txt”之結果

10.perceptron1.txt: 輕鬆分割。



圖十二、運行”perceptron1.txt”之結果

11.perceptron2.txt: 無法分割。



圖十三、運行”perceptron2.txt”之結果

1. 實驗結果分析及討論

我發現透過增加訓練次數同時減少學習率，能讓w的變化更趨於平緩，讓w能完美分割資料點的機率增加，而增加感知機層數能讓目標函數更高維，使用激活函數能讓目標函數更有彈性，但受限於時間，目前的程式碼是單層的感知機，也沒有使用到激活函數。

由於是採用單層感知機，因此面對到perceptron2、2Ccircle1這兩題data set會無解，而2Circle1以及 2CloseS3則是因為些許資料點無法完全分割，所以精準度無法達到百分百，至於如何解決上面這四題，我認為應該使用激活函數(activate function)以及提高感知機層數，好找到更彈性更高維的目標函數。