### 75

Capítulo 8

Mapas autoorganizados

* Estructura SOM
* Formación de un SOM
* Funciones vecinales
* Cálculo de errores SOM

Hasta ahora este libro se ha centrado en la red neuronal feedforward. En este capítulo veremos un tipo diferente de red neuronal. Este capítulo se centrará en el Mapa autoorganizado (SOM). Aunque el SOM se puede considerar un tipo de red neuronal feedforward, se utiliza de manera muy diferente. Un SOM se utiliza para la clasificación no supervisada.

Los mapas autoorganizados comienzan con pesos aleatorios, al igual que las redes neuronales que hemos visto hasta ahora. Sin embargo, los medios por los cuales estos pesos están organizados para producir una salida significativa son muy diferente de una red neuronal feedforward. SOM hace uso de entrenamiento sin supervisión.

EL entrenamiento no supervisado funciona de manera muy diferente al entrenamiento supervisado que hemos estado utilizando hasta ahora. Los conjuntos de entrenamiento no supervisados solo especifican la entrada a la red neuronal. No se proporciona una salida ideal. Debido a esto, el SOM aprende a agrupar o mapear sus datos en un número especifico de clases.

El número de neuronas de salida en el SOM define en cuántas clases de salida se desea agrupar los datos de entrada. El número de neuronas de entrada define los atributos de cada uno de los elementos del conjunto de entrenamiento sen los que desea agrupar la red neuronal.

### 76 Mapas autoorganización

Esto es exactamente lo mismo que era cierto para las redes neuronales feedforward. Debe presentar datos a la red neuronal a través de un número fijo de neuronas de entrada.

Los SOM solo se utilizan para la clasificación, que es dividir la entrada en clases. Los SOM no se utilizan generalmente para la regresión. La regresión es donde la red neuronal predice uno o más valores numéricos. Con un SOM, simplemente proporciona el número total de clases, y el SOM automáticamente sus datos de entrenamiento en estas clases. Además, el SOM debe ser capaz de organizar los datos con los que nunca se capacitó en estas clases.

# Estructura SOM

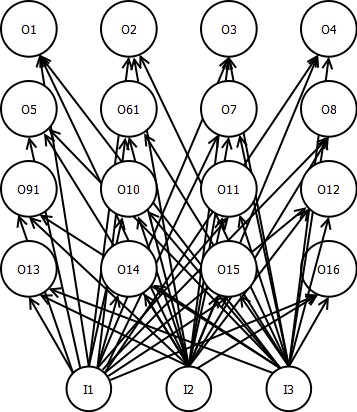
El SOM tiene una estructura algo similar a las redes neuronales que ya hemos visto en este libro. Sin embargo, hay algunas diferencias muy importantes. Algunas de las diferencias entre SOM y redes neuronales tradicionales feedforward se muestran aquí.

* + - Las redes SOM no tienen neuronas sesgadas
    - Las redes SOM no tienen capas ocultas
    - Las redes SOM no tienen funciones de activación
    - Las redes SOM deben tener más de una neurona de salida
    - Producir neuronas en una celosía 1d, 2d, 3d, etc

Considere un SOM que tiene 16 neuronas de salida y tres neuronas de entrada. Este SOM está diseñado para clasificar los datos en tres grupos. Los elementos de datos entrantes que se clasificarán tienen seis valores cada uno. Este SOM se puede ver en la figura 8.1.

### EstructuraSOM 77

**Figura 8.1:** Un SOM 2D con 16 salidas y 3 entradas



Como se puede ver en el diagrama anterior no hay capas ocultas y no hay neuronas de sesgo. También notarás que las neuronas de salida están dispuestas en una celosía. En este caso las neuronas de salida están dispuestas en una cuadrícula 2D. Esta cuadrícula se volverá muy importante cuando se entrena la red neuronal. Las neuronas cerca unas de otras en la cuadrícula serán entrenadas juntas. La cuadrícula no necesita ser 2D, las cuadrículas pueden ser 1D, 3D o de un número aún mayor de dimensiones. La topología de esta cuadrícula se define mediante la función de vecindad que se utiliza para entrenar la red neuronal. Las funciones vecinales se describirán más adelante en este capítulo.

Se puede considerar que un SOM reduce un alto número de dimensiones a un número menor de dimensiones. El mayor número de dimensiones es proporcionado por las neuronas de entrada. Una dimensión para cada neurona de entrada. El menor número de dimensiones es la configuración de la celosía de la neurona de salida. En el caso de la Figura 8.1, estamos reduciendo tres dimensiones a dos.

En el Capítulo 1 vimos cómo una red neuronal feedforward calcula su salida. El método por el cual un SOM calcula su salida es muy diferente. La salida de un SOM es la neurona de salida "ganadora" para una entrada dada. Esta neurona ganadora también se llama la mejor unidad de coincidencia, o BMU. Veremos cómo calcular la BMU en la siguiente sección.

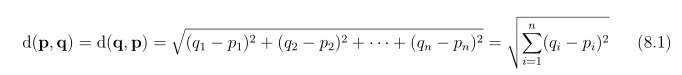
* + 1. **Mejor unidad** **a juego**

La mejor unidad de coincidencia es la neurona de salida cuyos pesos coinciden más estrechamente con la entrada que se proporciona a la red neuronal. Considere el SOM mostrado en el cuadro 8.1. Hay 16 neuronas de salida y 3 neuronas de entrada. Esto significa que cada una de las 16 neuronas de salida tiene 3 pesos, uno de cada una de las neuronas de entrada. La entrada al SOM también sería de tres números, ya que hay tres neuronas de entrada. Para determinar la BMU determinamos la neurona de salida cuyos tres pesos coinciden más estrechamente con los tres valores de entrada introducidos en el SOM.

Es bastante fácil calcular la BMU. Para ello, recorremos todas las neuronas de salida y calculamos la distancia euclidiana entre los pesos de la neurona de salida y los valores de entrada. Cualquiera que sea la neurona de salida tiene la distancia euclidiana más baja es la BMU. La distancia euclidiana es simplemente la distancia entre dos puntos multidimensionales. Si usted está tratando con dos dimensiones, la distancia euclidiana es simplemente la longitud de una línea trazada entre los dos puntos.

La distancia euclidiana se utiliza a menudo en el aprendizaje automático. Es una forma rápida de comparar dos matrices de números que tienen el mismo número de elementos. Considere tres matrices, denominada matriz **a**, matriz **b** y matriz **c**. La distancia euclidiana entre la matriz **a** y la matriz **b** es 10. La distancia euclidiana entre la matriz **a** y la matriz **c** es 20. En este caso, el contenido de la matriz a coincide más estrechamente con la matriz b que con la matriz c.

Ahora veremos cómo calcular realmente la distancia euclidiana. La ecuación 8.1 muestra esta fórmula.



d(**p***,*

**q**) = d(**q***,*

**p**) = q(*q*1 − *p*1

)2 + (*q*2 − *p*2

)2 +

· · ·

+ (*qn* − *pn*

)2 =

*N*

(*qi pi*

 −

*i*=1

)2 (8.1)

La ecuación anterior nos muestra la distancia euclidiana **d** entre dos matrices **p** y **q**. La ecuación anterior también establece que **d(p,q)** es la misma que **d(q,p)**. Esto simplemente afirma que la distancia es la misma sin importar en qué extremo comiences. El cálculo de la distancia euclidiana no es más que sumar los cuadrados de la diferencia de cada elemento de matriz. Finalmente, se toma la raíz cuadrada de esta suma. Esta raíz cuadrada es la distancia euclidiana. La neurona de salida con la distancia euclidiana más baja es la BMU.

# Entrenamiento de un SOM

En los capítulos anteriores aprendimos varios métodos para entrenar un trabajo de red neuronal feedforward. Aprendimos sobre técnicas como backpropagation, RPROP y LMA. Todos estos son métodos de entrenamiento supervisados. Los métodos de entrenamiento supervisados funcionan ajustando los pesos de una red neuronal para producir la salida correcta para una entrada determinada.

Un método de capacitación supervisado no funcionará para un SOM. Las redes SOM requieren entrenamiento sin supervisión. En esta sección aprenderemos a entrenar un SOM con un método sin supervisión. La técnica de entrenamiento generalmente utilizada para redes SOM se resume en la Ecuación 8.2.

*Wv*(*t* + 1) = *Wv*(*t*) + *ḥ*(*v,* *t*)*α*(*t*)(*D*(*t*) − *Wv*(*t*)) (8.2)

W(t+1): Peso final

W(t): Peso Actual

Teta(v,t): function vecinal

:Tasa de aprendisaje

D(t): Entrada

La ecuación anterior muestra cómo los pesos de una red neuronal SOM se actualizan a medida que avanza el entrenamiento. La iteración de entrenamiento actual se observa por la letra **t**, y la siguiente iteración de entrenamiento se observa por **t+1**. La ecuación 8.2 nos permite ver cómo se ajusta el peso **v** para la siguiente iteración de entrenamiento.

La variable **v** indica que estamos realizando la misma operación en cada peso. La variable **W** representa los pesos. El símbolo **theta** es una función especial, denominada función de vecindad. La variable **D** representa la entrada de entrenamiento actual al SOM.

El símbolo alfa denota una tasa de aprendizaje. La tasa de aprendizaje cambia para cada iteración. Esta es la razón por la ecuación 8.2 muestra la tasa de aprendizaje con el símbolo t, ya que la tasa de aprendizaje se adjunta a la iteración. Se dice que la tasa de aprendizaje de un SOM está disminuyendo monotónicamente. Monotónicamente disminuyendo simplemente yoy que la tasa de aprendizaje sólo cae, y nunca aumenta.

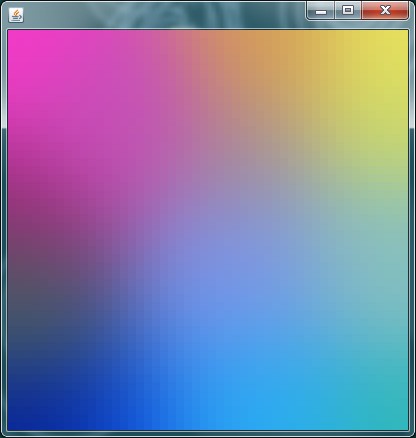
## Ejemplo de formación som

Ahora veremos cómo entrenar realmente un SOM. Aplicaremos la ecuación presentada en la sección anterior. Sin embargo, lo abordaremos más desde una perspectiva de algoritmo para que podamos ver la estrategia de aprendizaje real detrás de la ecuación. Para ver el SOM en acción necesitamos un ejemplo muy simple. Vamos a utilizar un SOM muy similar al que vimos en la figura 8.1. Este SOM intentará hacer coincidir los colores. Sin embargo, en lugar de la celosía 4x4 que vimos en la Figura 8.1, tendremos una celosía de 50x50. Esto resulta en un total de 2.500 neuronas de salida.

El SOM utilizará sus tres neuronas de entrada para que coincidan con los colores de las neuronas de 2.500 output. Las tres neuronas de entrada contienen los componentes rojo, azul y verde del color que se está presentando actualmente al SOM. Para el entrenamiento, generaremos 15 colores aleatorios. El SOM aprenderá a agrupar estos colores.

Este tipo de formación se demuestra en uno de los ejemplos de Encog. Puede ver la salida de este programa de ejemplo en la Figura 8.2.

**Figura 8.2:** Mapeo de colores



Como se puede ver en la figura anterior, colores similares se agrupan. Además, hay 2.500 neuronas de salida, y sólo 15 colores que fueron entrenados. Esta red podría reconocer potencialmente hasta 2.500 colores. El hecho de que entrenamos con sólo15 colores significa que tenemos bastantes neuronas de salida no utilizadas. Estas neuronas de salida aprenderán a reconocer colores que están cerca de los 15 colores con los que entrenamos.

Lo que realmente está viendo en la Figura 8.2 son los pesos de la red SOM que se ha entrenado. Como se puede ver, a pesar de que el SOM sólo fue entrenado para reconocer 15 colores, es capaz de reconocer bastantes colores. Cualquier nuevo color proporcionado al SOM se asignará a uno de los 2.500 colores vistos en la imagen anterior. El SOM puede ser entrenado para reconocer más clases de las que proporciona sus datos de capacitación proporcionados. Este es desafiante el caso en la Figura 8.2. Las neuronas de salida no utilizadas terminarán aprendiendo a reconocer los datos que caen entre los elementos del conjunto de entrenamiento más pequeño.

## Formación del ejemplo SOM

Ahora veremos cómo se entrena la red SOM para el ejemplo de colores. Para comenzar con todos los pesos de la red SOM se aleatorizan a valores entre -1 y +1. Ahora se genera un conjunto de entrenamiento para 15 colores aleatorios. Cada uno de estos 15 colores aleatorios tendrá tres valores de entrada. Cada uno de los tres valores de entrada será un valor aleatorio entre -1 y +1. Para los valores rojo, verde y azul -1 representa que el color está totalmente apagado, y +1 representa que el color está totalmente encendido.

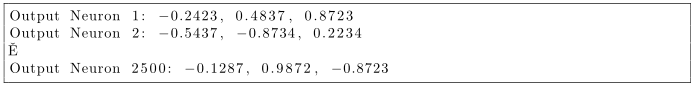
Veremos cómo se entrena el SOM para uno de estos 15 colores. El mismo proceso se utilizaría para los 14 colores restantes. Considere si estábamos capacitando al SOM para el siguiente aleatorio Color.

−1, 1, 0 . 5

Veremos cómo se formará el SOM con este elemento de formación.

## Ejemplo de cálculo de BMU

El primer paso sería comparar esta entrada con cada neurona de salida en el SOM y encontrar la mejor unidad de coincidencia (BMU). BMU se debatió anteriormente en este capítulo. La BMU se puede calcular encontrando la distancia euclidiana euclidiano más pequeña en SOM. Los pesos aleatorios SOM se muestran aquí.



Por supuesto, nos estamos saltando bastantes de las neuronas de salida. Normalmente, calcularía la distancia euclidiana para las 2.500 neuronas de salida. Sólo calcular la distancia euclidiana para las tres neuronas anteriores debe darle una idea de cómo se hace esto. Usando la Ecuación 8.1 calculamos la distancia euclidiana entre la entrada y la neurona.



A Similar proceso enlatar ser usado Para calcular neurona Dos.



Del mismo modo, también podemos calcular neuronas 2.500.



Ahora que hemos calculado todas las distancias euclidianas, podemos determinar la BMU. La BMU es neurona uno. Esto se debe a que la distancia de 0,9895 es la más baja. Ahora que tenemos una BMU, podemos actualizar los pesos.

## Ejemplo de funciones vecinales

Ahora recorreremos todos los pesos de todo el SOM y usaremos la Ecuación 8.2 para actualizarlos. La idea es que modificaremos la neurona BMU para que se parezca más a la entrada de entrenamiento. Sin embargo, además de modificar la neurona BMU para que se parezca más a la entrada de entrenamiento, también modificaremos las neuronas en el vecindario alrededor de la BMU para ser más como la entrada también. Sin embargo, cuanto más lejos sea una neurona de la BMU, menos impacto tendrá este cambio de peso.

Determinar la cantidad de cambio que le sucederá a un peso es el trabajo de la función neighborhood. Cualquier función de base radial (RBF) se puede utilizar como una función neighborhood. Una función de base radial (RBF) es una función de valor real cuyo valor depende únicamente de la distancia del origen.

La función gaussiana es la opción más común para una función de vecindad. La función gaussiana se muestra en la ecuación 8.3.



*f* (*x*1, *x*2*,* *...,* *xn*) = *e−v,* (8.3)

Esta ecuación continúa de la siguiente manera.



*v* =  *xi* − *c*

*i<n*

2*w*2

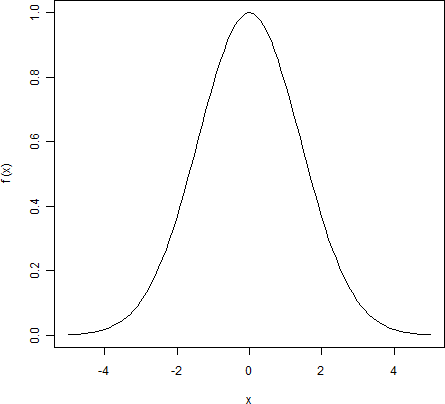
2

*i*=0

(8.4)

Puede ver la función gaussiana graficada en la Figura 8.3. Donde **n** es el número de dimensiones, **c** es el centro, **w** es el ancho de la curva gaussiana. El número de dimensiones es igual al número de neuronas de entrada. El ancho comienza en algún número fijo y disminuye a medida que avanza el aprendizaje. Para la iteración de entrenamiento final, el ancho debe ser uno.

**Figura 8.3:** Función gaussiana graficada



Desde la figura anterior, se puede ver que la función gaussiana es una función de base radial. El valor solo depende de la distancia del origen. Es decir, **f(x)** tiene el mismo valor independientemente de si **x** es -1 o +1.

Mirando la Figura 8.3, se puede ver cómo la función gaussiana escala la cantidad de entrenamiento recibido por cada neurona. La BMU tendría una distancia cero consigo misma, por lo que la BMU recibiría una formación completa de 1.0. A medida que te alejas más de la BMU, en cualquier dirección, la cantidad de entrenamiento se cae rápidamente. Como una neurona que era -4 o +4 de la BMU recibiría casi ningún entrenamiento en absoluto.

La función gaussiana no es la única función disponible para el entrenamiento SOM. Otra elección com- mon es el Ricker wavelet, o "sombrero mexicano" función de vecindario. Esta función generalmente sólo se conoce como la función "Sombrero Mexicano" en las Américas, debido a su parecido con un "sombrero". En la nomenclatura técnica esta función se conoce como la wavelet Ricker, donde seemplea confrecuencia para modelar datos sísmicos. La ecuación para la función Sombrero mexicano se muestra en la Ecuación 8.4.

*f* (*x*1, *x*2*,* *...,* *xn*) = (1 − *v*)*e−*0*.* 5*v,* (8.5)

Esta ecuación continúa con lo siguiente.

*i<n*

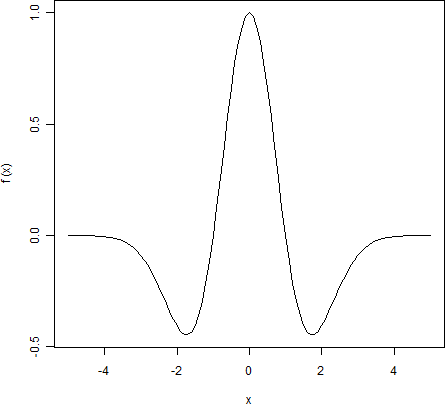
 −

*v* = (*x c*)2 (8.6)

*i*=0

Verá el valor de la función de vecindad Mexican Hat cuando examine su gráfico en la Figura 8.4.

**Figura 8.4:** Gráfico del sombrero mexicano



Al igual que antes, la BMU está en el origen de **x**. Como se puede ver en la tabla anterior, la función sombrero mexicano en realidad castiga las neuronas justo en el borde del radio de la BMU. Su parte del aprendizaje será en realidad menos de cero. A medida que se aleja aún más de la BMU, el aprendizaje vuelve de nuevons a cero, y permanece en cero. Sin embargo, las neuronas justo en el borde tendrán aprendizaje negativo aplicado. Esto puede hacer que el SOM a las clases se diferencie mejor entre sí.

## Ejemplo de actualización de peso

Ahora que hemos visto cómo calcular la BMU y la función de vecindad, finalmente estamos listos para calcular la actualización de peso real. La fórmula para la actualización de peso se dio en la Ecuación 8.2. Ahora calcularemos los componentes de esta ecuación para un

peso actualizar. recordar ese el Pesos dado anterior en éste capítulo. neurona Uno Tenía el siguiente Pesos.

Salida Neurona 1: −0.2423, 0.4837, 0.8723

La neurona uno fue la BMU cuando se le proporcionó la siguiente entrada.

−1, 1, 0.5

Como puede ver, la BMU es algo similar a la entrada. Durante el entrenamiento, ahora queremos modificar la BMU para que se parezca aún más a la entrada. También modificaremos a los vecinos para recibir parte de este aprendizaje también.

El aprendizaje "definitivo" es simplemente copiar la entrada en los pesos. A continuación, la distancia euclidiana de la BMU se convertirá en cero y la BMU está perfectamente entrenada para este vector de entrada. Sin embargo, no queremos ir tan extremo. Simplemente queremos mover los pesos en la dirección de la entrada. Aquí es donde entra en vigor la ecuación 8.2. Usaremos la Ecuación 8.2 para cada peso en el SOM, no sólo los pesos BMU. Para este ejemplo, vamos a iniciar con el BMU. Calculando la actualización al primer peso, que es -1, tenemos.



*w* = −0*.* 2423 + (*N* ∗ *r* ∗ (−1Ḥ(−0*.* 2423))) (8.7)

Antes de calcular la ecuación anterior, echaremos un vistazo a lo que realmente estamos haciendo. Mira el término en la extrema derecha. Estamos tomando la diferencia entre la entrada y el peso. Como dije antes, lo último es asignar la entrada al wocho. Si simplemente agregamos el último término al peso, el peso sería el mismo que el vector de entrada.

No queremos ser tan extremos. Por lo tanto, escalamos esta diferencia. Primero lo escalamos por la tasa de aprendizaje, que es la variable **r**. Luego también lo escalamos por el resultado de la función de vecindad, que es **N**. Para la BMU, **N** es el valor 1.0 y no tiene ningún efecto. Si la tasa de aprendizaje fuera 1.0, también, entonces la entrada sería realmente copied al peso. Sin embargo, una tasa de aprendizaje nunca está por encima de 1.0. Además, la tasa de aprendizaje normalmente se deteriora a medida que avanzan las iteraciones de aprendizaje. Teniendo en cuenta que tenemos una tasa de aprendizaje de 0,5, nuestra actualización de peso se convierte en la siguiente.

*w* = −0*.* 2423 + (1*.* 0 ∗ 0*.* 5 ∗ (−(− 1) – (− 0*.* 2423)) = −0*.* 62115 (8.8)

Como se puede ver, el peso pasó de -0.2423 a -0.62115. Esto acercó el peso a la entrada de -1. Podemos realizar una actualización similar para los otros dos pesos que se alimentan en la BMU. Esto se puede ver aquí.

*w* = 0*.*4837+(1*.*0∗0*.*5∗(1−0*.*4837)) = 0*.*74185

(8.9)

*w* = 0*.*8723+(1*.*0∗0*.*5∗(0*.*5−0*.*8723)) = 0*.*68615

Como puede ver, ambos pesos se acercaron a los valores de entrada.

La función de vecindad es siempre 1.0 para la BMU. Sin embargo, considere si vamos a calcular la actualización de peso para Neuron Two, que no es la BMU. Tendríamos que calcular la función de vecindad, que se dio en la ecuación 8.3. Esto supone que usamos una función de barrio gaussiana. También se podría utilizar la función Sombrero mexicano.

El primer paso es calcular **v**. Aquí usamos un ancho **w** de 3.0. Cuando se utiliza Gaussian para una función de vecindad, el centro **c** siempre es 0.0.

*v* = ((*x*1Ḥ0)*/*2*w*2) + (((*x*2Ḥ0)*/*2*w*2) (8.10)

Conectaremos estos valores. Los valores **x1** y **x2** especifican cuán lejos está la neurona actual de la BMU. El valor de **x1** especifica la distancia de columna y el valor de **x2** especifica la distancia de fila. Debido a que la neurona dos está en la misma fila, entonces **x2** será cero. La neurona dos es sólo una columna hacia delante de la BMU, por lo que **x1** será 1. Esto nos da el siguiente cálculo para v.

*v* = ((0 − 0)*/*(2 ∗ 3)2) + ((1 − 0)*/*(2 ∗ 3)2) = 0*.* 0277 (8.11)

Ahora que v se ha calculado podemos calcular el Gaussian.

*exp*(−*v*) = 0*.* 9726 (8.12)

Como se puede ver, una neurona tan cerca de la BMU obtiene casi toda la formación que la BMU recibió. Aparte de usar el valor anterior para la función de vecindario, el peso cal- culation para la neurona dos es el mismo que la neurona uno.

# Cálculo de errores SOM

Mientras entrenamos redes neuronales feedforward normalmente calcularíamos un número de error para indicar si el entrenamiento es exitoso. Un SOM es una red neuronal no supervisada. Dado que no está supervisado, el error no se puede calcular por medios normales. De hecho, debido a que no está supervisado, realmente no hay ningún error en absoluto.

### Capítulo Summary 87

Si no se conocen, los datos ideales entonces ere no es nada contra lo que calcular un error. Sería útil ver algún tipo de número para indicar la progresión del entrenamiento.

Muchas implementaciones de red neuronal ni siquiera informan de un error para el SOM. Addi- tionally, muchos artículos y libros sobre SOM no proporcionan una manera de calcular un error. Como resultado, no hay ninguna manera estándar de calcular un error para un SOM.

Sin embargo, hay una manera de informar de un error. El error de un SOM se puede considerar como la peor (o más larga) distancia euclidiana de cualquiera de las mejores unidades coincidentes. Este es un número que debe disminuir a medida que avanza la capacitación. Esto puede darle una indicación of cuandosu SOM ya no está aprendiendo. Sin embargo, es sólo una indicación. No es una verdadera tasa de error. También es más una "distancia" que un porcentaje de error.

# 8.4 Resumen del capítulo

Este capítulo introdujo una nueva red neuronal y un método de entrenamiento. El mapa de autoorganización organiza automáticamente los datos en clases. No se proporcionan datos ideales; el SOM simplemente agrupa los datos de entrada en clases similares. El número de clases de agrupación debe remain constante.

La formación de un SOM requiere que determine la mejor unidad de coincidencia (BMU). La BMU es la neurona cuyos pesos coinciden más estrechamente con la entrada que estamos entrenando. Los pesos de la BMU se ajustan para que coincidan más estrechamente con la entrada que estamosinsinuiendo. Además, también se ajustan los pesos cercanos a la BMU.

En este próximo capítulo veremos otra arquitectura de red neuronal. En este capítulo usamos. Este tipo de red neuronal se denomina red neuronal RBF. Las redes RBF son similares a las redes neuronales feedforward en el sentido de que pueden tener múltiples neuronas de salida que producen números de punto flotante. Las redes RBF no clasifican como las de SOM. Sin embargo, las redes RBF hacen uso de las mismas funciones de base radial que utiliza un SOM.