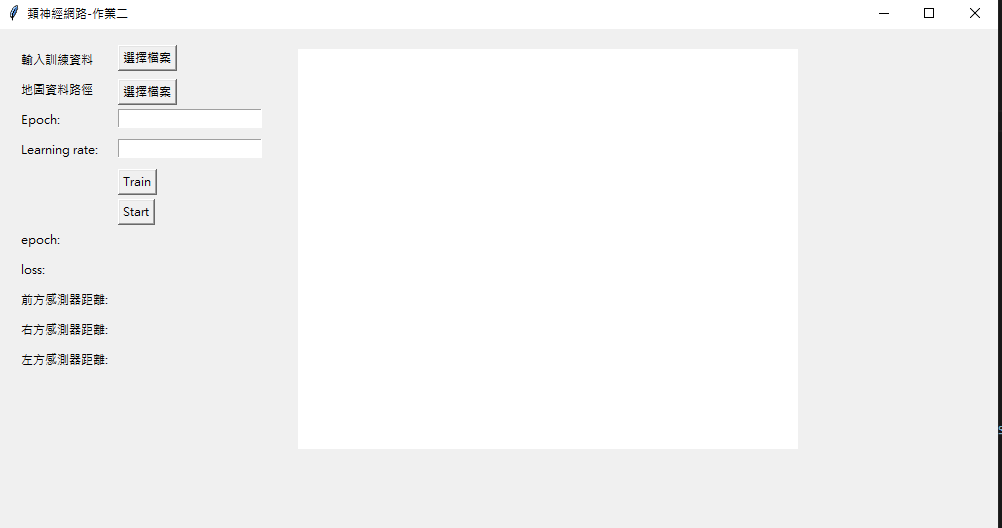
**類神經網路作業二報告**

1. 程式介面與操作說明

點開/dist/main/main.exe可直接使用程式，詳細操作可以看111522101\_HW2.mkv，需要把軌跡圖清空在選一次地圖資料路徑就好。

Track檔會存在/dist/main/底下



2.選擇軌跡資料=>軌道座標點.txt

1.選擇訓練資料=>train4dAll.txt、train6dAll.txt

3.輸入epoch

4.輸入LR

顯示車道與移動軌跡

5.train

7.開始模擬

6.每10個顯示

Epoch與loss

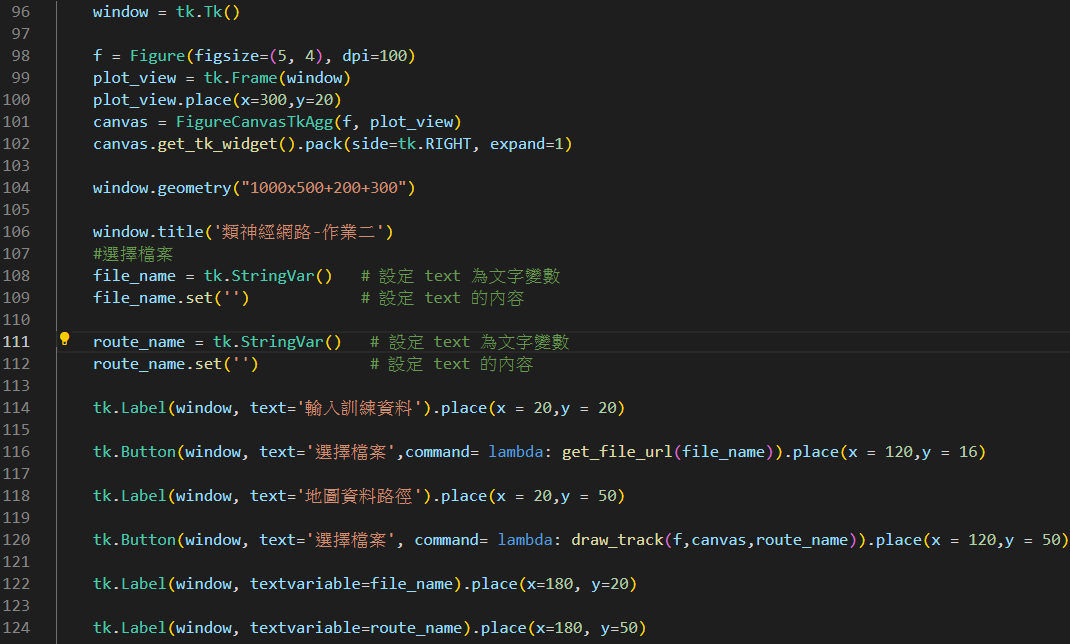
8.顯示每一時刻感測器偵測距離

1. 程式碼說明

本次作業使用四個class和一個main當作主要架構，class部份分別是Dataprocessor 、Kmeans、model、simulator，以下會分別進行講解

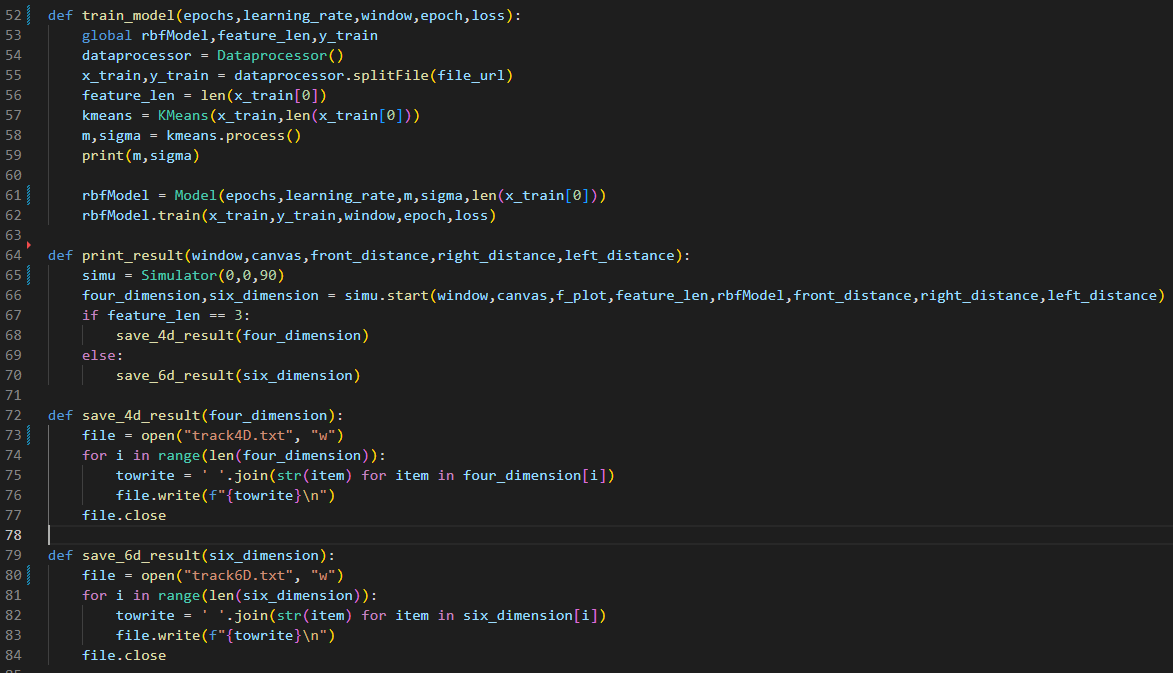
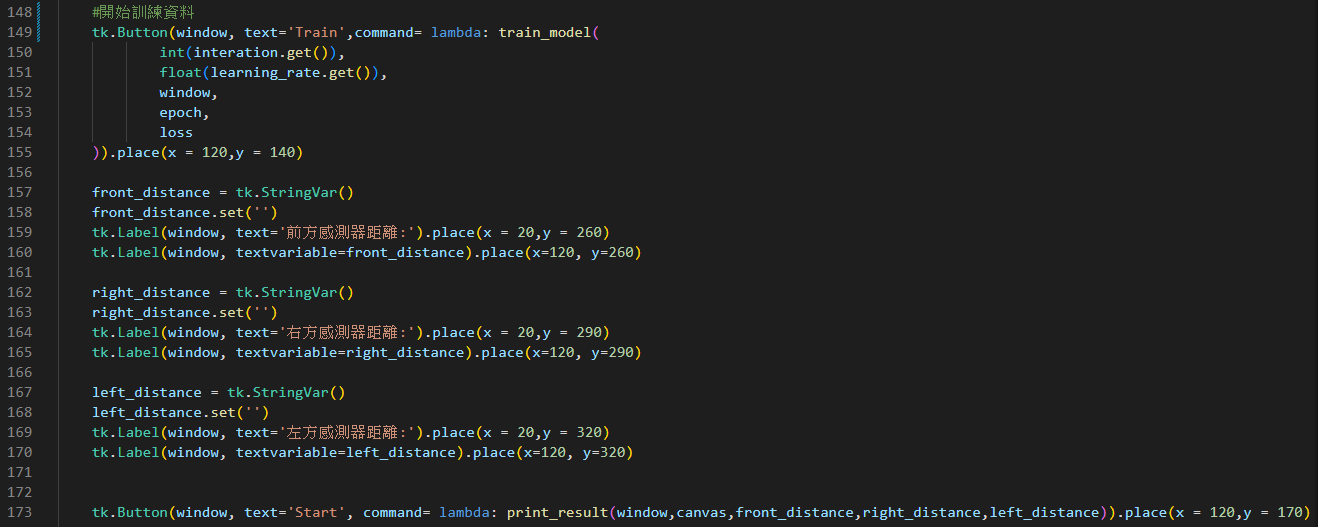
* Main.py

main.py主要是負責介面的設計與class的調動，GUI主要是使用tkinter進行撰寫，在button的部分分別會調用get\_file\_url() 取得訓練檔案資料與draw\_track繪製路徑圖



Train\_model()會取得學習率與epoch次數進行model的訓練

print\_result()會使用調用模擬程式開始進行感測器的計算與路徑的繪製



使用Dataprocessor呼叫將資料分成input與label

使用Kmeans挑出rbfn的初始值m跟sigma

將資料丟進RBFN model進行訓練

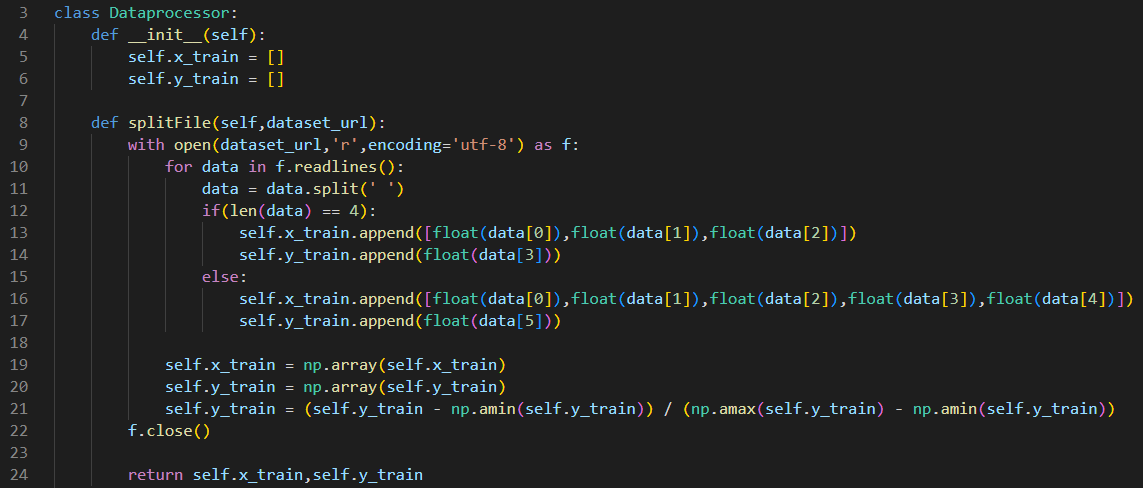
將訓練好的model與GUI所需設定的物件放入

simulator，並取得路徑與方向盤資訊

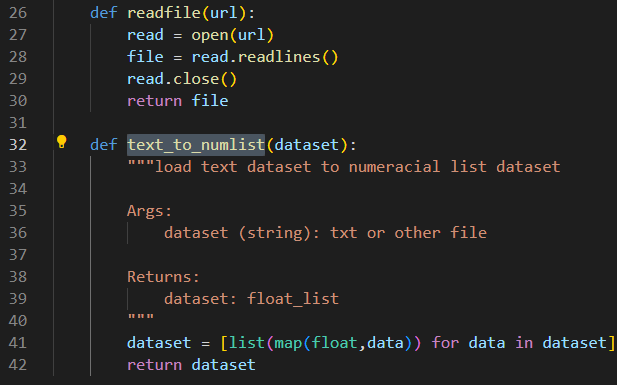
看維度繪製檔案

* Dataprocessor

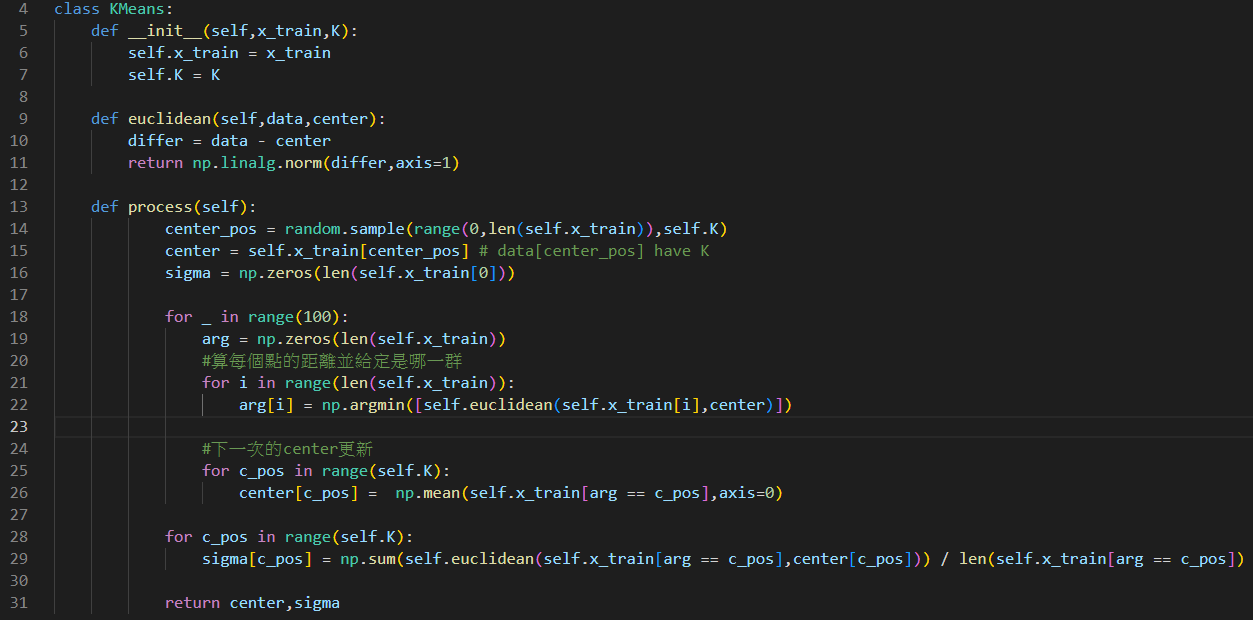
將資料分成input跟label，會根據輸入的維度不同做不同處理，並且會將label使用min max scaler正規化，以便後續model訓練。



Min max scaler

讀檔與將資料從string轉成float

* Kmeans



隨機挑三個點當中心點，並且初始化

Center與sigma

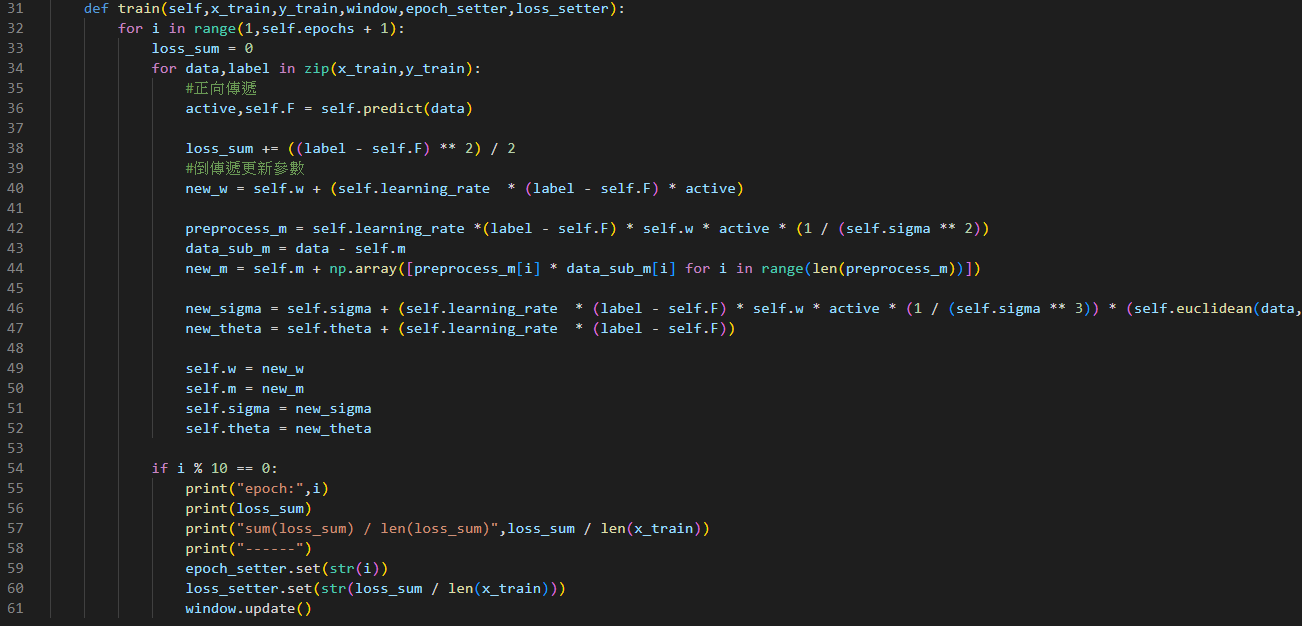
中心點更新，一開始將資料屬於哪

個群先記錄在arg，再把根據arg把

群體內的點做mean

算sigma

* Model

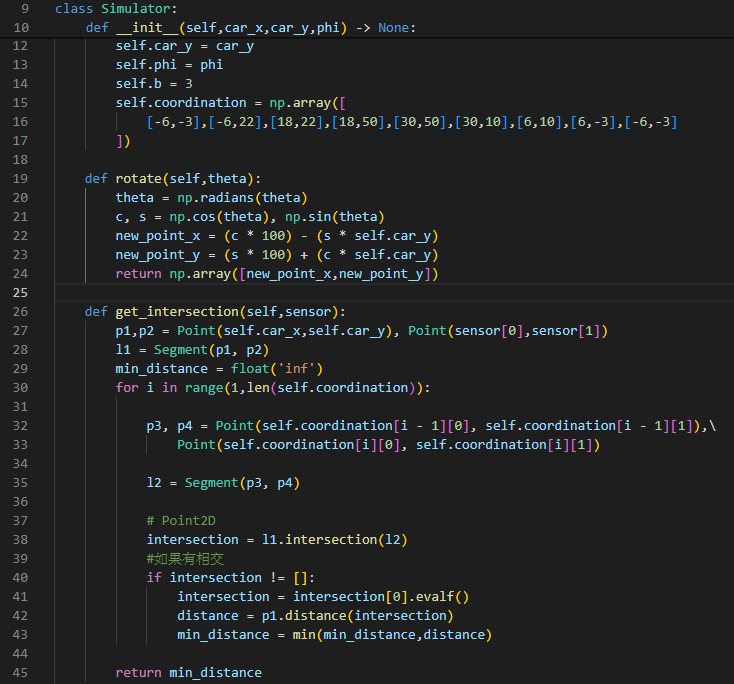


Back probagation

Forward

每10次epoch會印一次loss在GUI上

* Simulator



使用旋轉矩陣，將點

定在車的右邊100，然後

用phi將車子面相對的位置

取得車與自定義的右邊100遠的線使用旋轉矩陣轉成感測器的方向形成的線段與車道邊界相交的點與車位置的距離，挑最短的回傳



取得各個感測器抓到與車道的最短距離

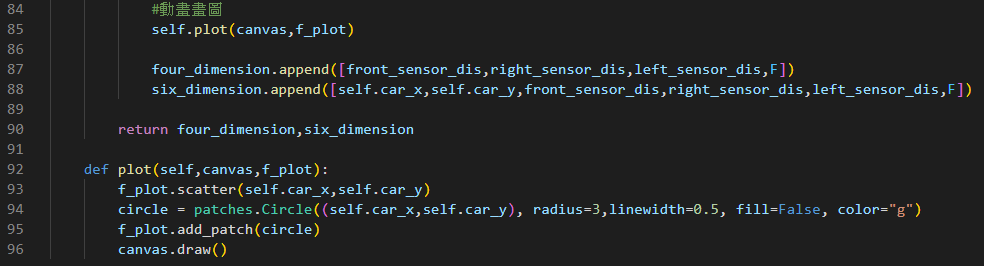
將感測器抓到的資料丟到model進行預測，並將得到的結果正規化回來

更新車的位置與phi

基本的碰撞偵測

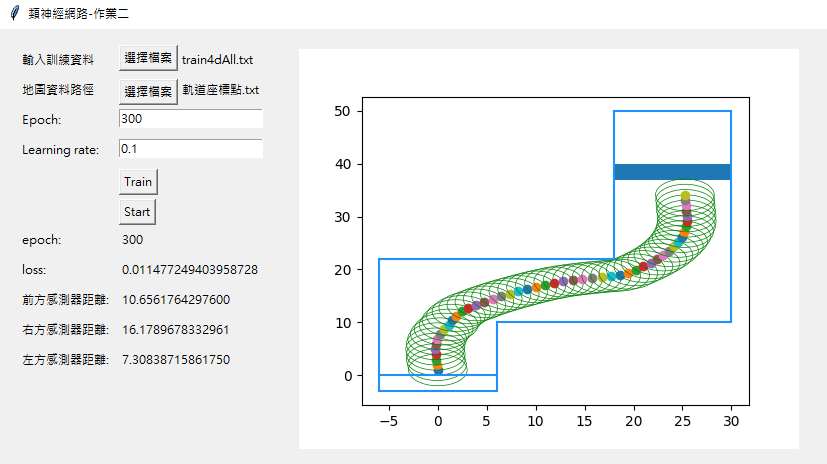


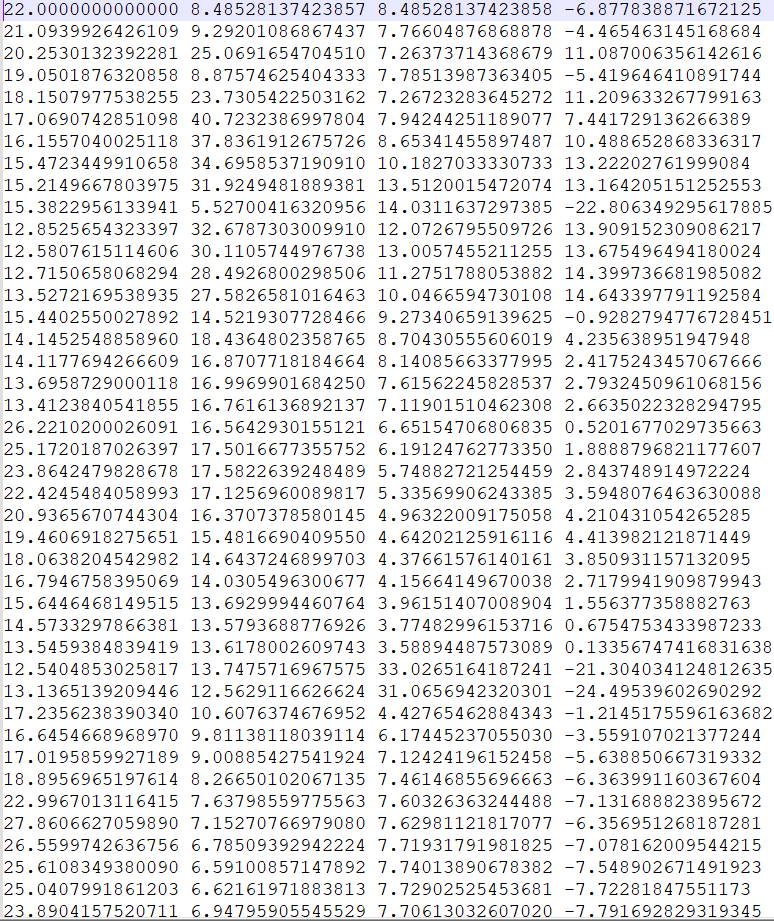
將車目前的位置進行畫圖，並將半徑畫出來



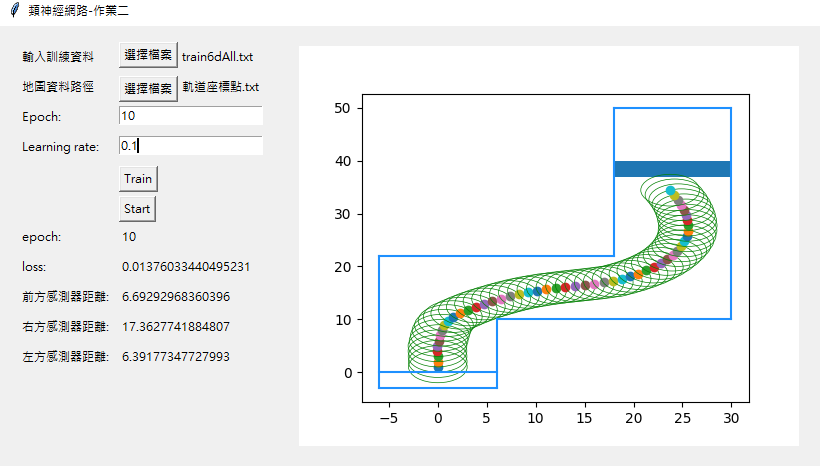
1. 實驗結果(包含移動軌跡截圖)

Train4dAll.txt Track4D.txt

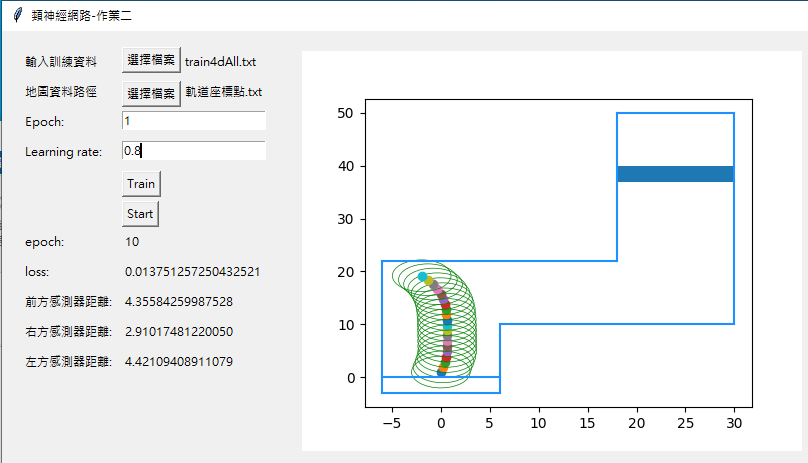


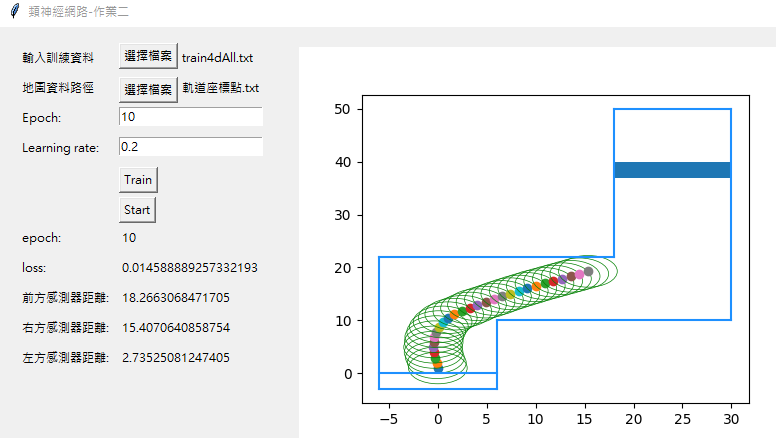


Traind6All.txt Track6D.txt



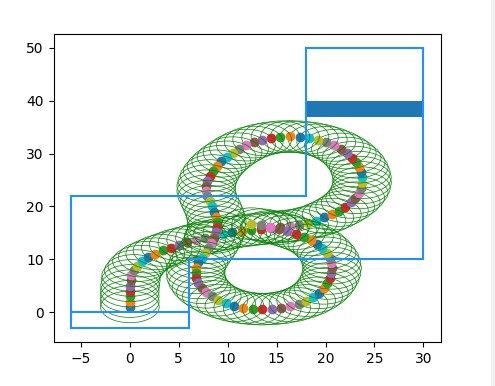
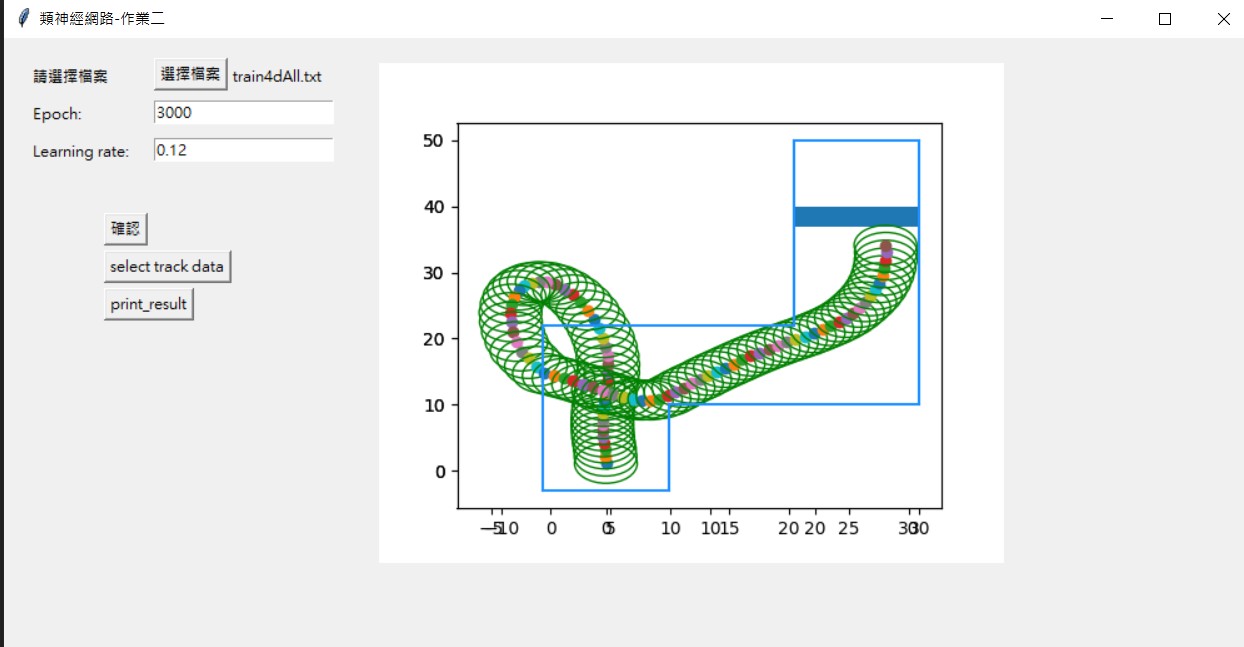


碰撞偵測



1. 分析

在4D的資料下，目前試下來Epoch數量需要比6D的多不少，在4D的情況下epoch拉到100 – 200而learning rate為0.1的情況下都還會撞牆，測試下來epoch = 300 learning rate = 0.1效果是最好的，但是epoch拉到太高，則會overfitting，在還沒有碰撞偵測的時候，發生過以下例子，大約是epoch 1000 以上就會發生



而在6D的情況下，因為資料量比較大的關係，只需要不多epoch就能不撞牆的到達終點，在epoch = 10，learning rate = 0.1的情況下就可以訓練完畢，不過由於Kmeans的關係，如果中心點取的不好，那10個epoch有時候還是發現撞牆的現象，而epoch拉到100左右，就大致上都能夠不出錯。

1. 有做的加分
2. RBFN
3. 模擬程式