計算型智慧 作業三

110403518 資工 4B 林晉宇

一、程式介面說明

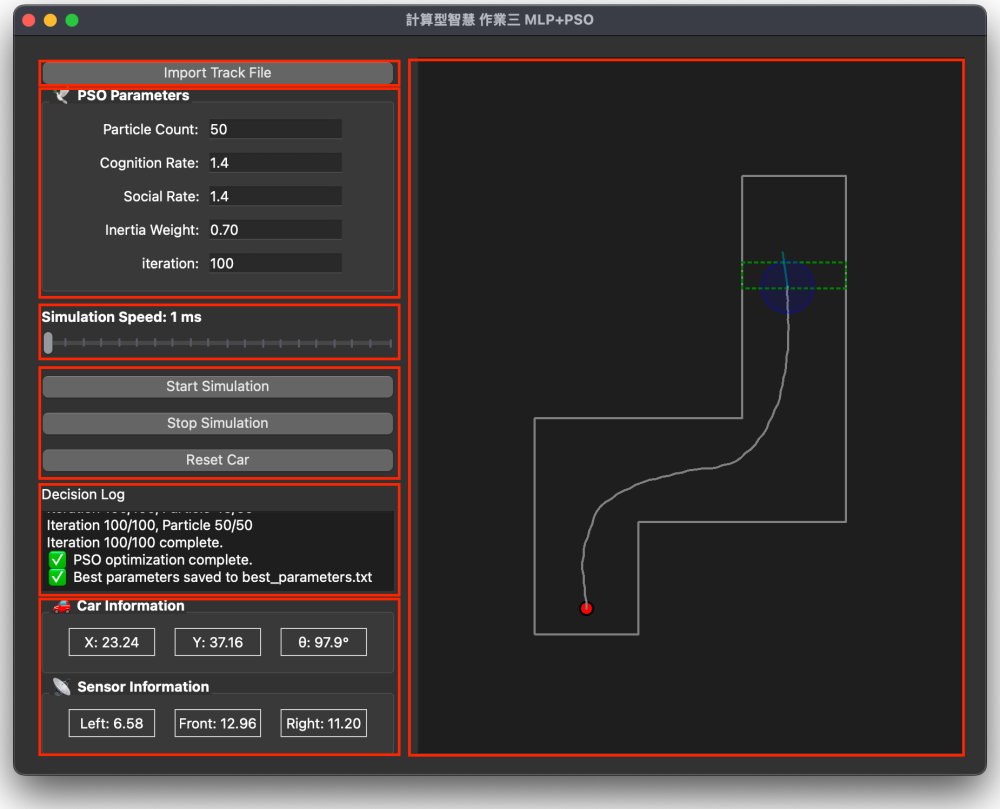
本次作業以 Python 的 PyQT5 開發互動式的 GUI 介面（圖一），整體介面分為左右兩區，操作方式為先匯入軌道座標檔，接著調整參數以及訓練速度，在按下「Start Training」。

左側功能區：

1. **匯入軌道**：點選「Import Track File」可讀入軌道.txt 檔案，並顯於右方顯示軌道
2. **PSO 超參**：調整粒子數量、認知學習率、社會學習率、慣性權重、訓練次數。
3. **訓練速度**：最低可設為 1ms 以達到快速訓練效果（加速動畫）。
4. **控制按鈕**：提供開始、暫停訓練（Stop Training）與重置車輛（Reset Car）。
5. **Decision Log**：記錄訓練過程中的決策狀態。
6. **車輛資訊**：即時顯示車輛的座標 (x, y) 及角度 θ。
7. **感測器資訊**：顯示三個方向（左前右）對邊界的偵測距離。

右側畫布區：

1. 顯示匯入的軌道圖形，包含白色邊界線、紅色起點、綠色終點區域。
2. 車輛以圓形表示，朝向以藍色線段表示。
3. 訓練過程中，車輛行經路徑會留下白色軌跡。

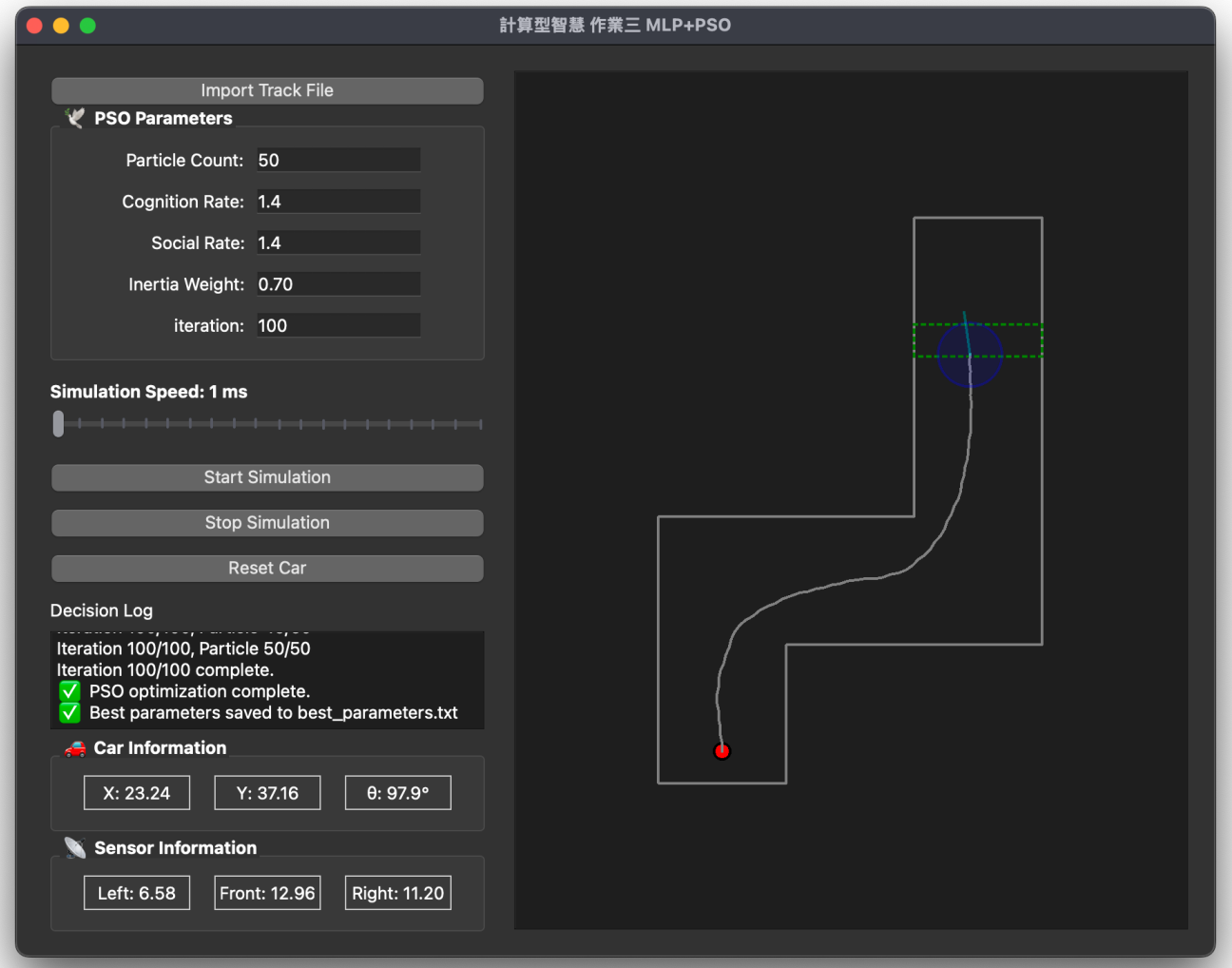


圖一、GUI 介面展示

二、實驗結果

經過大量實驗後，最終發現趨於穩定的設定如下，基於這個設定，測試十次的失敗次數大概是一到兩次，下圖為移動軌跡截圖（圖二）：

1. **粒子數量**：50 個粒子
2. **認知學習率**：1.4
3. **社會學習率**：1.4
4. **慣性權重**：0.7
5. **訓練回合**：100 回合。



圖二、移動軌跡截圖（測試階段）

三、PSO 實作細節

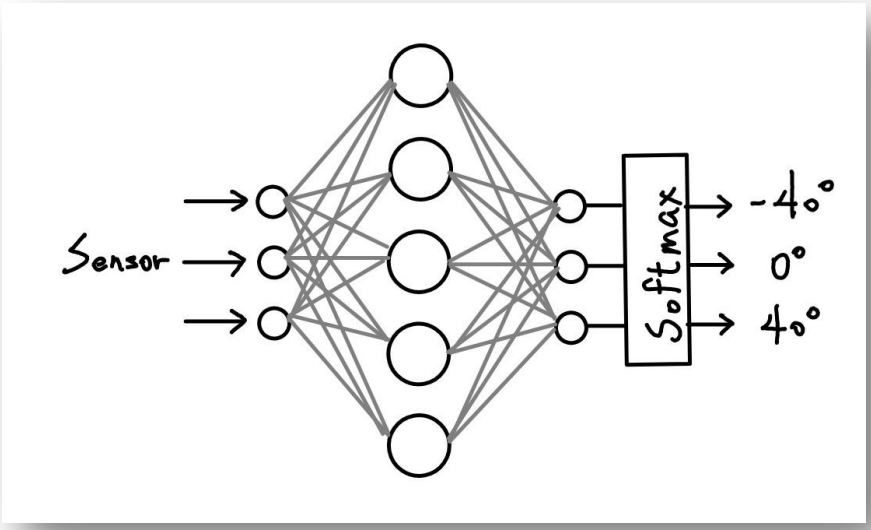
1. **模型架構**：我結合 PSO 與 MLP 來實作這次的作業，PSO 的參數如上述所寫，MLP 架構的部分採用輸入層（3 維）、隱藏層（5 維+ReLU）、輸出層（3 維+Softmax）。 這個模型總共的參數會有 38 個（3 x 5 + 3 + 5 x 3 + 3），一開始會先隨機初始化參數，接著再藉由粒子的探索，基於 fitness function 找出更好的 fitness 的參數。

   甲、**輸入層**：輸入左、前、右方的感測器距離，形成一個三維的向量。

   乙、**隱藏層**：包含五個感知機，每一個感知幾輸出會結合 ReLU 激活函數。

   丙、**輸出層**：輸出三維的資訊，在結合 Softmax 激活函數，形成一個機率分佈，代表往左-40 度、往前 0 度、往右 40 度的機率分別為何。車子下一步行走的角度為機率乘上角度，如下公式：

$$\theta = \sum_{i=1}^{3} p_i \cdot \theta_i$$



圖三、MLP 模型架構圖（一層隱藏層）

2. **PSO 程式**：位於 pso.py 檔案，核心函數為 evaluate_particle_step 以及 optimize_step 這兩者，代表 PSO 的每一個粒子更新的方式以及每跑完一次 iteration，所有粒子更新的方式。

   甲、**evaluate_particle_step()**: 負責在 iteration 中一個粒子的 step 中計算下一步的角度，並計算當下的 fitness，如果優於之前的最佳，則更新。

   乙、**optimize_step()**: 負責在執行完一個 iteration 後，將所有粒子的參數更新，依照標準 PSO 更新公式（c_1, c_2 爲認知與社會學習率、$\mathrm{r}_1, \mathrm{r}_2$ 爲 $[0,1]$ 爲之間的隨機變數、w 爲慣性權重）：

$$v_i = w \cdot v_i + c_1 \cdot r_1 \cdot (p_i - x_i) + c_2 \cdot r_2 \cdot (g - x_i)$$

$$x_i = x_i + v_i$$

3. **Fitness function 設計**：我的設計會是讓 PSO 去探索盡可能小的值為目標。

   甲、**撞到牆**：+100

   乙、**抵達終點**：-100

   丙、**每走一步**：-1 + 與終點距離。（期望車子可以越往終點的方向走）

四、分析與探討

針對這次作業，我主要有以下幾點觀察與思考：

1. **隨機性仍是主導因素**：經過大量實驗後，最終版本的模型在部分時候，仍會遇到 PSO 粒子探索不到終點的時候，我發現這與粒子一開始初始化以及在更新參數時的隨機性仍有很大的關聯。

2. **Fitness function 對於模型訓練速度的影響**：在設計模型與算法時，我也意識到 Fitness function 對於 PSO 訓練模型的重要性，起初我的 Fitness function 只有單純的撞牆與抵達終點的判斷，就會發現即便初始化再多的粒子與再多的 iteration，車子都會漫無目的的行走；後來將 Fitness function 判斷規則加入與終點距離的因素，車子才有逐漸邁向終點的趨勢，即便少量的粒子數量與 iteration 也可以很容易地到達終點。

3. **Gradient Descent 泛化能力的強大**：這次作業是我第一次嘗試使用 Gradient Descent 以外的最佳化演算法來訓練模型參數，也讓我深刻體會到 Gradient Descent 在泛化能力上的強大。使用 PSO 時，我必須依據這次作業的情境設計 Fitness Function，而 Gradient Descent 則可以根據 Loss Function 直接進行梯度下降更新，可以通用於所有任務。此外，PSO 在收斂速度與穩定性上仍面臨不少挑戰。這次作業只有 38 個參數，但若要調整數千萬個參數，PSO 就可能顯得力不從心。1986 年 Rumelhart、Hinton 與 Williams 發表的經典論文《Learning representations by back-propagating errors》，Backpropagation 登上舞台，Gradient Descent 成為現代深度學習的核心方法，確實是實至名歸。

綜合上述，通過課堂理解 PSO 與不同最佳化演算法的運作方式，近一步結合多層感知幾（MLP）透過 PSO 訓練模型參數，在這過程認知各種超參、fitness function、激活函數對於訓練的影響。