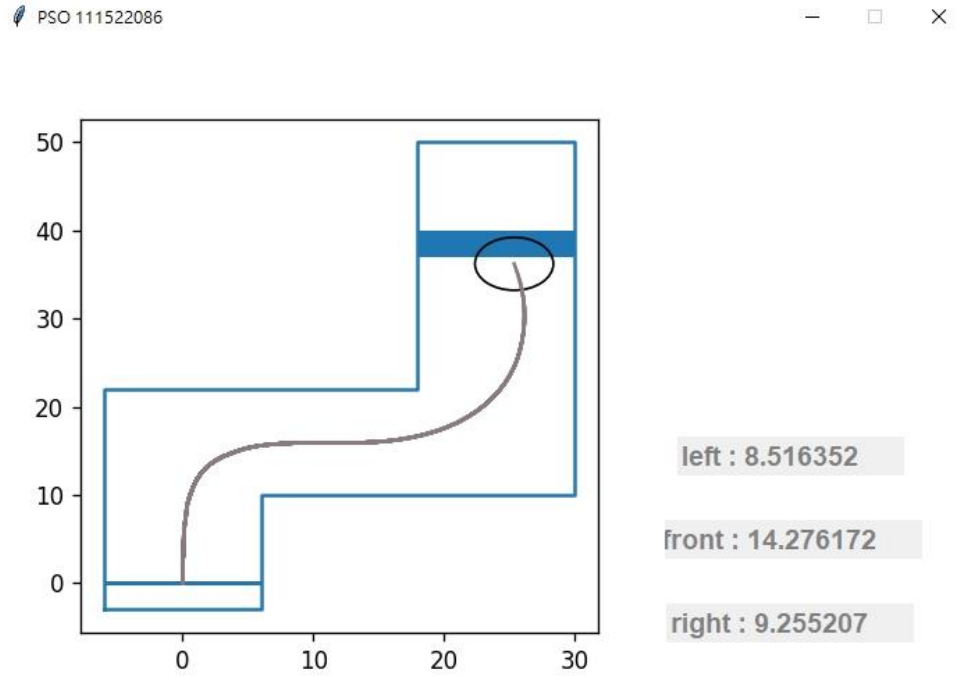
計算型智慧 HW3 – PSO

111522086 林思婷

一、 程式介面說明



藍色細線為軌道的牆壁，黑色圓圈為自走車，起點置於軌道座標(0,0)的位置，終點線位於藍色粗線位置處。程式執行後會自動以動畫顯示自走車每一步的位置並在介面右側顯示左/右前方 45 度角、正前方等三個測量距離 sensor 所測量到的距離。

此次作業共有六份 python 檔，分別為：

1. psoParameter (PSO 實作細節)
2. generatePSO
3. pso (particle 初始化、管理和設置，計算適應值)
4. RBF
5. geometry
6. playground

前 4 份檔案利用基因演算法訓練 RBFN 的網路參數

執行 2.檔案，讀進 train4D.txt 檔等 ground truth 進行訓練，訓練完後得一組網路參數，實作的細節將在第三節中詳細說明。

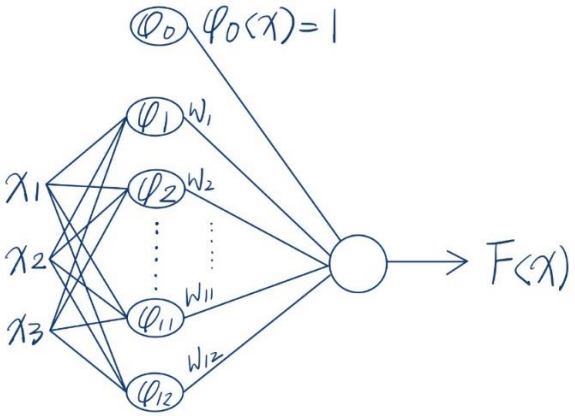
後 4 份檔案將 sensor 偵測所得距離輸入 RBFN 並輸出方向盤角度

二、 實驗結果

如一、圖所示，在移動的過程中均能根據 sensor 回傳距離改變方向盤的角度，以避免觸碰軌道牆壁，順利抵達終點線。

三、 基因演算法實作細節

此次作業採用實數型基因演算法，下圖為 RBFN



$$F(\underline{x}) = \sum_{j=1}^{J} w_j \varphi_j(\underline{x}) + \theta = \sum_{j=0}^{J} w_j \varphi_j(\underline{x})$$

此 RBFN 選用高斯型基底函數

$$\varphi_j(\underline{x}) = \exp\left(-\frac{\left\| \underline{x} - \underline{m}_j \right\|^2}{2\sigma_j^2} \right)$$

我們需將 RBFN 的所有可訓練的參數作為一個粒子(particle)的內容

故此粒子的位置和速度皆為 1 + J + J * xDim + J 維度，如下所示：

$\theta, w_1, w_2, w_3, \ldots\ldots, w_{j-1}, w_j$（此次作業 J=12、xDim=3）

$m_{11}, m_{12}, \ldots\ldots, m_{1j}, m_{21}, m_{22}, \ldots\ldots, m_{2j}, m_{31}, m_{32}, \ldots\ldots, m_{3j}$

$\sigma_1, \sigma_2, \sigma_3, \ldots\ldots, \sigma_{j-1}, \sigma_j$

各維度的初始值範圍：

θ, w_i：0~1

m_{ij}：0~30

σ_i：10^{-6}~1（分母不可為 0）

適應函數為：

$$E(n) = \frac{1}{2} \sum_{1}^{N} (y_n - F(\underline{x}_n))^2$$

N：作業 1 產生的 N 筆成功到達目的訓練資料

y_n：表示訓練資料的方向盤期望輸出值

目標即利用 PSO 找出一個最佳的 particle(其適應值越小越好)

以下詳述 PSO 實作細節

- PoolSize (族群數): 128
- MaxIteration: 30
- phi1: 0.4、phi2: 0.6

起初隨機產生初始族群(初始位置依據不同維度給予相對應範圍的值)

可參考上頁各維度的初始值範圍

Step1.計算各個粒子的適應函數值(此次作業適應值應越小越好)

Step2.所有粒子依據適應值作排序

Step3.記錄p_i(自己過去找到最好的)和p_j(族群過去找到最好的)

Step4.根據下方公式計算下一個時間的速度的位置，(粗體為向量

$$\mathbf{v}_i(t) = \mathbf{v}_i(t-1) + \varphi_1\left(\mathbf{p}_i(t) - \mathbf{x}_i(t-1)\right) + \varphi_2\left(\mathbf{p}_g(t) - \mathbf{x}_i(t-1)\right)$$

$$\mathbf{x}_i(t) = \mathbf{x}_i(t-1) + \mathbf{v}_i(t)$$

重複上述步驟直到迭代完畢(此次作業設定迭代 30 次)

四、 分析

這次有改變 RBFN 隱藏層神經元的數量，從原本 3 個增加至 12 個，因為沒有修過類神經網路，所以沒有想到增加神經元的數量可以提升網路架構輸入與輸出之間映射關係的能力(較高維度的隱藏層空間可以有較為準確的預測值)。一開始以 3 個神經元大概訓練 10 次僅成功抵達終點一次，增加至 12 個神經元後訓練 5 次均成功抵達。

也嘗試過加入之前所讀的論文使用的方法(加入 SA 和 DLS)

A Novel Hybrid Particle Swarm Optimization Algorithm for Path Planning of UAVs

SA: 一開始讓 p_g 能夠接受次優解，避免陷入局部最佳解

DLS: 抽換各維度的值比較適應值，若較好則替換(如下圖)

但此方法將增加可調整參數數量，故後來未採用。

```
/*Acceleration of convergence based on dimensional
  learning*/
if (Fit(x_i^{t+1}) > p_i^t) do: count_i = count_i + 1;
    if count_i > m do: count_i = 0;
        for j = 1 : D do:
            Replace the j-th dimension of x_i^t with the j-th
            dimension of g^t, denoted as temp;
            if Fit(temp) < Fit(x_i^t) do:
                x_i^t = temp;
            end if
        end for
    end if
end if
```