一、 問題

「放射狀基底函數網路」，基本上，其網路架構如圖 1 所示，為兩層的網路；假設輸入維度是 p ，以及隱藏層類神經元的數目是 J ，那麼網路的輸入可以表示成：

$$F(\underline{x}) = \sum_{j=1}^{J} w_j \varphi_j(\underline{x}) + \theta$$
$$= \sum_{j=0}^{J} w_j \varphi_j(\underline{x}) \qquad (3.24)$$

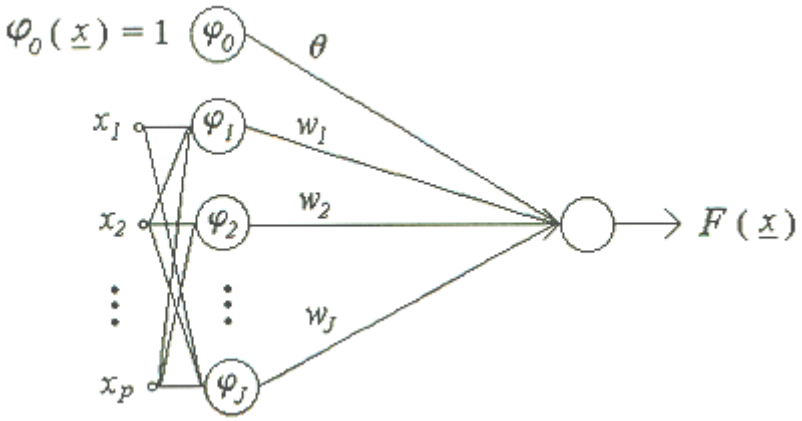


圖 1：放射性基底函數網路的架構。

其中選用高斯型基底函數：

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

其中 $\underline{x}$ = (x1,x2,...,xp) 、$\underline{m}_j$ = (m_{j1},m_{j2},...,m_{jp}) 、‖ ‖ 代表向量絕對值

適應函數為：

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

N：作業 1 產生的 N 筆成功到達目的訓練資料(換不同起始點)

y_n：表示訓練資料的方向盤期望輸出值

<P.s.如果配合 wj 值的範圍為 0~1 之間，在此則必須把 yn 由-40~+40 度正規化到 0~1 之間；如果不想正規化 就必須把 wj 的值範圍調整到 -40~40 之間>

請用實數型基因演算法，找出 wj , mj , σj，在不同的數字 J 下，最好的基因向量 (例如 J 為 9、輸入 x 為 3 維向量，則表示基因向量是 1+9+3x9+9=46 維度的向量，請注意這裡不是指族群數；又例如 J 為 7、輸入 x 為 3 維向量，則表示基因向量是 1+7+3x7+7=36 維度的向量)下，評估函數 E(式 1)為越小越好。

其中基因向量維度公式為 1+J + p*J+J = (p+2)*J+1 維向量(θ,w_1 , w_2 ,.., w_J , m_{11}, m_{12},.., m_{1p}, m_{21}, m_{22},.., m_{2p},.., m_{J1}, m_{J2},..., m_{Jp}, σ_1, σ_2, ..., σ_J)。

θ範圍為 0~1 之間;

Wj 範圍為 0~1 (或是-40~40)之間

<P.s.此需配合訓練集的 yn 跟 F(n)值範圍，所以皆需正規化到 0~1 之間；若不正規化，wj 的值範圍為 -40~40 之間>

mj 範圍跟 X 範圍一樣，如以提供的範例檔則為 0~30；

σj 範圍為 0~10 之間；也可以設定更大的範圍做探討。

由於 $\varphi_j(\underline{x})$ 的值介於 0~1 之間，故需把輸出的 F(x) 由 -40~+40 度 正規化到 0~1 之間，再去計算，而期望輸出值 yn 也必須調整到 0~1 之間來做訓練。如沒有正規化 0~1，則把 Wj 範圍：-40~40 來解決(而不是 0~1 之間)。

需要有複製，輪盤式選擇，競爭式選擇，交配，突變。

須設定疊代次數，族群大小，突變機率，交配機率。

最後訓練完成的 F(x)當作規則 請跑出車子軌跡。

二、 程式設計

首先，先設計出「放射狀基底函數網路」，其中， x =(x1, x2, x3)，x1 為前方感測器距離，x2 為右方感測器距離，x3 為左方感測器距離。輸入維度 p=3，隱藏神經元數 J=3。接下來再利用實數型基因演算法，演化出最佳的 w_j , m_j , σ_j。利用上次作業 1 執行完畢會產生名為 " TrainingData.txt " 的訓練資料。

Step1. 根據族群大小，隨機產生出 n 組(p+2)*J+1 維向量(θ,w_1 , w_2 ,.., w_J , m_{11}, m_{12},.., m_{1p}, m_{21}, m_{22},.., m_{2p},.., m_{J1}, m_{J2},..., m_{Jp}, σ_1, σ_2, ..., σ_J)，每組向量皆視為一個基因

Step2. 利用競爭式選擇將**最優良的**基因(E(n)值最小的)優先複製 10 個到交配池中，再利用輪盤式選擇決定剩下 n-10 個基因複製到交配池中(可重複複製)

Step3. 根據交配機率，來做基因間的交配，其中σ是隨機選取的微量實數，在此設定為-0.1~0.1 之間

$$\begin{cases} \underline{x_1'} = \underline{x_1} + \sigma(\underline{x_1} - \underline{x_2}) \\ \underline{x_2'} = \underline{x_2} - \sigma(\underline{x_1} - \underline{x_2}) \end{cases}$$

Step4. 再根據突變機率，來做突變，其中 s 控制所加入雜訊之大小，在此設定為-0.1~0.1 之間

$$\underline{x} = \underline{x} + s \times random\_noise$$

Step5. 最後，從留下來的 n 個基因中選取 Error(n)值最小的作為最佳的基因

由於一開始的 n 個基因是隨機產生的，所以演化完畢後不一定能得出 Error(n)值很小的基因，若 Error(n)值過大，這時會重新隨機產生出新的族群，並再作一次基因演算法。

三、 主要程式

複製

```
public void reproduction(double[] En) {
        double sumf = 0;
        double[] copyP = new double[groupNum];
        double[] Roulette = new double[groupNum];

        CrossoverPool = new LinkedList<GenePair>();

        for (int i = 0; i < groupNum; i++) {
            sumf += En[i];
        }
        Roulette[0] = 0;
        double bestGene = 0;
        int bestIndex = 0;
        for (int i = 0; i < groupNum; i++) {
            copyP[i] = En[i] / sumf; //複製機率
            if (i != 0) {
                Roulette[i] = Roulette[i - 1] + copyP[i];
            }
            // System.out.println(copyP[i]);
            if (bestGene < copyP[i]) {
                bestGene = copyP[i];
                bestIndex = i;
            }
        }


        /**** 競爭式選擇 *****/
        for (int i = 0; i < 10; i++) {
            GenePair g = new GenePair(gPairs.get(bestIndex).getDNA());
            CrossoverPool.add(g);
        }

        /***** 輪盤式選擇 *****/
        int k = 10;
        while (k < groupNum) {
            double Ran = Math.random();
            int j = 1;
            while (Ran > Roulette[j]) {
```

```
                j++;
                if (j == groupNum) {
                    break;
                }
            }
            int copyNum = (int) Math.round(groupNum * copyP[j - 1]);
            for (int m = 0; m < copyNum; m++) {
                CrossoverPool.add(gPairs.get(j - 1));
                k++;
            }
        }
        while (CrossoverPool.size() > groupNum) {
            CrossoverPool.removeLast();
        }
    }
```

交配

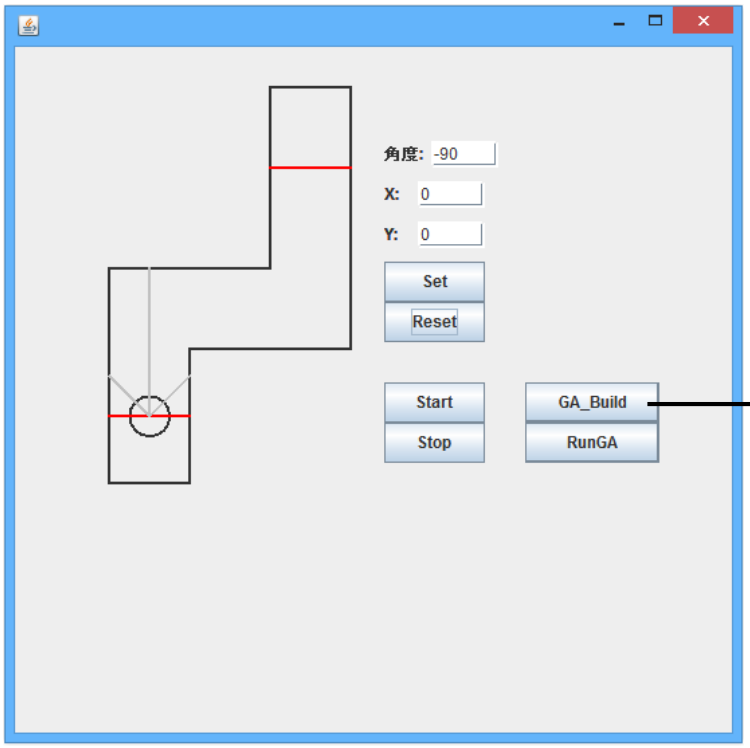
```
public void crossover(double crossoverPro) {
        for (int i = 0; i < CrossoverPool.size(); i++) {
            if (Math.random() < crossoverPro) {
                int crossoverPair;
                do {
                    crossoverPair = (int) Math.floor(Math.random() * groupNum);
                } while (crossoverPair == i);

                /***** GeneCrossOver *****/
                double[] x1 = CrossoverPool.get(i).getDNA();
                double[] x2 = CrossoverPool.get(crossoverPair).getDNA();

                double phi = (Math.random() - 0.5) * 2 * 0.1; // -0.1~0.1 之間
                for (int j = 0; j < x1.length; j++) {
                    x1[j] = x1[j] + phi * (x1[j] - x2[j]);
                    x2[j] = x2[j] - phi * (x1[j] - x2[j]);
                }
                CrossoverPool.get(i).setDNA(x1);
                CrossoverPool.get(crossoverPair).setDNA(x2);
            }
        }
    }
```

突變

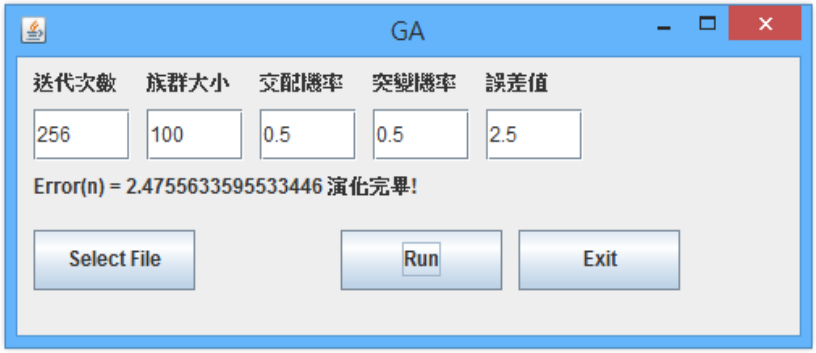
```
public void mutation(double mutationPro) {
        for (int i = 0; i < CrossoverPool.size(); i++) {
                if (Math.random() < mutationPro) {
                        double s = (Math.random() - 0.5) * 2 * 0.1;
                        double[] x = CrossoverPool.get(i).getDNA();
                        for (int j = 0; j < x.length; j++) {
                                if (Math.random() < mutationPro) {
                                        x[j] = x[j] + s * Math.random() * x[j];
                                }
                        }
                        CrossoverPool.get(i).setDNA(x);
                }
        }
}
```

四、執行畫面

主畫面



GA_Build



備註：

1. Error(n)小於誤差值時才會停止重新隨機產生出新的族群
2. 演化完畢後的結果有機率會無法走到終點
3. 由於地圖等比例放大五倍，故 mj 範圍為 0~190，σj 範圍為 0~65
4. 壓縮檔中附有 Demo 影片