2.5 REDES COMPETITIVAS

2.5.1 Antecedentes. En las redes con aprendizaje competitivo (y cooperativo), suele decirse que las neuronas compiten (y cooperan) unas con otras con el fin de llevar a cabo una tarea dada. Con este tipo de aprendizaje se pretende que cuando se presente a la red cierta información de entrada, sólo una de las neuronas de salida de la red, o una por cierto grupo de neuronas, se active (alcance su valor de respuesta máximo). Por tanto las neuronas compiten para activarse quedando finalmente una, o una por grupo, como neurona vencedora y el resto quedan anuladas y siendo forzadas a sus valores de respuesta mínimos.

La competición entre neuronas se realiza en todas las capas de la red, existiendo en estas redes neuronas con conexiones de autoexitación (signo positivo) y conexiones de inhibición (signo negativo) por parte de neuronas vecinas.

El objetivo de este aprendizaje es categorizar (clusterizar) los datos que se introducen en la red, de esta forma las informaciones similares son clasificadas formando parte de la misma categoría y por tanto deben activar la misma neurona de salida. Las clases o categorías deben ser creadas por la propia red, puesto que se trata de un aprendizaje no supervisado a través de las correlaciones entre los datos de entrada.

A principios de 1959, Frank Rosenblatt creó su simple clasificador espontáneo, una red de aprendizaje no supervisado basado en el Perceptrón, el cual aprendía a clasificar vectores de entrada en dos clases con igual número de términos.

A finales de los años 60's y principios de los 70's, Stephen Grossberg introdujo muchas redes competitivas que usaban inhibición lateral obteniendo buenos resultados. Algunos de los comportamientos útiles obtenidos por él, fueron la supresión del ruido, aumento del contraste y normalización de vectores.

En 1973, Christoph Von Der Malsburg introduce la regla del mapa de organización propia, que permitía a la red clasificar entradas en las cuales las neuronas que estuviesen en un vecindario cercano a la neurona ganadora, respondieran a entradas similares. La topología de esta red imitaba de alguna forma las estructuras encontradas en la corteza visual de los gatos, estudiada por David Hubel y Torten Wiesel. Su regla de aprendizaje generó gran interés, pero ésta utilizaba un cálculo no local para garantizar que los pesos fueran normalizados, este hecho hacia este modelo biológicamente poco posible.

Grossberg extendió el trabajo de Von Der Malsburg, redescubriendo la regla Instar. Grossberg mostró que la regla Instar removió la necesidad de renormalizar los pesos, porque los vectores de pesos que aprendían a reconocer vectores de entrada normalizados, automáticamente se normalizarán ellos mismos.

El trabajo de Grossberg y Von Der Malsburg enfatizó la posibilidad biológica de sus redes. Otro exitoso investigador, Tuevo Kohonen ha sido también un fuerte proponente de las redes competitivas, sin embargo su énfasis ha sido en aplicaciones para ingeniería y en descripciones de eficiencia matemática de las redes. Durante la década de los 70 Kohonen desarrolló una versión simplificada de la regla Instar, inspirada también en la red de Von Der Malsburg y Grossberg, de esta forma encontró una manera muy eficiente de incorporar topología a una red competitiva.

Otra forma de aplicar este tipo de aprendizaje fue propuesta por Rumelhart y Zisper [32] en 1985, quienes utilizaban redes multicapa dividiendo cada capa en grupos de neuronas, de tal forma que éstas disponían de conexiones inhibitorias con otras neuronas de su mismo grupo y conexiones excitadoras con las neuronas de la siguiente capa; en una red de este tipo, después de recibir diferentes informaciones de entrada, cada neurona en cada grupo se especializa en la respuesta a determinadas características de los datos de entrada.

En este tipo de redes cada neurona tiene asignado un peso total (suma de todos los pesos de las conexiones que tiene a su entrada), el aprendizaje afecta sólo a las neuronas ganadoras (activas), en las que se redistribuye el peso total entre sus conexiones y se sustrae una porción de los pesos de todas las conexiones que llegan a la neurona vencedora, repartiendo esta cantidad por igual entre todas las conexiones procedentes de unidades activas, por tanto la variación del peso de

una conexión entre una unidad i y otra j será nula si la neurona j no recibe excitación por parte de la neurona i (no vence en presencia de un estímulo por parte de i) y se modificará (se reforzará) si es excitada por dicha neurona.

Una variación del aprendizaje supervisado aplicado a redes multicapa consiste en imponer una inhibición mutua entre neuronas únicamente cuando están a cierta distancia unas de otras (suponiendo que las neuronas se han dispuesto geométricamente, por ejemplo formando capas bidimendisionales), existe entonces un área o región de vecindad alrededor de las neuronas que constituye un grupo local.

Fukushima [11] empleó esta idea en 1975 para una red multicapa llamada Cognitron, fuertemente inspirada en la anatomía y fisiología del sistema visual humano y en 1980 el mismo Fukushima [12] en una versión mejorada de la anterior a la que llamó Necognitron, presentó una variación de esta red utilizando aprendizaje supervisado. El Necognitron disponía de un gran número de capas con arquitectura muy específica de interconexiones entre ellas y era capaz de aprender a diferenciar caracteres, aunque estos se presentasen a diferente escala, en diferente posición o distorsionados.

El aspecto geométrico de la disposición de neuronas de una red, es la base de un caso particular de aprendizaje competitivo introducido por Kohonen en 1982 conocido como feature mapping (mapas de características), aplicado en redes con

una disposición bidimensional de las neuronas de salida, que permiten obtener mapas topológicos o topográficos (Topology Preserving Maps, Topographics Maps, Self Organization Maps) en los que de algún modo estarían representadas las características principales de las informaciones presentadas a la red. De esta forma, si la red recibe informaciones con características similares, se generarían mapas parecidos, puesto que serían afectadas neuronas de salidas próximas entre sí.

2.5.2 Red de Kohonen. Existen evidencias que demuestran que en el cerebro hay neuronas que se organizan en muchas zonas, de forma que las informaciones captadas del entorno a través de los órganos sensoriales se representan internamente en forma de mapas bidimensionales. Por ejemplo, en el sistema visual se han detectado mapas del espacio visual en zonas del córtex (capa externa del cerebro), también en el sistema auditivo se detecta una organización según la frecuencia a la que cada neurona alcanza mayor repuesta (organización tonotópica).

Aunque en gran medida esta organización neuronal está predeterminada genéticamente, es probable que parte de ella se origine mediante el aprendizaje, ésto sugiere que el cerebro podría poseer la capacidad inherente de formar mapas topológicos de las informaciones recibidas del exterior, de hecho esta teoría podría explicar su poder de operar con elementos semánticos: algunas áreas del cerebro simplemente podrían crear y ordenar neuronas especializadas o grupos con

características de alto nivel y sus combinaciones, en definitiva se construirían mapas especiales para atributos y características.

A partir de estas ideas Tuevo Kohonen [24] presentó en 1982 un sistema con un comportamiento semejante, se trataba de un modelo de red neuronal con capacidad para formar mapas de características de manera similar a cómo ocurre en el cerebro; el objetivo de Kohonen era demostrar que un estímulo externo (información de entrada) por sí solo, suponiendo una estructura propia y una descripción funcional del comportamiento de la red, era suficiente para forzar la formación de los mapas.

Este modelo tiene dos variantes denominadas LVQ (Learning Vector Quantization) y TPM (Topology Preserving Map) o SOM (Self Organizing Map), ambas se basan en el principio de formación de mapas topológicos para establecer características comunes entre las informaciones (vectores) de entrada a la red, aunque difieren en las dimensiones de éstos, siendo de una sola dimensión en el caso de LVQ y bidimensional o tridimensional en la red SOM. Estas redes se tratarán con mayor profundidad en secciones posteriores.

El aprendizaje en el modelo de Kohonen es de tipo Off-line, por lo que se distingue una etapa de aprendizaje y otra de funcionamiento. En la etapa de aprendizaje se fijan los valores de las conexiones (feedforward) entre la capa de entrada y la salida. Esta red utiliza un aprendizaje no supervisado de tipo competitivo, las neuronas de la capa de salida compiten por activarse y sólo una de ellas

permanece activa ante una determinada información de entrada a la red, los pesos de las conexiones se ajustan en función de la neurona que haya resultado vencedora.

Durante la etapa de entrenamiento, se presenta a la red un conjunto de informaciones de entrada (vectores de entrenamiento) para que ésta establezca en función de la semejanza entre los datos las diferentes categorías (una por neurona de salida), que servirían durante la fase de funcionamiento para realizar clasificaciones de nuevos datos que se presenten a la red. Los valores finales de los pesos de las conexiones entre cada neurona de la capa de salida con las de entrada se corresponderán con los valores de los componentes del vector de aprendizaje que consigue activar la neurona correspondiente. En el caso de existir más patrones de entrenamiento que neuronas de salida, más de uno deberá asociarse con la misma neurona, es decir pertenecerán a la misma clase.

En este modelo el aprendizaje no concluye después de presentarle una vez todos los patrones de entrada, sino que habrá que repetir el proceso varías veces para refinar el mapa topológico de salida, de tal forma que cuantas más veces se presenten los datos, tanto más se reducirán las zonas de neuronas que se deben activar ante entradas parecidas, consiguiendo que la red pueda realizar una clasificación mas selectiva.

Un concepto muy importante en la red de Kohonen es la zona de vecindad, o vecindario alrededor de la neurona vencedora i^*, los pesos de las neuronas que se encuentren en esta zona a la que se le dará el nombre de $\boldsymbol{X}(q)$, serán actualizados junto con el peso de la neurona ganadora, en un ejemplo de aprendizaje cooperativo.

El algoritmo de aprendizaje utilizado para establecer los valores de los pesos de las conexiones entre las N neuronas de entrada y las M de salida es el siguiente:

1. En primer lugar se inicializan los pesos (w_{ij}) con valores aleatorios pequeños y se fija la zona inicial de vecindad entre las neuronas de salida.
2. A continuación se presenta a la red una información de entrada (la que debe aprender) en forma de vector $\boldsymbol{p} = (p_1, p_2, ..., p_n)$, cuyas componentes p_i serán valores continuos.
3. Puesto que se trata de un aprendizaje competitivo, se determina la neurona vencedora de la capa de salida, ésta será aquella i cuyo vector de pesos $\boldsymbol{w}_i$ (vector cuyas componentes son los valores de los pesos de las conexiones entre esa neurona y cada una de las neuronas de la capa de entrada) sea el más parecido a la información de entrada $\boldsymbol{p}$ (patrón o vector de entrada). Para ello se calculan las distancias o diferencias entre ambos vectores, considerando una por una todas las neuronas de salida, suele utilizarse la

distancia euclídea o la siguiente expresión que es similar a aquella, pero eliminando la raíz cuadrada:

$$d_i = \sum_{j=1}^{N} \left(p_j - w_{i\,j} \right)^2 \qquad 1 \le i \le M \tag{2.5.1}$$

p_j: componente i-ésimo del vector de entrada

w_{ij}: peso de la conexión entre la neurona j de la capa de entrada y la neurona i de la capa de salida.

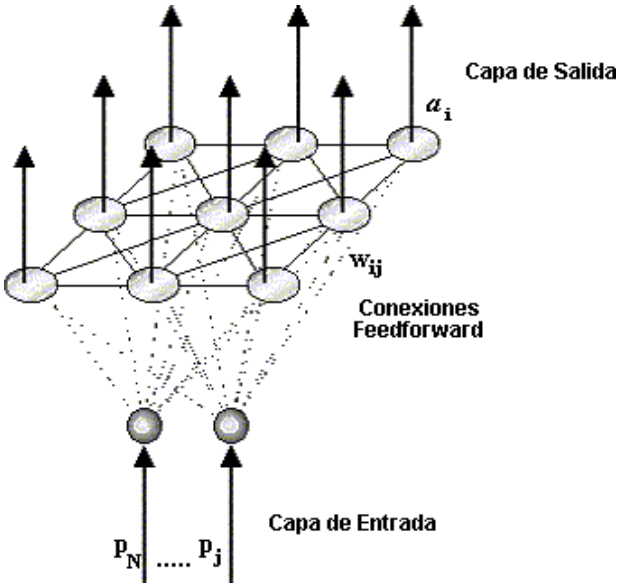

Figura 2.5.1 Conexiones de una red de Kohonen

4. Una vez localizada la neurona vencedora (i^*), se actualizan los pesos de las conexiones entre las neuronas de entrada y dicha neurona, así como los de las conexiones entre las de entrada y las neuronas vecinas de la vencedora, en realidad lo que se consigue con esto es asociar la información de entrada con

una cierta zona de la capa de salida. Esto se realiza mediante la siguiente ecuación

$$w(q)= w(q-1)+a(q)(p(q)-w(q-1)) \quad para\ i \in \boldsymbol{X}(q) \qquad (2.5.2)$$

El tamaño de $\boldsymbol{X}(q)$ se puede reducir en cada iteración del proceso de ajuste de los pesos, con lo que el conjunto de neuronas que pueden considerarse vecinas cada vez es menor como se observa en la figura 2.5.2, sin embargo en la práctica es habitual considerar una zona fija en todo el proceso de entrenamiento de la red.

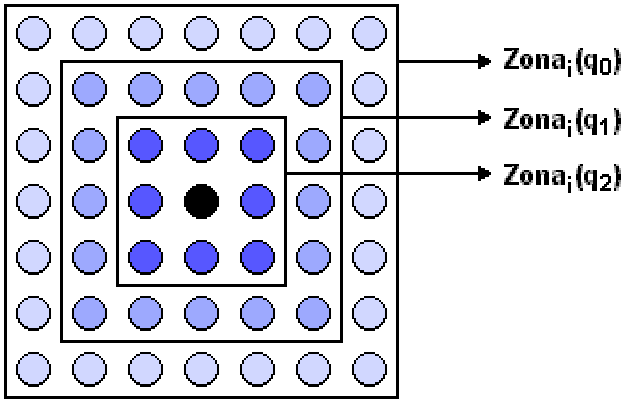

Figura 2.5.2 Posible evolución de la vecindad en una red de Kohonen

El término $\alpha(q)$es el coeficiente de aprendizaje o parámetro de ganancia, con un valor entre 0 y 1 el cual decrece con el número de iteraciones (q) del proceso de entrenamiento, de tal forma que cuando se ha presentado un gran

número de veces todo el juego de patrones de aprendizaje su valor es prácticamente nulo, con lo que la modificación de los pesos es insignificante.

Para hallar α suele utilizarse una de las siguientes expresiones [20]:

$$\alpha(q) = \frac{1}{q} \qquad \alpha(q) = \alpha_1\left(1 - \frac{q}{\alpha_2}\right) \tag{2.5.3}$$

Siendo α_1 un valor de 0.1 ó 0.2 y α_2 un valor próximo al número total de iteraciones del aprendizaje, que por lo general se toma como 10000 para esta red.

5. El proceso debe repetirse, volviendo a presentar todo el juego de patrones de aprendizaje $p_1, p_2..,p_n$ hasta obtener la salida deseada.

Como la regla Instar, la regla de Kohonen habilita a los pesos de una neurona a aprender un vector de entrada y de esta forma resolver aplicaciones de reconocimiento de patrones. A diferencia de la regla Instar, el aprendizaje no es proporcional a la salida de la neurona $a_i(q)$, en lugar de ello el aprendizaje ocurre cuando la neurona i sea miembro del conjunto $\boldsymbol{X}(q)$, si la regla Instar es aplicada a una capa de neuronas cuya función de transferencia solamente retorna valores de 0 o 1 (por ejemplo *hardlim*), la regla de Kohonen es equivalente a la regla Instar.

En definitiva lo que hace una red de Kohonen es realizar una tarea de clasificación, puesto que la neurona de salida activada ante una entrada representa la clase a la que pertenece dicha información de entrada, además ante otra entrada parecida se activa la misma neurona de salida, u otra cercana a la anterior debido a la semejanza entre las clases, así se garantiza que las neuronas topológicamente próximas sean sensibles a entradas físicamente similares; por esta causa la red es especialmente útil para establecer relaciones desconocidas previamente entre conjuntos de datos.

2.5.3 Red de Hamming. La red de Hamming ilustrada en la figura 2.5.3 es uno de los ejemplos más simples de aprendizaje competitivo, a pesar de ello su estructura es un poco compleja ya que emplea el concepto de capas recurrentes en su segunda capa y aunque hoy en día en redes de aprendizaje competitivo se ha simplificado este concepto con el uso de funciones de activación más sencillas, la red de Hamming representa uno de los primeros avances en este tipo de aprendizaje, convirtiéndola en un modelo obligado de referencia dentro de las redes de aprendizaje competitivo. Las neuronas en la capa de salida de esta red compiten unas con otras para determinar la ganadora, la cual índica el patrón prototipo más representativo en la entrada de la red, la competición es implementada por inhibición lateral (un conjunto de conexiones negativas entre las neuronas en la capa de salida).

Esta red consiste en dos capas; la primera capa, la cual es una red Instar, realiza la correlación entre el vector de entrada y los vectores prototipo, la segunda capa

realiza la competición para determinar cuál de los vectores prototipo está más cercano al vector de entrada.

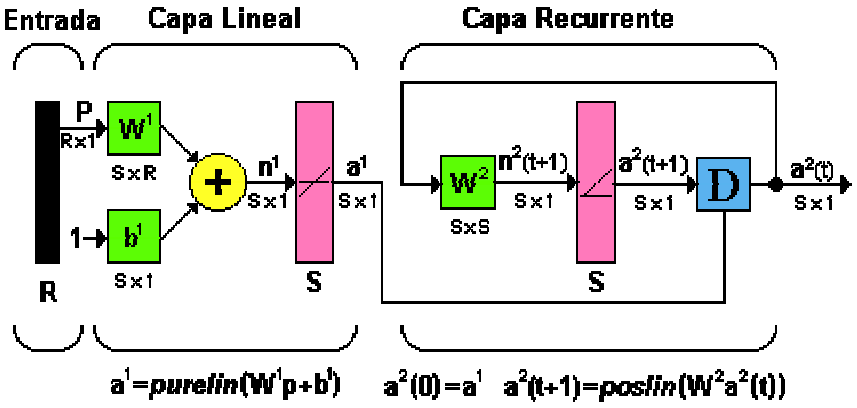

Figura 2.5.3 Red de Hamming

Capa 1:

La red Instar es capaz de clasificar solo un patrón; para que múltiples patrones sean reconocidos se necesitan múltiples Instar y es precisamente de esa forma como está compuesta la primera capa de la red de Hamming. Para una mejor comprensión de su funcionamiento se partirá de unos vectores prototipo que la red debe clasificar

$$\{ p_1, p_2 ,..., p_Q\} \qquad (2.5.4)$$

La matriz de pesos $\boldsymbol{W}^1$ y el vector de ganancias $\boldsymbol{b}^1$ para la capa uno serán:

$$\boldsymbol{W^1} = \begin{bmatrix} {}_1\boldsymbol{W}^T \\ {}_2\boldsymbol{W}^T \\ \vdots \\ {}_S\boldsymbol{W}^T \end{bmatrix} = \begin{bmatrix} \boldsymbol{p}_1^T \\ \boldsymbol{p}_2^T \\ \vdots \\ \boldsymbol{p}_Q^T \end{bmatrix}, \quad \boldsymbol{b}^1 = \begin{bmatrix} R \\ R \\ R \\ R \end{bmatrix} \tag{2.5.5}$$

Donde cada fila de $\boldsymbol{W}^1$ representa un vector prototipo, el cual deseamos reconocer y cada elemento $\boldsymbol{b}^1$ es igual al número de elementos en cada vector de entrada (R) (el número de neuronas S es igual al número de vectores prototipo Q). Así la salida de la primera capa será:

$$\boldsymbol{a^1} = \boldsymbol{W^1 p} + \boldsymbol{b^1} = \begin{bmatrix} \boldsymbol{p}_1^T \boldsymbol{p} + \boldsymbol{R} \\ \boldsymbol{p}_2^T \boldsymbol{p} + \boldsymbol{R} \\ \vdots \\ \boldsymbol{p}_Q^T \boldsymbol{p} + \boldsymbol{R} \end{bmatrix} \tag{2.5.6}$$

La salida de la capa 1 es igual al producto punto de los vectores prototipo con la entrada más el vector $\boldsymbol{R}$; este producto indica cuan cercano está cada vector de entrada a los patrones prototipo.

Capa 2:

La red Instar emplea una función de transferencia *poslin* para decidir si el vector de entrada estaba lo suficientemente cerca al vector prototipo. En la capa 2 de la red de Hamming se utilizan múltiples Instar, así se determinará por medio de una

capa competitiva el patrón prototipo más cercano. Las neuronas en esta capa son inicializadas con la salida de la capa en realimentación, la cual índica la correlación entre los patrones prototipo y el vector de entrada. Las neuronas compiten unas con otras para determinar una ganadora; después de la competición sólo una neurona tendrá una salida no cero. La neurona ganadora índica cual categoría de entrada fue presentada a la red (cada vector prototipo representa una categoría).

La salida de la primera capa a^1 es usada para inicializar la segunda capa:

$$\boldsymbol{a}^2(0) = \boldsymbol{a}^1 \tag{2.5.7}$$

La salida de la segunda capa está determinada de acuerdo a la siguiente relación recurrente:

$$\boldsymbol{a}^2(t+1) = poslin\,(\boldsymbol{W}^2\boldsymbol{a}^2(\mathrm{t})) \tag{2.5.8}$$

Los pesos de la segunda capa $\boldsymbol{W}^2$ son fijados de tal forma que los elementos de la diagonal sean 1, y los elementos por fuera de la diagonal tengan pequeños valores negativos.

$$w_{ij}^2 = \begin{Bmatrix} 1, \ si \quad i = j \\ -\varepsilon \ \ de\,otra \ \ forma \end{Bmatrix} \qquad Donde \ \ 0 < \varepsilon < \frac{1}{s-1} \tag{2.5.9}$$

Esta matriz produce un efecto inhibitorio, en el cual la salida de cada neurona tiene un efecto inhibitorio sobre todas las otras neuronas. Para ilustrar este efecto sustituimos los valores de peros de 1 y $-\varepsilon$ por los apropiados elementos de $\boldsymbol{W}^2$; reescribiendo la ecuación de salida de la red para una sola neurona se tiene :

$$a^2{}_i(t+1) = poslin \left(a_i^2(t) - \varepsilon \sum_{j \neq i} a_j^2(t) \right) \tag{2.5.10}$$

En cada iteración, cada salida de la neurona se decrementará en proporción a la suma de las salidas de las otras neuronas. La salida de la neurona con la condición inicial más grande se decrementará más despacio que las salidas de otras neuronas; eventualmente cada neurona tendrá una salida positiva y en ese punto la red habrá alcanzado el estado estable.

2.5.4 Estructura general de una red competitiva. En las redes asociativas, se vio como la regla instar puede aprender a responder a un cierto grupo de vectores de entrada que están concentrados en una región del espacio. Supóngase que se tienen varias instar agrupadas en una capa, tal como se muestra en la figura 2.5.4, cada una de las cuales responde en forma máxima a un cierto grupo de vectores de entrada de una región distinta del espacio.

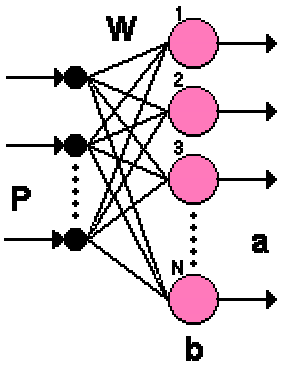

Figura 2.5.4 Instar agrupadas en una capa

Se puede decir que esta capa de Instars clasifica cualquier vector de entrada, porque la Instar con la mayor respuesta para alguna entrada dada es la que identifica a la región del espacio en la cual yace el vector de entrada. En lugar de examinar la respuesta de cada instar para determinar cuál es la mayor, la labor de clasificación sería más fácil si la Instar de mayor respuesta fuera la única unidad que tuviese una salida no nula; esto se puede conseguir si las instar compiten unas con otras por el privilegio de la activación, este es el principio de las redes competitivas.

Las neuronas de la segunda capa de la red de Hamming, están en competición porque cada neurona se excita a sí misma e inhibe a todas las otras neuronas, para simplificar la discusión se definirá una nueva función de transferencia que hace el trabajo de una capa recurrente competitiva

$$\boldsymbol{a = compet\,(n)} \qquad (2.5.11)$$

Donde $\boldsymbol{a}$ es la salida total de la red y $\boldsymbol{n}$ es la entrada neta a la función de transferencia, *Compet* es una función de transferencia que encuentra el índice i^* de la neurona con la entrada neta más grande y fija su salida en uno, todas las otras neuronas tienen salida 0.

$$a_i = \begin{cases} 1, & i = i^* \\ 0, & i \neq i^* \end{cases} \qquad Donde \ \ n_{i^*} \geq n_i, \forall i, \qquad i^* \leq i, \ \forall n_i = n_{i^*} \qquad (2.5.12)$$

Reemplazando la capa recurrente de la red de Hamming, con una función de transferencia competitiva, la presentación de una capa competitiva se simplifica de la siguiente manera.

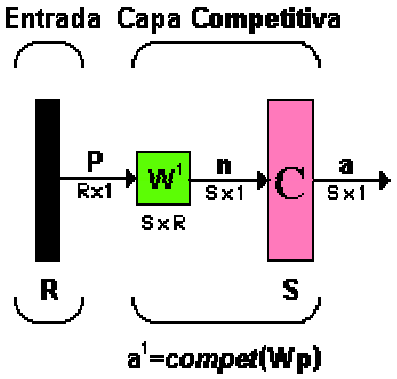

Figura 2.5.5 Capa Competitiva

Como con la red de Hamming, los vectores prototipo son almacenados en las filas de $\boldsymbol{W}$. La entrada neta $\boldsymbol{n}$ calcula la distancia entre el vector de entrada $\boldsymbol{p}$ y cada prototipo $\boldsymbol{w_i}$ (asumiendo que los vectores tiene longitudes normalizadas $\boldsymbol{L}$). La

entrada neta n_i de cada neurona es proporcional al ángulo θ_i entre $\boldsymbol{p}$ y el vector prototipo $\boldsymbol{w_i}$:

$$\boldsymbol{n = Wp} = \begin{bmatrix} {}_1\boldsymbol{w}^T \\ {}_2\boldsymbol{w}^T \\ {}_S\boldsymbol{w}^T \end{bmatrix} \boldsymbol{p} = \begin{bmatrix} {}_1\boldsymbol{w}^T\,\boldsymbol{p} \\ {}_2\boldsymbol{w}^T\,\boldsymbol{p} \\ {}_S\boldsymbol{w}^T\,\boldsymbol{p} \end{bmatrix} = \begin{bmatrix} \boldsymbol{L^2 cos} & \theta_1 \\ \boldsymbol{L^2 cos} & \theta_2 \\ \boldsymbol{L^2 cos} & \theta_S \end{bmatrix} \qquad (2.5.13)$$

La función de transferencia competitiva asigna una salida de 1 a la neurona cuyo vector de pesos apunte en la dirección más cercana al vector de entrada

$$\boldsymbol{a = compet\,(n)} \qquad (2.5.14)$$

2.5.5 Regla de aprendizaje. En este punto es posible diseñar una red competitiva que realice clasificaciones correctas fijando el valor de las filas de $\boldsymbol{W}$ en los valores del vector prototipo esperado, sin embargo es deseable tener una regla de aprendizaje que pueda entrenar los pesos en una red competitiva sin conocer los vectores prototipo, una de estas reglas es la Instar estudiada es el numeral 2.4.3

$$_1w(q) = {}_1w(q\text{-}1) + a(q)(p(q) - {}_1w(q\text{-}1)) \qquad (2.5.15)$$

Para redes competitivas a tiene un valor diferente de cero solamente para la neurona ganadora ($i=i^*$), de esta forma los mismos resultados serán obtenidos utilizando la regla de Kohonen,

$$_i\boldsymbol{w}(q) = {}_i\boldsymbol{w}(q\text{-}1) + \alpha\,(\boldsymbol{p}(q) - {}_i\boldsymbol{w}(q\text{-}1)) = (1\text{-}\alpha)\,{}_i\boldsymbol{w}\,(q\text{-}1) + \alpha\,\boldsymbol{p}(q) \qquad (2.5.16)$$

y

$$w(q)=w(q\text{-}1) \quad i \neq i^* \qquad (2.5.17)$$

Así, la fila de la matriz de pesos que esté más cerca al vector de entrada (o tenga el producto punto más grande con el vector de entrada) se moverá hacía el vector de entrada. Este se mueve a lo largo de la línea entre la fila anterior del vector de pesos y el vector de entrada, como puede verse en la figura 2.5.6

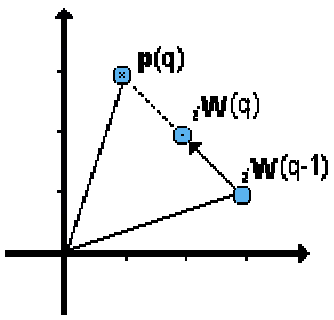

Figura 2.5.6 Representación gráfica de la regla de Kohonen

Para demostrar como trabaja una red competitiva, se creará una red que clasifique los siguientes vectores:

$$p_1 = \begin{bmatrix} -0.216 \\ 0.993 \end{bmatrix}, p_2 = \begin{bmatrix} 0.216 \\ 0.993 \end{bmatrix}, \quad p_3 = \begin{bmatrix} 0.993 \\ 0.216 \end{bmatrix}$$

$$p_4 = \begin{bmatrix} 0.993 \\ -0.216 \end{bmatrix}, p_5 = \begin{bmatrix} -0.622 \\ -0.873 \end{bmatrix}, p_6 = \begin{bmatrix} -0.873 \\ -0.622 \end{bmatrix}$$

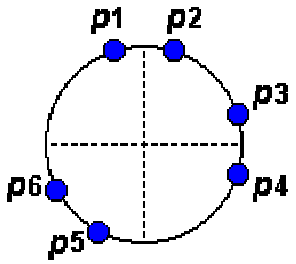

Figura 2.5.7 Vectores de entrada

La red tendrá tres neuronas, por lo tanto los vectores serán clasificados en tres clases o grupos, ésta es una de las principales características de las redes competitivas, ellas pueden agrupar los patrones de entrada en clases que no se conocen. Los pesos normalizados escogidos aleatoriamente son:

$${}_1w = \begin{bmatrix} 0.7071 \\ -0.7071 \end{bmatrix}, \; {}_2w = \begin{bmatrix} 0.7071 \\ 0.7071 \end{bmatrix}, \; {}_3w = \begin{bmatrix} -1.000 \\ 0.000 \end{bmatrix}, \; \boldsymbol{W} = \begin{bmatrix} {}_1\boldsymbol{w}^T \\ {}_2\boldsymbol{w}^T \\ {}_3\boldsymbol{w}^T \end{bmatrix}$$

Los vectores de datos y los pesos asignados pueden visualizarse en la figura 2.5.8

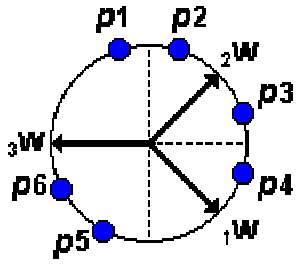

Figura 2.5.8 Vectores de entrada y vector de pesos

Se presenta a la red el vector p_2

$$\boldsymbol{a} = compet(\boldsymbol{Wp_2}) = compet\left(\begin{bmatrix} 0.7071 & -0.7071 \\ 0.7071 & 0.7071 \\ -1.000 & 0.000 \end{bmatrix}\begin{bmatrix} 0.216 \\ 0.993 \end{bmatrix}\right)$$

$$\boldsymbol{a} = compet\left(\begin{bmatrix} -0.5494 \\ 0.8549 \\ -0.2160 \end{bmatrix}\right) = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

El vector de peso de la segunda neurona estaba más cercano a $p_{2,}$ por lo tanto ganó la competición (i^*=2) y su salida es 1. Ahora se aplicará la regla de Kohonen a la neurona ganadora con una rata de aprendizaje $\alpha = 0.5$

$$_2\boldsymbol{w}^{\,nuevo} = {}_2\boldsymbol{w}^{\,anterior} + \alpha\,(\boldsymbol{p}_2 - {}_2\boldsymbol{w}^{\,anterior})$$

$$_2\boldsymbol{w}^{nuevo} = \begin{bmatrix} 0.7071 \\ 0.7071 \end{bmatrix} + 0.5\left(\begin{bmatrix} 0.216 \\ 0.993 \end{bmatrix} - \begin{bmatrix} 0.7071 \\ 0.7071 \end{bmatrix}\right) = \begin{bmatrix} 0.9527 \\ 0.5641 \end{bmatrix}$$

La regla de Kohonen hace que ${}_2w$ tienda hacia p_2 como puede verse en la figura 2.5.9, si continuamos escogiendo vectores de entrada aleatoriamente y presentándoselos a la red, en cada iteración el vector de pesos se acercará más al vector de entrada.

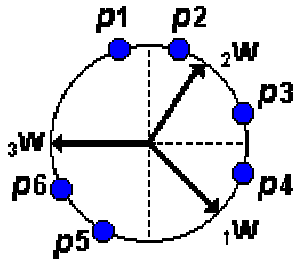

Figura 2.5.9 Proceso de entrenamiento

Cada vector de pesos apuntará hacia una clase diferente del vector de entrada, convirtiéndose en un prototipo para esa clase. Cuando termine el proceso de entrenamiento, los pesos finales se verán como aparece en la figura 2.5.10

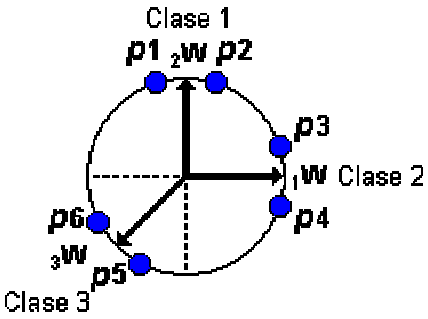

Figura 2.5.10 Pesos Finales

2.5.6 Problemas de las redes Competitivas. Las redes competitivas, son bastante eficientes para resolver problemas de clasificación, sin embargo

presentan algunos problemas. El primero es la elección de una rata de aprendizaje que permita hallar un punto de equilibrio entre velocidad de convergencia y la estabilidad final de los vectores de peso. Una rata de aprendizaje cercana a cero, torna el aprendizaje muy lento pero garantiza que cuando un vector haya alcanzado el centro de la clase objetivo, se mantendrá allí indefinidamente. En contraste, una rata de aprendizaje cercana a uno genera un aprendizaje muy rápido, pero los vectores de peso continuarán oscilando aún después de que se haya alcanzado convergencia. La indecisión que se presenta al escoger la rata de aprendizaje puede ser empleada como una ventaja si se inicia el entrenamiento con una rata de aprendizaje alta y se decrementa en el transcurso del proceso de entrenamiento cuando sea necesario, desafortunadamente esta técnica no funciona si la red necesita continuamente ser adaptada a nuevos argumentos de los vectores de entrada (caso en que la red se trabaje 0n-line). Un ejemplo de este problema se visualiza en la figura 2.5.11

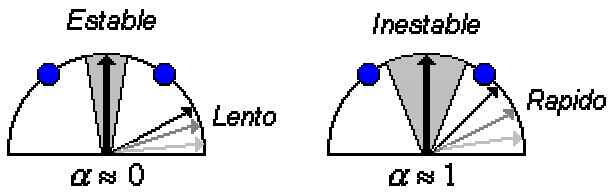

Figura 2.5.11 Variación de la rata de aprendizaje

Un problema de estabilidad más serio, ocurre cuando las clases están muy juntas; en ciertos casos, un vector de pesos tratando de apuntar hacia una clase determinada, puede entrar al territorio de otro vector de pesos. En la figura 2.5.12,

pueden observarse con círculos azules, como dos vectores de entrada son presentados repetidas veces a la red; el resultado, es que los vectores de pesos que representan las clases de la mitad y de la derecha se encuentran a la derecha. Con seguridad, se presentará el caso en que una de las clases de la derecha será clasificada por el vector de pesos del centro

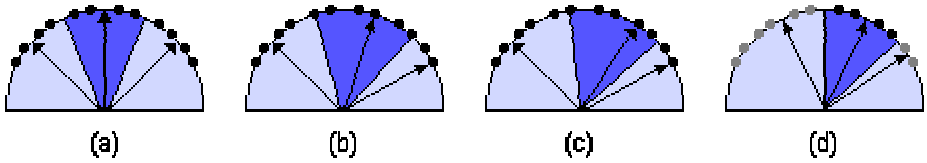

Figura 2.5.12 Aprendizaje Inestable

Un tercer problema con redes competitivas, es que es posible que el vector de pesos inicial de una neurona se encuentre muy lejos de cualquiera de los vectores de entrada y por lo tanto nunca gane la competición. La consecuencia será, la "muerte" de la neurona, lo que por supuesto no es recomendable.

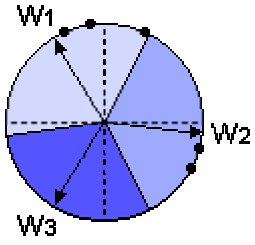

Figura 2.5.13 Causa de la muerte de una neurona

En la figura 2.5.13 el vector de peso w_3 nunca ganará la competición, sin importar cual sea el orden en que se le presenten los vectores de entrada. Una solución a

este problema, consiste en adicionar una ganancia negativa a la entrada neta de cada neurona y decrementar así la ganancia total cada vez que la neurona gane la competición; esto hará que difícilmente una neurona gane varias veces la competición, a este mecanismo se le llama “conciencia”.

Una capa competitiva tiene tantas clases como neuronas, lo que podría complicar algunas aplicaciones, especialmente cuando el número de clases no se conoce de antemano. En capas competitivas, cada clase consiste de una región convexa del espacio de entrada, las capas competitivas no pueden formar clases con regiones no convexas o clases que sean la unión de regiones no conectadas.

2.5.7 Mapas de auto organización (SOM). 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 mejicano debido a su forma.

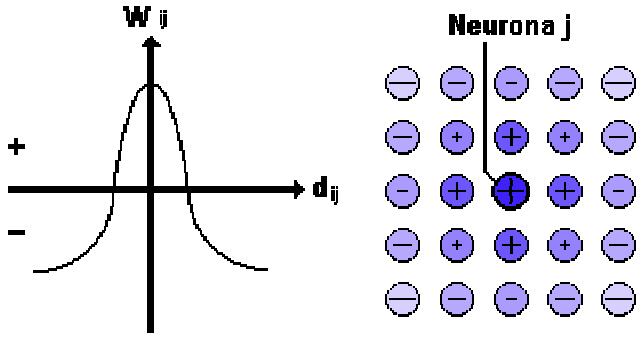

Figura 2.5.14 *on-center/off-surround*; para capas biológicas

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). 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

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

Donde el vecindario N_i^* contiene el índice para todas las neuronas que se encuentren a un radio "d" de la neurona ganadora i^*

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

Cuando un vector $\boldsymbol{p}$ es presentado, los pesos de la neurona ganadora y de sus vecinas tenderán hacia $\boldsymbol{p}$, el resultado es que después de muchas presentaciones las neuronas vecinas habrán aprendido 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.

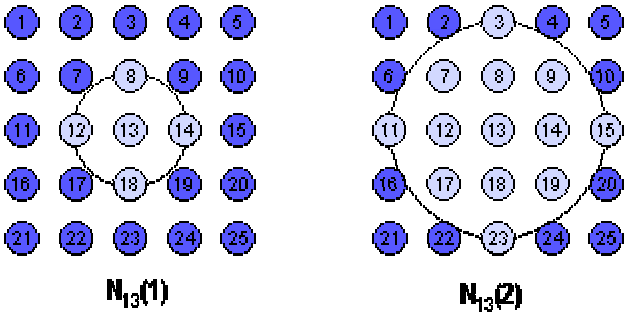

Figura 2.5.15 Vecindarios

Estos vecindarios pueden definirse como sigue:

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

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

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

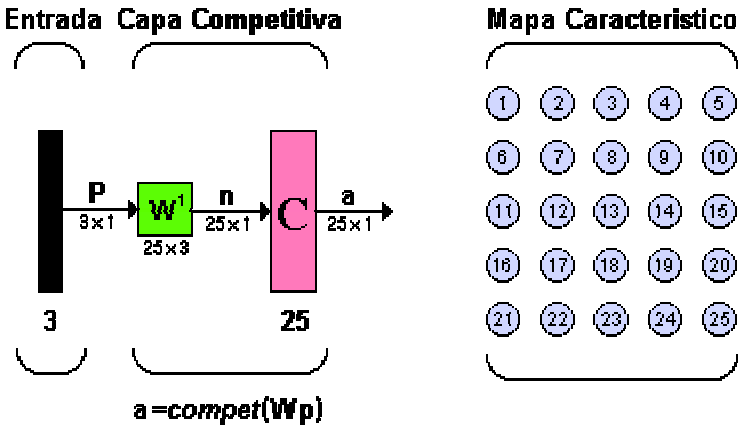

Figura 2.5.16 Mapa de auto organización

2.5.8 Learning Vector Quantization (LVQ). Esta red es un híbrido que emplea tanto aprendizaje no supervisado, como aprendizaje supervisado para clasificación de patrones

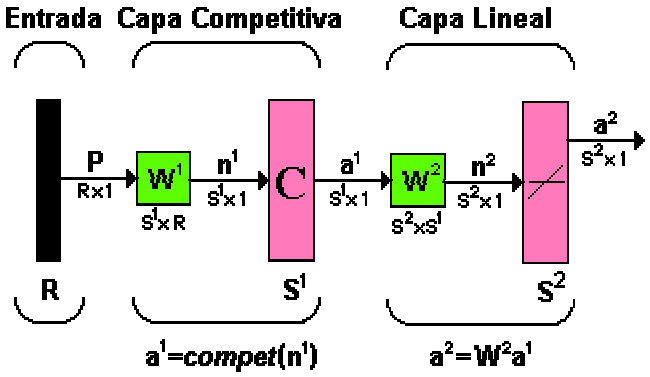

Figura 2.5.17 Red LVQ

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^1 debe 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,

$$\boldsymbol{n}_i^1 = -\begin{bmatrix} \left\| {}_1\boldsymbol{w}^1 - \boldsymbol{p} \right\| \\ \left\| {}_2\boldsymbol{w}^1 - \boldsymbol{p} \right\| \\ \vdots \\ \left\| {}_{S^1}\boldsymbol{w}^1 - \boldsymbol{p} \right\| \end{bmatrix} \qquad (2.5.21)$$

La salida de la primera capa de la red LVQ es,

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

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 *sub-clase*, y de esta forma muchas neuronas (subclases), conforman una clase.

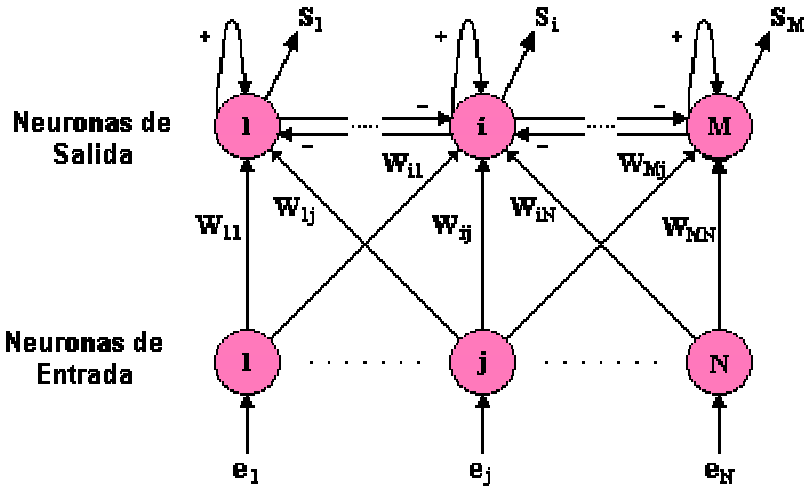

Figura 2.5.18 Comportamiento de las neuronas en una red LVQ

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.

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

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:

$$\left\{ \boldsymbol{p}_1 = \begin{bmatrix} \sqrt{0.74} \\ 0 \\ \sqrt{0.74} \end{bmatrix}, \quad \boldsymbol{t}_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \right\} \qquad (2.5.25)$$

Antes de que suceda el aprendizaje, cada neurona en la segunda capa es asignada a una neurona de salida, así se genera la matriz $\boldsymbol{W}^2$; 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 $\boldsymbol{W}^2$ son cero excepto los que cumplan la siguiente condición:

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

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

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

1. En cada iteración, un vector de entrada $\boldsymbol{p}$ es presentado a la red y se calcula la distancia a cada vector prototipo.
2. Las neuronas ocultas compiten, la neurona i^* gana la competición y el i^*-ésimo elemento de $\boldsymbol{a}^1$ se fija en 1.
3. $\boldsymbol{a}^1$ 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 $\boldsymbol{p}$ 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 $\boldsymbol{p}$ es clasificado correctamente los pesos de la neurona ganadora ${}_{i^*}w^1$ se hacen tender hacia $\boldsymbol{p}$.

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

Segundo, si $\boldsymbol{p}$ es clasificado incorrectamente una neurona equivocada ganó la competición y por lo tanto sus pesos ${}_{i*}w^1$ se alejan de $\boldsymbol{p}$.

$$ {}_{i*}\boldsymbol{w}(q) = {}_{i*}\boldsymbol{w}(q-1) - a(q)\,(\boldsymbol{p}\,(q) - {}_{i*}\boldsymbol{w}(q-1)) \;\; si \; a^2{}_{k*} = 1 \;\neq\; t_{k*} = 0 \qquad (2.5.28) $$

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:

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

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:

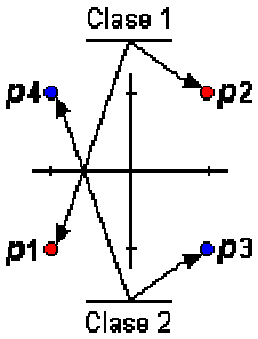

Figura 2.5.19 Posición de los patrones de entrada

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:

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

$\boldsymbol{W^2}$ 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.

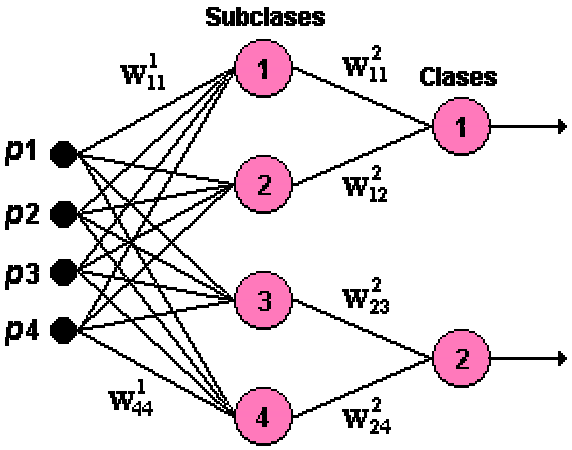

Figura 2.5.20 Esquema de la red LVQ que solucionará el ejemplo

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

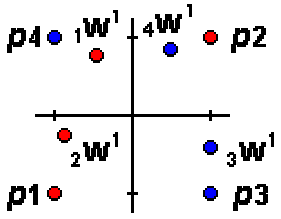

Figura 2.5.21 Estado inicial del vector de peso

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_3 a la red:

$$\boldsymbol{a^1} = compet(\boldsymbol{n^1}) = compet\left(\begin{bmatrix} -\left\| {}_1\boldsymbol{w^1} - \boldsymbol{p_3} \right\| \\ -\left\| {}_2\boldsymbol{w^1} - \boldsymbol{p_3} \right\| \\ -\left\| {}_3\boldsymbol{w^1} - \boldsymbol{p_3} \right\| \\ -\left\| {}_4\boldsymbol{w^1} - \boldsymbol{p_3} \right\| \end{bmatrix}\right)$$

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

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 $\boldsymbol{W}^2$

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

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.

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

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

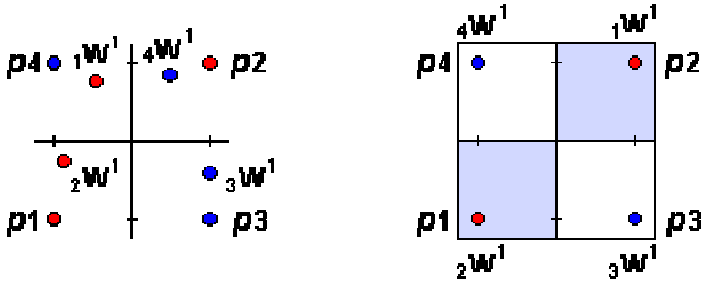

Figura 2.5.22 Resultado después de la primera y después de muchas iteraciones

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 $\boldsymbol{p}_1$ y $\boldsymbol{p}_2$ perteneciente a la clase uno son visualizadas en azul y los vectores $\boldsymbol{p}_3$ y $\boldsymbol{p}_4$ pertenecientes a la clase dos pueden verse en blanco.