朝陽科技大學

資訊管理系碩士在職專班一年級

91(下)Neural Network Homework 3

Self-Organization Map

指導老師：李麗華教授
學生姓名：朱　孝　國
學號：９１５４６１０

中 華 民 國 九 十 二 年 五 月

目錄

第一章 SOM作業說明

本作業的目的旨在學習unsupervised learning的特性，本作業指定利用SOM (Self-Organization Map)做為下面解題的工具.

本作業的二個主題,分述如下:

1.請設計一個具有6x6 output、 2個input、亂數設定weights的SOM network。請學習(learning)所附的資料檔: 92SOMdata-1.xls 其topological mapping的型態。 所附資料檔應有60個input patterns，同學利用SOM將這60個input做學習及聚類(學習的過程則會逐漸與input pattern的topological位置相仿)，建議採用的learning rate 應由1逐漸降至0.1，而R 則應大約由5慢慢的降至 0.5以下左右. (實際上最好的learning rate 及 R-Rate,仍要由同學多方測試來決定)。 本作業最理想的狀態是以30個cluster出現,如果同學可以將其topological relation map畫出來可以加分.

2.請根據所附資料檔92SOMdata-2.xls利用SOM自行設計 network架構，並達成將資料做clustering的分類工作. 所以附資料檔應有50個data patterns，本作業是要讓同學觀察SOM的clustering功能，經由同學的網路參數設定及調整，同學應可逐漸明白如何做出正確的clustering.

3.完成的作業,原則上先上機demo給老師看您的輸出結果(可帶notebook至課堂上demo)，demo後如無問題，再繳交一份報告，內容大要如下:

(1)本作業各個主題說明

(2)SOM的架構設計及所運用的weights、參數(R, η)等設定說明

(3)執行的測試成果，聚類的數量 (如果可以畫出圖更好)

(4)本作業的分析、經驗與心得.

第二章 SOM程式設計說明

2.1 系統功能說明

本系統以SOM理論為依據，使用Perl script開發，搭配Gnuplot繪圖工具實作做拓樸與聚類之功能，可跨平臺使用於Linux/Unix/Windows之命令列模式。

系統提供使用者可自行設定下列參數

1. 可設定輸入層單元個數(input nodes)
2. 可設定學習循環次數(learning cycle)
3. 可設定誤差函數的最小目標值(goal)
4. 可設定學習速率(learning rate)初值,變動率,與最小值
5. 可設定半徑範圍(R.Factor)初值,變動率,與最小值
6. 可設定圖形顯示之範圍(X,Y)
7. 可設定權重最佳化的範圍

其他輔助功能

1. 系統自動判別pattern格式讀入，如'1.2 0.8'，'1.20,0.8'皆可接受。
2. 系統不受限pattern形狀為矩陣或陣列，一律以串流方式讀入，再依指定的node數作切割輸入。
3. 系統提供學習進度百分比之顯示。
4. 搭配GPL工具 Gnuplot做圖形變動的繪製。
5. 針對傳統SOM會造成拓樸對應的學習不良狀況，提供最佳化解決方案。

2.2 執行環境說明

	Linux	Windows
程式語言	Perl (內建)	ActivePerl(http://www.activestate.com)
繪圖工具	Gnuplot	Win32 v3.7(http://www.gnuplot.info)

2.3 使用方式說明

- 學習訓練模式(Learning Mode)：

```
perl som.pl -i input.txt [-u som.ini][-o 1]
```

- 辨識回想模式(Recalling Mode)：

```
perl som.pl -t som.log -u som.ini
```

- 參數說明：

-i	輸入pattern檔	-u	SOM參數設定檔
-t	儲存之權重檔	-o	設定最佳化旗標 : 1
-h	SOM程式說明		

2.4 系統流程說明

茲以pseudo code表現如下 :

```
if (未正確使用參數) {
    顯示 求助畫面() ;
    結束 ;          }
如果有使用者設定檔讀入之 ;
if (模擬回想) {
   讀入之前訓練過的網路與權重 ;
   從命令列取得欲辨識的資料, 否則要求使用者輸入 ;
   尋找最相近的權重 ;
   將勝出的權重寫入som_simu檔, 並建立與gnuplot之連結後將之繪圖出來 ;
   結束 ;     }
if (學習訓練) {
   讀入欲辨識的input檔 ;
   if (欲辨識的pattern 總node數 % input node數 ≠ 0) {
       結束 ;                                          }
   隨機設定權重 ;
   產生權重檔並建立與gnuplot之連結後將之繪圖出來 ;
   for x=1 to 學習循環次數            {
       for y=1 to 所有欲學習的輸入pattern數 {
           計算加總輸出層的值(權重與輸入pattern的距離) ;
           找出最小值的輸出單元為優勝 ;
           加總優勝單元的誤差 ;
           以優勝單元為中心計算半徑範圍;
           更新半徑範圍內的權重 ;
           每%1進度則顯示之 ;                  }
       if (有設定最佳化旗標 and 學習循環次數超過50 and 每5次學習循環) {
           找出遠離輸入值的權重予以重新隨機設定 ;                        }
       50次學習循環內,每1次學習循環 產生權重檔並繪圖出來 ;
       50次學習循環以上,每2次學習循環 產生權重檔並繪圖出來 ;
       學習率*=學習變動率 但不得小於學習率最小值 ;
       半徑範圍*=半徑範圍變動率 但不得小於半徑範圍最小值 ;
       每100次學習循環顯示誤差值 ;
       若誤差值小於goal,則跳出迴圈 ; }
   產生權重檔並繪圖出來 ;
   關閉與gnuplot的連結 ;
   顯示最後的學習循環次數與誤差值 ;
   學習完成警示聲提示 ; }
```

第三章 程式執行結果

3.1 第一小題(拓樸)

3.1.1 使用參數

參數	
輸入層單元數	2
輸出層單元數(6x6)	36
半徑範圍(R.Factory)初值	5
半徑範圍(R.Factory)變動率	0.8
半徑範圍(R.Factory)最小值	0.1
學習率(learning rate)初值	1
學習率(learning rate)變動率	0.99
學習率(learning rate)最小值	0.1
學習循環	600
最終誤差值	11.95

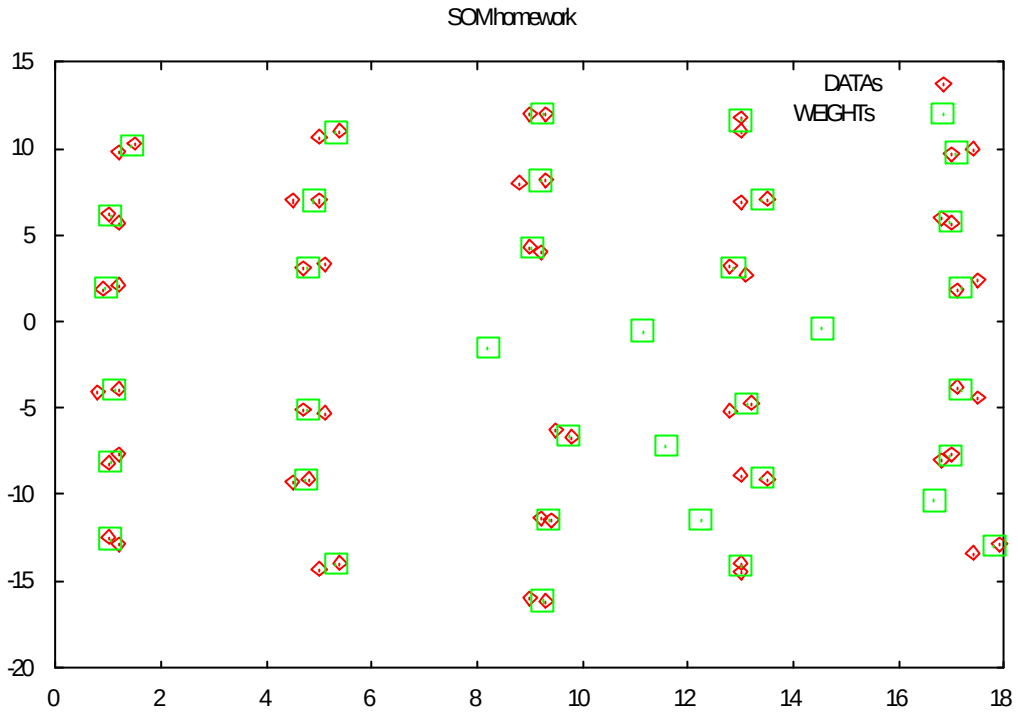
3.1.2 學習訓練畫面

學習訓練指令下達畫面

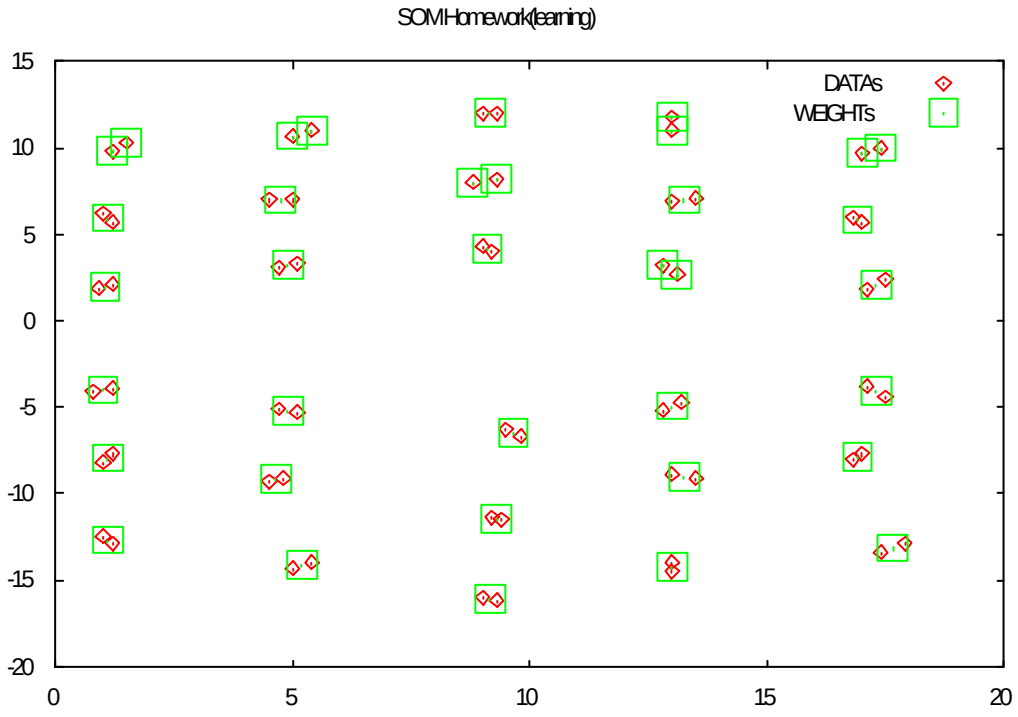
```
D:\perl\neural\som>perl som.pl -i som1.txt -u som1.ini -o 1
------------------------------------------------------------
2,6,5,0.8,0.1,1,0.99,0.1,600,0,20,-20,15,0.38
------------------------------------------------------------
cycle=100, error=16.49   16%
cycle=200, error=14.78   33%
cycle=300, error=14.27   50%
cycle=400, error=13.69   66%
cycle=500, error=12.94   83%
progress : 30240/36000  84%
```

Ps:下達最佳化的參數才能完全match, 如圖2

標準Kohonen演算法學習畫面(圖1)



經過最佳化的學習畫面(圖2)



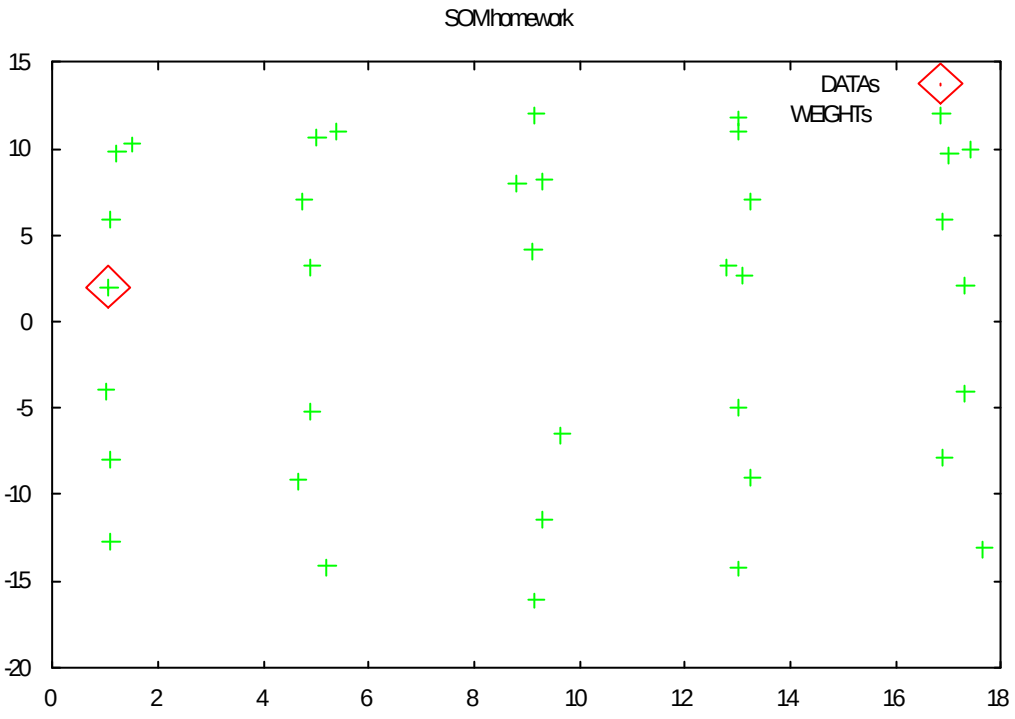
3.1.3 辨識回想畫面

辨識回想指令下達畫面

```
D:\perl\neural\som>perl som.pl -t som.log -u som1.ini
------------------------------------------------------------
2,6,5,0.8,0.1,1,0.99,0.01,600,0,20,-20,15,0.38
------------------------------------------------------------
Please input data:-7.5 3.5

Cluster is [1,1] => 1.04 2
```

辨識回想畫面(圖3)



3.2 第二小題(聚類)

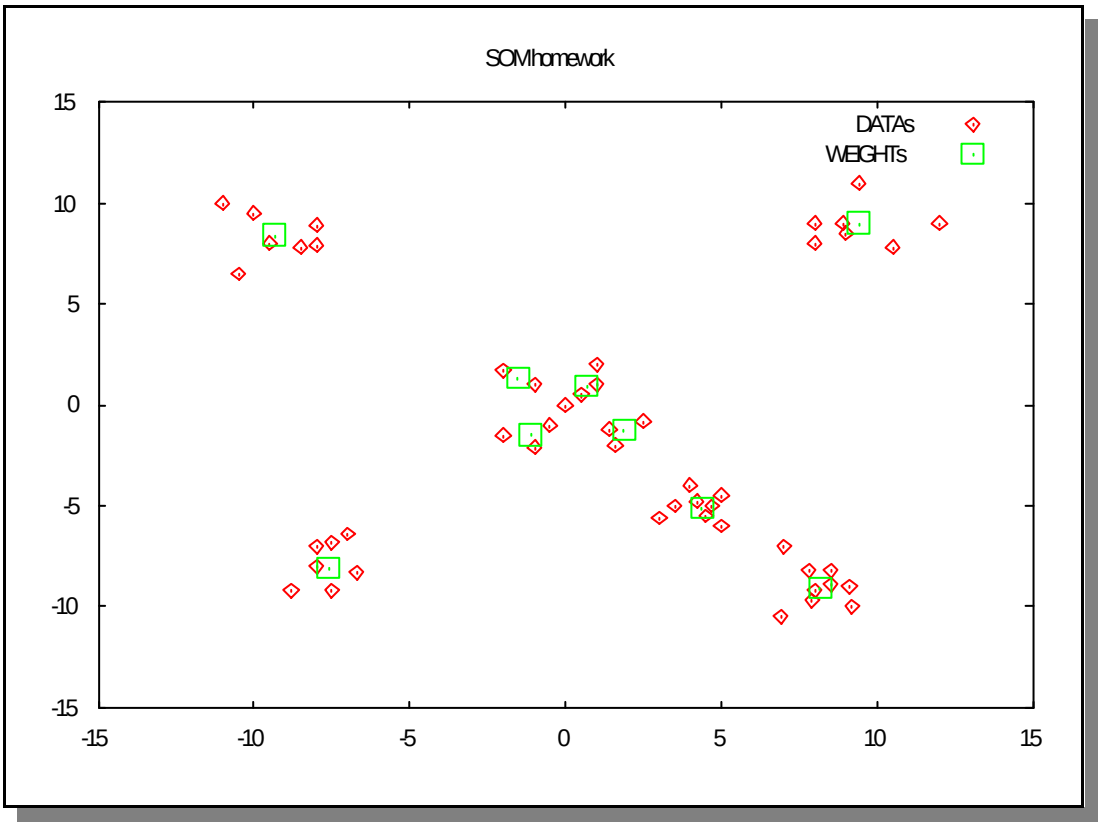
3.2.1 使用參數

參數	
輸入層單元數	2
輸出層單元數(3x3)	9
半徑範圍(R.Factory)初值	2
半徑範圍(R.Factory)變動率	0.8
半徑範圍(R.Factory)最小值	0.1
學習率(learning rate)初值	1
學習率(learning rate)變動率	0.99
學習率(learning rate)最小值	0.1
學習循環	200
最終誤差值	57.26

3.2.2 學習訓練畫面

訓練過程畫面(命令列)

```
D:\perl\neural\som>perl som.pl -i som2.txt -u som2.ini
----------------------------------------------------------------
2,3,2,0.8,0.1,1,0.99,0.1,200,-15,15,-15,15,0.7
----------------------------------------------------------------
cycle=100, error=62.96349%
progress : 5800/10000  58%
```



3.2.3 辨識回想畫面

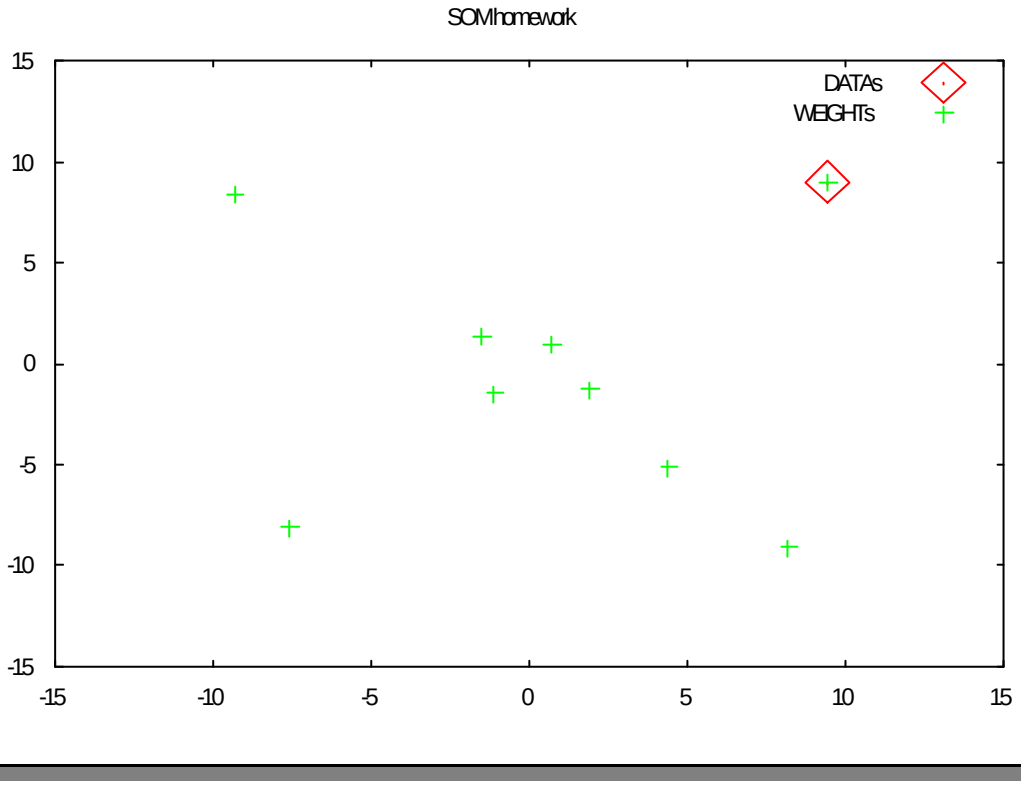
辨識回想指令下達畫面

```
D:\perl\neural\som>perl som.pl -t som.log -u som2.ini
------------------------------------------------------------
2,3,2,0.8,0.1,1,0.99,0.1,200,-15,15,-15,15,0.7
------------------------------------------------------------
Please input data:5 8

Cluster is [0,0] => 9.41 9.03
```

Ps:無須最佳化就能完美聚類

辨識回想畫面(圖2)



第五章 結論與心得

5.1 開發過程

這次的作業因為有上次BPN作業的經驗,開發的速度快多了,大概也花了一星期的時間，同樣是以Perl撰寫，寫作過程中遇到3個較難的地方。

- 沒有弄清楚誤差函數的正確意涵，所以多花了一些時間將之加入，作為程式判停的條件。
- 因Perl無法動態表現輸出單元移動的樣子，所以花了一些時間研究gnuplot這套GPL的繪圖程式。
- 因為老師DEMO了學長的成品，提出了一種[弄亂]的演算法，為此也花了一些時間來實作。

5.2 心得

- **利用gnuplot,excel快速得知題目的聚類形狀**

  在解SOM時,要如何決定輸出層的大小呢?可直接將輸入丟給gnuplot或excel為我們畫出原始資料的聚類情形，如此能較快決定輸出層的網路大小。

- **誤差收斂到一定範圍後就算再增加學習循環次數誤差也無法降低**

  以習題1舉例，學習循環從600次增加到10000次,其誤差仍然在10以上。

- **鄰近半徑的收縮程度要快於學習率的折減程度將有助於收斂**

  根據作業的經驗來看，設定鄰近範圍的收斂速率為0.8，學習率的收斂速率為0.99，較能順利的映射。

- **雖然誤差不容易收斂到很小的數值，但已是正確的拓樸或聚類，所以速度較 BPN模型為快**

  尤其在以很少的輸出單元來代表一組聚類時，如作業2，只需200次學習循環即可作出正確的聚類，雖然誤差高達57.26。

- **最佳化的拓樸排列不是萬靈丹，亦即將不在正確位置的輸出單元弄亂有下列需要注意的地方**

  1. 必須在一定的學習循環後，再進行最佳化，亦即先以Kohonen原本的演算法將大多數的輸出單元位置決定好，剩下對不準的單元再予以隨機給定位置。
  2. 在隨機給定位置時需注意鄰近半徑與學習率的值也要調整，以免原來的拓樸遭受大量的位移。
  3. 若以一般Kohonen演算法即能得出正確結果，建議不要啟動最佳化程序。
  4. 對於作業1最佳化輸出單元時其調整半徑為>=0.38，而作業2的最佳化調整半徑為>=0.7，因此可說明在拓樸作業所設的最佳化調整半徑通常小於聚類作業。
  5. 以本作業而言，最佳化對於拓樸較有意義，聚類無須最佳化就能有好結果。

5.3 結論

透過本次作業的網路參數設定與調整，我可以體會SOM模擬腦細胞物以類聚的特性，主要是由於鄰近區域的觀念，還有其網路拓樸的觀念更讓我了解到當聚類的數目到達一個程度後會有拓樸的功能，特別是體會了所謂非監督式學習的奧妙之處。