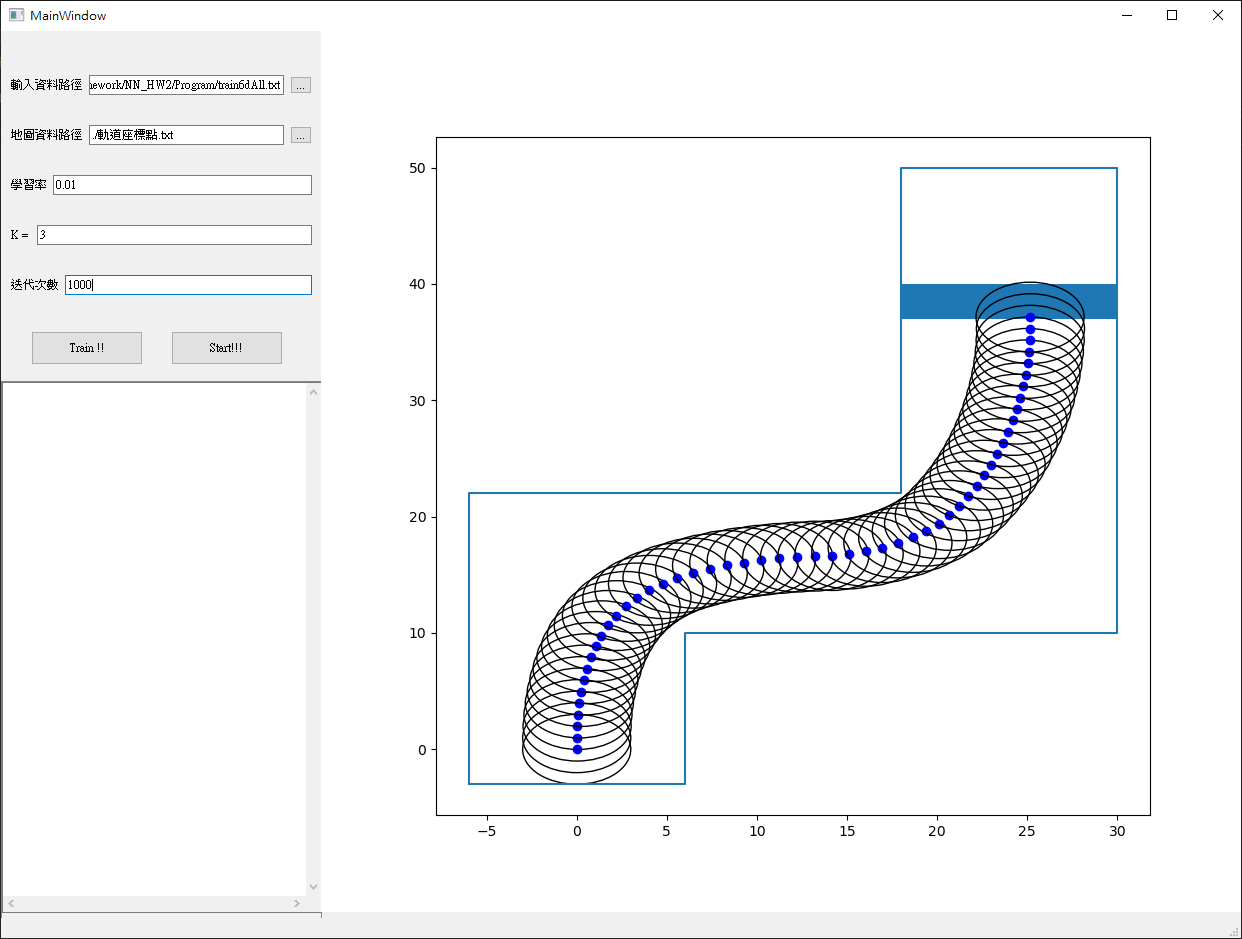
作業二：自駕車

111522094 資工碩一 凃建名

1. 程式介面說明:



輸入路徑資料: 可以輸入要進入RBFN訓練的資料

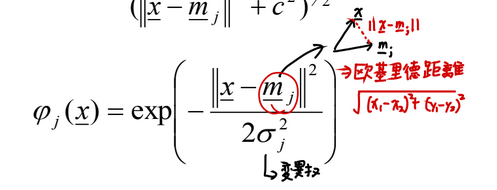
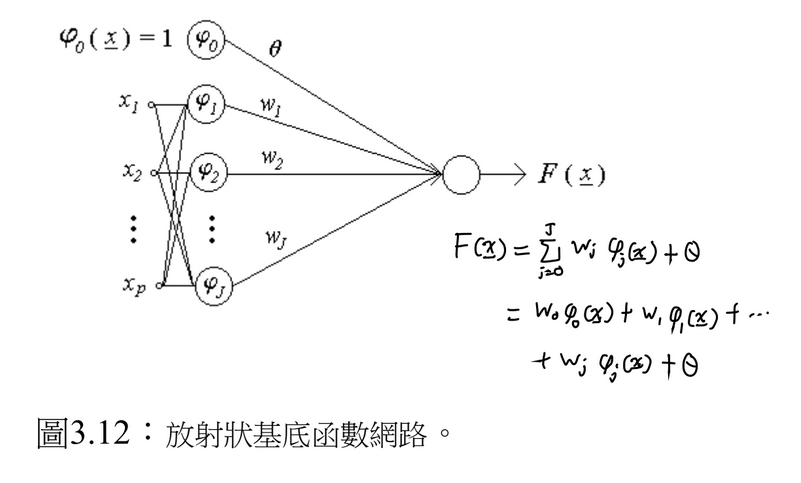
地圖資料路徑: 可以選擇地圖的資料路徑

學習率: 設定RBFN的learning rate

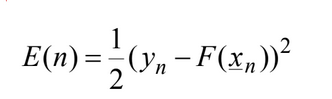
K: 設定將RBFN中的m初始化的K-means演算法的群聚數量

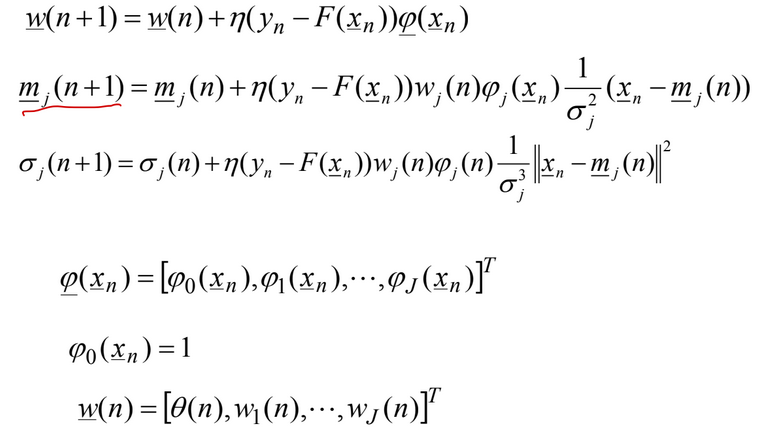
迭代次數: RBFN需要跑幾個epoch

1. 程式碼簡介:
   1. rbfn.py:



實作RBFN的py檔，我先用K-means演算法的群聚中心和群聚與群聚中心的平均距離來初始化m和σ，之後計算結合m、σ、X 三個矩陣來計算出φ矩陣，之後計算F =φ‧W + θ完成前傳導的過程。

利用來當作loss值並實作



來完成倒傳遞的調整。

* 1. drawplot.py

畫出GUI、模擬車輛移動的程式，

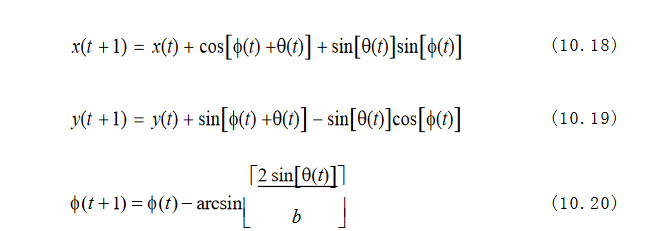
Sensor的作法:

事先設定一個固定的vector = [100, 0]

在利用旋轉矩陣，計算長度為100、角度為Φ(t)的rotated\_vector

之後計算所有邊界與a[rotated\_vector] + [車輛位置]的所有交點，選出與車輛位置最近的交點的歐基里德距離作為該方向sensor偵測出來的距離

車輛的移動方程式:



其中 Φ(t) 是模型車與水平軸的角度， b 是模型車的長度，x 與 y 是模型車的座標位置， θ(t) 是模型車方向盤所打的角度

碰撞偵測:

由三個Sensor回傳回來的距離判斷是否小於車身的半徑3

* 1. main.py

UI的主程式

* 1. toolkit.py

放有歐基里德距離計算、旋轉矩陣計算、向量和直線交點計算的程式

1. 實驗結果
   1. 實驗一:

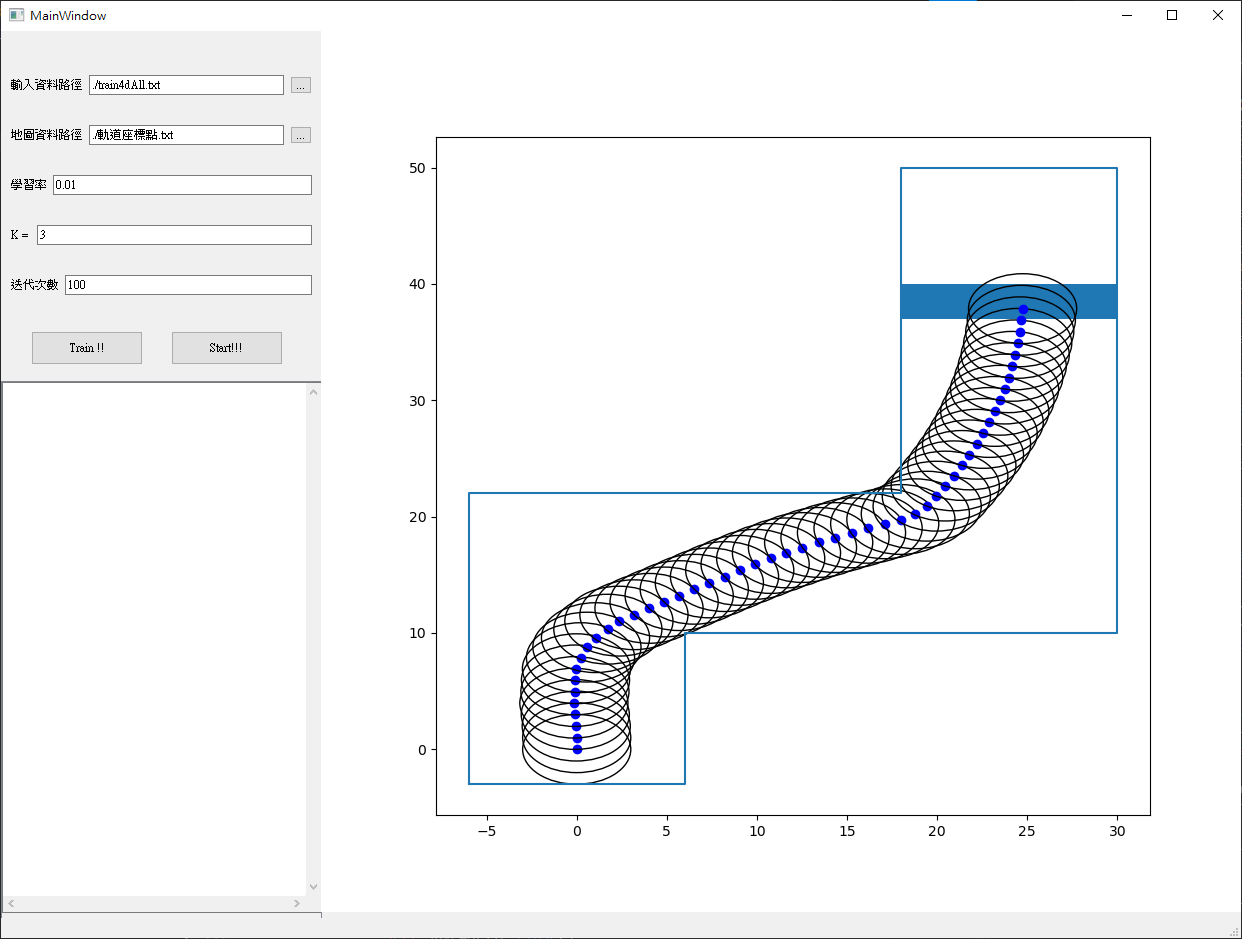
使用資料集: train4dAll.txt

Learning rate: 0.01

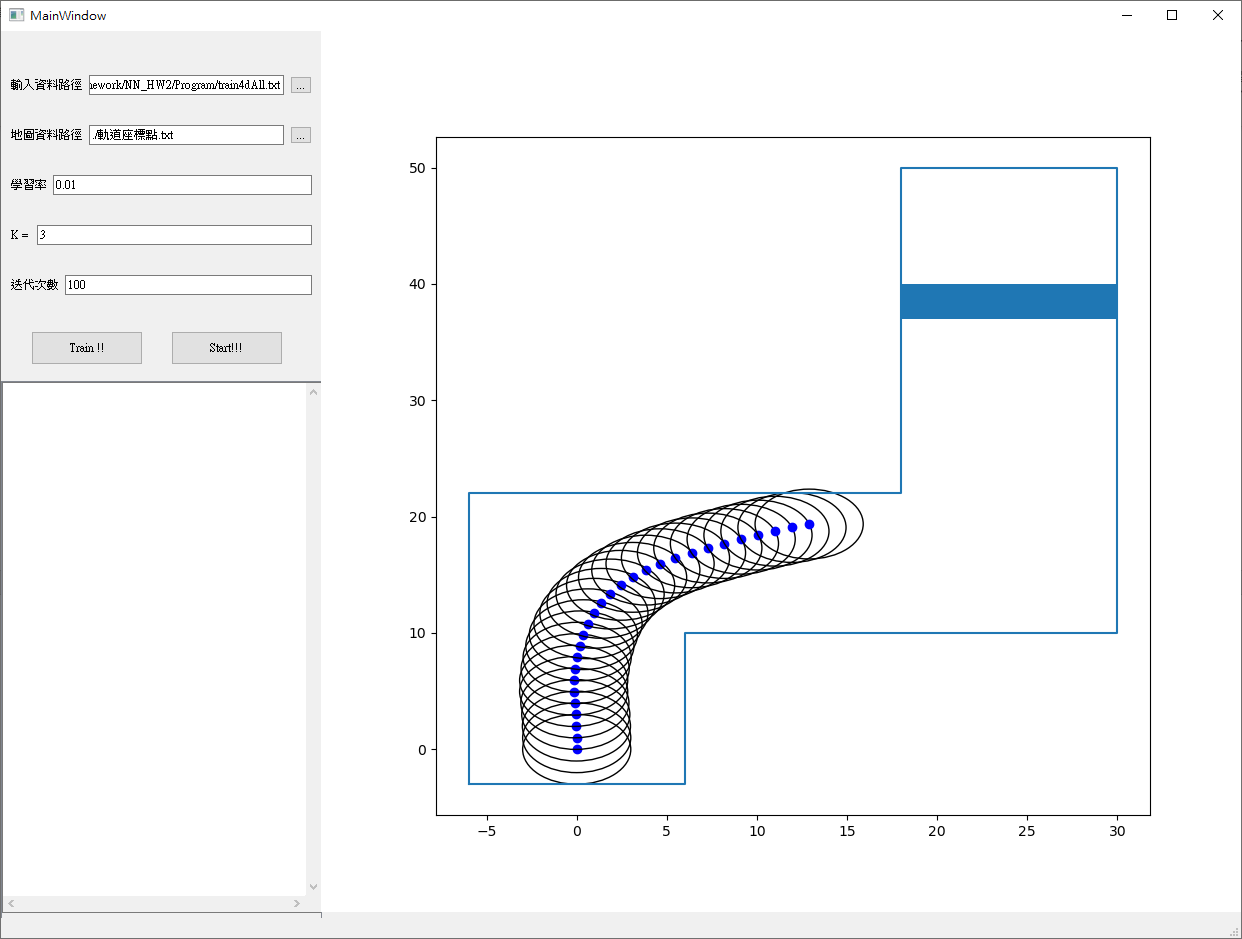
K = 3

迭代次數 = 100

優點:訓練速度快



缺點: 自走車碰壁機率較高(大概:19/20)



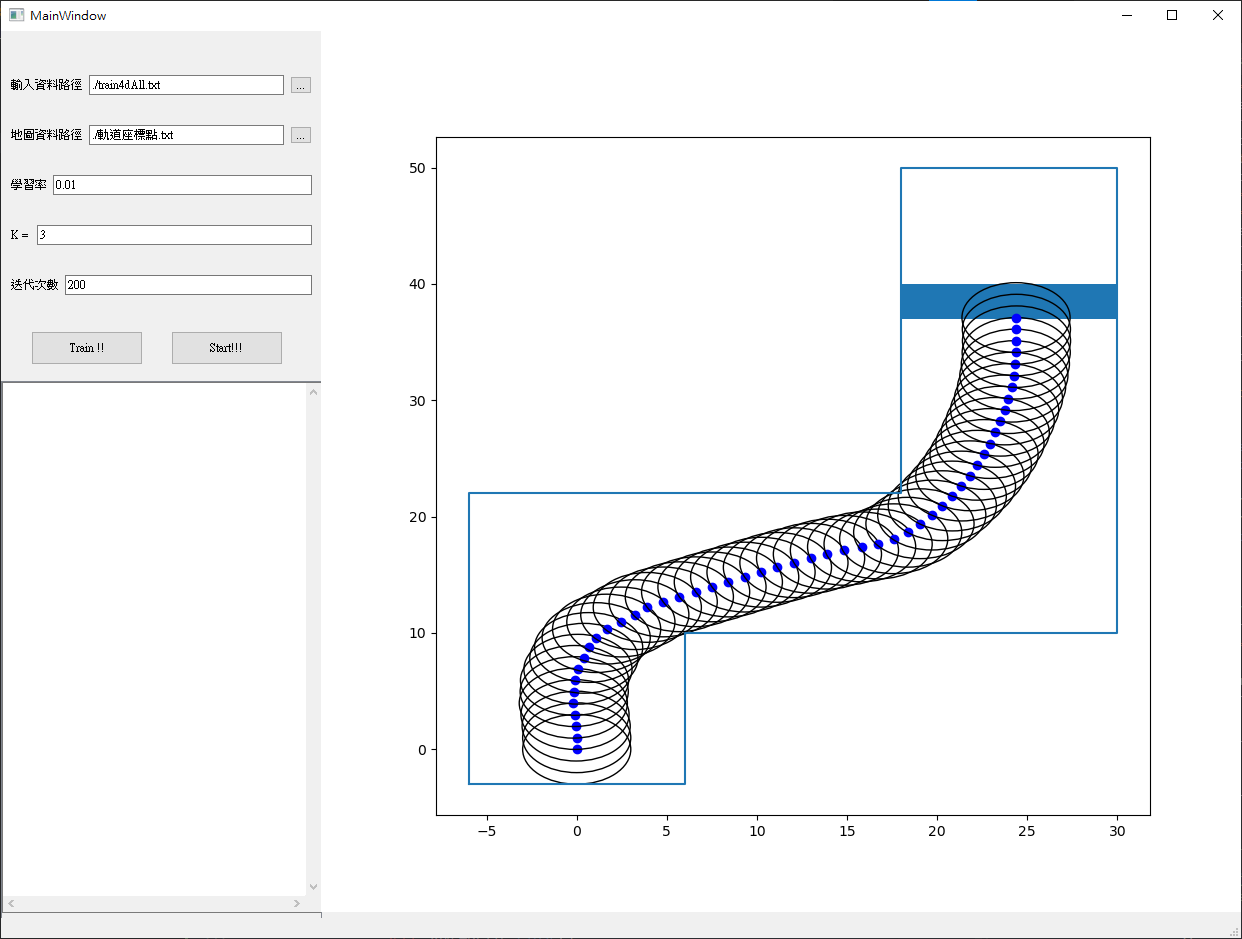
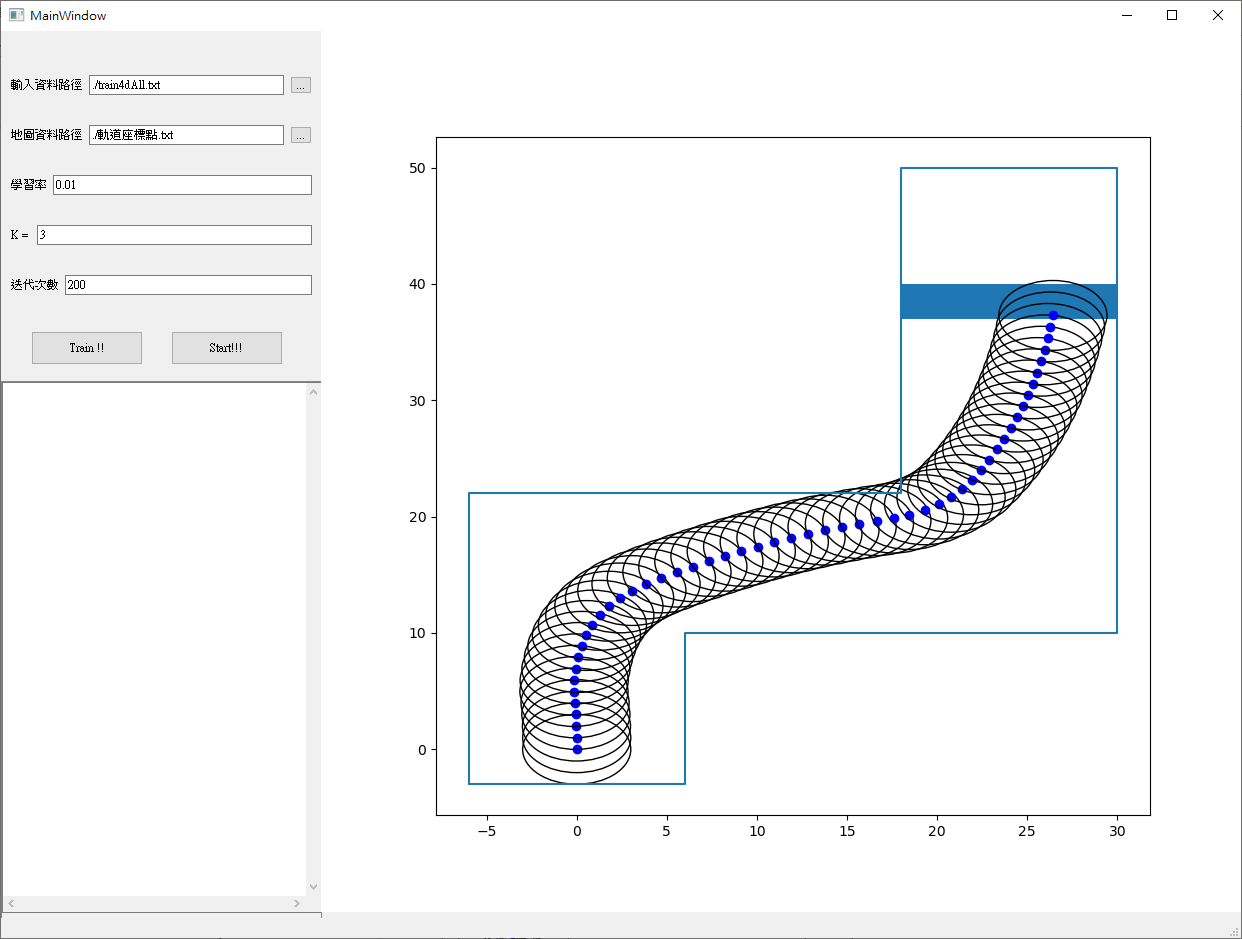
* 1. 實驗二:

使用資料集: train4dAll.txt

Learning rate: 0.01

K = 3

迭代次數 = 200



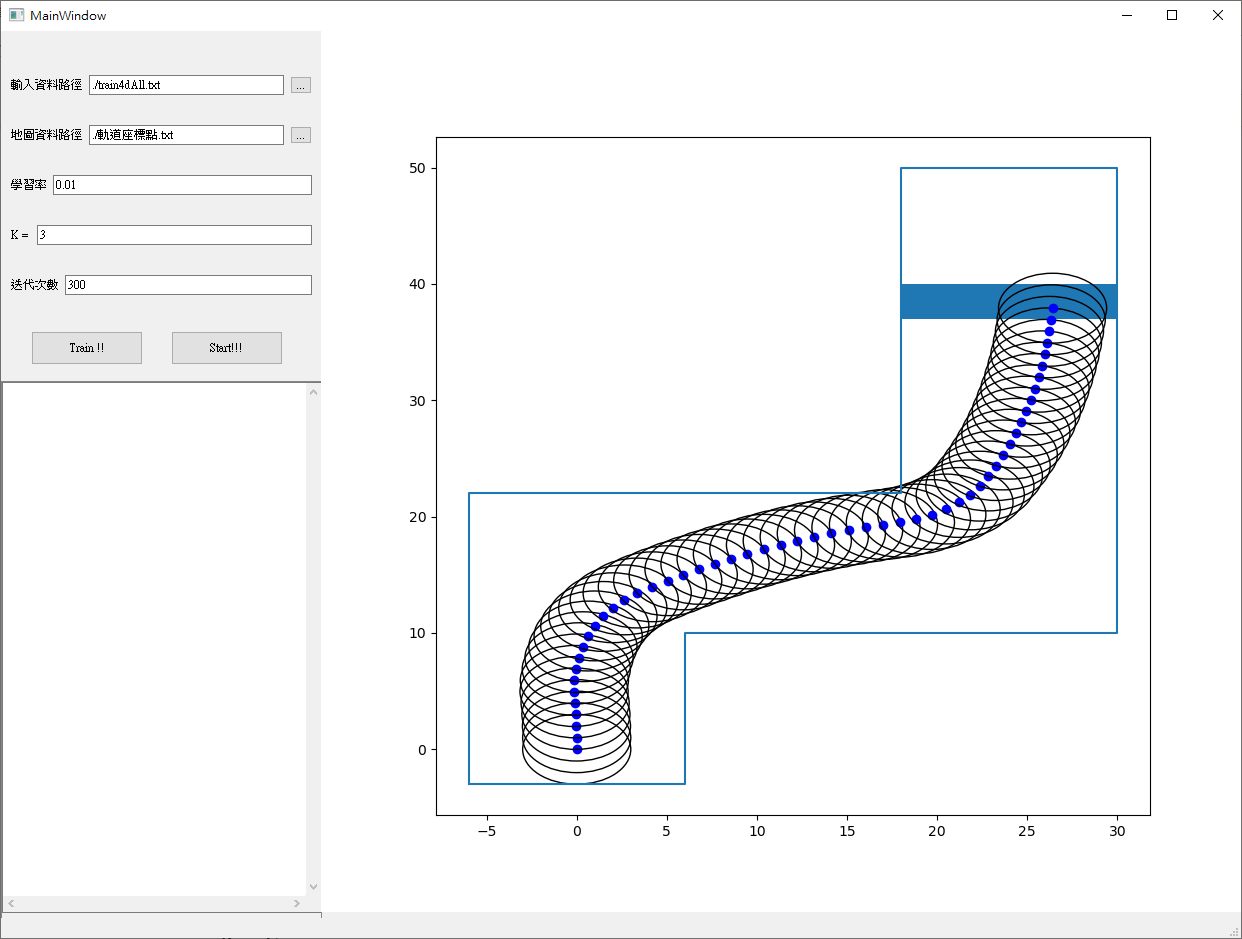
* 1. 實驗三

使用資料集: train4dAll.txt

Learning rate: 0.01

K = 3

迭代次數 = 300



優點: 成功機率高且軌跡較實驗一漂亮

缺點: 需要時間較實驗一長

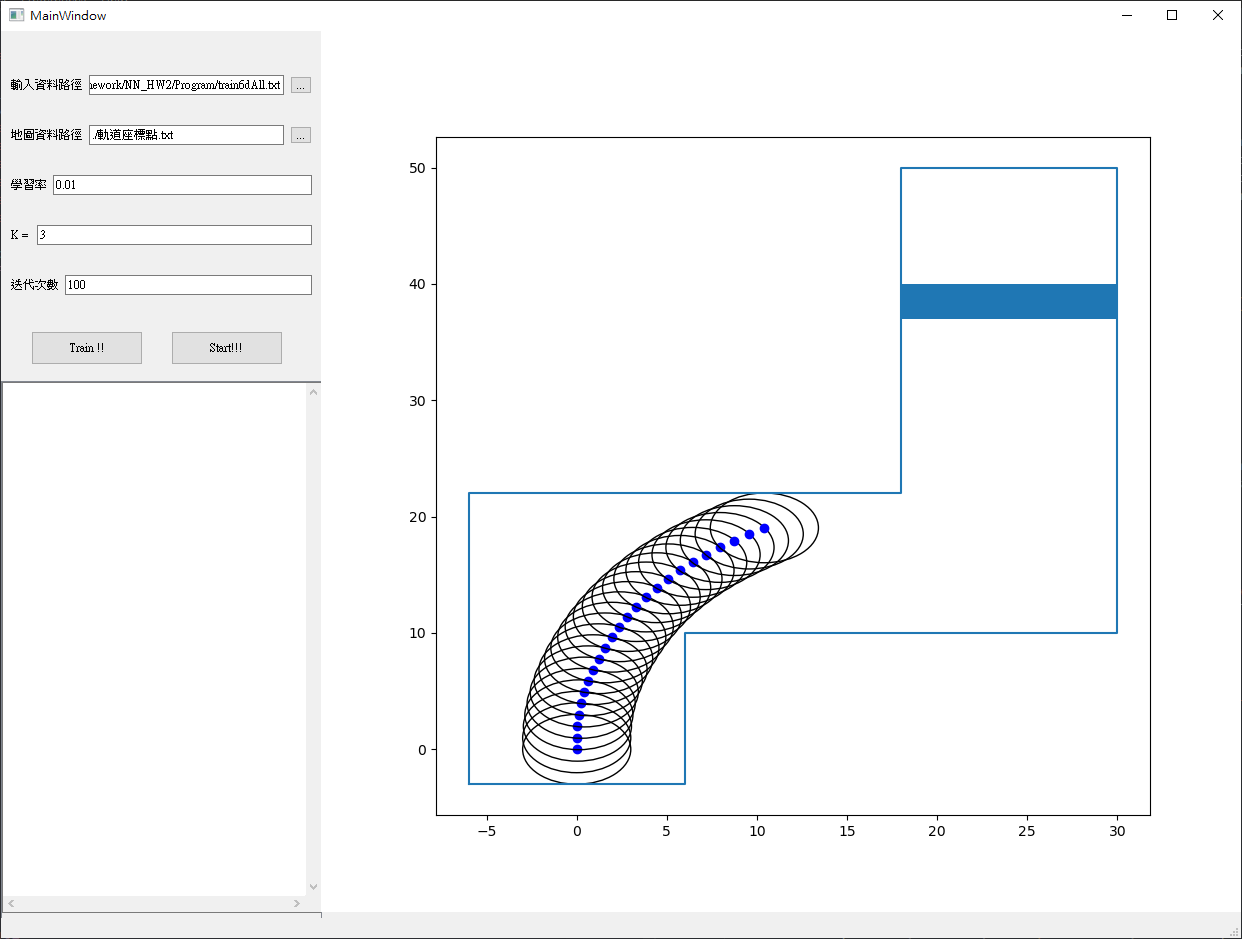
* 1. 實驗四:

使用資料集: train6dAll.txt

Learning rate: 0.01

K = 3

迭代次數 = 100



結果: 失敗機率高

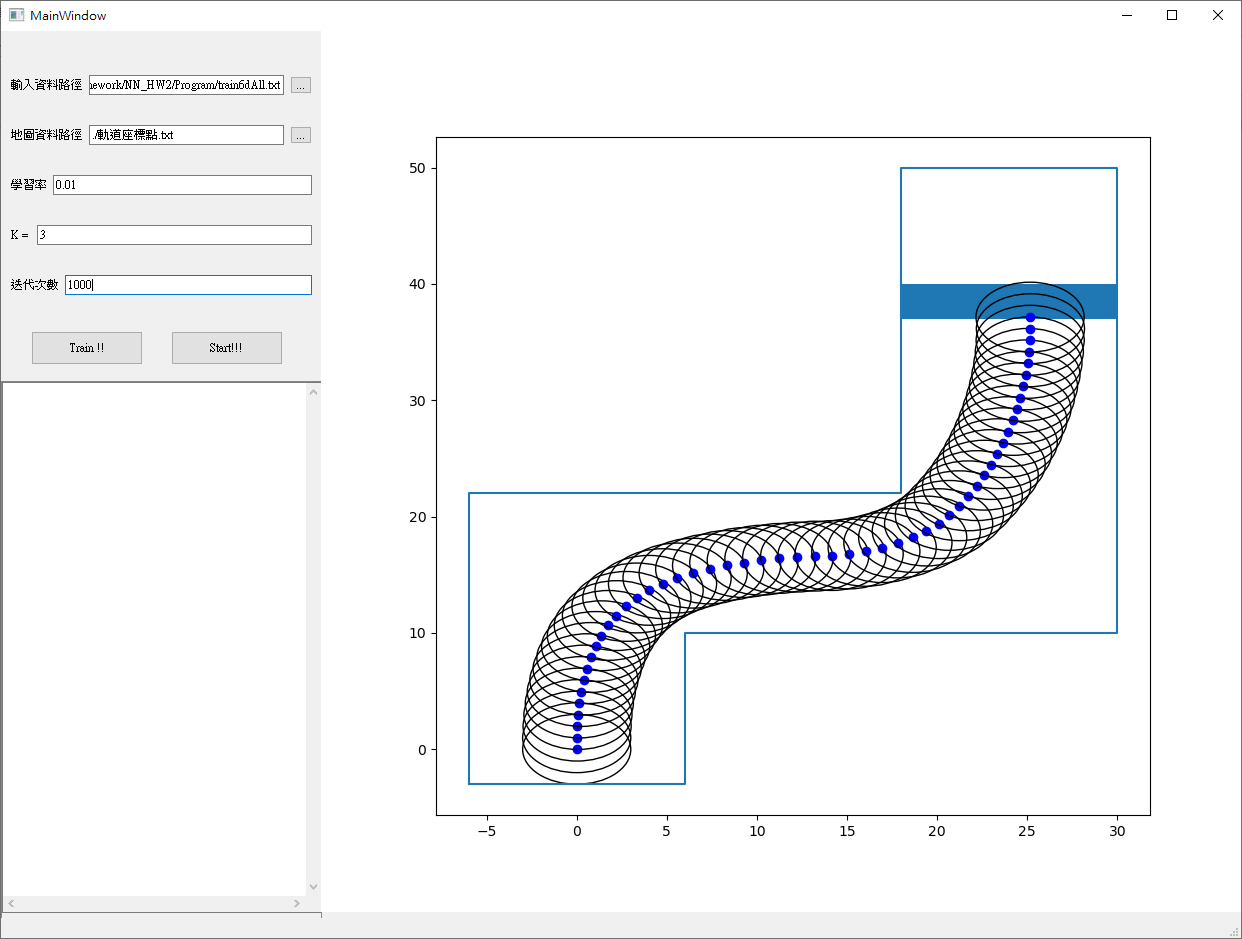
* 1. 實驗五:

使用資料集: train6dAll.txt

Learning rate: 0.01

K = 3

迭代次數 = 1000



優點: 得出的結果最好、連牆角都不會碰撞到

缺點: 需要時間過長(大概1分鐘至3分鐘之間)

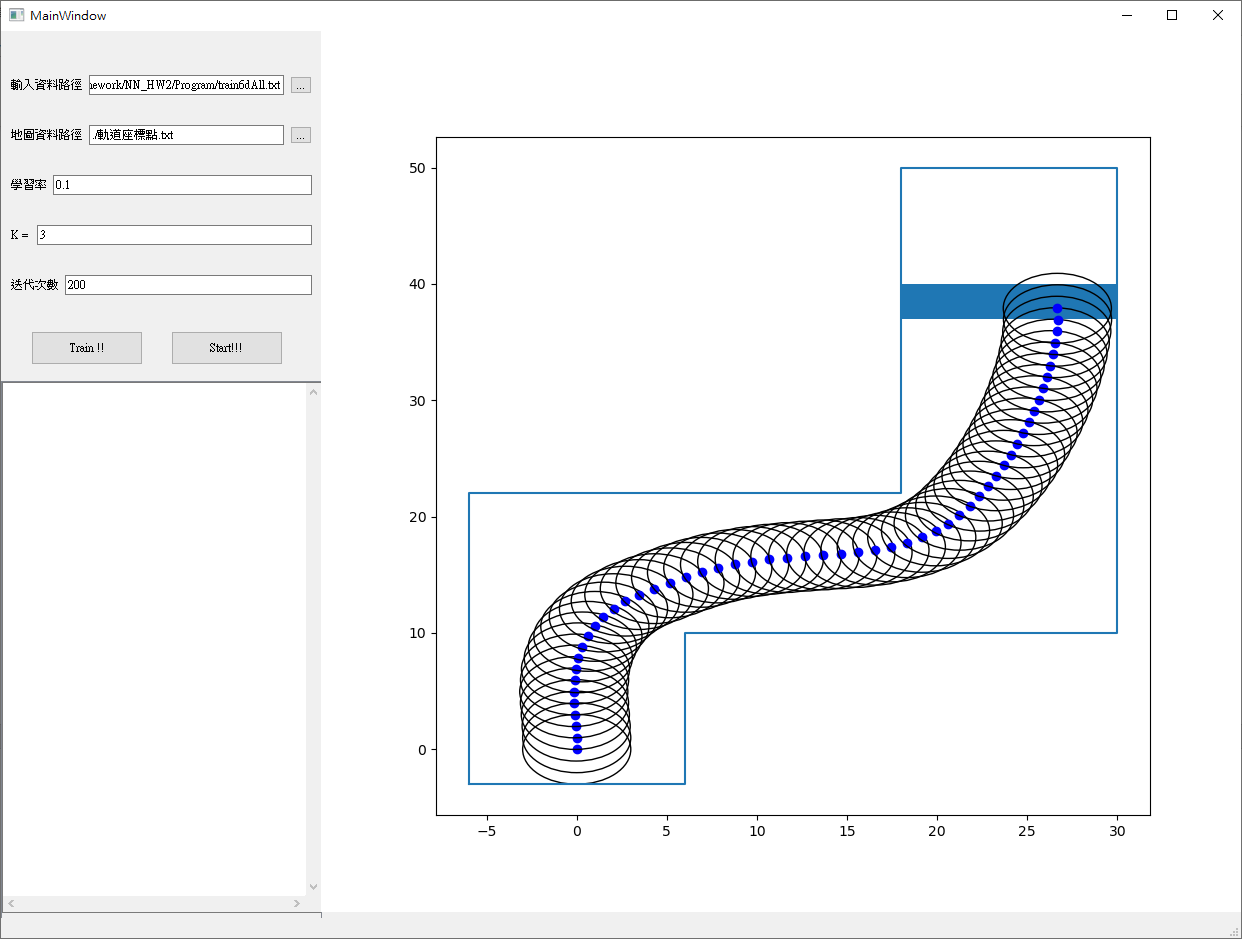
* 1. 實驗六:

使用資料集: train6dAll.txt

Learning rate: 0.1

K = 3

迭代次數 = 200



結果: 所需時間短、軌跡完美、沒有撞到任何牆

分析:

從實驗來看RBFN 的 loss會隨著learning rate的降低而下降，目前找到適合的learning rate 為 0.1 or 0.01，對比於MLP RBFN的速度較快，並且也能跑出非線性的預測。

如果單論這次自走車最好的參數為

使用資料集: train6dAll.txt

Learning rate: 0.1

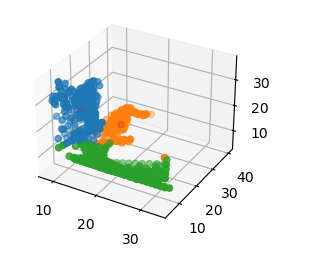
K = 3

迭代次數 = 200

train6dAll.txt 由於比 train4dAll.txt 多了x, y 的位置資料所以預測出的軌跡相較於train4dAll.txt也會比較漂亮

而learning rate 目前觀察0.1會下降快速，但是如果調到0.3的時候loss就會暴增，調到0.01時則下降速度較為緩慢

K = 3 從資料的分布來看分三群顯然是比較合理的選擇



迭代次數的話目前觀察則是越大越好，但是相對付出的時間成本就會更高。

如果使用的訓練資料集為train4dAll.txt

目前檢測最好、最有效率的結果為

Learning rate = 0.01

K=3

Epochs = 200~300