Capítulo 5

MAPAS AUTO-ORGANIZADOS DE KOHONEN

Introducción

Los Mapas Auto-organizados (SOM por su nombre en inglés Self-Organizing Maps) fueron presentados por Teuvo Kohonen en 1982, por lo que también reciben el nombre de Mapas Auto-organizados de Kohonen o Redes Neuronales de Kohonen, estos mapas están inspirados en la capacidad del cerebro humano de reconocer y extraer rasgos y características relevantes del mundo que los rodea. En la década de los ochenta, el profesor Kohonen propuso una red neuronal artificial capaz de aprender la estructura topológica de un conjunto de datos a través de un proceso de autoorganización de las neuronas de la red.

Desde el punto de vista bio-inspirado, esta red posee un nivel de aprendizaje de mayor evolución que el supervisado de los capítulos previos, puesto que la red, de alguna manera, es capaz de encontrar de manera autónoma la estructura de los datos, lo que nos facilita descubrir la información relevante de estos datos sin necesidad de entregarle pares ordenados donde se define una salida correspondiente a su entrada, como en el caso del aprendizaje supervisado de las redes MLP. Por primera vez en este libro nos encontramos con una red neuronal cuyo aprendizaje es No–Supervisado.

Teuvo Kohonen
Profesor Emérito de la Academia de Finlandia, en 1982 propuso los denominados Mapas Auto-organizados (SOM).
Los mapas auto-organizados se basan en la propiedad que tiene el cerebro para formar grupos de neuronas que procesan información del mismo tipo.
Los mapas auto-organizados se han aplicado a problemas diversos desde la minería de datos, hasta el procesamiento de imágenes.

La idea básica del SOM es crear una imagen de un espacio multidimensional de entrada en un espacio de salida de menor dimensión. Se trata de un modelo con dos capas de neuronas, como observamos en la figura 5.1, la primera capa es de entrada y la segunda de procesamiento. Las neuronas de la capa de entrada se limitan a recoger y canalizar la información. La capa de salida o procesamiento, está ligada a la capa de entrada a través de los pesos sinápticos de las conexiones; su principal tarea es la de realizar una proyección del espacio *n-dimensional* de entrada en una espacio *m-dimensional* de salida, conservando las características esenciales de los datos gracias a la relación de vecindad que se establece en las neuronas de la capa de salida. Al final, obtenemos un mapa autoorganizado que nos muestra un conjunto de sectores que agrupa los datos con características comunes.

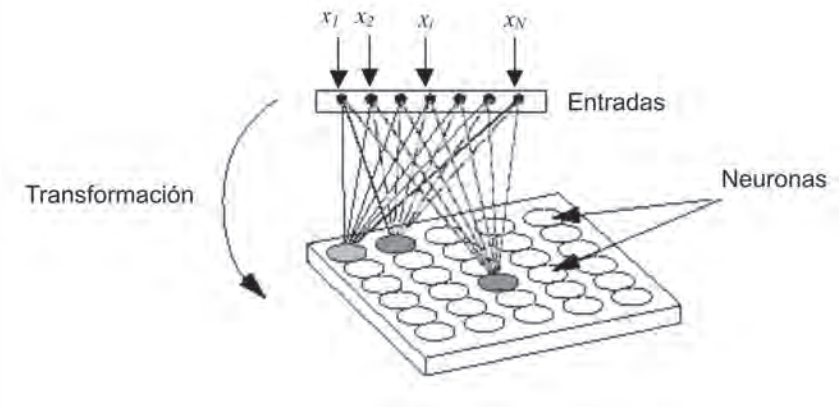

Figura 5.1 Mapa Auto Organizado

En otros tipos de redes neuronales no se tienen en cuenta las relaciones topológicas o geométricas que hay entre los nodos, y no es relevante establecer algún nivel de vecindad entre neuronas cercanas de una misma capa para establecer una relación bilateral entre ellas. En el SOM este aspecto de cercanía si lo vamos a considerar, es importante en la estructura de la red, y será fundamental en la manera como van a interactuar las neuronas.

Veremos que no va a existir la misma relación entre dos nodos cercanos y dos que están muy alejados el uno del otro, así que el objetivo será entrenar la red de manera que las salidas cercanas correspondan a entradas cercanas. En la mayor parte de los casos utilizaremos la distancia euclideana para tener una medida del nivel de relación que hay entre estos nodos.

Se puede decir que el SOM es una clase especial y única de red neuronal, ya que construye un mapa que auto-organiza la topología de las neuronas de un espacio multidimensional, de manera que las distancias entre los datos se conservan.

Estos mapas clasifican automáticamente la información de entrada permitiendo organizarla, de esta manera se pueden observar relaciones importantes entre los datos y descubrir otras relaciones subyacentes entre ellos. El SOM sirve entonces como una herramienta de agrupación de datos de dimensiones altas y es fácil de visualizar debido a su forma (usualmente) de dos dimensiones.

El modelo bioinspirado de Kohonen

El cerebro está organizado en zonas especializadas en procesar la diversa información que le llega. Si aceptamos una simplificación en los conceptos biológicos, podemos decir que en el cerebro hay una distribución predefinida de sus neuronas, orientadas a procesar información específica. De todas maneras, esto no significa que dichas neuronas no puedan procesar información de otro tipo, por ejemplo, cuando una persona pierde uno de sus sentidos. Coloquialmente se dice que un invidente desarrolla otros sentidos, pero en realidad lo que sucede es que en el cerebro ocurre una reorganización de la manera como se procesa la información sensorial proveniente de los sentidos que funcionan para suplir los datos faltantes por la visión, en este caso.

El cerebro se especializa por regiones para procesar la información proveniente de las diferentes partes del cuerpo humano. A estas regiones se les conoce como mapas sensoriales y están conformados por un conjunto de neuronas asociadas entre sí; como ejemplo podemos mencionar los siguientes:

- Mapa Somatosensorial que genera una imagen en el cerebro de la piel del cuerpo humano.
- Mapa Retinotópico que procesa la información proveniente de la visión.
- Mapa Tonotópico que procesa la información proveniente del oído.

Aunque genéticamente venimos predispuestos para que estos mapas realicen tareas como la de diferenciar sonidos, para el caso del mapa tonotópico, el aprendizaje tiene un papel fundamental en la especialización de estas zonas. Por ejemplo, cuando escuchamos una orquesta sinfónica, en general somos capaces de diferenciar los ins-trumentos de cuerda de los de viento, ninguno de nosotros confundiría el sonido de una trompeta con el de un violín; pero, ¿qué ocurre cuando se interpretan instrumentos similares, por ejemplo, un saxo tenor y un saxo barítono? Tal vez, ya no seamos capaces de diferenciarlos. Sin embargo, un estudioso de la música, a través de muchos años de entrenamiento logra no sólo diferenciarlos, sino separar las notas musicales que interpretan e identificarlas como un DO, un RE o un MI, para citar algunas.

Con esto lo que queremos mostrar es como un adecuado proceso de aprendizaje ayuda perfeccionar el trabajo realizado por estas zonas del cerebro que hemos denominado Mapas Sensoriales

En síntesis, podemos afirmar que el cerebro humano posee la capacidad inherente de formar mapas topológicos a partir de la información del mundo que lo rodea, por lo que Kohonen presentó un modelo con capacidad para formar mapas de características, cuyo objetivo es mostrar que un estímulo exterior (información de entrada), por sí sólo, suponiendo una estructura propia y una descripción funcional del comportamiento de la red, es suficiente para forzar la formación de mapas.

Arquitectura de la red

El mapa auto-organizado de Kohonen está constituido por dos niveles de neuronas, el de entrada y el de salida. Pero sólo en el nivel de salida se genera un procesamiento de información, por lo que recibe el nombre de Capa de Salida y la red pertenece al tipo Monocapa. La conectividad es total, es decir, todas las neuronas de la capa de salida reciben los estímulos de las neuronas de entrada.

Las neuronas de entrada reciben la información de los datos provenientes del exterior de la red neuronal y a través de las conexiones sinápticas envían esta información a la capa de procesamiento de la red.

Kohonen propuso inicialmente una red con una capa de salida, figura 5.2, donde las neuronas están organizadas en una fila similar a un vector, a esta estructura la denominó LVQ (Linear Vector Quantization). A través de un vector de entrada, presentamos a la red un estímulo, que es modificado por los pesos sinápticos de las conexiones y se le presentan a la capa de neuronas de procesamiento, resultando un vector de salida $\boldsymbol{y}$.

Como veremos más adelante, las neuronas compiten entre sí para activarse, ganando aquella cuyo vector de pesos sinápticos asociado, esté más

cercano al vector de entrada, es decir, se mide la distancia entre el vector de entrada y los vectores de pesos sinápticos asociados a las ***m*** neuronas de salida. Aquella neurona cuyo vector de pesos diste menos del vector de entrada, será la ganadora y modificará su vector de pesos con el fin de representar ese dato de entrada.

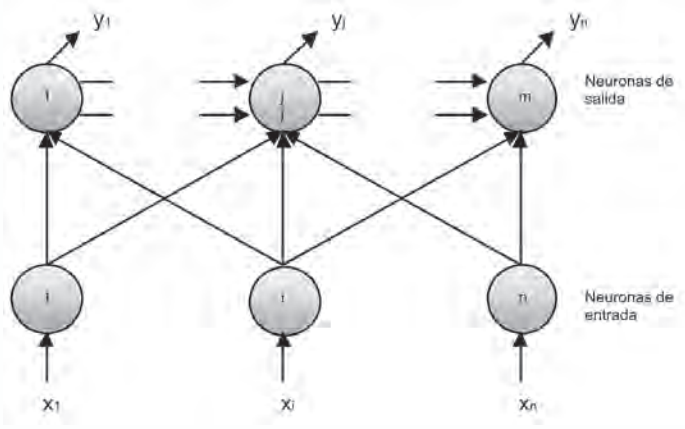

Fig. 5.2 Arquitectura de una red simple de Kohonen

La arquitectura del mapa bidimensional SOM de la figura 5.3 es similar a la del LVQ, pero las neuronas de salida se organizan en forma de matriz. A las neuronas organizadas de esta manera es lo que conocemos como el mapa de la red. En general, este tipo de red permite tomar patrones de entrada de dimensión ***n*** y visualizar los grupos que hay en los mismos usando una estructura bidimensional o de mapa.

La red neuronal tipo SOM tiene *N* neuronas en el nivel de entrada, *M* neuronas en la capa de salida, se establece una conectividad desde la capa de entrada hacia la capa de salida (*Feedforward*) y en la fase de entrenamiento, se establece en cada neurona una influencia de las neuronas vecinas, lo que se conoce como vecindad.

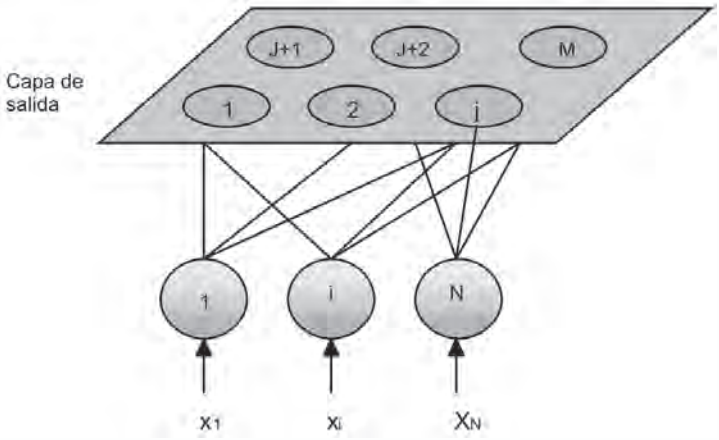

Fig. 5.3 Arquitectura de una red en dos dimensiones

Cuando Kohonen planteó su modelo de mapas auto-organizados, también tuvo en cuenta que en el cerebro, no sólo se activa ante un estímulo, una única neurona, sino que lo hace toda una región; propiedad que garantiza la robustez de nuestro cerebro y la respuesta adecuada frente a un estímulo aunque éste posea niveles de incertidumbre. Pensemos, en cómo es nuestra voz al iniciar la mañana, a media mañana o en la noche… muy seguramente es diferente, ¿coincidimos en esta apreciación? Pero de todas maneras las personas que nos conocen bien, nos reconocen, pese a que nuestra voz tiene diversos matices. Pero existe un cambio mucho más pronunciado, y es cuando hablamos por teléfono: Muchas veces nos ha pasado que a personas que conocemos poco no las reconocemos cuando nos hablan por teléfono, pero a nuestros seres queridos, los identificamos sin ningún tipo de duda. Justamente ésta es una capacidad de nuestro cerebro de responder adecuadamente a estímulos con niveles de incertidumbre y se debe, en parte, a que no hay una única neurona que responde al estímulo o que haya aprendido a reconocer un único matiz de voz, sino que hay toda una región que lo hace y a esto es lo que intentaremos emular con una Función de Vecindad en el proceso de aprendizaje.

En la figura 5.4, observamos la función de vecindad, en los espacios bidimensional y tridimensional. Se puede observar que el efecto de la vecindad es mayor mientras más cerca esté de la neurona ganadora, que está ubicada en el centro de la función de vecindad.

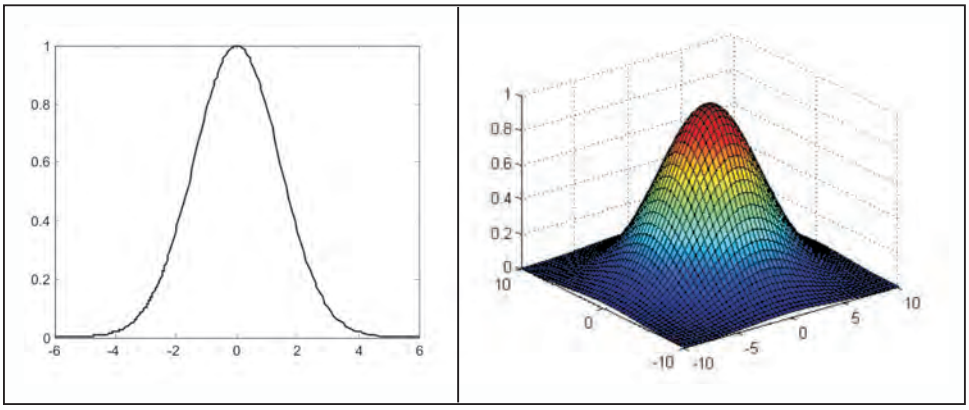

Fig. 5.4 Ejemplos de Funciones de Vecindad para Mapas Auto-Organizados

Algoritmo de aprendizaje

Consideraciones iniciales

Antes de llevar a cabo el proceso de aprendizaje de la red, debemos definir los parámetros que caracterizan la arquitectura de la red, es decir, asignar el número de neuronas (N) del nivel de entrada y la dimensión la capa de salida de la red (M). El número de neuronas de entrada es igual a la dimensión del vector de características que define el problema que se va a solucionar con la red. La capa de salida conforma el Mapa Auto-Organizado de Kohonen y, en general, las M neuronas del mapa se distribuyen en una matriz de dimensión M_f x M_c, donde M_f es el número de filas del mapa y M_c el número de columnas.

El siguiente paso en el proceso de entrenamiento es inicializar los pesos de la red. Estos pesos generalmente los inicializamos de manera aleatoria en un rango comprendido entre cero y uno si los datos de entrenamiento han sido normalizados. Si los datos de entrenamiento no han sido normalizados, los pesos de la red se pueden inicializar aleatoriamente alrededor de la media de dicho conjunto de datos, aunque nuestra recomendación es que los datos estén normalizados.

Modelo matemático

El aprendizaje en el modelo auto-organizado de Kohonen está regido por la ecuación 5.1 que define la variación de los pesos, $\Delta \boldsymbol{w_r}$, en este algoritmo. En donde la neurona ganadora y sus vecinas, modifican su vector de pesos sumándole una fracción de la distancia existente entre el vector de entrada y el vector de pesos en el instante t del algoritmo.

$$\Delta \mathbf{w}_r = \alpha(t) h_{rs}(t)(\mathbf{x} - \mathbf{w}_r) \tag{5.1}$$

donde,

$\boldsymbol{x}$	: Vector de Entrada
$\Delta \boldsymbol{w}_r$	: Variación del vector de pesos para la neurona *r-ésima*
α	: Rata de aprendizaje
$\boldsymbol{h}_{rs}$	: Función de Vecindad
$\boldsymbol{w}_r$	: Vector de pesos de la neurona *r-ésima*
$\boldsymbol{t}$	: Índice de iteración

La rata de aprendizaje $\alpha(t)$ se calcula con base en la ecuación 5.2, donde α_i y α_f las corresponden a las ratas de aprendizaje inicial y final, respectivamente; $\boldsymbol{t}_{max}$, es el número máximo de iteraciones. Con esta expresión lo que buscamos es que la rata de aprendizaje siga una forma exponencial, figura 5.5., con el fin de obtener al inicio del proceso fuertes variaciones en los pesos y, a medida que evolucione el algoritmo, el tamaño de los cambios de los pesos se atenúe para garantizar que al iniciar el algoritmo las neuronas se distribuyan rápidamente entre los datos más representativos de la base de entrenamiento y al finalizar, cuando las neuronas ya hayan aprendido la distribución de los datos, las modificaciones de los pesos sean más pequeñas con el fin de solo llevar a cabo un ajuste fino de los pesos.

$$\alpha(t) = \alpha_i \left(\frac{\alpha_f}{\alpha_i} \right)^{\frac{t}{t_{max}}} \tag{5.2}$$

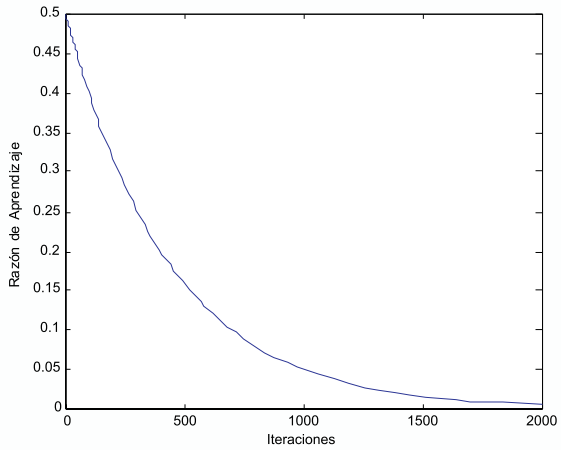

Fig. 5.5 Variación de la Razón de Aprendizaje α(t)

La función de vecindad se define con la ecuación 5.3, donde *d* es la distancia euclideana entre la neurona ganadora *s* y la neurona *r* a la cual se le modifican los pesos. El rango de vecindad $\sigma(t)$ es variable y se define con la ecuación 5.4, donde σ_i y σ_f corresponden a los rangos de vecindad inicial y final, respectivamente.

$$h_{rs}(t) = e^{\left(\frac{-d\ (r,s)^2}{2\sigma(t)^2}\right)} \tag{5.3}$$

$$\sigma(t) = \sigma_i \left(\frac{\sigma_f}{\sigma_i}\right)^{\frac{t}{t_{max}}} \tag{5.4}$$

La vecindad es una función exponencial cuya característica hace que las neuronas más alejadas de la unidad ganadora, se vean afectadas en sus pesos sinápticos en una menor proporción que las más cercanas; es importante resaltar como podemos modificar en el proceso de entrenamiento la magnitud de la región de vecindad a través del parámetro $\sigma(t)$. De hecho, cuando iniciamos el entrenamiento de la red, definimos este parámetro con un valor grande, el cual se va disminuyendo a medida que la red progresa en su aprendizaje, lo cual nos garantiza estabilidad en la convergencia del algoritmo de aprendizaje de la red.

Ejemplo

Una vez definidos los valores iniciales de los pesos, procedemos a entrenar la red neuronal, cuya explicación la haremos apoyados en un sencillo ejemplo de clasificación de patrones. En este ejemplo, un mapa auto-organizado es capaz de detectar dos grupos de patrones presentes en un espacio bidimensional como el mostrado en la figura 5.6; naturalmente, este análisis es extensible a un espacio de dimensión *N*.

En la figura, los puntos rojos y azules representan los grupos que la red debe clasificar, es decir, las dos clases de puntos que le vamos a presentar a la red en su entrenamiento. Los puntos verdes representan las posiciones aleatorias, en el espacio bi-dimensional, de las neuronas N_1 y N_2 al iniciar el proceso de aprendizaje.

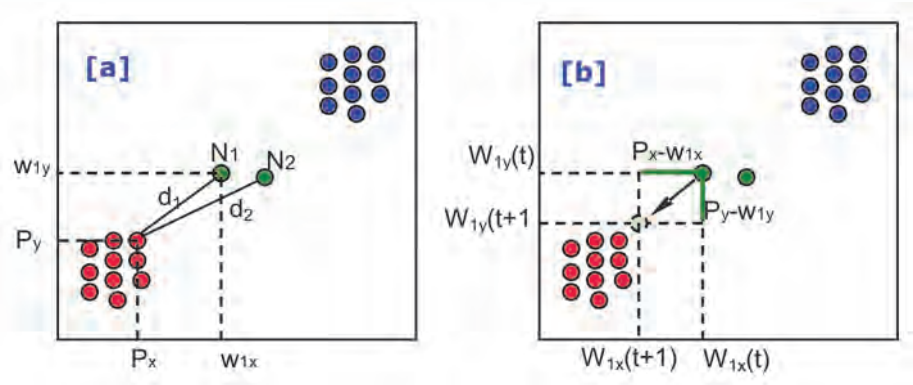

Fig. 5.6 Ilustración del algoritmo de entrenamiento de los SOM

El proceso de entrenamiento comienza seleccionando aleatoriamente alguno de los patrones de entrada, en este caso escogemos un patrón de la clase Rojo, cuyas coordenadas se representan como *(Px , Py),* figura 5.6.a. Las neuronas compiten entre sí para decidir cual de ellas es la más cercana al patrón seleccionado y para tal fin, calculamos las distancias d_1 y d_2, definidas entre el patrón y las neuronas N_1 y N_2 respectivamente. Tal como observamos en la figura, la neurona N_1 es la ganadora pues es la más cercana al patrón de aprendizaje utilizado. Los pesos de la neurona ganadora los representamos con W_{1x} y W_{1y}, cuyos valores debemos modificar con el fin de acercar esta neurona hacia el patrón seleccionado para el entrenamiento.

En la figura 5.6.b, ilustramos como la neurona ganadora modifica sus pesos para acercarse al patrón de entrenamiento presentado, lo que genera un *"movimiento"* de la neurona en el espacio de entrada. La modificación de los pesos de la neurona ganadora se lleva a cabo con base en las ecuaciones 5.5 y 5.6, que provienen de la ecuación 5.1, cuando se desglosa para cada una de las coordenadas.

$$w_{1x}(t+1) = w_{1x}(t) + \alpha(t) h_{rg} \left[P_x - w_{1x}(t) \right] \tag{5.5}$$

$$w_{1y}(t+1) = w_{1y}(t) + \alpha(t) h_{rg} \left[P_y - w_{1y}(t) \right] \tag{5.6}$$

Donde, $W_{1x}(t)$ y $W_{1y}(t)$ representan el valor de los pesos antes de la modificación; $W_{1x}(t+1)$ y $W_{1y}(t+1)$ representan el valor de los pesos después de la modificación y P_x y P_y definen las coordenadas del patrón de entre-

namiento seleccionado. En la figura también mostramos los vectores P_x-W_{1x} y P_y-W_{1y} que son los que determinan la dirección hacia donde evoluciona el nuevo peso. La neurona perdedora N_2 en este caso no sufre modificación de los pesos y el efecto final, es que se queda ubicada en sitio donde comenzó en el proceso de aprendizaje.

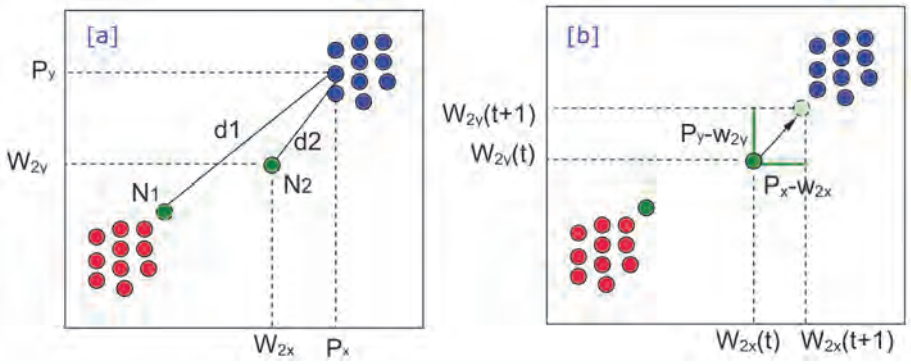

Figura 5.7 Ilustración del algoritmo de entrenamiento de los SOM

En la figura 5.7.a, presentamos las posiciones de las neuronas *N1* y *N2* luego de la primera iteración del proceso de aprendizaje, para continuar con el proceso de entrenamiento seleccionamos otro de los patrones, en este caso seleccionamos un patrón del grupo AZUL de coordenadas *Px* y *Py*. Las neuronas compiten ahora para determinar cuál está más cerca del patrón seleccionado, los valores d_1 y d_2 de la figura 5.7.a corresponden a las distancias del patrón seleccionado a las neuronas N_1 y N_2. Como se desprende de la figura, en este paso la neurona N_2 es la ganadora pues está mas cerca del patrón de aprendizaje utilizado, los pesos de la neurona ganadora se representa con W_{2x} y W_{2y} y son los que modificaremos.

En la figura 5.7.b, ilustramos como la neurona ganadora modifica sus pesos para acercarse al patrón de entrenamiento presentado, lo que genera un *"movimiento"* de la neurona en el espacio de entrada. La modificación de los pesos de la neurona ganadora se lleva a cabo con base en las ecuaciones 5.7 y 5.8.

$$w_{2x}(t+1) = w_{2x}(t) + \alpha(t) h_{rg} \left[P_x - w_{2x}(t) \right] \tag{5.7}$$

$$w_{2y}(t+1) = w_{2y}(t) + \alpha(t) h_{rg} \left[P_y - w_{2y}(t) \right] \tag{5.8}$$

Donde, $W_{2x}(t)$ y $W_{2y}(t)$ representan el valor de los pesos antes de la modificación; $W_{2x}(t+1)$ y $W_{2y}(t+1)$ representan el valor de los pesos después de la

modificación y P_x y P_y definen las coordenadas del patrón de entrenamiento seleccionado. En la figura también mostramos los vectores P_x-W_{2x} y P_y-W_{2y} que son los que determinan hacia donde va a generarse el nuevo peso. La neurona perdedora N_1 en este caso no sufre modificación de los pesos y el efecto final, es que se queda ubicada en sitio donde comenzó en el proceso de aprendizaje.

La figura 5.8 nos ilustra con claridad como las neuronas se acercan hacia los datos de entrada para intentar representar la información presente en ellos. En la primera, vemos la representación aleatoria y luego de dos iteraciones, las neuronas se han acercado lo suficiente a los datos como para representarlos en un proceso de decisión.

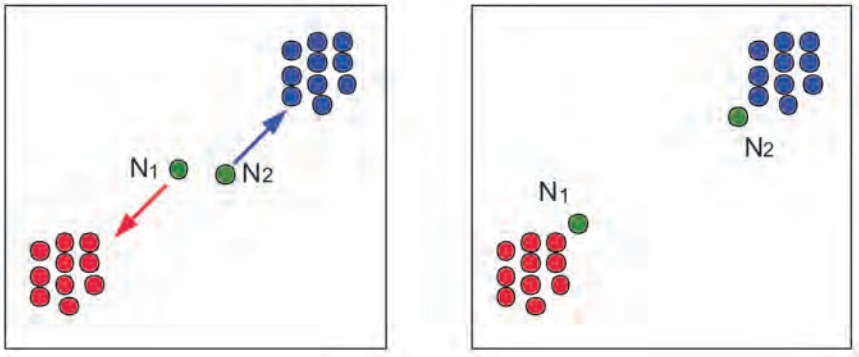

Fig. 5.8 Estado Inicial y Final de las Neuronas en el proceso de Aprendizaje del Mapa de Kohonen

En síntesis, hemos podido ver podemos ver cómo las neuronas se "mueven" en el espacio de entrada para ubicarse sobre los datos de entrada. Por ejemplo, si a un Mapa Auto-organizado de Kohonen le presentamos un conjunto de datos que tienen forma de cuadrícula, en la figura 5.9.a observamos la disposición desordenada de las neuronas de un mapa de Kohonen cuando inicia el proceso de aprendizaje y en la figura 5.9.b, vemos como las neuronas adoptan la forma cuadriculada, luego, de finalizar el proceso de aprendizaje.

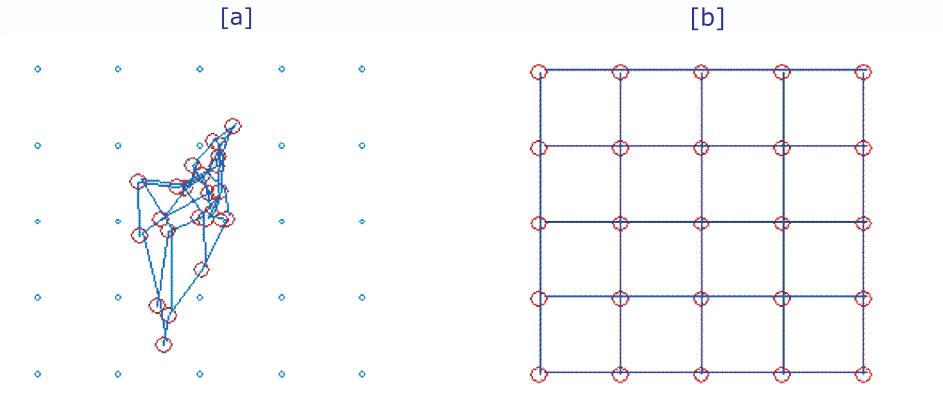

Fig. 5.9 Capacidad de auto-organización de los Mapas de Kohonen (Caso Cuadrícula)

Adicionalmente, en la figura 5.10 vemos un nuevo ejemplo de auto-organización del Mapa de Kohonen, en el cual le hemos presentado a la entrada un conjunto de datos con forma de toroide.

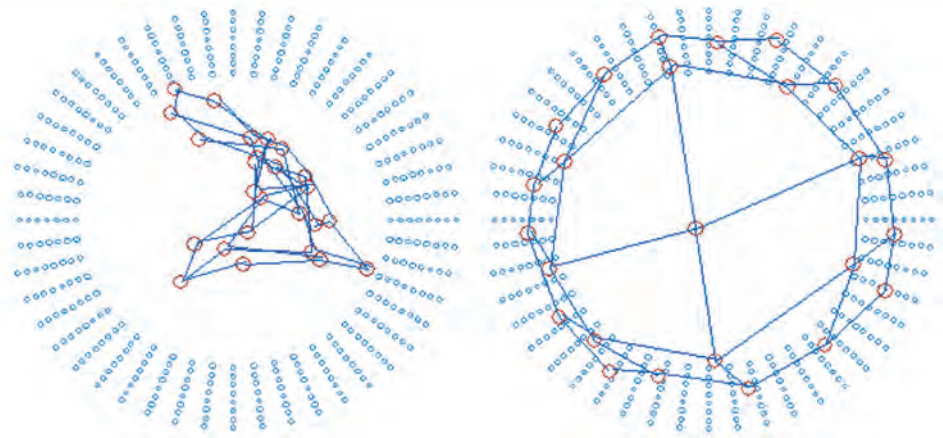

Fig. 5.10 Capacidad de auto-organización de los Mapas de Kohonen (Caso Toroide)

Pasos del algoritmo

MAPAS AUTO-ORGANIZADOS DE KOHONEN
ALGORITMO DE APRENDIZAJE

- Paso 1
  Definimos la arquitectura de la red, con N neuronas en la capa de entrada y en la capa de salida un mapa de M_f x M_c neuronas.
  Cada neurona la definimos con el vector de pesos sinápticos, w_i tiene la misma dimensión R^n del espacio de entrada, cuyos valores iniciales se asignan aleatoriamente.
  Definimos los parámetros de control: σ_i , σ_f, α_i , α_f y t_{max}

- Paso 2
  Seleccionamos un vector de entrada x de manera aleatoria, tal que pertenezca al conjunto de patrones de entrenamiento.

- Paso 3
  Determinamos el índice de la neurona ganadora s con base en la mínima distancia entre el vector de entrada y los vectores de pesos de las neuronas.

$$s = \arg\min \left\| \mathbf{x} - \mathbf{w}_i \right\|$$

- Paso 4
  Modificamos los pesos de la neurona *r-ésima* de acuerdo con:

$$\Delta \mathbf{w}_r = \alpha(t) h_{rs}(t)(\mathbf{x} - \mathbf{w}_r)$$

  donde,

$$h_{rs}(t) = e^{\left(\frac{-d_1(r,s)^2}{2\sigma(t)^2}\right)}, \quad \alpha(t) = \alpha_i \left(\frac{\alpha_f}{\alpha_i}\right)^{\frac{t}{t_{max}}}, \quad \sigma(t) = \sigma_i \left(\frac{\sigma_f}{\sigma_i}\right)^{\frac{t}{t_{max}}}$$

- Paso 5
  Incrementamos el parámetro t

  $t=t+1$

- Paso 6
  Si $t < t_{max}$ retornamos al Paso 2.

PRINCIPIO DE FUNCIONAMIENTO

Una vez hemos finalizado el proceso de entrenamiento de la red neuronal artificial, es común utilizarla para clasificar un conjunto de datos y conocer a que clase pertenece un patrón de entrada. En este tipo de ejemplos, iniciamos presentando el patrón de entrada a la red, lo que origina una competencia entre las diferentes neuronas de la capa de salida, que intentan determinar cual de las neuronas está a menor distancia del patrón presentado, en el espacio *N* dimensional. Cuando el algoritmo define la neurona ganadora, inmediatamente podemos determinar a que clase pertenece el patrón de entrada y, entonces, se finaliza el proceso de clasificación.

Si formalizamos matemáticamente el procedimiento, al Mapa de Kohonen le presentamos el patrón de entrada *k-ésimo* $\mathbf{x}^k=[x_1^k, x_2^k,..., x_i^k,..., x_N^k]$, para calcular la mínima distancia respecto del vector de pesos w_i. La neurona que tenga la mínima distancia, se constituye en la neurona ganadora y definirá la clase a la cual pertenece el patrón de entrada.

Retomemos el ejemplo de la sección 4.3, donde entrenamos una Red de Kohonen con un conjunto de datos que tiene dos clases definidas. En la Figura 5.11.a, vemos el estado final del entrenamiento, donde las neuronas N_1 y N_2, se ubican en los conjuntos de datos rojo y azul, respectivamente. Si le presentamos a la red un nuevo patrón de coordenadas P_x y P_y, representado en la figura como un círculo de color amarillo, la neurona ganadora será la N_1. En la figura 5.11.b, observamos como el patrón presentado ahora es asignado a la clase rojo debido a que la Neurona N_1 fue la ganadora.

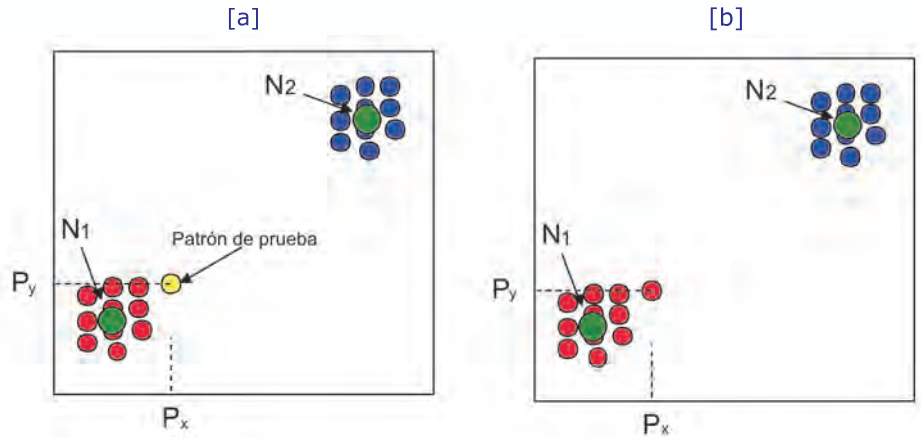

Fig. 5.11 Principio de Funcionamiento del Mapa de Kohonen

Aproximación práctica

Capacidad de reconocer grupos de patrones de un mapa de kohonen

En este primer proyecto tenemos un conjunto de datos conformado por dos clases, el objetivo es diseñar y entrenar una red competitiva para que clasifique estos puntos, para lo cual presentamos el siguiente programa escrito en MATLAB®.

```
%SomClasificador.m
% Este programa nos permite clasificar un conjunto de puntasen el
plano

close all;
clear;

% Generamos el conjunto de patrones
px1=0.05*randn(1,20);
py1=0.05*randn(1,20);
px2=0.05*randn(1,20)+1;
py2=0.05*randn(1,20)+1;

p = [px1 px2; py1 py2];

% Visualizamos los patrones a ser clasificados.
figure
plot(p(1,:),p(2,:),'ob');
hold on;

% Creamos el Mapa de Kohonen
red = newsom(minmax(p),2);

disp('Pesos y Bias Iniciales')
Wini = red.IW{1}
Bini=red.b{1}

W(:,:,1) = red.IW{1};
B(:,:,1)=red.b{1};

% Graficamos las neuronas en la figura
plot(W(:,1),W(:,2),'or');
title('Posición Inicial de las Neuronas')
```

```
hold off;
disp('Oprima una Tecla');
pause;

% Ejecutamos el proceso de aprendizaje de la red
red.trainParam.epochs = 25;
for i=1:20
red = train(red,p);
W(:,:,i+1) = red.IW{1};
B(:,:,i+1)=red.b{1};
end;

% Visualizamos los pesos y umbrales de la red
disp('Pesos y Bias Finales')
Wfin = red.IW{1}
Bfin=red.b{1}

disp('Vectores')
a = sim(red,p)
disp('Indices')
ac = vec2ind(a)

% Graficamos la evolución del proceso de aprendizaje
figure
plot(p(1,:),p(2,:),'ob');
hold on;
X1=W(1,1,:);
X1a=reshape(X1,1,21);
Y1=W(1,2,:);
Y1a=reshape(Y1,1,21);
plot(X1a,Y1a,'or');
plot(X1a,Y1a);

X2=W(2,1,:);
X2a=reshape(X2,1,21);
Y2=W(2,2,:);
Y2a=reshape(Y2,1,21);
plot(X2a,Y2a,'or');
plot(X2a,Y2a);
title('Evolución de las Neuronas')
hold off;
```

En la figura 5.12, observamos la ejecución de este programa y cómo las neuronas del Mapa de Kohonen parten de un sitio aleatorio para con migrar hacia los dos conjuntos de datos a medida que llevamos a cabo el proceso de aprendizaje para cumplir con el objetivo de clasificarlos.

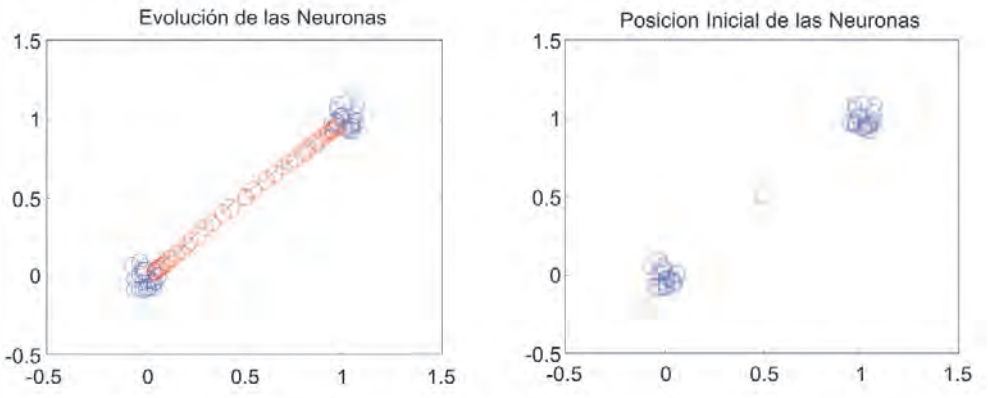

Fig. 5.12 Clasificación de Patrones usando un Mapa de Kohonen

Capacidad de autoorganización de los mapas de kohonen usando MATLAB®

Ejemplo 1: puntos en forma de arco

Con el fin de mostrar la capacidad de autoorganización de los Mapas de Kohonen, vamos a ejecutar el siguiente programa, por medio del cual le presentamos a la entrada de la red un conjunto de puntos que en el plano siguen la forma de un arco, en seguida llevamos a cabo su entrenamiento y visualizaremos el resultado del aprendizaje que hace esta red.

```
%SomArco.m
% Programa que ilustra la capacidad de auto-organización de
los Mapas de Kohonen
% Las neuronas al finalizar seguirán la forma de arco que se
definió en los puntos.
close all;
clear;

% Definimos los patrones de entrenamiento
  angles = 0:0.5*pi/15:0.5*pi;
   P = [sin(angles); cos(angles)];
```

```
% Creamos el Mapa de Kohonen
   net=newsom([0 1;0 1],[10]);

% Visualizamos los datos de entrenamiento y el estado inicial de las
% neuronas de la red
 figure;
  plot(P(1,:),P(2,:),'og');
  hold on;
  plotsom(net.iw{1,1},net.layers{1}.distances);
  disp('Oprima una tecla para continuar');
  pause;

% Definimos los parámetros de entrenamiento
  net.trainParam.epochs=1000;
  net.trainParam.show=100;
% Entrenamos la red
  net=train(net,P);

% Visualizamos los datos de entrenamiento y el estado final de las
% neuronas de la red
  figure;
  plot(P(1,:),P(2,:),'og');
  hold on;
  plotsom(net.iw{1,1},net.layers{1}.distances);
```

En la figura 5.13, vemos el resultado de ejecutar el anterior programa, en donde al iniciar tenemos que todas las neuronas del Mapa de Kohonen están centradas en el punto (0.5, 0.5) del plano. A medida que el programa ejecuta el proceso de entrenamiento, podemos observar como las neuronas paulatinamente buscan los datos de entrada y van asumiendo la forma de arco, hasta que al finalizar obtendremos un resultado como el de la figura, en donde las neuronas han modificado su estructura inicial para asumir la forma de arco que le presentan los datos.

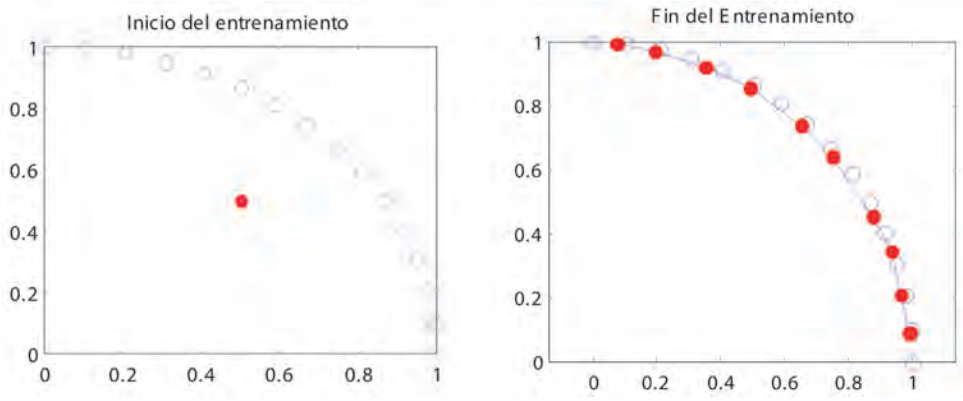

Fig. 5.13 Resultado del proceso de aprendizaje

Ejemplo 2: puntos en forma de cuadrícula

Como un segundo ejemplo para mostrar la capacidad de auto-organización de los Mapas de Kohonen, vamos a ejecutar el siguiente programa, por medio del cual le presentamos a la entrada de la red un conjunto de puntos que en el plano siguen la forma de una cuadrícula, para luego llevar a cabo su entrenamiento y visualizar el resultado del aprendizaje que hace esta red.

```
%SomCuadricula.m
% Programa que ilustra la capacidad de auto-organización de los
Mapas de Kohonen.
% Las neuronas al finalizar seguirán la forma de cuadrícula
definida en los puntos.

close all;
clear;

% Definimos los patrones de entrenamiento
cont=0;
for i=0:0.2:1
  for j=0:0.2:1
    cont=cont+1;
    X(cont)=i;
    Y(cont)=j;
  end;
end;
 P=[X;Y];
```

```
%Creamos el Mapa de Kohonen
net=newsom([0 1;0 1],[5 5],'gridtop');

% Visualizamos los datos de entrenamiento y el estado inicial de
las
% neuronas de la red
   figure;
  plot(P(1,:),P(2,:),'ob');
  hold on;
  plotsom(net.iw{1,1},net.layers{1}.distances);
  disp('Oprima una tecla para continuar');
  pause;

% Definimos los parámetros de entrenamiento
  net.trainParam.epochs=2000;
  net.trainParam.show=100;

%Entrenamos la red
  net=train(net,P);

% Visualizamos los datos de entrenamiento y el estado final de las
% neuronas de la red
  figure;
  plot(P(1,:),P(2,:),'ob');
  hold on;
  plotsom(net.iw{1,1},net.layers{1}.distances);
```

En la figura 5.14, vemos el resultado de ejecutar el anterior programa, en donde al iniciar tenemos que todas las neuronas del Mapa de Kohonen están centradas en el punto (0.5, 0.5) del plano. A medida que el programa ejecuta el proceso de entrenamiento podemos observar como las neuronas paulatinamente buscan los datos de entrada y van asumiendo la forma de cuadrícula, hasta que al finalizar obtendremos un resultado como el de la figura, en donde las neuronas han modificado su estructura inicial para asumir la forma de cuadrícula que le presentan los datos.

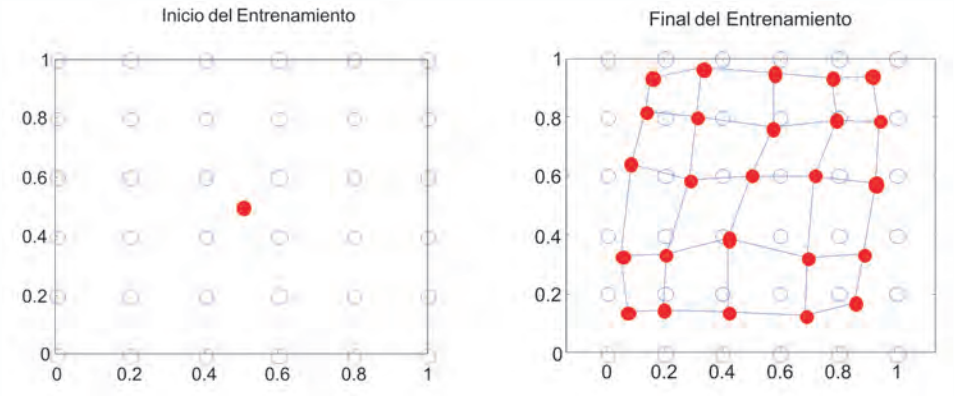

Fig. 5.14 Resultado del proceso de aprendizaje

Capacidad de autoorganización de los mapas de kohonen usando uv-srna

Vamos a ilustrar la capacidad de auto-organización de los Mapas de Kohonen, usando el simulador de redes neuronales UV-SRNA que tiene desarrollado un módulo para simular este tipo de redes competitivas. La interfaz gráfica con el usuario de este módulo lo presentamos en la figura 5.15.

Fig. 5.15 Interfaz de usuario del Móulo RNA Kohonen de UVSRNA

Para seguir la misma metodología propuesta en la sección 6.2, proponemos que el Mapa de Kohonen aprenda una configuración de puntos en forma de arco, por lo creamos un archivo de patrones como el mostrado en la

figura 5.16, que contiene 16 patrones de dos coordenadas, cuyos valores debemos listar tal como ilustra la figura con el fin de ser leídos por UV-SRNA.

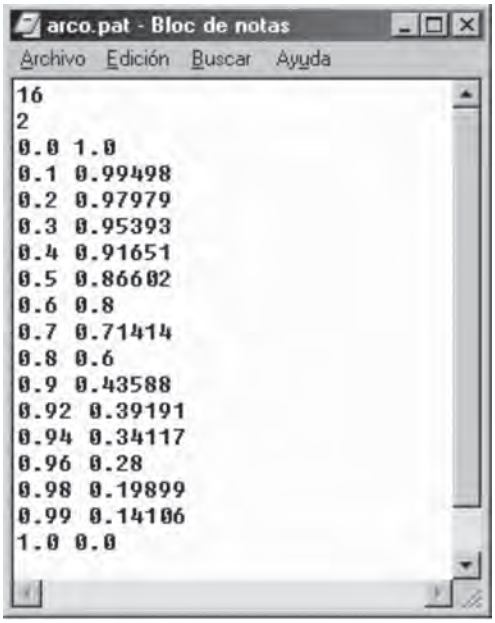

Fig. 5.16 Archivos de patrones.

Una vez editado el archivo de patrones procedemos a invocarlo desde UV-SRNA. Por defecto UV-SRNA crea un Mapa de Kohonen de 25 neuronas organizadas en una matriz de 5x5; pero debido a que los puntos usados definen una curva y no una región en el plano, debemos cambiar la arquitectura de la red a un Mapa de Kohonen de 1x10 neuronas, usando la interfaz de la figura 5.17.

Fig. 5.17 Interfaz para definir la arquitectura de la red

Luego de definir la arquitectura de la red procedemos a su entrenamiento, usando la opción *"entrenar"* de UV-SRNA, quien nos responde mostrándonos la evoluación del proceso de entrenamiento desde un estado inicial de las neuronas, donde su organización es aleatoria, hasta que finalmente éstas adoptan la forma de arco definida por los puntos de entrenamiento. En la figura 5.18, presentamos el avance del entrenamiento cuando se han ejecutado 10 iteraciones y en la figura 5.19, el estado final de la organización de las neuronas luego de 600 iteraciones.

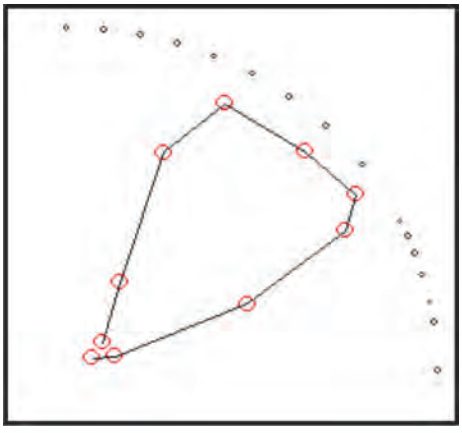

Fig. 5.18 Avance del Entrenamiento con 10 iteraciones

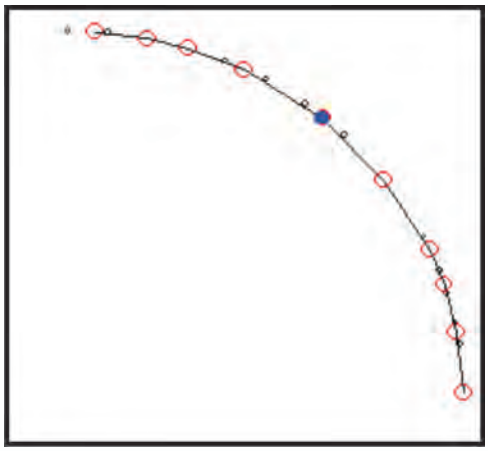

Fig. 5.19 Situación final del entrenamiento

En seguida procederemos a entrenar a un Mapa de Kohonen con un conjunto de datos en forma de cuadrícula, con base en el archivo de la figura 5.21.

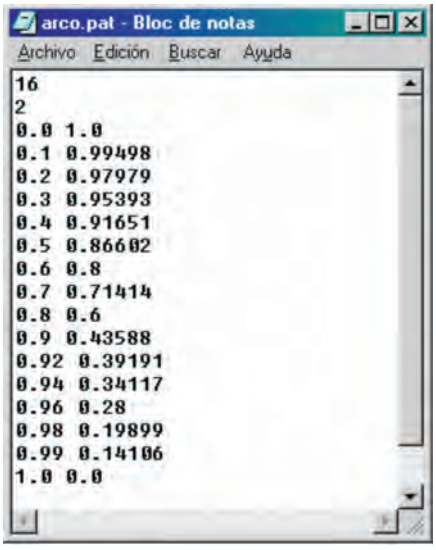

Fig. 5.20 Archivos de patrones.

Por defecto UV-SRNA crea un Mapa de Kohonen de 25 neuronas organizadas en una matriz de 5x5, que por la naturaleza de este ejercicio mantendremos y procedemos a su entrenamiento usando la opción *"entrenar"* de UV-SRNA, quien nos responde mostrándonos la evoluación del proceso de

entrenamiento desde un estado inicial de las neuronas, donde su organización es aleatoria, hasta que finalmente éstas adoptan la forma de cuadrícula definida por los puntos de entrenamiento. En la figura 5.21, presentamos el avance del entrenamiento cuando se han ejecutado 10 iteraciones y, en la figura 5.22, el estado final de la organización de las neuronas, luego, de 3000 iteraciones.

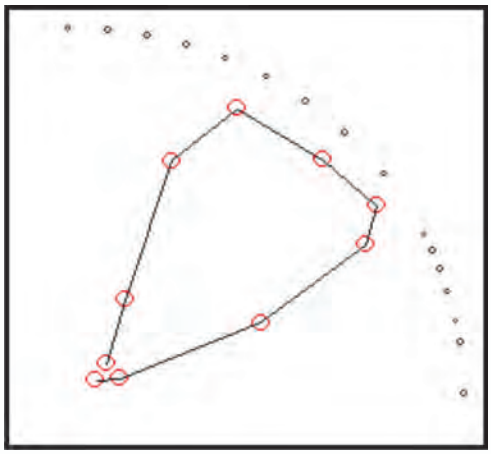

Fig. 5.21 Avance del Entrenamiento con 10 iteraciones

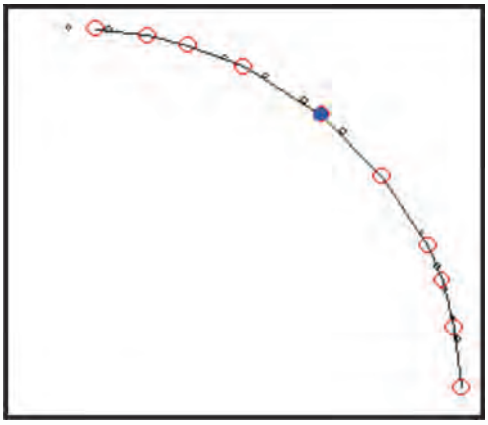

Fig. 5.22 Situación final del entrenamiento

Clasificación de patrones usando mapas de kohonen

El problema del iris fue presentado en la sección 6.7 del capítulo 4, cuando lo resolvimos utilizando una red tipo MLP, Ahora vamos a resolverlo usando un mapa auto-organizado. Para este caso presentamos al mapa todos los ejemplos disponibles para visualizar cuál es la organización que adquie-

ren la neuronas del mapa auto-organizado luego del proceso de aprendizaje. En el siguiente programa escrito para MATLAB®, presentamos los pasos necesarios para el entrenamiento de un Mapa de Kohonen.

```
%Programa que soluciona el problema de Clasificación en la
Base de Datos IRIS

close all;
clear;
clc;
Nneuronas=[10 10]
datosauxiris = load('-ascii','iris_data2.txt');
datosiris = datosauxiris(:,1:4)';
red = newsom(minmax(datosiris),Nneuronas,'gridtop','dist',0.9,
1000,0.1,0);
red.trainParam.epochs=1500;
red.trainParam.show=100;
red=train(red,datosiris);
yred=sim(red, datosiris);
indices=vec2ind(yred);
Mapa=zeros(10,10);
for (i=1:50)
     fila=floor(indices(i)/10)+1;

     columna=round(rem(indices(i),10));
     if columna==0
      columna=10;
   end;
   Mapa(fila,columna)=1;
end;

for (i=51:100)
   fila=floor(indices(i)/10)+1;
   columna=round(rem(indices(i),10));
   if columna==0
      columna=10;
   end;
   Mapa(fila,columna)=2;
end;

for (i=101:150)
   fila=floor(indices(i)/10)+1;
```

```
        columna=round(rem(indices(i),10));
        if columna==0
            columna=10;
        end;
        Mapa(fila,columna)=3;
    end;
    imagesc (Mapa(1:11,1:10)); figure(gcf)
    figure
    plotsomplanes(red)
```

En la figura 5.23, visualizamos la organización con que quedaron las neuronas del mapa de acuerdo a las tres clases definidas para este problema. En los colores rojo, amarillo y azul claro, vemos las neuronas que se activan de acuerdo a las diferentes clases. Podemos resaltar que las neuronas que definen una clase comparten una distribución espacial continua y bien definida. El color azul oscuro representa las neuronas que no se activaron.

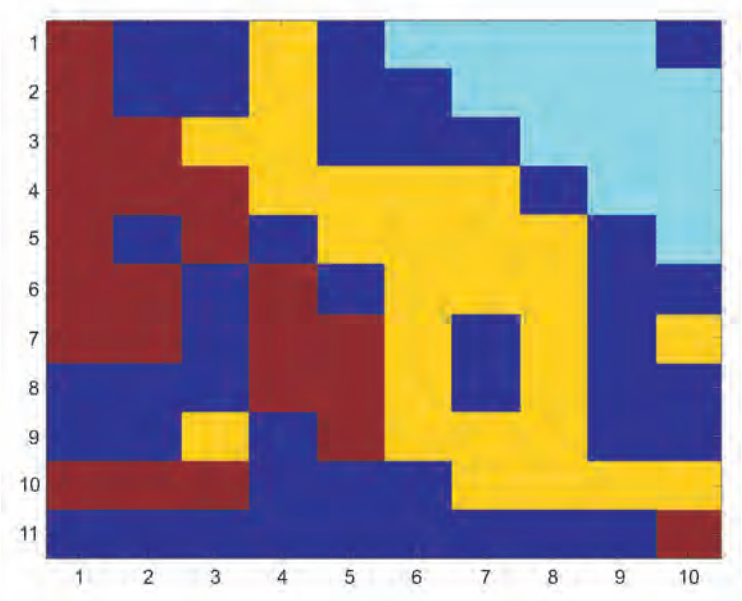

Fig. 5.23 Visualización de la categorias detectadas por el mapa auto-organizado

En la figura 5.24, mostramos la manera como cada una de las cuatro entradas afecta las diferentes zonas del mapa auto-organizado. El color rojo indica que en dicha región existe una alta influencia o valor de dicha característica. El color negro indica que la influencia de dicha característica

es nula. El color amarillo nos indica una influencia intermedia. Esta visualización es lo que se ha llamado el "*análisis de planos*" y, el mismo, nos permite detectar cuáles entradas o características tiene mayor o menor influencia en la determinación de las diferentes clases por parte del mapa auto-organizado.

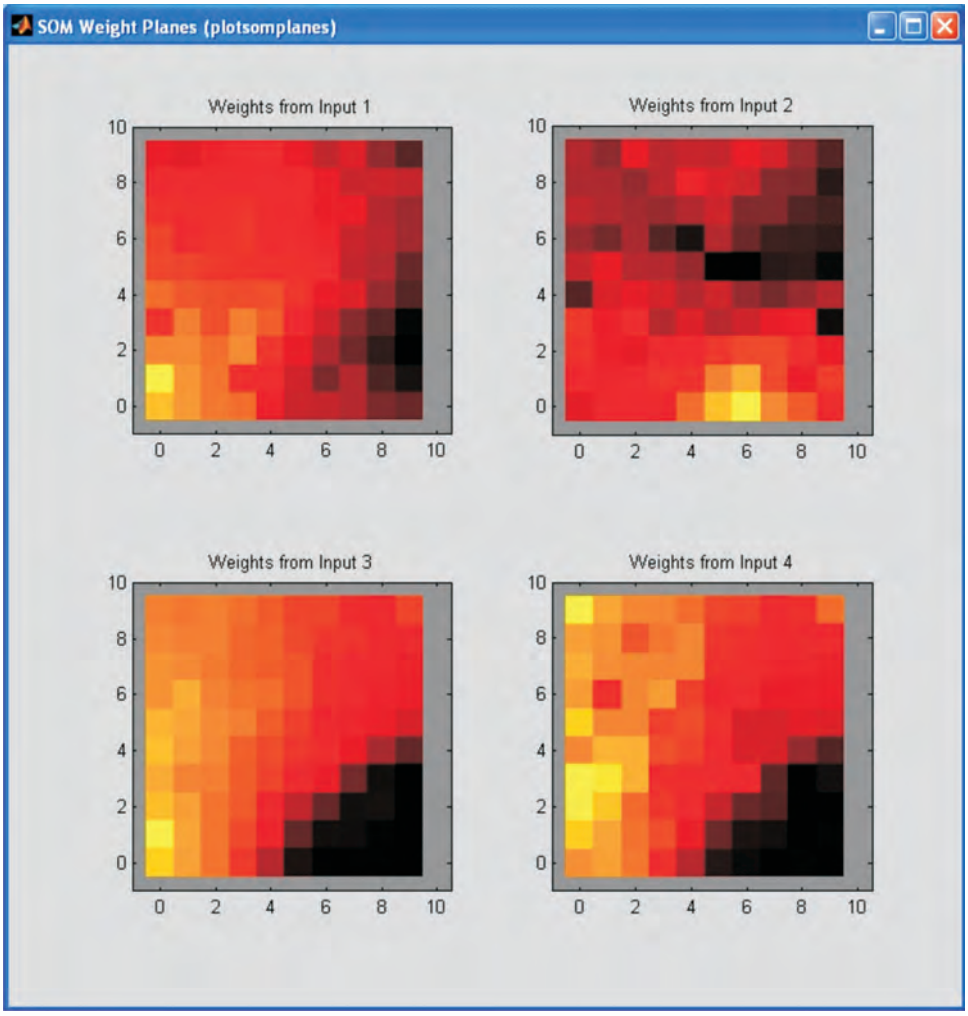

Fig. 5.24 Análisis de planos para las difentes entradas de la red neuronal

Proyectos propuestos

1. Repita el ejercicio del numeral 3.1.1 pero realizando el entrenamiento en UVSRNA.
2. Repita el ejercicio del numeral 3.1.3 pero realizando el entrenamien-

to en UVSRNA.

3. Suponga que tiene unos puntos distribuidos en un plano formando cinco grupos bien definidos como se muestra en la figura 5.9 Realice el entrenamiento en MATLAB® y en UVSRNA de un mapa auto-organizado que realice la sectorización de dichos grupos.

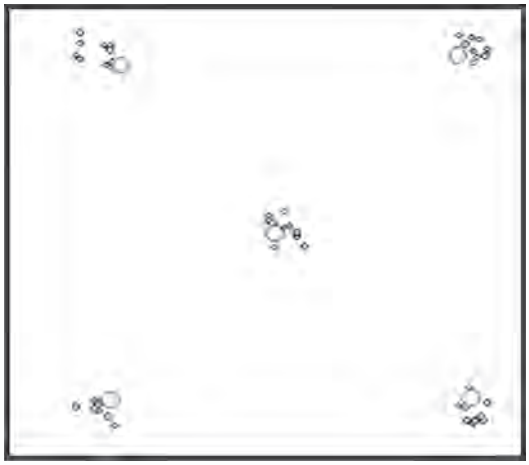

Fig. 5.25 Distribución de 5 grupos bien definidos en el plano

4. Suponga que tiene ocho grupos de bien definidos en un espacio 3D, realice la generación de patrones y el entrenamiento de la red en MATLAB® que permite sectorizar dichos patrones.
5. Suponga que tiene una distribución de puntos en el plano de forma triangular (figura 5.26). Genere dichos patrones y entrene una red en UVSRNA que se ajuste dicha distribución.

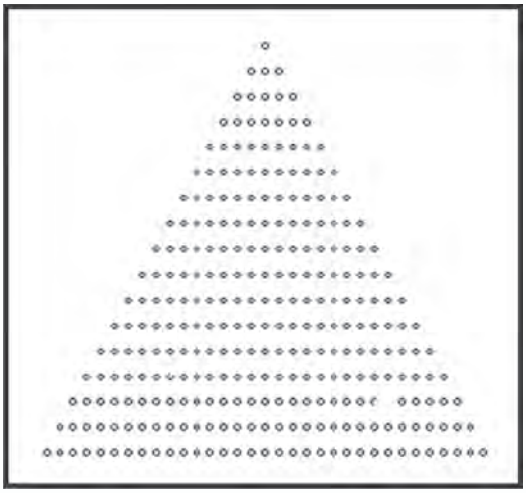

Fig. 5.26 Distribución triángular de puntos en el plano