Universidad Autónoma del Estado de Hidalgo

Escuela Superior Huejutla

Universidad Autónoma del Estado de Hidalgo - Huejutla

Área Académica: ICBI, Sistemas Computacionales

Tema: Mapas auto organizados

Profesor: M.C.C Víctor Tomás Tomás Mariano

Alumnos:

Leticia Hernandez Hernandez.

Agustin Escamilla Hernández

Periodo: Julio-Diciembre 2011

Resumen.

Se describe las características de los mapas auto organizados, su regla de aprendizaje, su algoritmo de entrenamiento, con la finalidad de resolver problemas de clasificación.

Abstract

It describes the characteristics of self-organizing maps, the learning rule, the training algorithm, in order to solve classification problems.

Keywords: Neural Networks,

Learning Vector Quantization

Se cree que algunos sistemas biológicos realizan sus operaciones siguiendo un método de trabajo que algunos investigadores han llamado, on-center/off-surround; este término describe un patrón de conexión entre neuronas, cada neurona se refuerza a ella misma (center) mientras inhibe a todas las neuronas a su alrededor (surround).

En las redes competitivas biológicas, lo que sucede realmente es que cuando una neurona se refuerza a ella misma, refuerza también las neuronas que están cerca; la transición entre reforzar las neuronas “vecinas” o inhibirlas, se realiza suavemente a medida que la distancia entre las neuronas aumenta. De esta forma el proceso on-center/off-surround; para redes biológicas sigue el comportamiento señalado en la figura 2.5.14, función que habitualmente es referida como sombrero mexicano debido a su forma.

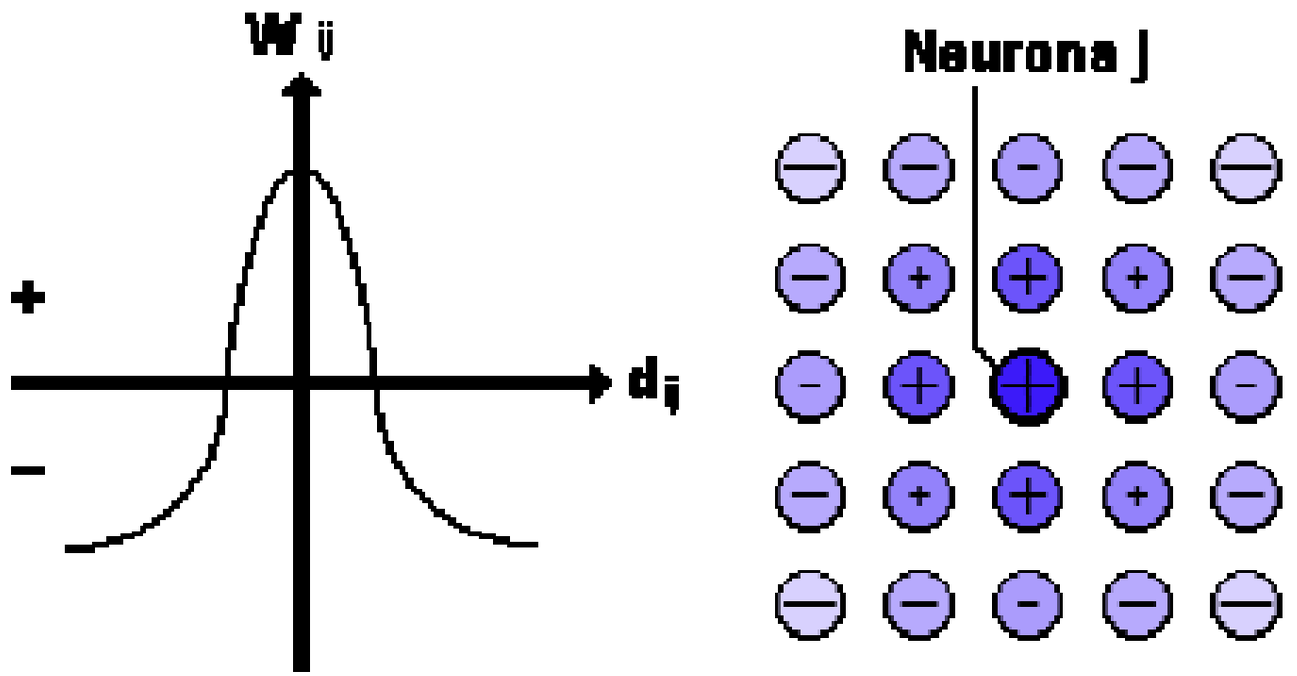

Figura 2.5.14 *on-center/off-surround*; para capas biológicas

Tratando de emular la actividad biológica, sin tener que implementar conexiones on-center/off-surround; de realimentación no lineal, Kohonen diseñó la red conocida como mapa de auto organización (SOM).

Mapa de auto organización (SOM)

- Esta red determina primero la neurona ganadora i* usando el mismo procedimiento que las redes competitivas, luego los vectores de pesos de todas las neuronas que se encuentren en una región cercana “vecindario”, serán actualizados mediante la regla de Kohonen.

$$ {}_i\boldsymbol{w}(q) = {}_i\boldsymbol{w}(q\text{-}1) + \alpha\,(\boldsymbol{p}(q) - {}_i\boldsymbol{w}(q\text{-}1)) \quad para\ i \in N_i{}^*(d) \qquad (2.5.18) $$

- Donde el vecindario Ni* contiene el índice para todas las neuronas que se encuentren a un radio “d” de la neurona ganadora i*

$$ N_i(d) = \{ j, d_{ij} \le d \} \qquad (2.5.19) $$

- Cuando un vector p es presentado, los pesos de la neurona ganadora y de sus vecinas tenderán hacia p, el resultado es que despu muchas presentaciones las neuronas vecinas habrán apre vectores similares que cada una de las otras.

- El concepto de vecindario es ilustrado en la figura 2.5.15; para la primera figura se ha tomado un vecindario de radio d =1 alrededor de la neurona 13; para la segunda figura se ha tomado un vecindario de radio d =2.

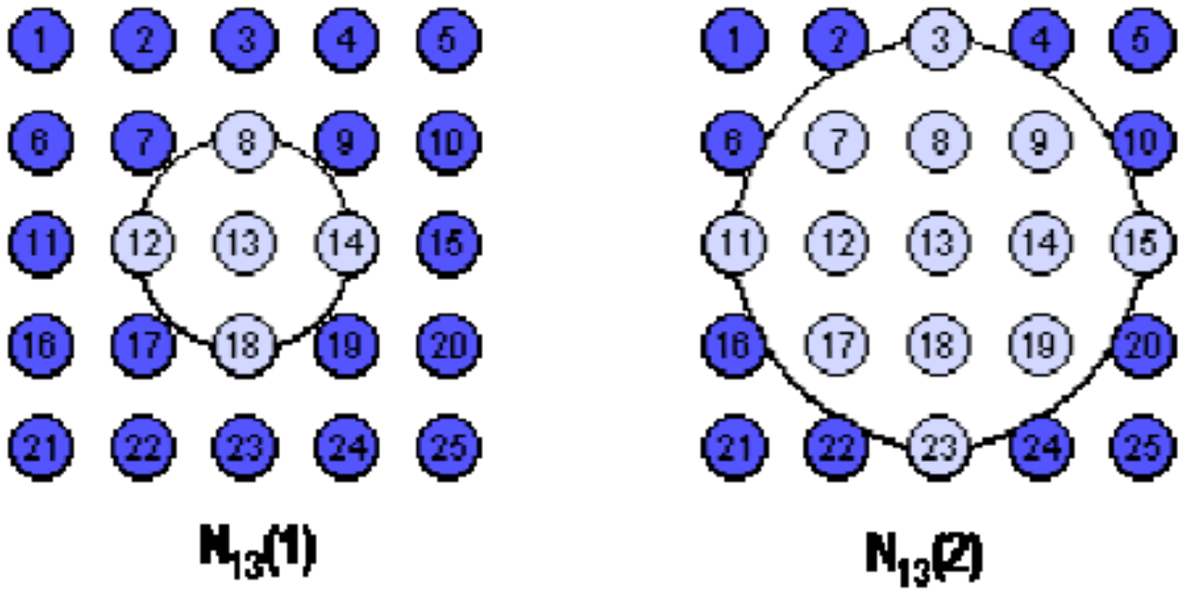

Figura 2.5.15 Vecindarios

- Estos vecindarios pueden definirse como sigue:

$$N_{13}(1) = \{8,12,13,14,18\} \qquad (2.5.20)$$

$$N_{13}(2) = \{3,7,8,9,11,12,13,14,15,17,18,19,23\}$$

- El vecindario puede determinarse en diferentes formas; Kohonen, por ejemplo ha sugerido vecindarios rectangulares o hexagonales para lograr alta eficiencia; es importante destacar que el rendimiento de la red no es realmente sensitivo a la forma exacta del vecindario.

- La figura 2.5.16 ilustra un mapa de auto organización de dos dimensiones:

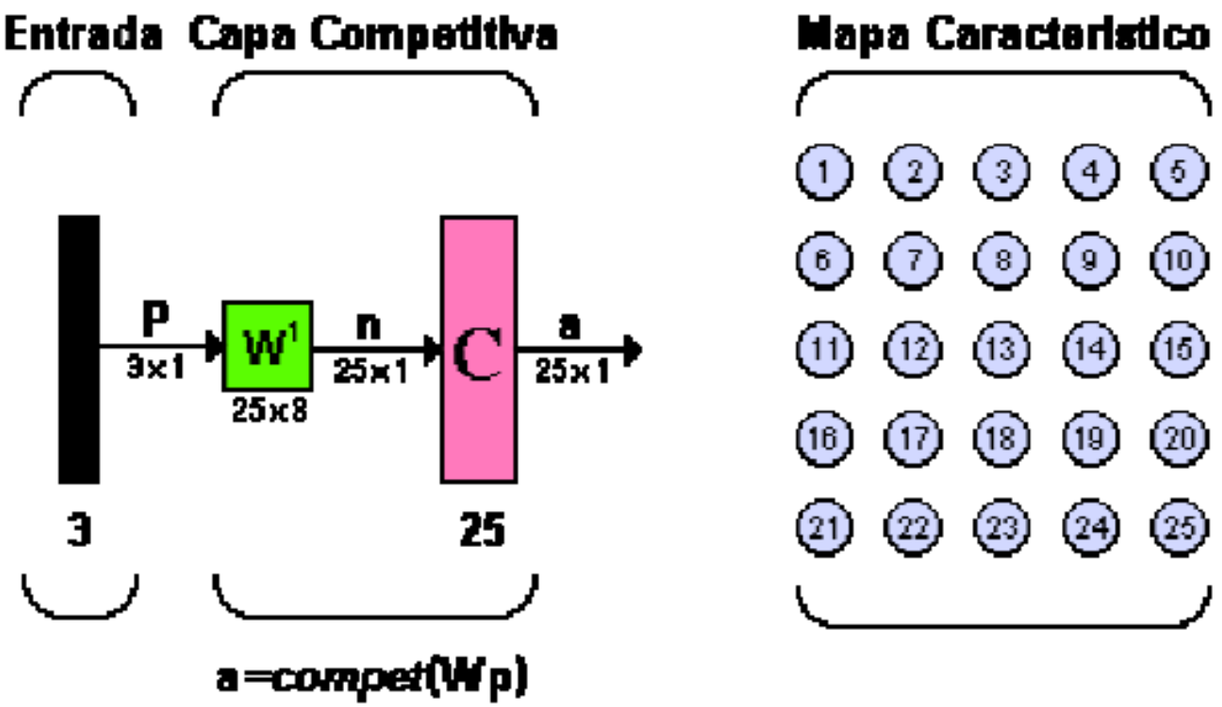

Figura 2.5.16 Mapa de auto organización

- Ejecutar nnd14fm1, nnd14fm2, en Matlab.

Learning Vector Quantization (LVQ).

- Esta red es un híbrido que emplea tanto aprendizaje no supervisado, como aprendizaje supervisado para clasificación de patrones.

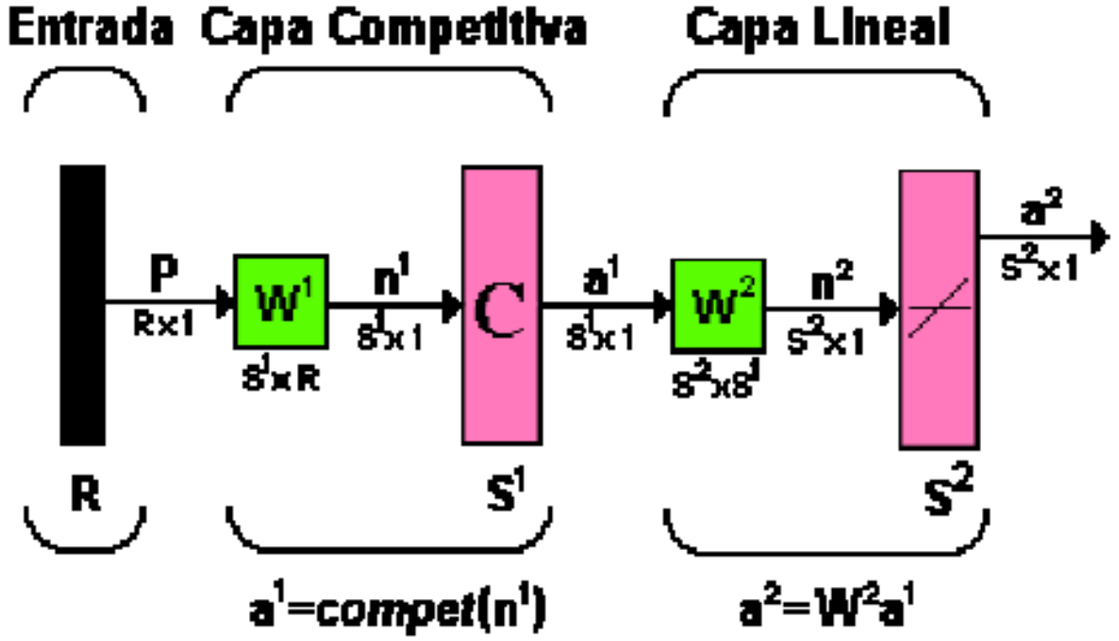

Figura 2.5.17 Red LVQ

- En la red LVQ, cada neurona de la primera capa es asignada a una clase, después cada clase es asignada a una neurona en la segunda capa. El número de neuronas en la primera capa, S^1debe ser mayor o al menos igual que el número de neuronas en la segunda capa, S^2.

- Al igual que con redes competitivas, cada neurona en la primera capa de la red LVQ aprende un vector prototipo, el cual permite a la neurona clasificar una región del espacio de entrada, sin embargo en lugar de calcular la distancia entre la entrada y el vector de pesos por medio del producto punto, la red LVQ calcula la distancia directamente.

- Una ventaja de hacer el cálculo de la distancia directamente, es que los vectores no necesitan ser normalizados, cuando los vectores son normalizados la respuesta de la red será la misma sin importar la técnica que se utilice.

- La entrada neta a la primera capa de la red LVQ es entonces,

$$n_i^1 = -\begin{bmatrix} \| {}_1w^1 - p \| \\ \| {}_2w^1 - p \| \\ \vdots \\ \| {}_{S^1}w^1 - p \| \end{bmatrix} \qquad (2.5.21)$$

- La entrada neta a la primera capa de la red LVQ es entonces,

$$a^1 = compet\,(n^1) \qquad (2.5.22)$$

- Así, la neurona cuyo vector de pesos este cercano al vector de entrada tendrá salida 1 y las otras neuronas, tendrán salida 0; en este aspecto la red LVQ se comporta igual a las redes competitivas, la única diferencia consiste en la interpretación, mientras que en las redes competitivas la salida no cero representa una clase del vector de entrada, para el algoritmo LVQ, indica mas bien una subclase, y de esta forma muchas neuronas (subclases), conforman una clase.

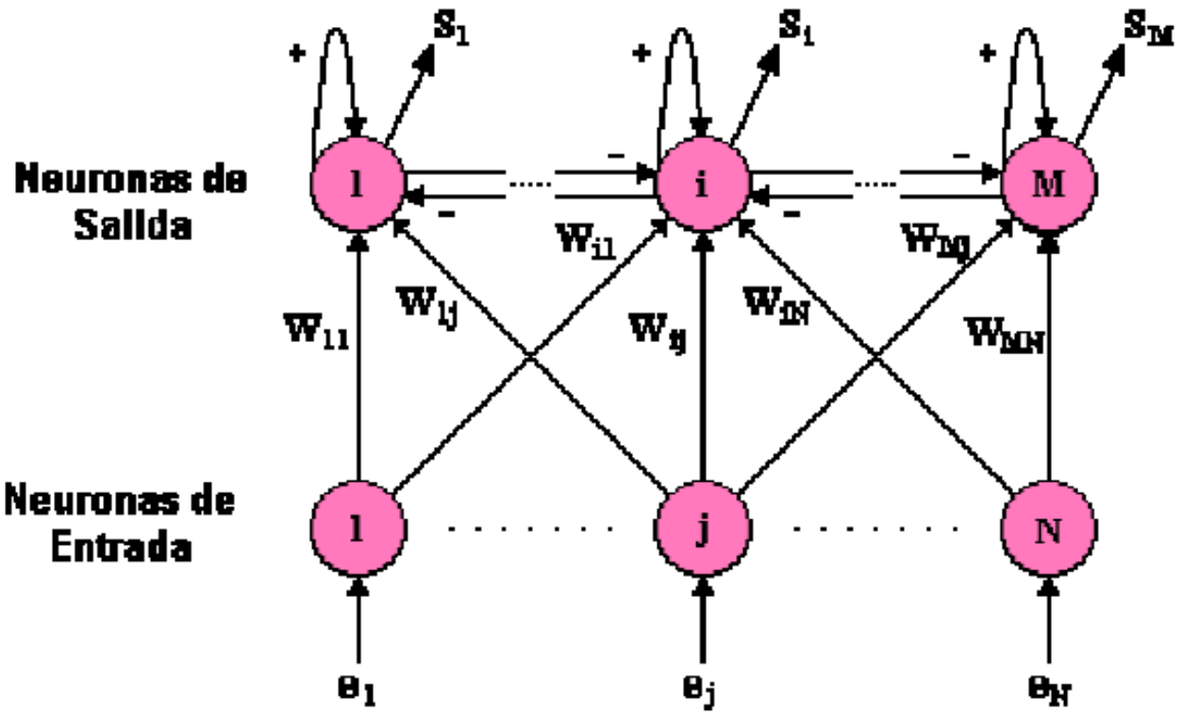

Figura 2.5.18 Comportamiento de las neuronas en una red LVQ

- La segunda capa de la red LVQ es usada para combinar subclases dentro de una sola clase, esto es realizado por la matriz de pesos W^2. Las columnas de W^2 representan las subclases y las filas representan las clases, W^2 tiene un solo 1 en cada columna, todos los demás elementos son cero, la fila en la cual se presenta el 1 índica cual es la clase a la que la subclase pertenece.

$$W^2_{ki} = 1 \Rightarrow \textit{la subclase } i \textit{ pertenece a la clase } k \qquad (2.5.23)$$

- Una propiedad importante de esta red, es que el proceso de combinar subclases para formar clases, permite a la red LVQ crear clases más complejas. Una capa competitiva estándar tiene la limitación de que puede crear sólo regiones de decisión convexas; la red LVQ soluciona esta limitación.

- La red LVQ combina aprendizaje competitivo con aprendizaje supervisado, razón por lo cual necesita un set de entrenamiento que describa el comportamiento propio de la red:

$$\{p_1, t_1\}, \{p_2, t_2\}, \ldots, \{p_Q, t_Q\} \qquad (2.5.24)$$

- Para ilustrar el desempeño de la red LVQ, se considerará la clasificación de un vector particular de tres elementos dentro de otro de cuatro clases, de esta forma:

$$\left\{ p_1 = \begin{bmatrix} \sqrt{0.74} \\ 0 \\ \sqrt{0.74} \end{bmatrix}, \; t_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \right\} \qquad (2.5.25)$$

- Antes de que suceda el aprendizaje, cada neurona en la segunda capa es asignada a una neurona de salida, así se genera la matriz W^2; por lo general, igual número de neuronas ocultas son conectadas a cada neurona de salida, para que cada clase pueda ser conformada por el mismo número de regiones convexas.

- Todos los elementos de W^2 son cero excepto los que cumplan la siguiente condición:

$$Si\ la\ neurona\ i\ es\ asignada\ a\ la\ clase\ k \Rightarrow w^2{}_{ki}=1 \qquad (2.5.26)$$

- Una vez W^2 ha sido definida, nunca será alterada. Los pesos ocultos W^1 son actualizados por medio de la regla de Kohonen.

La regla de aprendizaje del algoritmo LVQ, trabaja de la siguiente manera:

En cada iteración, un vector de entrada p es presentado a la red y se calcula la distancia a cada vector prototipo.

Las neuronas ocultas compiten, la neurona i* gana la competición y el i*-ésimo elemento de $\boldsymbol{a^1}$ se fija en 1.

a^1 es multiplicada por $\boldsymbol{W^2}$para obtener la salida final $\boldsymbol{a^2}$, la cual tiene solamente un elemento no cero, k*, indicando que el patrón p está siendo asignado a la clase k*.

La regla de Kohonen es empleada para mejorar la capa oculta de la red LVQ, en dos formas:

Primero, si p es clasificado correctamente los pesos de la neurona ganadora ${}_{i^*}w^1$ se hacen tender hacia p.

$$ {}_{i}*\boldsymbol{w}(q) = {}_{i}*\boldsymbol{w}(q-1) - a(q)\,(\boldsymbol{p}\,(q) - {}_{i}*\boldsymbol{w}(q\text{-}1))\;\; si\;\; a^2{}_k = t_{k^*} = 1 \qquad (2.5.27) $$

- El resultado será que cada neurona se moverá hacia los vectores que cayeron dentro de la clase, para la cual ellos forman una subclase y lejos de los vectores que cayeron en otras clases.

- Se ilustrará el funcionamiento de la red LVQ, buscando que clasifique correctamente los siguientes patrones, cuyas clases se han definido arbitrariamente:

$$clase\ 1:\ \left\{\boldsymbol{p}_1=\begin{bmatrix}-1\\-1\end{bmatrix},\ \boldsymbol{p}_2=\begin{bmatrix}1\\1\end{bmatrix}\right\},\quad clase\ 2:\ \left\{\boldsymbol{p}_3=\begin{bmatrix}1\\-1\end{bmatrix},\ \boldsymbol{p}_4=\begin{bmatrix}-1\\1\end{bmatrix}\right\}$$

- Los vectores esperados asociados a cada una de las entradas son:

$$\left\{\boldsymbol{p}_1=\begin{bmatrix}-1\\-1\end{bmatrix},\boldsymbol{t}_1=\begin{bmatrix}1\\0\end{bmatrix}\right\}\left\{\boldsymbol{p}_2=\begin{bmatrix}1\\1\end{bmatrix},\boldsymbol{t}_2=\begin{bmatrix}1\\0\end{bmatrix}\right\}\left\{\boldsymbol{p}_3=\begin{bmatrix}1\\-1\end{bmatrix},\boldsymbol{t}_3=\begin{bmatrix}0\\1\end{bmatrix}\right\}\left\{\boldsymbol{p}_4=\begin{bmatrix}-1\\1\end{bmatrix},\boldsymbol{t}_4=\begin{bmatrix}0\\1\end{bmatrix}\right\}$$

- La posición inicial de los patrones de entrada es la siguiente:

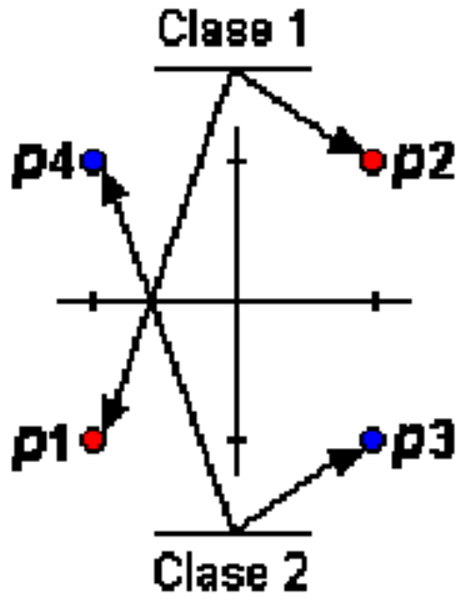

Figura 2.5.19 Posición de los patrones de entrada

- Si se escogen dos subclases para cada una de las dos clases existentes, tendremos entonces cuatro subclases, lo que determina que deben haber cuatro neuronas en la capa oculta. La matriz de pesos para la capa de salida es:

$$W^2 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

- W^2 conecta las neuronas ocultas 1 y 2 a la neurona de salida 1 y las neuronas ocultas 3 y 4 a la neurona de salida 2. Cada clase será formada por dos regiones convexas.

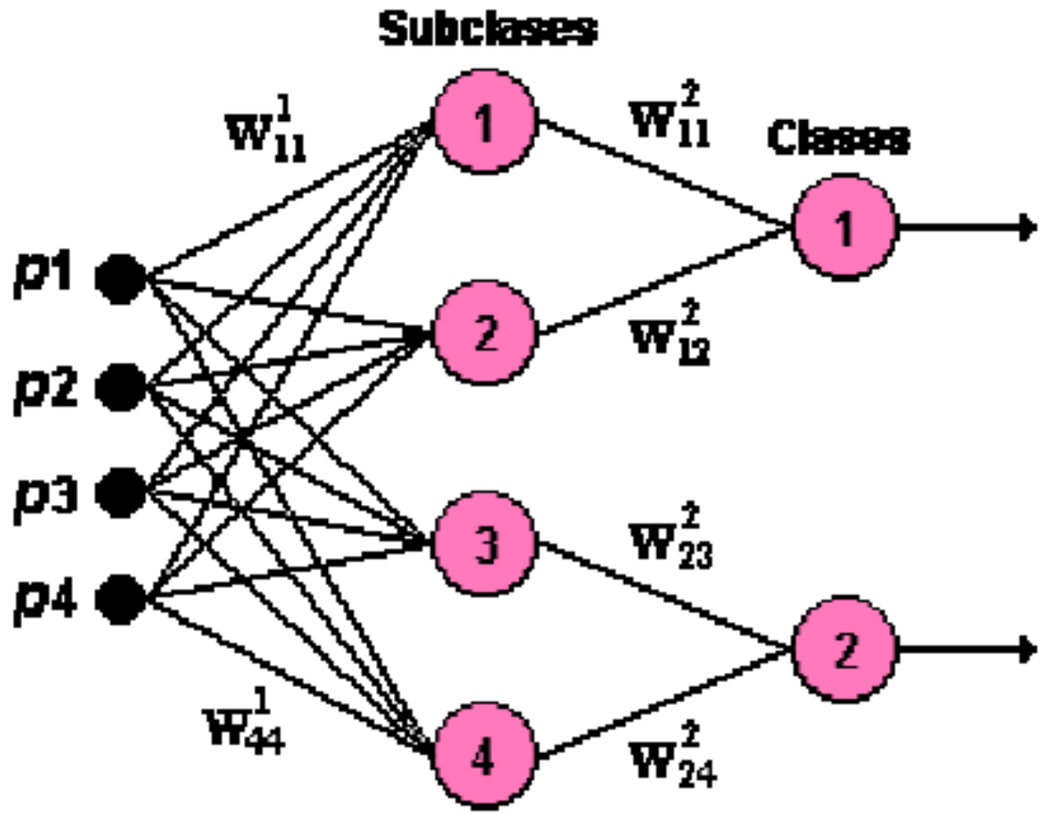

Figura 2.5.20 Esquema de la red LVQ que solucionará el ejemplo

- $\boldsymbol{W^1}$ será inicializada con valores aleatorios, de la siguiente forma:

$$ {}_1w^1 = \begin{bmatrix} -0.673 \\ 0.970 \end{bmatrix}, \; {}_2w^1 = \begin{bmatrix} -0.969 \\ -0.379 \end{bmatrix}, \; {}_3w^1 = \begin{bmatrix} 1.002 \\ 0.140 \end{bmatrix}, \; {}_4w^1 = \begin{bmatrix} 0.785 \\ 0.955 \end{bmatrix} $$

- La posición inicial de estos vectores de pesos, se observa en la figura 2.5.21

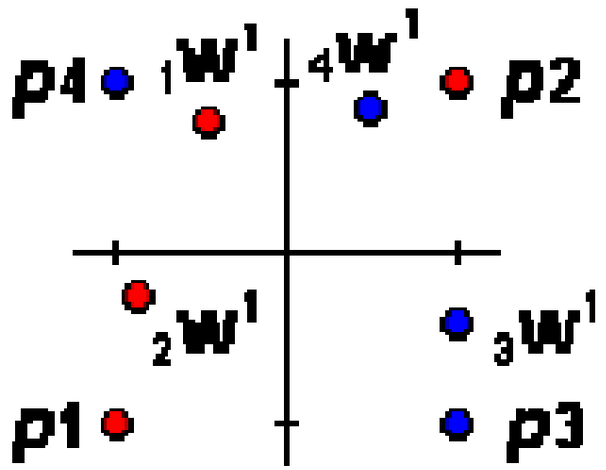

Figura 2.5.21 Estado inicial del vector de peso

- Un vector de entrada diferente será presentado en cada iteración, se encontrará
- su respuesta y luego se actualizarán los pesos correspondientes. Se presentará inicialmente p_3a la red:

$$a^1 = compet(n^1) = compet\left(\begin{bmatrix} -\|{}_1w^1 - p_3\| \\ -\|{}_2w^1 - p_3\| \\ -\|{}_3w^1 - p_3\| \\ -\|{}_4w^1 - p_3\| \end{bmatrix}\right)$$

$$a^1 = compet\left(\begin{bmatrix} -\|[-0.673 \quad 0.970]^T - [1 \quad -1]^T\| \\ -\|[-0.969 \quad -0.379]^T - [1 \quad -1]^T\| \\ -\|[1.002 \quad 0.140]^T - [1 \quad -1]^T\| \\ -\|[0.7805 \quad 0.955]^T - [1 \quad -1]^T\| \end{bmatrix}\right) = compet\left(\begin{bmatrix} -2.5845 \\ -2.0646 \\ -1.1400 \\ -1.9668 \end{bmatrix}\right) = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

- La tercera neurona oculta ha estado más cerca del vector p_3 y de esta forma ya
- se determinó la subclase, ahora determinamos la clase a la cual pertenece multiplicando a^1 por w^2.

$$a^2 = \boldsymbol{W}^2 a^1 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

- La salida de la red índica que p_3 es un miembro de la clase 2, lo cual es correcto por lo tanto ${}_3w^1$ es desplazado en la dirección de p_3.

$${}_3w^1(1) = {}_3w^1(0) + \alpha\left(p_3 - {}_3w^1(0)\right)$$

$${}_3w^1(1) = \begin{bmatrix} 1.002 \\ 0.140 \end{bmatrix} + 0.5\left(\begin{bmatrix} 1 \\ -1 \end{bmatrix} - \begin{bmatrix} 1.002 \\ 0.140 \end{bmatrix}\right) = \begin{bmatrix} 1.001 \\ -0.430 \end{bmatrix}$$

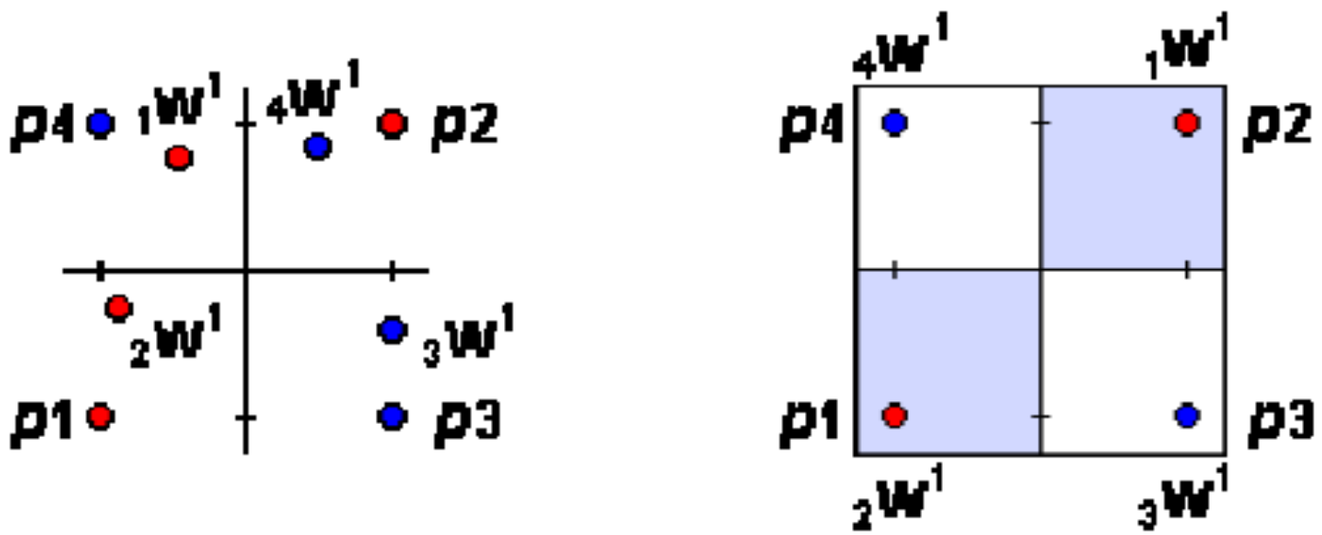

Figura 2.5.22 Resultado después de la primera y después de muchas iteraciones

El diagrama al lado izquierdo de la figura 2.5.22, muestra como el vector peso ${}_3w^1$ es actualizado después de la primera iteración; el diagrama de la derecha, muestra la localización de los pesos después de que el algoritmo ha alcanzado convergencia, además en esta parte de la gráfica puede verse como las regiones del espacio de entrada son clasificadas. Los vectores de entrada p_1 y p_2 perteneciente a la clase uno son visualizadas en azul y los vectores p_3 y p_4 pertenecientes a la clase dos pueden verse en blanco.

- Ejecutar nnd14lv1, nnd14lv2, en Matlab.

Bibliografía

Neural Network Design. [Martin T. Hagan, Howard B, Demuth, Mark Beale – PWS Publishing Company].