訓練過程如下

1.對資料前處理，包括把它拆成trainset/testset

2.初始化網路，隨機產生權重

3.把資料標準化後，逐筆跑forward propagation計算出output  
4.逐筆backward propagation 計算出delta  
5.更新權重

1.處理data:

Input:檔案路徑

Output:四個set : dataset ,dataset(打亂的) ,trainset , testset

把data重新從0開始分組ex:組別1,2,4->組別0,1,2

整理完的格式如下

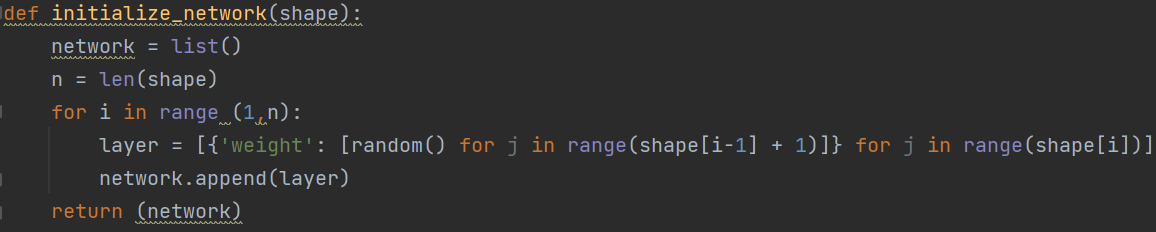
0 1 0

1 1 1

-> [[0,1,0],[1,1,1]]

Code就不貼上來了

2.初始化網路:



Input : 網路形狀

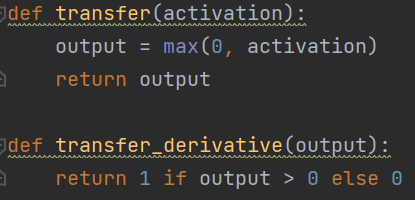
ex.[16,12]，兩層hidden layer，第一層16個神經元，第二層12個

Output:網路

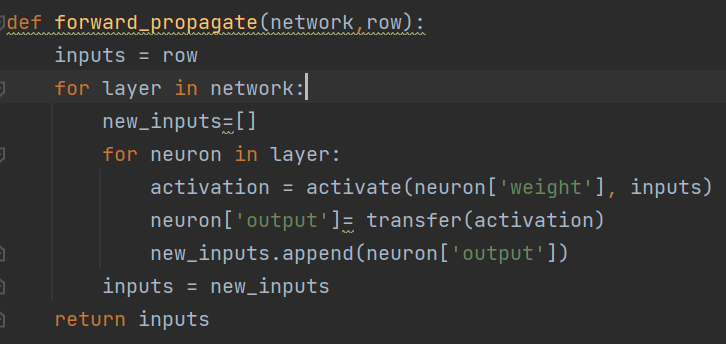
產生神經元之間的weight跟bias，初始值隨機

3.activation function and activation function微分:

使用Relu



4.Forward propagation:



Input: 網路，一筆data

把一筆資料通過網路，並且記錄每一層的output

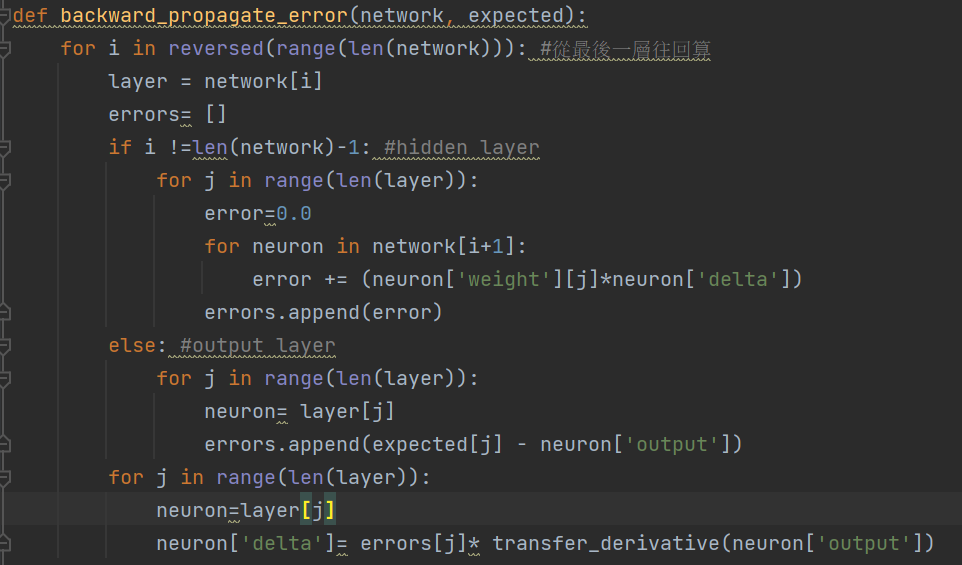
5.Backward propagation:

Input : network ,expeted(他是一個one hot vector，該比資料屬於的分類是1，其他是0)

從output layer往回計算到input layer

計算出delta值，記錄起來，等等更新權重

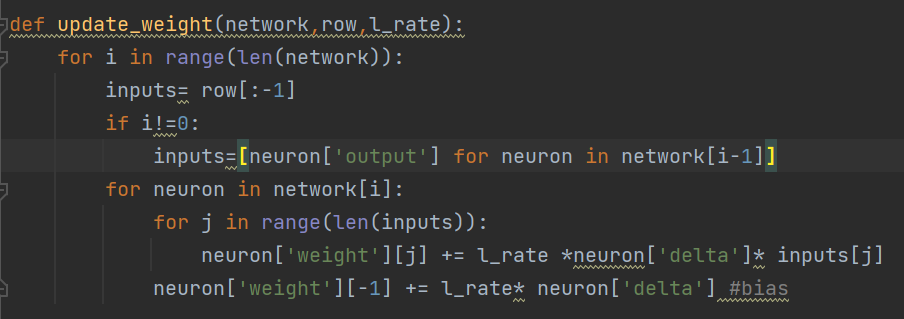
分成hidden layer 跟output layer，討論  
詳細運作流程跟老師的ptt差不多，



6.Update weight:

Input: network, 一筆資料,學習率

用delta值、weight、學習率更新權重

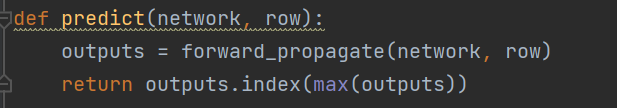


7.Predict:

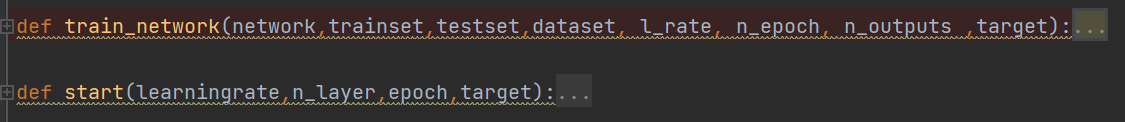
Input:資料，網路

Output:預測的結果

用forward propagation預測，不會進行權重更新:



8.開始訓練



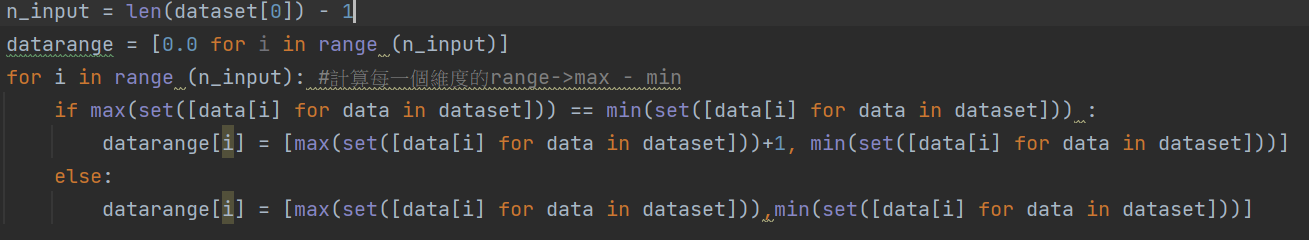
按開始訓練後會input使用者輸入的參數到start

Start會call train\_network

train\_network會再把資料正規化後開始以上步驟，根據使用者輸入的參數開始訓練並印出結果，start會輸出最後的weight

這兩個func主要就是call上面的func，code有點長就不貼上來了

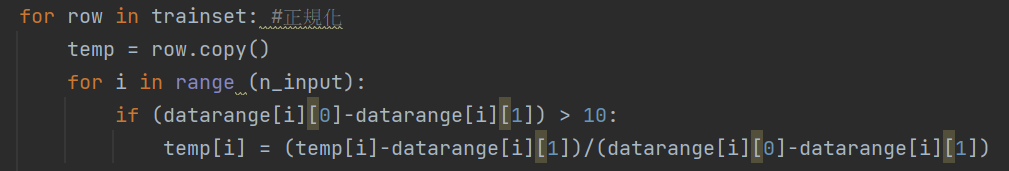
正規化



計算出每一個維度的range = min – max

把資料正規化的(0,1)

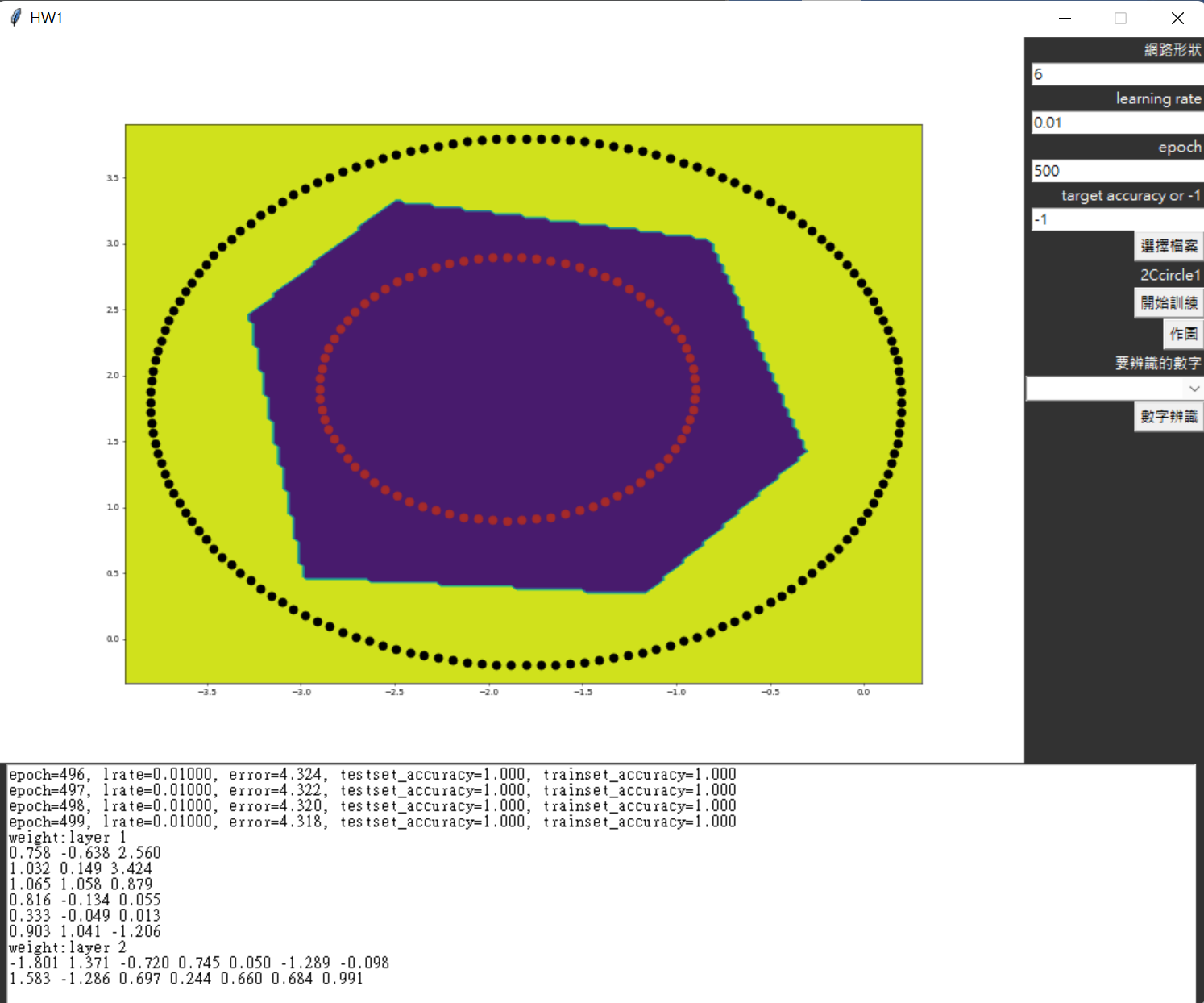
如果資料只有0,1或者min – max < 10就不做正規化了



Temp儲存正規化後的數據

code再往下的部分就是gui的部分，用tkinter模組製作的，不特別說明

gui操作:



我是用色塊交界來表示decision boundary，生成一堆點(網狀)去跑predict，所以三維的作圖並沒有色塊顯示decision boundary，不然作圖的時間會太久

tkinter

網路形狀輸入範例:

6 -> 一層hidden layer 六個神經元  
6,8 -> 兩層hidden layer 第一層六個 第二層八個神經元

Learning rate:

別設太大，可參考我測試結果中的數據

Epoch: 跑幾輪

Target : 達到多少正確率就停止執行，可輸入0~1，沒有target填-1

要辨識的數字、數字辨識是給數字辨識用的

請訓練完再按作圖

訓練結果的word檔有我用的參數

作業提供的全部dataset皆可以訓練

可畫出三維的點，可顯示數字辨識，可處理多維資料，可識別多群資料