

類神經網路

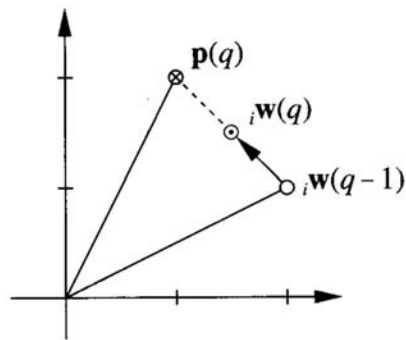
Self-Organizing Map Neural Network

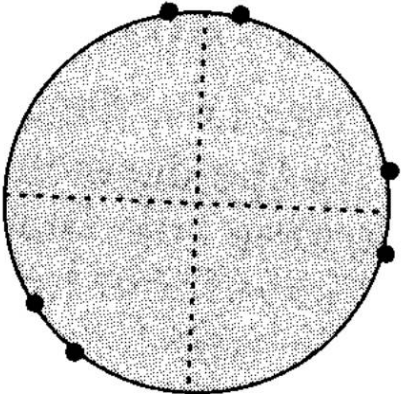
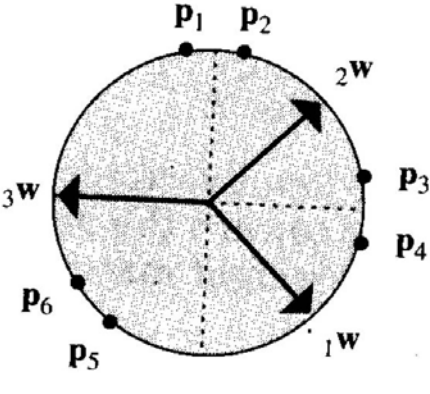
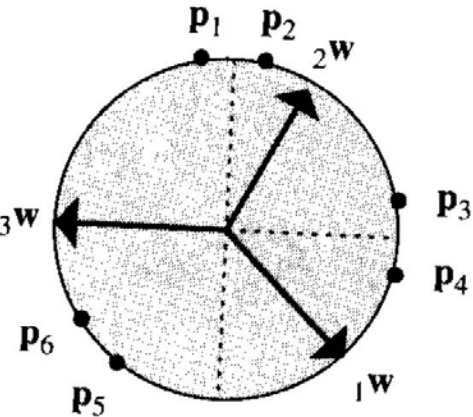
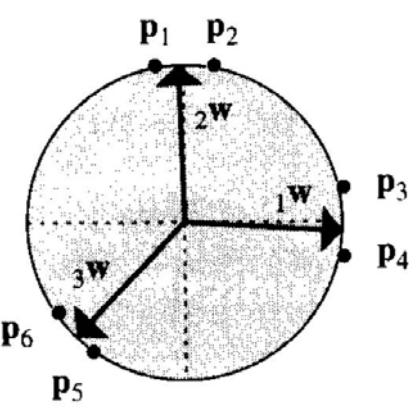
陳慶瀚

義守大學電機系

2002年3月20日

競爭式學習 (Competitive Learning) 原理



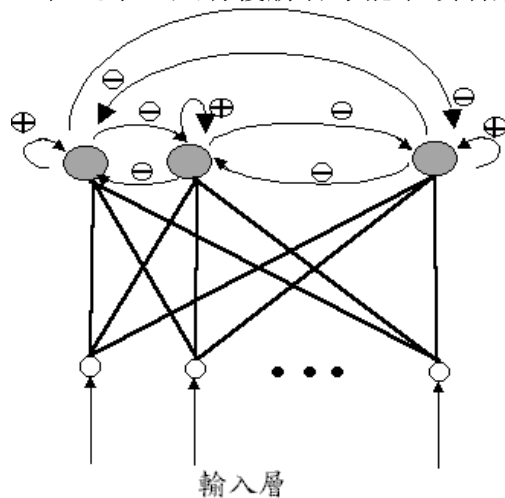
<p>1. 起始階段： 6 個學習範例在單位圓的投影</p>	<p>2. 隨機設定三個 cluster 中心</p>
	
<p>3. 調整三個 cluster 中心 (學習)</p>	<p>4. 學習完成</p>
	

競爭式學習本質上就是一種自組織 (Self-Organization) 學習的方式，它的基本理念是在未經標示的樣本群 (Unlabelled Samples) 尋找某些相似的特徵、規則或是關係，然後再將這些有共同特色的樣本聚集成同類。

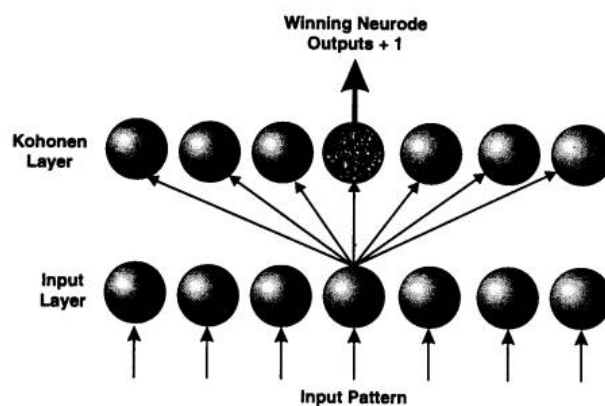
假設在輸出神經元間存在某種形態的競爭，而在競爭的群體之中，只有一個神經元會被激發成活化狀態 (Active State)，即輸出為 1，而其他神經元會會被抑

制成休止狀態 (Inactive State)，即輸出為 0；當競爭完成之後，只有獲勝神經元才會進行調整（即學習），落敗神經元則保持不變。

競爭式學習法有時又稱為「贏者全拿」(Winner-take-all) 學習法。正如名稱所形容的，當輸入呈現時，網路上的類神經元會彼此競爭，以便獲得被活化的機會，而這個機會只賦予對輸入最有反應的那個類神經元，亦即輸出值最大者，這個被稱作得勝者的類神經元，其鏈結值會被調整，以便更增加其與此時輸入之間的相似性。然而，這種「贏者全拿」的特性該如何實現呢？在實際的網路中，我們則是利用側向抑制(Lateral Inhibition)架構來達到這個目的。側向抑制主要是藉由輸出神經元的交互作用來找出最大值；由圖 1 可以看出，每個輸出神經元會產生自我激發訊號(正值)，並試圖抑制其它神經元的活動(負值訊號)，在這個競爭過程之中，只有獲勝者才能繼續活動。



一維的 Kohonen 模型



Winner 的決定

假設在此網路中有 K 個類神經元，如果

$$\underline{w}_k^T(n)\underline{x}(n) = \max_{i=1,2,\dots,K} \underline{w}_i^T(n)\underline{x}(n)$$

那麼第 k 個類神經元為得勝者。

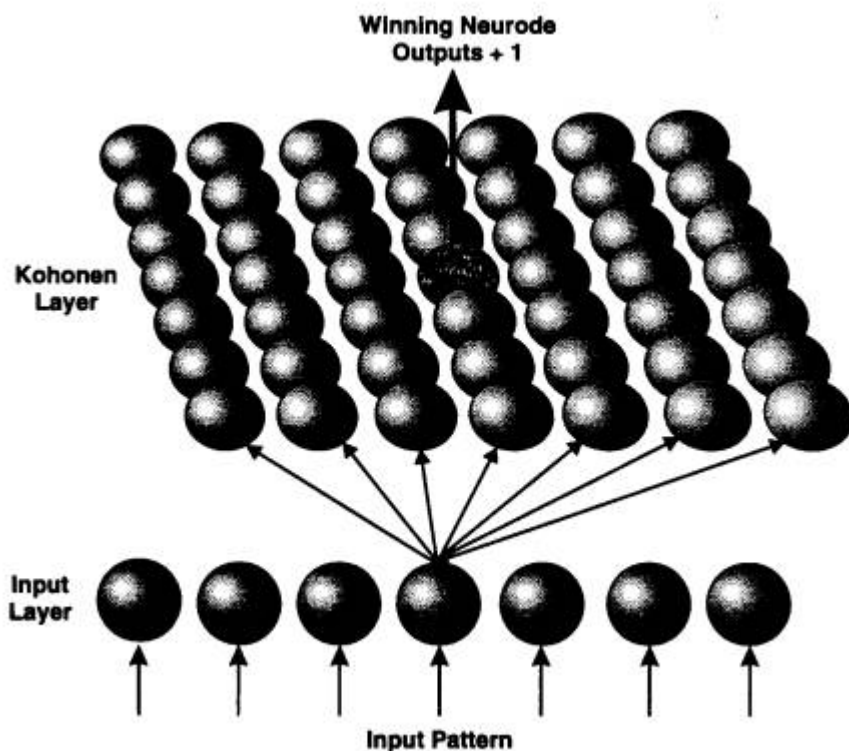
鏈結值之調整(學習)

$$\underline{w}_j(n+1) = \begin{cases} \underline{w}_j(n) + \eta(\underline{x}(n) - \underline{w}_j(n)) & \text{if } j = k \\ \underline{w}_j(n) & \text{if } j \neq k \end{cases}$$

其中 $\underline{x} = (x_1, x_2, \dots, x_p)^T$ 代表網路的輸入向量， $\underline{w}_j = [w_{j1}, w_{j2}, \dots, w_{jp}]^T$ 代表第 j 個類神經元的鏈結值向量， P 代表維度， η 是學習率參數， n 代表離散時間。有一點值得注意的是通常每個類神經元的鏈結值會被「正規化」(Normalize) 成長度為 1 的單位向量。因此，得勝者選取的規則可更改為：
更改為：

若 $\|\underline{x} - \underline{w}_k\| < \|\underline{x} - \underline{w}_j\|$ 則類神經元 k 為得勝者

二維的 Kohonen 模型

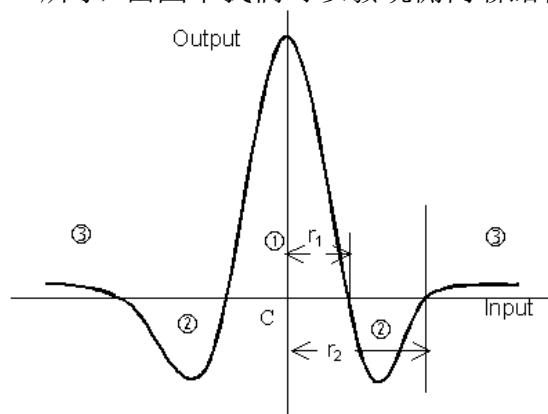


自組織特徵映射 (Self-organizing Feature Maps) 網路 (一般簡稱為 SOFM 或 SOM)，是基於「競爭式學習」的一種網路，也就是說輸出層的類神經元們彼此競爭，以爭取被活化的機會，而這全靠輸出層類神經元間的「側向抑制聯結」來實現。然而，它與一般競爭式學習神經網路稍微不同的是，其競爭方式採用「有福同享」的方式，而一般競爭式學習神經網路則採用「贏者通吃」。也就是說，自我組織特徵映射在競爭之後，不只獲勝神經元有資格學習，它週圍的神經元也能夠學習，這種學習方式就是墨西哥帽的側向作用函數。

在自組織特徵映射中，輸出層的類神經元是以矩陣方式排列於一維或二維的空間中，並且根據目前的輸入向量彼此競爭以爭取得到調整鏈結值向量的機會，而最後輸出層的類神經元會根據輸入向量的「特徵」以有意義的「拓樸結構」

(Topological Structure) 展現在輸出空間中。而這種陣列的拓樸關係，簡單地來說，就是神經元之間的鄰居關係。由於所產生的拓樸結構圖可以反應出輸入向量本身的特徵，因此我們將此種網路稱為自我組織特徵映射網路。

依據生理學和解剖學所得到的證據顯示，通常在許多生物的腦部組織中，會有大量的神經元，彼此之間有側向聯接，並形成二維的層狀結構，而側向聯接的強度與神經元間的距離有關，由觀察發現，側向聯結的回饋量通常是以「墨西哥帽函數」來代表，如圖 1.4 所示，由圖中我們可以發現側向聯結作用的三個不同區域：



- 一、具有一短距離的側向激發作用區域，圖中標示為 1 的區域。
- 二、具有一較大的側向抑制作用區域，圖中標示為 2 的區域。
- 三、一個強度較小的激發作用區域，其涵蓋區域包圍著抑制區域，圖中標示為 3 的區域。

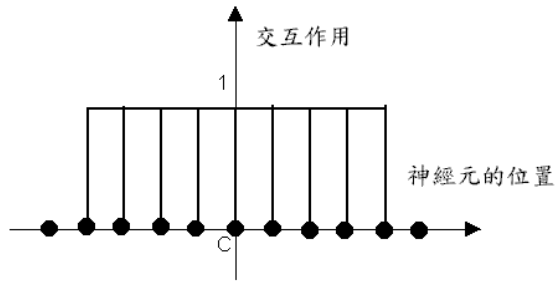
我們可以看出，當神經元被激發時，它附近的神經元會受到不同程度的激勵或抑制作用。其中，位在半徑 r_1 之內的神經元，將受到比較大的激勵作用，而且這個激勵作用會隨著神經元距離的增加而減少。而位在半徑 r_1 和 r_2 之間的神經元，將受到負向的抑制作用。至於半徑 r_2 以外的神經元，則只能感受到很小的激勵作用。

由以上的觀察我們亦可發現，存在於神經元之間的側向聯結作用不再只是一種單純的競爭關係，而是一種競爭（抑制作用）加上合作（激發作用）的關係。對於這種特殊的作用，Kohonen 提出了一種簡化的非監督式類神經網路來模擬上述神經元間之側向聯結作用：

- 一、我們可以簡化墨西哥帽函數的側向聯結作用。也就是說我們可以用被激發類神經元的「鄰近區域」的概念來取代側向聯結作用，他假設在「鄰近區域」內的交互作用具有相同的振幅如圖 1.5 所示。
- 二、對側向聯結鏈結值的修正則以可變的「鄰近區域」來加以取代；也就是說，當鄰近區域設定的越大就表示側向聯結的正回授越強，相反地，當鄰近區域設定的越小也就表示側向聯結的負回授越強。
- 三、我們可以將被激發類神經元 j 的輸出理想化為：

$$Out_j = \begin{cases} 1, & \text{當第 } j \text{ 個類神經元位於鄰近區域內} \\ 0, & \text{當第 } j \text{ 個類神經元位於鄰近區域外} \end{cases}$$

其中 1 為活化函數的極限值。



SOM 學習演算法

- 一、由類神經元所構成的一維或二維的矩陣，這些類神經元的輸出值，可以顯示出那一個類神經元對目前的輸入最有反應。
- 二、尋找出所謂的「得勝者」(Winner)，亦即反應最強烈的類神經元。
- 三、調整得勝者及其鄰居的鏈結值，以便使得被調整過的類神經元對此輸入向量能較調整前更有反應。

我們將輸入向量表示為：

$$\underline{x} = [x_1, x_2, \dots, x_p]^T$$

第 j 個類神經元的鏈結值向量為：

$$\underline{w}_j = [w_{j1}, w_{j2}, \dots, w_{jp}]^T, j = 1, 2, \dots, N$$

我們計算 \underline{x} 與 \underline{w}_j 的內積，也就是 $\underline{w}_j^T \underline{x}$ ，便可以很直接地找出 \underline{x} 與那一個鏈結值向量最匹配，但是，通常我們並不計算內積，反而是將鏈結值向量，正規化成長度為 1 的基本向量之後，找出那一個正規化後的鏈結值與輸入向量的距離最短，此類神經元便是得勝者，可以用下列式子表示此過程：

$$j^* = \underset{j}{\operatorname{argmin}} \|\underline{x} - \underline{w}_j\|, j = 1, \dots, N^2$$

其中 $\|\cdot\|$ 表示歐幾里德範數，而第 j^* 個類神經元就是與當時輸入向量最

匹配的類神經元，我們稱第 j^* 個類神經元為對輸入向量 \underline{x} 的「得勝者」。由上式中可知，特徵映射是將連續的輸入空間映射至離散的類神經元，而

網路的輸出可依據問題的需要，以對輸入向量 \underline{x} 最匹配的類神經元 j^* 作

為網路的輸出，或是以對輸入向量 \underline{x} 具有最小歐幾里德距離的鏈結值向量來作為網路的輸出。由於採用歐幾里德範數來量測輸入向量與鏈結值的相似性；因此，我們只要使得神經網路的鏈結值能夠近似圖樣的機率密度函數，並使它們能夠照一定的順序排列，即可達到映射的目的。

但若光靠調整得勝者的鏈結值是無法將空間中的拓樸關係表現於類神經元間的鄰近關係的，這也就是為何“贏者全拿”的競爭式學習法無法發展出拓樸映射圖的原因；我們可以藉由定義“鄰近區域”函數的方法來取代較複雜的側向連接的回授功能。

在競爭式學習法和鄰近函數的配合之下，才能發展拓樸映射圖，

演算法流程：

步驟一

初始化：將鏈結值向量 $\underline{w}_j(0)$ ，以隨機方式設定其值，但須注意所有的 N 個鏈結值向量之初始值都應不同， N 是類神經元的個數。

步驟二

輸入範例特徵向量：針對時間 n ，輸入向量

$$\underline{x} = (x_1, x_2, \dots, x_p)^T。$$

步驟三

尋找得勝者類神經元：以最小歐幾里德距離的方式找出，在時間 n 的得勝者類神經元 j^* ：

$$j^* = \underset{j}{\operatorname{argmin}} \|\underline{x} - \underline{w}_j\|, j = 1, \dots, N^2$$

步驟四

調整鏈結值向量：以下列公式調整所有類神經元的鏈結值向量：

$$\underline{w}_j(n+1) = \begin{cases} \underline{w}_j(n) + \eta(n)[\underline{x}(n) - \underline{w}_j(n)], & j \in N_{j^*}(n) \\ \underline{w}_j(n) & j \notin N_{j^*}(n) \end{cases}$$

其中 $\eta(n)$ 是學習率參數， $N_{j^*}(n)$ 是得勝者類神經元 j^* 的鄰近區域，兩者都是時間 n 的函數。

步驟五

回到步驟二，直到特徵映射圖形成後，演算法才予以終止。

參數的設定方式：

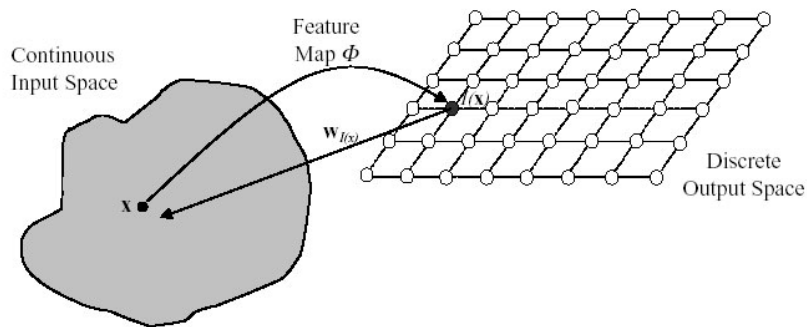
一、學習率參數 $\eta(n)$ ：用來調整鏈結值向量的學習率參數應該隨著時間而調整。舉例來說，在開始的 1000 次學習循環中，學習率參數應相當接近 1，然後隨著時間增加慢慢減小，但仍需保持在 0.1 以上；而學習率參數 $\eta(n)$ 的型式，可以是線性地遞減、指數遞減、或是和時間 n 成反比等都可以，而此時的學習循環，可以視為是演算法的「排列階段」(Ordering Phase)。往後其餘的學習循環主要的目的地是細部地調整特徵映射圖，因此稱之為演算法的「收斂階段」(Convergence Phase)；而此時的學習率參

數，應保持在相當小的數值(如 0.01 或更小)，而收斂階段通常需要大約數千次的學習循環。

二、鄰近區域 $N_{j^*}(n)$ ：鄰近區域函數通常採用包圍著得勝者類神經元 j^* 的正方形型式。而鄰近區域函數也可以採用其它的型式，如六邊形的型式、或是高斯函數的型式等；然而不管採用何種型式，鄰近區域的設定應於一開始時先包含了全部或較大範圍的類神經元，然後隨著時間的增加而慢慢縮減鄰近區域的大小。舉例來說，在「排列階段」時，鄰近區域可以線性地隨著時間 n 而縮減至一相當小的範圍；在「收斂階段」時，鄰近區域應保持只包含一個或兩個類神經元，甚至不需要鄰近區域裏的類神經元，只針對得勝者類神經元調整其鏈結值即可。

Properties of the Feature Map

Once the SOM algorithm has converged, the feature map displays important statistical characteristics of the input space. Given an input vector \mathbf{x} , the feature map Φ provides a winning neuron $I(\mathbf{x})$ in the output space, and the weight vector $\mathbf{w}_{I(\mathbf{x})}$ provides the coordinates of the image of that neuron in the input space.



Property 1 : Approximation of the Input Space

The feature map Φ represented by the set of weight vectors $\{\mathbf{w}_i\}$ in the output space, provides a good approximation to the input space.

We can state the aim of the SOM as storing a large set of input vectors $\{\mathbf{x}\}$ by finding a smaller set of prototypes $\{\mathbf{w}_i\}$ so as to provide a good approximation to the original input space. The theoretical basis of this idea is rooted in **vector quantization theory**, the motivation of which is dimensionality reduction or data compression. In effect, the goodness of the approximation is given by the total squared distance. In effect, the goodness of the approximation is given by the total squared distance

$$D = \sum_{\mathbf{x}} \|\mathbf{x} - \mathbf{w}_{I(\mathbf{x})}\|^2$$

which we wish to minimize. If we work through gradient descent style mathematics

we do end up with the SOM weight update algorithm, which confirms that it is generating a good approximation to the input space.

Property 2 : Topological Ordering

The feature map Φ computed by the SOM algorithm is topologically ordered in the sense that the spatial location of a neuron in the output lattice/grid corresponds to a particular domain or feature of the input patterns.

The topological ordering property is a direct consequence of the weight update equation that forces the weight vector $\mathbf{w}_I(\mathbf{x})$ of the winning neuron $I(\mathbf{x})$ to move toward the input vector \mathbf{x} . Remember that the weight updates also move the weight vectors \mathbf{w}_j of the closest neurons j along with the winning neuron $I(\mathbf{x})$. Together these cause the whole output space to become appropriately ordered. We can visualise the feature map Φ as an *elastic or virtual net* with a grid like topology. Each output node can be represented in the input space at coordinates given by their weights. Then if the neighbouring nodes in output space have their corresponding points in input space connected together, we can see directly the topological ordering.

Property 3 : Density Matching

The feature map Φ reflects variations in the statistics of the input distribution: regions in the input space from which the sample training vectors \mathbf{x} are drawn with high probability of occurrence are mapped onto larger domains of the output space, and therefore with better resolution than regions of input space from which training vectors are drawn with low probability.

We need to relate the probability distribution $p(\mathbf{x})$ of the input vectors to the *magnification factor* $m(\mathbf{x})$ of the feature map. Generally, for two dimensional feature maps the relation cannot be expressed as a simple function, but in one dimension we can show that

$$m(\mathbf{x}) \propto p^{2/3}(\mathbf{x})$$

So the SOM algorithm doesn't match the input density exactly, because of the power of 2/3 rather than 1. Computer simulations indicate similar approximate density matching in general, always with the low input density regions slightly over-represented.

Property 4 : Feature Selection

Given data from an input space with a non-linear distribution, the self organizing map is able to select a set of best features for approximating the underlying distribution.

This property is a natural culmination of properties 1 through 3. Remember how

Principal Component Analysis (PCA) is able to compute the input dimensions which carry the most variance in the training data. It does this by computing the eigenvector associated with the largest eigenvalue of the correlation matrix. PCA is fine if the data really does form a line or plane in input space, but if the data forms a curved line or surface (e.g. a semi-circle), linear PCA is no good, but a SOM will overcome the approximation problem by virtue of its topological ordering property. The SOM provides a discrete approximation of finding so-called *principal curves* or *principal surfaces*, and may therefore be viewed as a non-linear generalization of PCA.

練習

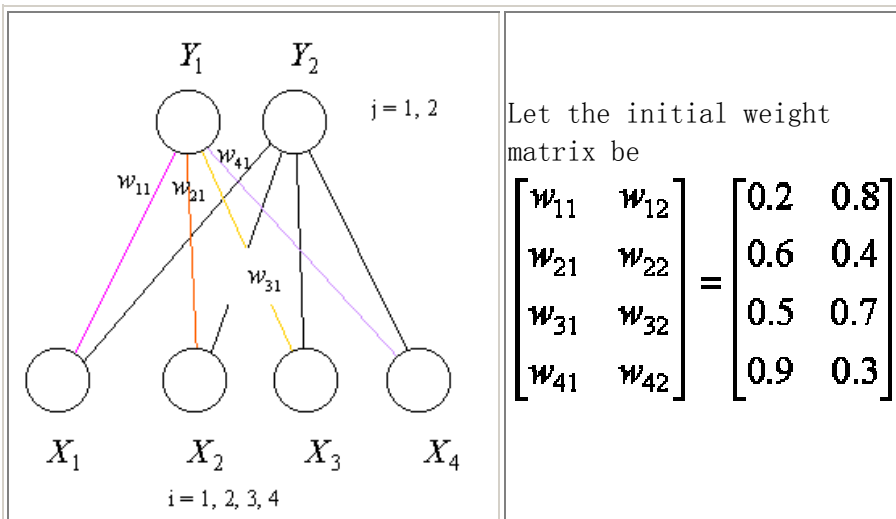
Consider a simple example in which there are only four input training patterns.

x_1	x_2	x_3	x_4
1	1	0	0
0	0	0	1
1	0	0	0
0	0	1	1

Let the learning rate at time $t + 1$ be given by $\alpha(t+1) = \frac{\alpha(t)}{2}$, and suppose $\alpha(t=0) = 0.6$

Let topological radius $R = 0$.

To make the problem very simple, suppose that there are only two neurons in the output layer as shown below:



Following the algorithm presented in the previous lecture:

For vector 1100

$D(1) = 1.86$, $D(2) = 0.98$ [Student exercise]

Hence $J = 2$. Note that $R = 0$, so we need not update the weights of any neighboring neurons.

Using $w_{ij}(new) = w_{ij}(old) + \alpha(x_i - w_{ij}(old))$, the new weight matrix is

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 0.2 & 0.92 \\ 0.6 & 0.76 \\ 0.5 & 0.28 \\ 0.9 & 0.12 \end{bmatrix}$$

For vector 0001

$$D(1) = 0.66, D(2) = 2.2768$$

Hence $J = 1$.

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 0.08 & 0.92 \\ 0.24 & 0.76 \\ 0.20 & 0.28 \\ 0.96 & 0.12 \end{bmatrix}$$

For vector 1000

$$D(1) = 1.8656, D(2) = 0.6768$$

Hence $J = 2$

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 0.08 & 0.968 \\ 0.24 & 0.304 \\ 0.20 & 0.112 \\ 0.96 & 0.048 \end{bmatrix}$$

For vector 0011

$$D(1) = 0.7056, D(2) = 2.724$$

Hence $J = 1$

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 0.032 & 0.968 \\ 0.096 & 0.304 \\ 0.680 & 0.112 \\ 0.984 & 0.048 \end{bmatrix}$$

$$\alpha(1) = \frac{\alpha(0)}{2} = \frac{0.6}{2} = 0.3$$

Now reduce learning rate (step 6):

It can be shown that after 100 presentations of all the input vector, the final weight matrix is

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 6.7 \times 10^{-17} & 1 \\ 2 \times 10^{-16} & 0.49 \\ 0.51 & 2.3 \times 10^{-16} \\ 1 & 1 \times 10^{-16} \end{bmatrix}$$

This matrix seems to converge to

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0.5 \\ 0.5 & 0 \\ 1 & 0 \end{bmatrix}$$

Cluster 1 Cluster 2

TEST NETWORK

Suppose the input pattern is 1100.

Then

$$D(j) = (w_{1j} - x_1)^2 + (w_{2j} - x_2)^2 + (w_{3j} - x_3)^2 + (w_{4j} - x_4)^2$$

$$D(1) = (0 - 1)^2 + (0 - 1)^2 + (0.5 - 0)^2 + (1 - 0)^2 = 3.25$$

$$D(2) = (1 - 1)^2 + (0.5 - 1)^2 + (0 - 0)^2 + (0 - 0)^2 = 0.25$$

Thus neuron 2 is the "winner", and is the localized active region of the SOM. Notice that we may label this input pattern to belong to cluster 2. For all the other patterns, we find the clusters are as listed below.

x_1	x_2	x_3	x_4	Cluster
1	1	0	0	2
0	0	0	1	1
1	0	0	0	2
0	0	1	1	1