第四章　自組性類神經網路演算法

類神經網路（Artificial Neural Network）為機器學習中一種功能強大的演算技術[17]。在人們逐漸了解腦細胞的思考與學習模式後，希望電腦也能用類似人類思考的方式來解決問題。類神經網路有大量相互連結的處理單元，通常是以平行的方式操作，可透過樣本或資料的訓練來展現出學習（Learning）、回想（Recall）、歸納推演（Generalization）的能力，所以用在分類（Classification）、最佳化（Optimization）、形式配套（Pattern matching）等都有很好的效果。

許多不同型態的自組性類神經網路都具有相同的特徵，包括：（1）能夠在輸入範例的學習過程中，產生自我組織性而不需要依靠目標輸出值的誤差來修正。（2）可以展現輸入範例的分布或相似性。（3）具有將輸入範例聚類到相似群組中的能力。

另外，在自組性類神經網路中，其中一種學習方式是屬於競爭性學習，這種法則是在學習過程中，只有唯一優勝的神經元進行權重之調整，但其餘的神經元則不被調整。自組性類神經網路中有三種類神經網路是屬於這類的網路，即自組性（特徵）映射（Self-Organizing Map, SOM）、學習向量量化（Learning Vector Quantization, LVQ）、調適性共振理論（Adaptive Resonance Theory, ART）。而本論文將介紹 SOM 與 LVQ 的理論，以及兩者結合之演算法。

第一節　SOM 基本理論

SOM 於 1982 年首先由 Kohonen 提出，屬於前饋非監督式神經網路，同時也是競爭式神經網路[18]。SOM 是以特徵映射的方式，將任意維度的輸入向量，映射至較低維度的特徵映射圖上，如一維向量方式或二維矩陣

方式排列形成拓樸層架構的映射圖，並且依據目前的輸入向量在神經元間彼此相互競爭，優勝的神經元可獲得調整連結權重向量的機會；而最後輸出層的神經元會依據輸入向量的特徵以有意義的拓樸結構在輸出空間中展現，由於所產生的拓樸結構圖可以反應所有輸入值間的分布關係，所以我們將該映射圖稱為拓樸圖（Topology），如下圖所示。

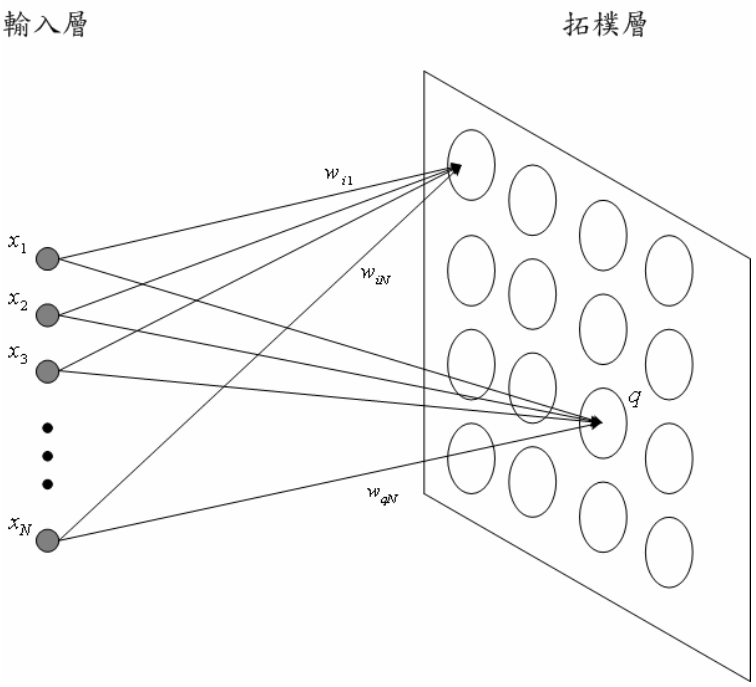

圖 4-1 SOM 架構圖

由上圖 4-1 所示，輸入層代表輸入變數向量 X ，每個輸入變數透過權重 w_{ij} 來連結輸出層的神經元，這些神經元代表著訓練範例輸入資料經由拓樸映射的結果。對於 N 個維度的輸入值而言，輸入向量 X 可表示為（4.1）式，而第 i 個神經元的連結權重 w_{ij} 可用（4.2）式表示。

$$X = \left[x_1, x_2, ..., x_N\right]^T \tag{4.1}$$

$$w_{ij} = \left[w_{i1}, w_{i2}, ..., w_{iN}\right]^T \qquad \text{if } i = 1, 2, ..., M \tag{4.2}$$

在（4.2）式中 M 表示 SOM 神經元的總數，而所有神經元連結權重的初始值可以隨機亂數產生，建議在輸入範圍的中心位置附近，給予一個很小的隨機亂數，如此在形成階段有較快的收斂速度。每一筆輸入向量透過連結權重與神經元相連接，各神經元間以競爭式學習法則來決定被調整的的神經元，也就是說，每筆輸入向量都必須尋找其對應的優勝神經元，即與該輸入向量最近似的神經元，進而調整神經元的連結權重。

決定優勝神經元的方式是比較各神經元的連結權重與輸入向量間的距離，並選取距離最近的神經元為優勝神經元，在此我們採用歐基里德距離公式來計算神經元與輸入向量之間的距離；首先，對於輸入向量 X 與所有神經元的連結權重 w，分別計算它們兩者之間的距離 $q(X)$，進行比較並選取距離最小的神經元稱為優勝神經元。公式如下所示：

$$q_c(X) = \min \left\| X - w_{cj} \right\| \qquad \text{if } c \in i \tag{4.3}$$

每一筆輸入向量所對應出的優勝神經元都不盡相同，因此每個神經元被調整的次數及時機也不一定相同或有規則可循，完全端視輸入向量間的分布關係，此學習過程與傳統聚類分析過程極為相似；但這樣的過程僅將每個神經元視為獨立的個體，而無法將神經元間的拓樸關係表現出來，因此，在 SOM 的學習過程中，有一個重要的關係存在於各神經元間，也就是說各神經元間有著鄰近的關係，讓優勝的神經元在進行連結權重調整時，也會將這樣的訊息傳遞給鄰近的神經元，讓鄰近的神經元也隨著進行連結權重的調整，如此有助於神經元間的拓樸映射關係。設定鄰近神經元間的鄰近關係之參數包括：

一、 鄰近中心

我們將以優勝的神經元為鄰近中心點，並以該中心來修正鄰近區域中所有的神經元。

二、 鄰近半徑

一開始我們先定義其鄰近中心的區域半徑，接著，我們試著先取較大的鄰近半徑值，隨著訓練次數或時間的增加，該鄰近半徑將會逐漸的縮小。

三、 鄰近區域

以鄰近中心為中心點，和中心點距離是以鄰近半徑為長度，所圍繞的區域範圍稱為鄰近區域。鄰近區域的形狀可以有不同的選擇，如矩形、六角形等，而該區域隨著訓練次數或時間的增加而逐漸縮小。

四、 鄰近函數

接下來我們可以根據前面三個定義的參數得到鄰近函數的表示式如下。

$$\eta_{qi} = \exp\left(-\left\|r_i - r_q\right\|^2 \Big/ R^2\right) \qquad (4.4)$$

其中η_{qi}表示第i個神經元與優勝神經元q之鄰近關係值，r_i是鄰近區域中其他神經元於拓樸圖上的位置，r_q則是優勝神經元於拓樸圖上的位置，R則是鄰近半徑。

因此，經由上述討論所獲得的四個鄰近參數值，我們最後可得到的權重值表示如下：

$$w_{ij}(k+1) = w_{ij}(k) + \Delta w_{ij} \qquad (4.5)$$

其中

$$\Delta w_{ij} = \mu(k)\eta_{qi}(k)\left[x(k) - w_{ij}(k)\right] \qquad (4.6)$$

（4.6）式中的$\mu(k)$為第k次迭代次數之學習速率，隨著訓練次數或時間的增加而逐次縮小，一般而言，我們可以定義$\mu(k)$為$0<\mu(k)<1$。

最後我們可以得到一個結論，就是對於一個一開始毫無秩序或任何拓

樸結構的類神經網路而言，由於競爭式學習與鄰近關係的表現，逐漸地形成各神經元間的拓樸結構，而神經元的連結權重經由調整的同時，也逐漸獲得訓練輸入資料的聚類結果。

第二節　LVQ 基本理論

跟前者 SOM 不同的 LVQ，其最大特色在於它是屬於監督式的學習，所以對於每一筆輸入樣本都應該要有相對應的實際之輸出值，而這一筆實際之輸出值也就代表著該輸入向量所隸屬的類別（Class）。雖然 LVQ 的演算模式也是將每一筆輸入向量各自映射到距離自己最近的特徵空間中，以達到群集分類的效果，但不同的是 LVQ 並無像 SOM 一樣有拓樸映射圖的存在，其類神經網路架構圖如下圖 4-2 所示。

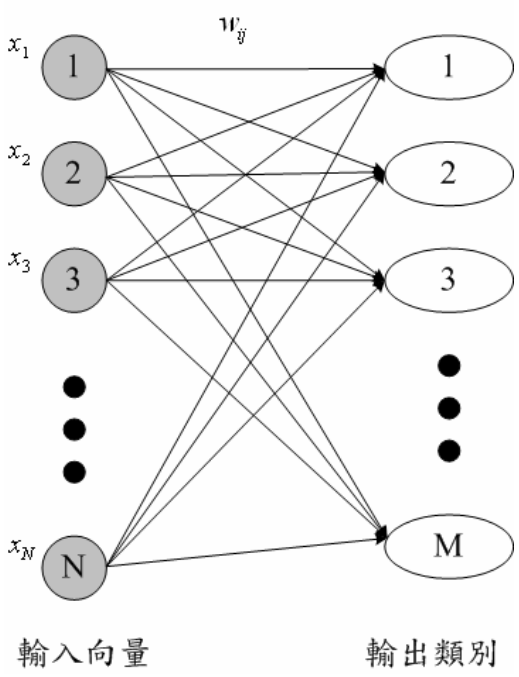

圖 4-2 LVQ 架構圖

我們由圖 4-2 知道，輸入向量 X 透過連結權重 w_{ij} 與輸出類別 Y 相連

接，其中 X 和 w_{ij} 的表示式就如同 SOM 之（4.1）式與（4.2）式，而 Y 之表示式如下：

$$Y=\left[y_1,y_2,...,y_M\right]^T \tag{4.7}$$

接著分別計算輸入向量與所有神經元的連結權重之間的距離 $q(X)$，計算方式同 SOM（其表示式如（4.3）式），進行比較並選取距離最小的神經元，也稱為優勝神經元。優勝神經元可代表輸入向量是屬於優勝神經元的那個類別，所以，在進行分類比較時，即可從優勝神經元所屬的類別與輸入向量所屬的類別進行比對，以判定分類的結果正不正確，來對連結權重進行不同策略的修正。其方法討論如下：

1. 我們首先定義輸入向量剛開始依據實際輸出所做的分類編號為 C_{xi}，接著定義由優勝神經元所得的類別編號為 C_{wq}。
2. 若優勝神經元的類別編號與輸入向量的類別編號是一致的，即 $C_{wq}=C_{xi}$，則採用（4.8）式來修正權重。

$$w_q(k+1)=w_q(k)+\mu(k)\left[X-w_q(k)\right] \tag{4.8}$$

3. 若優勝神經元的類別編號與輸入向量的類別編號屬於不相同的，即 $C_{wq}\neq C_{xi}$，則採用（4.9）式來修正權重。

$$w_q(k+1)=w_q(k)-\mu(k)\left[X-w_q(k)\right] \tag{4.9}$$

經由（4.8）式與（4.9）式的修正式得知，當分類正確時，神經元連結權重的修正方向是往輸入向量接近；反之，分類不正確時，神經元連結權重的修正方向則是遠離輸入向量。因此，透過這樣的策略便可將整個類神經網路的連結權重值逐漸地移向較佳的向量空間，以符合輸入向量的類別歸屬。

第三節　SOL 演算法

我們在此節討論 SOM 與 LVQ 的結合(Self-Organizing Map & Learning Vector Quantization)，簡稱為 SOL 模式，結合之目的就是希望能改善 SOM 於聚類上的錯誤[19]。SOL 的架構圖如下所示。

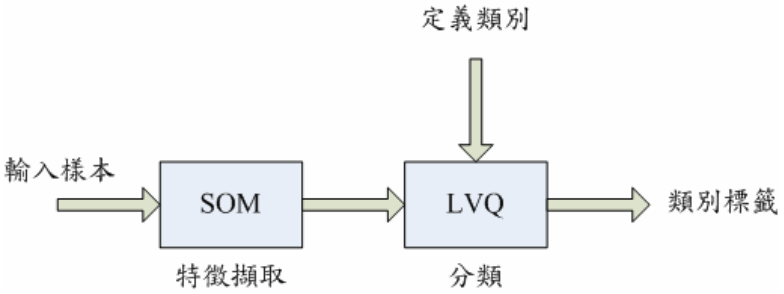

圖 4-3 SOL 之示意圖

當把大量事先已分類好的樣本，運用 SOM 做映射時，可以得到許多類別的群集映射於拓樸圖上，且每個群集都由一個或數個同類別的映射空間所組成。而輸入的差異會造成映射時產生誤差，為了克服聚類上映射錯誤的情況，因此結合 LVQ 以改善聚類上的錯誤情形。我們可以下圖 4-4 為例來說明 SOL 的作法。

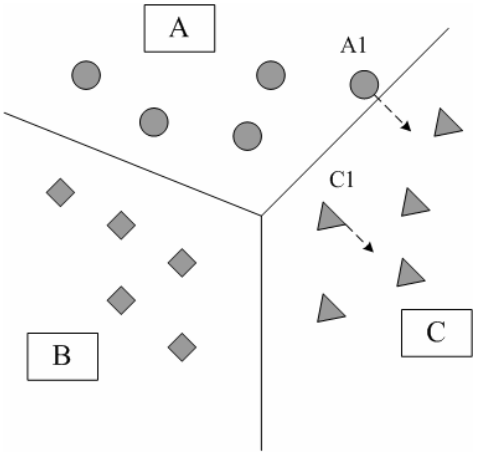

圖 4-4 SOL 特徵空間映射圖

首先假定某樣本輸入之向量空間，經由 SOL 映射之過程，可分為 A、B、C 三個群集。其中 A1 神經元剛開始雖然被分到 A 群集；但事實上 A1 神經元卻是比較相似 C 群集，此時就可以使用 SOL 的修正演算法，讓 A1 神經元逐漸遠離 A 群集而接近 C 群集，而圖中 C1 神經元也經由 SOL 的修正後，更加靠近 C 群集的特徵空間。

經由上述的說明例，我們可以逐步決定 SOL 的演算法則，其演算法步驟所示如下：

1. 首先初始化各神經元之連結權重值w，並且指定類別編號C_{xi}。
2. 決定鄰近區域中的鄰近半徑R與學習速率$\mu(k)$。
3. 設定演算法的停止條件，如演算次數，收斂值等。
4. 載入訓練範例之資料，使用歐基里德之距離公式$q(X)$計算出優勝神經元，並決定優勝神經元之類別編號C_{wq}。
5. 獲得優勝神經元後可以決定其鄰近關係值η_{qi}。

$$\eta_{qi} = \exp\left(-\left\|r_i - r_q\right\|^2 \Big/ R^2\right) \tag{4.10}$$

6. 以優勝神經元的鄰近關係值η_{qi}與類別判斷來修正優勝神經元之權重值，修正式如下。

$$\begin{cases} w_{ij}(k+1) = w_{ij}(k) + \Delta w_{ij} & \text{if } C_{wq} = C_{xi} \\ w_{ij}(k+1) = w_{ij}(k) - \Delta w_{ij} & \text{if } C_{wq} \neq C_{xi} \end{cases} \tag{4.11}$$

其中

$$\Delta w_{ij} = \mu(k)\eta_{qi}\left[x(k) - w_{ij}(k)\right] \tag{4.12}$$

7. 調整學習速率以及適當的縮小鄰近區域的範圍。
8. 檢查是否達到停止條件，若否，則回到步驟 4 之載入訓練範例資料的地方重新執行訓練。