自組織映射圖網路簡介

國立台中師範學院教育測驗統計研究所

陳桂霞  黃重光

壹、緒論

本文的目的希望以 Matlab 程式撰寫自組織映射圖網路(Self-Organizing Map)的程式，一方面將 Matlab 這套功能強大又好用的程式語言介紹給讀者，另一方面也將現今最流行的類神經網路(artificial neural network)理論中「自組織映射圖網路」做簡單介紹並示範給讀者。

一、為什麼選用 Matlab

Matlab 這套程式是由 MathWorks 這家公司所發展出來的程式語言，而為什麼筆者要選用這套程式語言呢？其原因如下：

（一）直譯式語言：可以免除使用 C 語言等這類程式需要編譯及連結的麻煩。

（二）多樣化功能函數：Matlab 所提供的函數相當的多，像是數學上要多項式求解、微積分方程式，或是物理、數位信號上的傅利葉轉換、漢明框、自相關函數等等，Matlab 皆能以相當簡單的方式撰寫出來，甚至直接提供其函數。

（三）強大的繪圖能力：繪圖的函式相當簡單而且功能強大，可以以相當容易的方式繪出工程、數學及各種研究上需要的圖形。
二、什麼是自組織映射圖網路(Self-Organizing Map,SOM)？

自組織映射圖網路是類神經網路的一種。類神經網路或譯為人工神經網路，是指模
仿生物神經網路的資訊處理系統。目前著名的類神經網路模式不下數十種，基本上可以
分成四類：1.監督式學習網路(Supervised learning network)，2.無監督式學習網路
(Unsupervised learning network)，3.聯想式學習網路(Associate learning network)，4.最適
化應用網路(Optimization application network)。而筆者這次所要介紹的自組織映射圖網
路便是屬於無監督式學習網路的一種。

三、本文架構

先對自組織映射圖網路的理論做一個簡單的介紹，並舉一個實際的例子，對於自組
織映射圖網路演算法的步驟，配合程式加以說明，最後展示程式的結果並加以討論。

貳、自組織映射圖網路簡介

自組織映射圖網路在 1980 年由 T. Kohonen 提出，它是屬於無監督式學習網路模
式的—種。它的基本原理是模仿人類大腦中，具有相似功能的腦細胞會聚集在一起的特
性，所發展出來的類神經網路。因此，只要在學習範例中有內在的聚類規則，那麼就適
用於自組織映射圖網路。茲將自組織映射圖路的架構及重要概念簡介如下：

一、網路架構

自組織映射圖網路主要由輸入層、輸出層(網路拓撲)及網路連結(加權)三層組成其
網路架構(如圖一所示)。輸入層即訓練範例的輸入向量，輸出層則是由網路拓撲所組成，
而網路連結則是每個輸出層單元與輸入層處理相連結的加權值所成的向量(也是形心座
標)。
二、重要概念

（一）輸入層：訓練範例的向量，一般是使用線性轉換函數，\( f(x) = x \)。

（二）輸出層：網路拓撲之座標，代表著輸入向量之聚類。

（三）網路拓撲：輸出層處理單元組成的座標系(網路座標)，可以是一維、二維甚至於多維，不過一般以使用二維較多，因其繪圖後較易明瞭。

（四）鄰近區域(neighborhood)：在網路拓撲中，某一輸出處理單元為中心的區域，稱為該單元之鄰近區域。

（五）鄰近中心：控制鄰近區域的中心位置的參數。

（六）鄰近半徑：控制鄰近區域大小的參數。

（七）鄰近距離：輸出處理單元在網路拓撲中距鄰近中心的距離。

（八）鄰近係數：控制鄰近區域內輸出處理單元相互影響程度的參數。鄰近係數是「鄰近半徑」與「鄰近距離」的函數。

（九）鄰近函數：控鄰近係數和「鄰近半徑」與「鄰近距離」關係式的函數。
參、網路演算法

類神經網路的演算法一般可分成學習過程和回想過程兩大部份。在自組織映射網路的學習過程主要是修正輸入層及輸出層的網路連結加權值；而回想過程則在判斷某一輸入向量 $X$ 輸出之網路拓撲座標 $Y$。以下就舉一個實例來說明。例：以三個三度空間的球體為例，將其投射在 $1 \times 3$ 的線性網路拓撲[1,2,3]上。

一、三個球體的程式分別為：

(一) $\left( x-15 \right)^2 + \left( y-15 \right)^2 + \left( z-15 \right)^2 \leq 25$

(二) $\left( x+15 \right)^2 + \left( y-15 \right)^2 + \left( z-15 \right)^2 \leq 25$

(三) $\left( x-15 \right)^2 + \left( y+15 \right)^2 + \left( z-15 \right)^2 \leq 25$

二、在三個球體中分別取 30 點，共九十點來當成訓練樣本點：

| (-15, 15, 14) | (-15, 15, 13) | (-15, 15, 12) | (-15, 15, 11) | (-15, 15, 10) | (-15, 15, 16) | (-15, 15, 17) | (-15, 15, 18) | (-15, 15, 19) | (-15, 15, 20) | (-15, 16, 15) | (-15, 17, 15) | (-15, 18, 15) | (-15, 19, 15) | (-15, 20, 15) | (-15, 14, 15) | (-15, 13, 15) | (-15, 12, 15) | (-15, 11, 15) | (-15, 10, 15) | (-16, 15, 15) | (-17, 15, 15) | (-18, 15, 15) | (-19, 15, 15) | (-20, 15, 15) | (-14, 15, 15) | (-13, 15, 15) | (-12, 15, 15) | (-11, 15, 15) | (-10, 15, 15) |
三、學習過程：

（一）設定網路參數

1. 設定網路拓撲矩陣為[1,2,3]的線性網路拓撲座標。
2. 鄰近半徑 \( R = 2 \)
3. 鄰近半徑縮小比例 \( R_{\text{rate}} = 0.8 \)
4. 學習速率 \( \text{ETA} = 1 \)
5. 學習速率縮小比例 \( \text{ETA}_{\text{rate}} = 0.8 \)
6. 設定網路收斂值=272

為了修改參數方便，將參數存成一般文字檔，如果要修改參數時，只要修改文字檔上的參數，而不用修改程式。以下以開啟網路拓撲座標的檔案為例，其程式碼及說明如下：

```plaintext
%% 開啟網路拓撲檔 'nettop.txt'
%% 設定 Global 變數 MAP,MAP_col,MAP_row

fid=0;  
while fid<1
    fiename='nettop.txt';
    [fid,message]=fopen(filename,'r');
    if fid==-1
        disp(m essage);  
    end
end
vardata=fscanf(fid,'%s');
i=findstr(vardata,'%');
vardata=vardata(1:i(1)-1); % 去除百分比後之說明
i=findstr(vardata,',');
j=findstr(vardata,');
MAP_col=length(findstr(vardata(1:j(1)),','))+1; % 網路拓撲之列數
MAP_row=length(j); % 網路拓撲之行數
i=[i,j];
i=sort(i);
temp(1)=str2num(vardata(1:i(1)-1));
for j=2:length(i)
    temp(j)=str2num(vardata(i(j-1)+1:i(j)-1));
```
end
for i=1:MAP_row
    for j=1:MAP_col
        MAP(i,j)=temp(j+(I-1)*MAP_col);       %% MAP 矩陣即為網路拓撲
    end
end
status=fclose(fid);

其它參數，筆者儲存於另一個文字檔，其開檔讀取方式與上面程式相同，故在此不多加贅述。

（二）以均佈隨機亂數設定加權值矩陣 W。

設定輸出層處理單元拓撲座標，筆者採用與上述相同的方法，將網路拓撲座標存成一文字檔，因此不列出其程式。筆者任意取了三個整數點分別是 (1,2,3)、(4,5,6)及(7,8,9)，當成網路拓撲的起始加權。

（三）輸入訓練範例的輸入向量 X：同上述方法，將 90 個樣本點存成文字檔後，再分別讀出即可。

（四）找出優勝的輸出層處理單元 Node[j*]

1. 計算訓練範例與各輸出層單元的距離

\[ \|X(c) - X(c_j)\| = \sum_i \left[ X_i(c) - X_i(c_j) \right]^2 \]

\[ X(c) = \text{訓練範例c的特徵向量} \]

\[ X(c_j) = \text{第j個輸出層處理單元對映之特徵向量} \]

\[ = \text{第j個輸出層單元與輸入層單元間的加權值。} \]

\[ X_i(c) = \text{訓練範例c的特徵向量的第i個元素。} \]

\[ X_i(c_j) = \text{第j個輸出層單元對應之特徵向量第i個元素。} \]

\[ = W_{ij} \]

2. 找出優勝單元(winner)

距離最短的輸出層單元稱為優勝單元。公式如下：
\[ \|X(c) - X(c_j)\| = \min_j \|X(c) - X(c_j)\| \]

則 \(j^*\) 個輸出層處理單元為優勝單元。

為顧及程式的完整性及可讀性，因此將步驟(四)到步驟(七)的程式碼同時列出如下：

```matlab
%% 程式開始
d=[];
while tracount ~= 0           %% 訓練次數
  for k=1:tsampnum
    %% 計算距離
    for i=1:W_row
      d(i)=0;
      for j=1:W_col
        d(i)=d(i)+power(x(k,j)-W(i,j),2);
      end
      d(i)=sqrt(d(i));
    end
    %% 找出優勝單元
    [y,z]=min(d);  %% z 為最小值之 index，也是其網路拓撲位置，y 為其值，加總
    totaldis=totaldis+y;
    %% 計算加權值改變量
    %% 拓撲座標位置
    for i=1:MAP_row
      for j=1:MAP_col
        if MAP(i,j)==z
          I_min=i;
          J_min=j;
        End
      end
    end
    %% 計算鄰近距離 r
    for i=1:MAP_row
      for j=1:MAP_col
        d(i,j)=sqrt(power(i-I_min,2)+power(j-J_min,2));
      end
    end
    for i=1:W_row
      for m=1:MAP_row
        for n=1:MAP_col
          if MAP(m,n)==I
            R_factor=exp(-d(m,n)/R);
          End
        end
      end
    end
  end
end
```
for j=1:W_col
    ETA_W(i,j)=ETA*(x(k,j)-W(i,j))*R_factor;
    W(i,j)=W(i,j)+ETA_W(i,j);
end
end
title('圖二 學習次數與收歛值分配圖');
xlabel('學習次數');
ylabel('收歛值');
hold on;
plot(tracount,totaldis,'r-o');
hold off;

if totaldis<Tranenum
    tracount=0;
else
    totaldis=0;
    ETA=ETA_rate*ETA;
    R=R_rate*R;
    tracount=tracount+1;
end
end

str=[
    num2str(W(i,j)) ','],
end

fid=0;  % 開檔 id 預設值
while fid<1
    filename='W.txt';
    [fid,message]=fopen(filename,'w');
    if fid==-1
        disp(message);  % % 開檔失敗
    end
    str=[',
        num2str(W(i,j)) ',',
    end

(五) 調整輸入層與輸出層間的連結加權值
\[ \Delta W_y = +\eta \cdot (X - W_y) \cdot R\_factor_j \]

其中 \( \eta \) = 學習速率

\( R\_factor_j \) = 第 \( j \) 輸出處理單元的鄰近係數

\[ = f(R,r_j) \]

\[ = \exp(-r[j]/R) \]

（六）重複步驟三至五，直到輸入所有訓練範例後，縮小學習速率 \( \eta \) 及鄰近半徑 \( R \)

\[ \eta = \eta\_rate \cdot \eta \]

\[ R = R\_rate \cdot R \]

（七）重複步驟三至步驟六，直到收斂：筆者是以各訓練範例到其形心的距離之和為其收斂之條件，如果收斂，則其距離應為最小值。在此例中筆者設定收斂值為 272。收斂後將加權矩陣 \( W \) 回存至檔案中。

四、回想過程：

為讓讀者充份了解程式回想的過程，因此筆者直接將回想的函數列於步驟的最後面，其變數的意義與學習程式中的變數意義相同

（一）讀入網路參數

（二）讀入加權值矩陣 \( W \)

並設定輸出層處理單元拓撲座標

\[ X\_Node[j] = j \]

（三）輸入一個測試範例的輸入向量 \( X \)

（四）找出優勝的輸出層處理單元 \( Node[j*] \)

\[ net[j] = \sum_i (X[i] - W[i][j][k])^2 \]
找出 net 值最小的隱藏層處理單元 Node[j*]，即

$$net[j^*] = \min_j net[j]$$

（五）計算輸出層輸出向量 Y

如果 j=j*，k=k*

则 Y[j][k]=1

否则 Y[j][k]=0

Function testsom()

%% 自組織映射圖測試程式 %%

global MAP MAP_col MAP_row;
global W W_col W_row;
x=input('請輸入測試向量');
ty=input('請輸入其所屬類型');

%% 計算距離
for i=1:W_row
    d(i)=0;
    for j=1:W_col
        d(i)=d(i)+power(x(1,j)-W(i,j),2);
    end
    d(i)=sqrt(d(i));
end

%% 找出優勝單元
[y,z]=min(d);  %% z 為最小值之 index，也是網路拓撲的位置，y 為其值，在此無作用
if z==ty
    ['您輸入的向量和類型是正確的']
else
    ['對不起，您輸入的向量和類型不符合']
end

肆、結果與討論

筆者將程式執行的結果，分列討論如下：
一、收斂速度：收斂結果如圖二，我們可以發現雖然起始的加權不在三個球中，但收斂的速度卻相當的快，只要學習八十多多次就收斂了，對於這樣的結果，可以說是非常的符合我們的期望。


三、學習後，回想的正確性：由於筆者為避免勝率不均的問題，而將三個球的樣本點穿插輸入。但見到圖三為樣本點經學習後的成果，確實相當令人激賞，因為其學習的正確率高達 100%。
四、再嘗試：筆者嘗試將網路拓撲擴大二維(3×3)的拓撲座標後，用同樣的九十個點再進行訓練。其起始的加權值如右：(1,2,3)、(4,5,6)、(7,8,9)、(10,11,12)、(13,14,15)、(16,17,18)、(19,20,21)、(22,23,24)、(25,26,27)，在訓練後其加權值更新如下：(-15.56,15,15)、(0.78,-0.68,15)、(15,-15,15)、(-11.48,15,15)、(6.26,8.21,14.66)、(14.99,-3.37,15.24)、(11.48,15,15)、(14.91,13.61,13.62)、(16.02,16.02,16.01)，我們將加權值和我們原來的圓心相比較後，可以發現第7、8、9 為一類，1、4 為一類，3 為第三類，而2、5及6則比較看不出是屬於那一類的。

![圖四 3x3 網路拓撲圖](image)

五、新發現：筆者再將訓練用的九十點，當作輸入點，輸入程式做回想，其結果如圖五：

![圖五 學習的正確性](image)

首先，我們可以看出其回想能力仍然是到達了100%。其次，我們可以發現「輸出
層單元在樣本空間中相近的，在網路拓撲中也是相鄰近的」。

這次以 Matlab 這個程式語言來撰寫自組織映射網路的程式，可以說是相當的成功。筆者希望能藉此拋磚引玉，一方面希望以後能看到更多的 Matlab 的作品，另外一方面，也期待能看到在神經網路這個新興的領域中有更多的發展與突破。

參考文獻

葉怡成（民 82）“類神經網路模式應用與實作”，儒林圖書有限公司。