計算型智慧作業一書面報告

Q-Learning

資電院不分系三 葉展維 110504514

壹、 程式介面說明

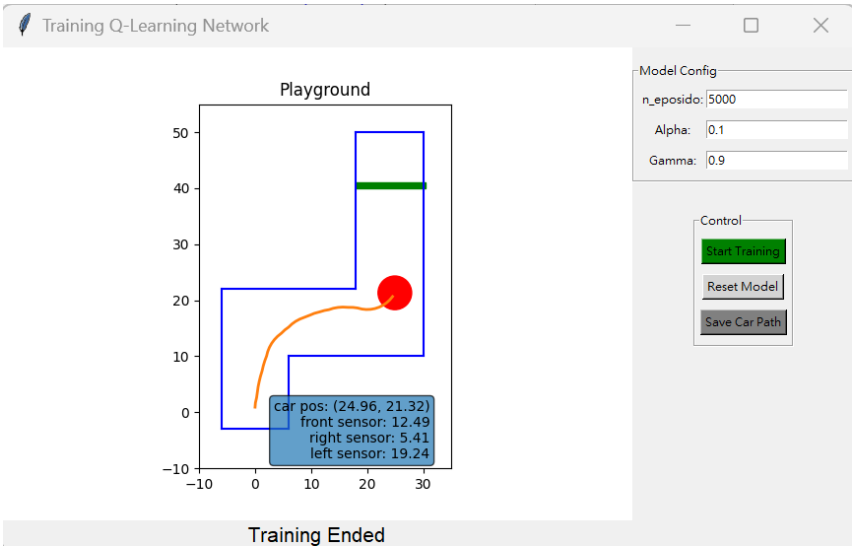

圖 1:系統 UI 介面

一、 介面主要分成四個部分

1 **模擬車輛顯示**: 顯示車輛的行走路徑以動畫的方式呈現，並顯示個步驟車輛的位置與前方、右方及左方的感測距離。

2 **Q-Learning 參數設定**：使用者可以自行設定迭帶次數(n_episode)、Alpha、Gramma 值。

3 **操作設定**：三個按鈕分別為訓練按鈕（Start Training)、重設 Q-Learning 模型（Reset Model)、儲存車輛移動路徑檔（Save Car Path)。按下訓練鈕後會根據前面訓練過的 Q-Learning 模型再重新跑過 n_episodes 次或直到模型成功達到終點。

4 **系統提示**：位於介面的最下方，會根據目前訓練狀態做提示（訓練中、訓練結束、錯誤訊息)。

二、 操作流程

1. 設定迭帶次數(n_episode)、Alpha、Gramma 值。

2. 按下訓練按鈕開始訓練。

3. 訓練結束後，左側會顯示車輛移動路徑的動畫。

4. 按下儲存車輛路徑可以儲存車輛行走路徑至 car_path.txt 。

貳、 程式碼介紹

一、 使用套件

1 **NumPy:** 負責 Q-Learning 中的運算

2 **Tkinter:** 負責 UI 介面顯示。

3 **Pyinstaller:** 負責將 Python 檔案包裝成可執行檔。

二、 專案結構

專案由兩個資料夾及一個主程式（如圖 2）。以下將分別介紹。

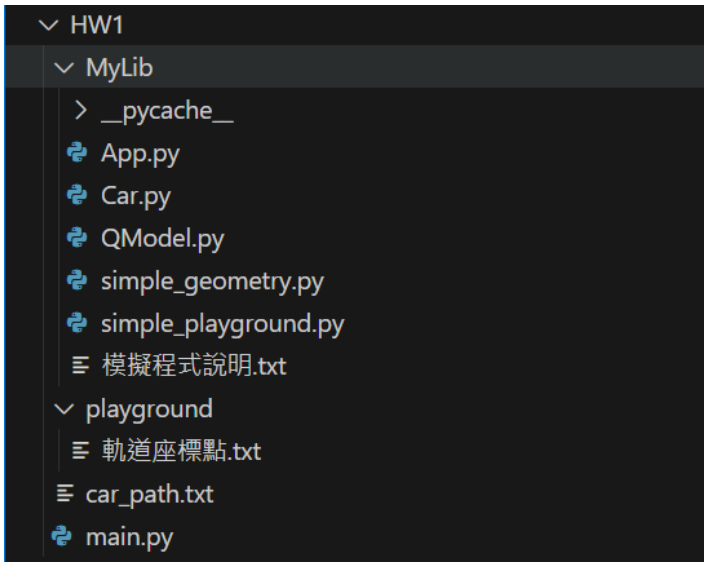

圖 2：專案架構圖

i. MyLib 資料夾：為了要讓程式碼具有好的可為維護性與可擴充性，因此將各部分任務分成 5 個 Python 檔案，寫成一個自建的 Library。

1. App.py：負責 UI 介面的初始化、模擬結果顯示、定義按鈕的功能、錯誤顯示、並且為與其他 Function 互動的統一介面。

2. Car.py：定義模擬車輛。(從範例程式，拆分出來)

3. Qmodel.py：定義 Q-Learning Model 的預測及更新 Q Table 的方式。

4. Simple_geometry.py：負責點和線的計算，例: 旋轉、距離、是否交集。(使用範例程式)

5. Simple_playground.py：定義 Playground Class，負責處理場地與車輛碰撞判斷。(使用範例程式，並修改環境的 Reward 及範例程式的 Bug)

ii. Playground 資料夾：存放軌道做標點.txt。

iii. Car_path.txt：將車輛的移動路徑紀錄。

iv. Main.py：整個專案的主程式。

三、 Q-Model 及 Q Table 的設計方式

使用離散化的 Q-Model 將車輛角度 Action 切分成 80 種，並根據車輛感測器測得的左中右距離設計成雙變數的狀態。

狀態 State：

中間感測器距離（範圍 0＜距離＜100，離散成 30 種）

左右感測器相減（範圍-100＜距離＜100，離散成 60 種）

總狀態數 30*60=1800

動作 Action：

車輛角度（範圍-40＜角度＜40，離散成 80 種）

總動作數 80

Q Table 維度 (30, 60, 80)，大小 30*60*80=144000

四、 車輛與環境 Reward 的關係式

由於目標是讓車輛盡可能的到達終點，因此 Reward 參考的是車輛與終點的距離。

獎勵機制

到達終點：+1000 分

懲罰機制

撞牆：-1000 分
車輛與終點距離: -(車輛中心與終點線的點到線距離) 分

移動步數：每一步 -1 分

參、 實驗結果與分析

模型大小 ：

Q-Table：(30,60,80)

訓練參數：

N_eposides：10000

Alpha：0.1

Gramma：0.9

訓練結果：成功

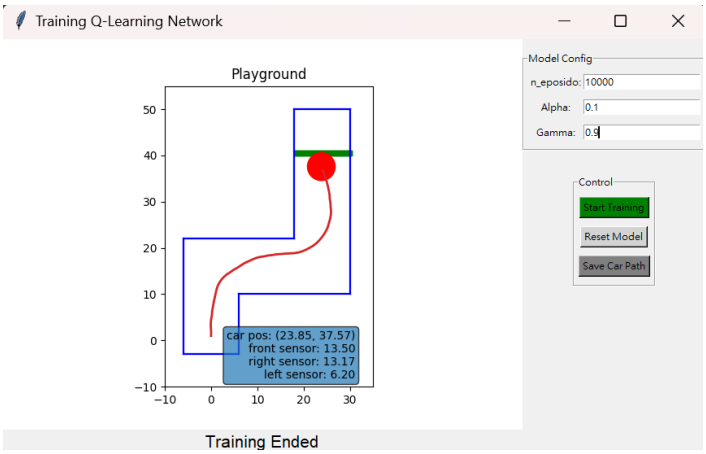

圖 3：訓練結果與車輛路徑

結果分析：

由於 Reward 的計算方式與終點之距離、移動步數相關，可以明顯看到車輛盡可能的減少移動路徑，轉彎時以相當靠牆的路徑移動(有點類似賽車的過彎)。

肆、 未來可改進方向

一、 讓使用者可調整 Q-Table 跟離散化大小
提供使用者更多控制權，以看出 Q-Table 和離散化精細度對整體運算速度和執行成功率的影響。

二、 使用其他訓練方法並進行比較
例如利用監督式學習、模糊系統其他最佳化演算法實作，並比較不同演算法所產生的路徑差異。