### 29

# Capítulo 4

# Retro-propagación

* Comprensión de los degradados
* Cálculo de gradientes
* Comprensión de la retro-propagación
* Impulso y tasa de aprendizaje

Hasta ahora sólo hemos visto cómo calcular la salida a partir de una red neuronal. La salida de la red neuronal es el resultado de aplicar la entrada a la red neuronal a través de los pesos de varias capas. En este capítulo veremos cómo se ajustan estos pesos para producir salidas más cercanas a la salida deseada.

Este proceso se llama entrenamiento. El entrenamiento es un proceso iterativo. Para hacer uso del entrenamiento, realiza múltiples iteraciones de entrenamiento. Se espera que las iteraciones de entrenamiento reduzcan el error global de la red neuronal. En el Capítulo 2 se examinó un error global y local.

# Comprensión de los degradados

El primer paso es calcular los gradientes de la red neuronal. Los degradados se utilizan para calcular la pendiente, o degradado, de la función de error para un peso determinado. Esto permite que el método de entrenamiento sepa aumentar o disminuir el peso. Hay un número de diferentes métodos de entrenamiento que hacen uso de degradados. Estos métodos de entrenamiento se denominan entrenamiento de propagación. Este libro discutirá los siguientes métodos de entrenamiento de propagación.

* + - Backpropagation
    - Propagación resiliente
    - Propagación rápida

Este capítulo se centrará en el uso de los degradados para entrenar la red neuronal utilizando la agregación de apoyo posterior. Los próximos capítulos se centrarán en los demás métodos de propagación.

## Qué es un gradiente

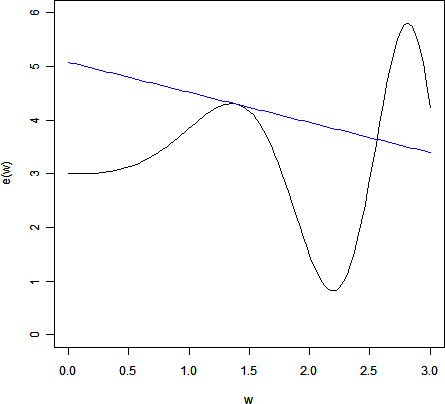
Comenzaremos mirando lo que es un gradiente. Básicamente el entrenamiento es una búsqueda. Está buscando el conjunto de pesos que hará que la red neuronal tenga el error global más bajo para un conjunto de entrenamiento. Si tuviéramos una cantidad infinita de recursos de cálculo, simplemente probaríamos todas las combinaciones posibles de pesos y veríamos cuál proporcionaba el mejor error global absoluto.

Debido a que no tenemos recursos informáticos ilimitados tendremos que tomar algún tipo de corte sh ort. Un acceso directo es en esencia todo lo que realmente son los métodos de entrenamiento de la red neuronal. Cada método de entrenamiento es una forma cortante de encontrar un conjunto óptimo de pesos sin realizar una búsqueda exhaustiva impostarle.

Considere un gráfico que muestre el error global de una red neuronal para cada peso posible.

Este gráfico podría tener un aspecto similar a la Figura 4.1.

**Figura 4.1:** Degradado



Mirando este gráfico se puede ver fácilmente el peso óptimo. El peso óptimo es la ubicación donde la línea tiene el valor y más bajo. El problema es que no podemos ver todo el gráfico. Esta sería la búsqueda exhaustiva mencionada anteriormente. Sólo vemos el error para el valor actual del peso. Sin embargo, podemos determinar la pendiente de la curva de error en un peso determinado. En el gráfico anterior vemos la pendiente de la curva de error en 1.5. La pendiente es dada por la línea recta que apenas toca la curva de error en 1.5. Esta es esta pendiente es el degradado. En este caso la pendiente es -0.5622.

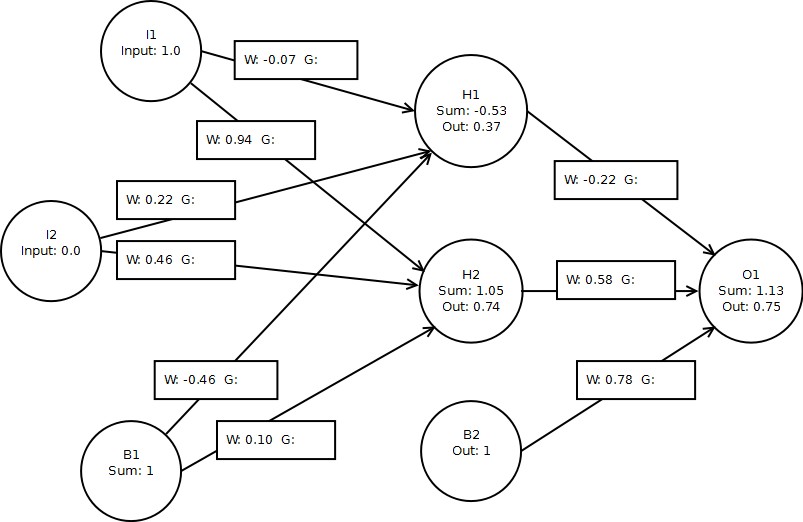
El degradado es la pendiente instantánea de la función de error en el peso especificado. Esta es la misma definición para la derivada que aprendimos en el Capítulo 3. El degradado es dado por la derivada de la curva de error en ese punto. Esta línea nos dice algo acerca de lo empinada que es la función de error en el peso dado.

Utilizado con una técnica de entrenamiento, esto puede proporcionar una idea de cómo se debe ajustar el peso para un error más bajo. Ahora que hemos visto lo que es un degradado, veremos cómo calcular realmente un degradado en la siguiente sección.

## Calculating Gradients

Ahora veremos cómo calcular realmente el gradiente. Calcularemos un gradiente individual para cada peso. Un peso es una conexión entre dos neuronas. Les mostraré las ecuaciones, así como cómo aplicarlas a unared real neural con números reales. La red neuronal que usaremos se muestra en la Figura 4.2.

**Figura 4.2:** Una red XOR



La red neuronal de arriba es una red típica de tres capas como hemos visto antes. Los círculos indican neuronas. Las líneas que conectan los círculos son las ponderaciones. Los rectángulos en el centro de las conexiones dan el peso para cada ión de conexión.

El problema al que debemos enfrentarnos ahora es cómo calcular el derivado parcial para la salida de cada neurona. Esto se puede lograr mediante un método basado en la regla de cadena de Cálculo. La regla de la cadena se discutió en el Capítulo 3. Comenzaremos con un elemento de conjunto de entrenamiento. Para la Figura 4.2 estamos proporcionando una entrada de **[1,0]** y esperando una salida de **[1]**. Se puede ver la entrada que se aplica en la figura anterior como la primera neurona de entrada tiene un valor de entrada de 1.0 y la segunda neurona de entrada tiene un valor de entrada de 0.0.

Esta entrada se alimenta a través de la red y, finalmente, produce una salida. El proceso exacto por el cual se calculan la salida y las sumas se cubrió en el Capítulo 1. Backpropa- gation tiene un pase hacia adelante y hacia atrás. Cuando seprodujo la pasada haciaadelante, se produjo cuando se calculó la salida de la red neuronal. Calcularemos los degradados solo para este elemento en el conjunto de entrenamiento. Otros elementos del conjunto de entrenamiento tendrán diferentes degradados. La forma en que se combinan los degradados para cada elemento de conjunto de entrenamiento individual se analizará más adelante en este capítulo cuando analicemos "Formación por lotes y en línea".

Ahora estamos listos para calcular los gradientes. Hay varios pasos involucrados en calcu- lating los gradientes para cada peso. Estos pasos se resumen aquí.

* Calcular el error, basado en el ideal del conjunto de entrenamiento
* Calcular el delta del nodo para las neuronas de salida
* Calcular el delta del nodo para las neuronas interiores
* Calcular gradientes individuales

Estos pasos se tratarán en las siguientes secciones.

## Cálculo del nodo Deltas

El primer paso es calcular un valor constante para cada guiño, o neurona, in la red neuronal. Comenzaremos con los nodos de salida y trabajaremos nuestro camino hacia atrás a través de la red neuronal. De aquí viene el término backpropagation. Inicialmente calculamos los errores para las neuronas de salida y propagamos estos errores hacia atrás a través de la red neuronal.

El valor que calcularemos para cada nodo se denomina delta del nodo. El término layer delta también se utiliza a veces para describir este valor también. Layer delta describe el hecho de que estas deltas se calculan una capa a la vez. El método para calcular las deltas de nodo varía en función de si está calculando para un nodo de salida o interior. Las neuronas de salida son obviamente, todos los nodos de salida. Las neuronas ocultas y de entrada son los nodos interiores. La ecuación para calcular el delta del nodo se proporciona en la ecuación 4.1.

*δ* −*EfiJ*

=

*i* *fJ* Σ *w δ*

*Yo*

*K*

*Ki*

*K*

, nodos de salida

, nodos más interier

(4.1)

Calcularemos el delta del nodo para todas las neuronas ocultas y no sesgadas. No hay necesidad de calcular el delta del nodo para las neuronas de entrada y sesgo. El delta del nodo se puede calcular fácilmente para las neuronas de entrada y sesgo utilizando la ecuación anterior. Sin embargo, estos valores no son necesarios para el cálculo del degradado. Como pronto verá, gradient calculación para un peso sólo mira la neurona a la que está conectado el peso. El sesgo y las neuronas de entrada son sólo el punto inicial para una conexión. Nunca son el punto final.

Comenzaremos usando la fórmula para las neuronas de salida. Observará que la fórmula utiliza un valor **E**. Este es el error de esta neurona de salida. Puede ver cómo calcular **E** a partir de la ecuación 4.2.

*E* = (*a* − *i*) (4.2)

Puede recordar una ecuación similar a la Ecuación 4.2 del Capítulo 3. Esta es la función de error. Aquí restamos el ideal de lo real. Por ejemplo, la red neuronal proporcionada anteriormente éste Se convierte el siguiente.

E = 0 . 7 5 − 1 . 00 = −0.25

Ahora que tenemos **E**, podemos calcular el delta del nodo para el primer (y único) nodo de salida. Rellenando la Ecuación 4.1 obtenemos lo siguiente.

−( −0.25) ∗ dA( 1 . 1 2 5 4 ) = 0 . 185 ∗ 0 . 25 = 0 . 0 5

El valor de 0.05 se utilizará para el nodo delta de la neurona de salida. En la ecuación anterior, **dA** representa el derivado de la función de activación. Para este ejemplo, estamos utilizando una función de activación sigmoid. La función de activación sigmoid se muestra en la ecuación 4.3.

1

*s*(*x*) =

1 + *e−x*

(4.3)

El derivado de la función sigmoide se muestra en la ecuación 4.4.

*fj*(*x*) = *s*(*x*) ∗ (1*.* 0 − *s*(*x*)( 4.4)

Ahora ese el nodo delta tiene sido calculado para el salida neurona Nosotros deber calcular eso para las neuronas interiores, así. La ecuación para calcular el delta del nodo para las neuronas interiores era con tal que en ecuación 4.1. Appling éste para el Primero escondido neurona Nosotros have el siguiente.

dA( suma o F H1) ∗ (O1 nodo delta ∗ peso o F H1 −*>* O1)

Usted notará que la ecuación 4.1 llamó para la suma de un adormecimientoer de los elementos basado en el número de conexiones entrantes para producir la neurona uno. Debido a que sólo hay una conexión entrante a la neurona de salida uno, solo hay un valor que sumar. Este valor es el producto de la neurona de salida delta de un nodo y el peso entre la neurona oculta uno y la neurona de salida uno.

Rellenando valores reales para la expresión anterior, nos quedamos con lo siguiente.

dA( −0 .53 ) ∗ ( 0 . 0 5 ∗ −0.22) = −0.0025

el valor -0.0025 es el nodo delta para el Primero escondido neurona. calculador el segundo escondido neurona sigue exactamente lo mismo forma como arriba. La segunda neurona se calcularía como Sigue.

dA( suma o F H2) ∗ (O1 nodo delta ∗ peso o F H2 −*>* O1)

Conectando números reales, tenemos lo siguiente.

dA( 1 . 0 5 ) ∗ ( 0 . 0 5 ∗ 0 . 5 8 ) = 0 . 0055

El valor de 0,0055 es el delta del nodo para la segunda neurona oculta.

Como se explicó anteriormente, no hay razón para calcular el delta del nodo para las neuronas de sesgo o las neuronas de entrada. Ahora tenemos todas las capas delta necesarias para calcular un gradiente para cada peso en la red neuronal. El cálculo de los degradados individuales se discutirá en la siguiente sección.

## Cálculo de los degradados individuales

Ahora podemos calcular los gradientes individuales. A diferencia de las deltas de capa, solo se utiliza una ecuación para calcular el degradado real. Un degradado se calcula utilizando la Ecuación 4.5.

*∂E*

*∂w (ik*)

= *δk*

· *oi*

(4.5)

La ecuación anterior calcula el derivado parcial del error (**E**) con respecto a cada pesoen- dividual. Los derivados parciales son los degradados. En el Capítulo 3 se examinaron los derivados parciales. Para determinar un gradiente individual multiplicar el delta del nodo para la neurona objetivo por el peso de la neurona de origen. En la ecuación anterior **k** representa la neurona objetivo y **yo** representa la neurona de origen.

Para calcular el gradiente para el peso de **H1** Para **O1** los siguientes valores serían usado.

salida ( h1 ) ∗ nodo delta ( o1 ) ( 0 . 3 7 ∗ 0 . 0 5 ) = 0 . 01677

Es importante tener en cuenta que en la ecuación anterior estamos multiplicando por la salida de oculto 1, no por la suma. Al tratar directamente con un derivado debe proporcionar la suma. De lo contrario, aplicaría indirectamente la función de activación dos veces. En la ecuación anterior no estamos tratando directamente con el derivado, por lo que utilizamos lasalida delnodo regula r. La salida del nodo ya ha aplicado la función de activación.

una vez el Gradientes son calculado el individual Posiciones de el Pesos No más tiempo materia. Nosotros enlatar simplemente pensar de el Pesos y Gradientes como soltero dimensional Matrices. el individual métodos de entrenamiento que veremos tratarán todos los pesos y gradientes iguales. No lo hace asunto si un peso es de una neurona de entrada o una neurona de salida. Sólo es importante que el peso correcto se utiliza con el degradado correcto. El orden de esto queight y gradiente arreglo es arbitrario. sin embargo Encog usos el siguiente orden para el encima neural red.

Peso/ Gradiente 0 :

Peso/ Gradiente 1 :

Peso/ Gradiente 2 :

Peso/ Gradiente 3 :

Peso/ Gradiente 4 :

Peso/ Gradiente 5 :

Peso/ Gradiente 6 :

Peso/ Gradiente 7 :

Peso/ Gradiente 8 :

Oculto 1 −*>* Salida 1

Oculto 2 −*>* Salida 1

Entrada 1 −*>* Oculto 1

Sesonía 2 −*>* Salida 1

Entrada 2 −*>* Oculto 1

Entrada 1 −*>* Oculto 2

Sesonía 1 −*>* Oculto 1

Entrada 2 −*>* Oculto 2

Sesonía 1 −*>* Oculto 2

Los diversos algoritmos de aprendizaje harán uso de las matrices de peso y degradado. También es importante tener en cuenta que se trata de dos matrices independientes. There es una matriz de peso, así como una matriz de degradado. Ambas matrices tendrán exactamente la misma longitud. Entrenar una red neuronal no es más que ajustar los pesos para proporcionar una salida deseable. En este capítulo veremos cómo backpropagation utiliza la matriz gradient para modificar la matriz de peso.

# Aplicación de la propagación posterior

La backpropagation es un método de entrenamiento simple que utiliza los gradientes calculados de una red neuronal para ajustar los pesos de la red neuronal. A medida que estos pesos se ajustan la red neuronal debe producir una salida más deseable. El error global de la red neuronal debe caer a medida que se entrena. Antes de que podamos examinar elproceso de actualización de pesaje de backpropagation t debo mirar dos maneras diferentes en que se pueden calcular los degradados.

## Formación por lotes y en línea

Ya hemos visto cómo calcular los degradados para un elemento de conjunto de entrenamiento individual. Más temprano en este capítulo vimos cómo podíamos calcular los gradientes para un caso en el que la red neuronal recibió una entrada de [1,0] y se esperaba una salida de [1]. Esto funciona bien para un único elemento de conjunto de entrenamiento. Sin embargo, la mayoría de los conjuntos de entrenamiento tienen muchos elementos. Hay dos maneras diferentes de manejar varios elementos de conjunto de entrenamiento. Estos enfoques two se denominan entrenamiento en línea y por lotes.

El entrenamiento en línea implica que modifique los pesos después de cada elemento del conjunto de entrenamiento. Con los degradados que obtuvo para el primer elemento de conjunto de entrenamiento, calcula y aplica un cambio a la ponderacións. El entrenamiento progresa al siguiente elemento del conjunto de entrenamiento y también calcula una actualización de la red neuronal. Esta formación progresa hasta que se ha utilizado cada elemento del conjunto de entrenamiento. En este punto se ha completado una iteración, o época, de entrenamiento.

El entrenamiento por lotes también hace uso de cada elemento del conjunto de entrenamiento. Sin embargo, los pesos no se actualizan para cada elemento de conjunto de entrenamiento. Más bien se suman los degradados para cada elemento del conjunto de entrenamiento. Una vez utilizado cada elemento del conjunto de entrenamiento, se pueden actualizar losn pesos de red eural. En este punto, la iteración se considera completa.

A veces se establece un tamaño de lote. Por ejemplo, es posible que tenga un tamaño de conjunto de entrenamiento de 10.000 elementos. Puede optar por actualizar los pesos de la red neuronal cada 1.000 elementos. Esto haría que los pesos de la red neuronal se actualizarían 10 veces durante la iteración de entrenamiento.

El entrenamiento en línea fue el método original utilizado para lagation backpropa. Sin embargo, el entrenamiento en línea es ineficiente, ya que la red neuronal debe actualizarse constantemente. Además, la capacitación en línea es muy difícil de implementar en una mansión multiproceso que aprovechará los procesadores multinúcleo. Debido a estas razones, el entrenamiento por lotes es generalmente preferible a la capacitación en línea.

## Actualización de peso de backpropagation

Ahora estamos listos para actualizar los pesos. Como se mencionó anteriormente, trataremos los pesos y gradientes como unamatriz dimensional single. Dé a estas dos matrices que estamos listos para calcular la actualización de peso para una iteración del entrenamiento de backpropagation. La fórmula para actualizar los pesos para la backpropagation se muestra en la ecuación 4.6.

∆*w*(*t*)

= *ε* *∂E*

*∂w*(*t*)

+ *α*∆*w*(*t-*1)

(4.6)

La ecuación anterior calcula el cambio de peso para cada elemento de la matriz de peso. También observará que la ecuación anterior requiere el cambio de peso de la iteración anterior. Estos valores deben mantenerse en otra matriz.

La ecuación anterior ca lcula el delta de peso para ser el producto del gradiente y la tasa de aprendizaje (representado por epsilon). Además, se añade el producto del cambio de peso anterior y el valor de momento (representado por alfa). La tasa de aprendizaje y el impulso son dos parámetros que se deben proporcionar al algoritmo de backpropagation. La elección de valores para la tasa de aprendizaje y el impulso es muy importante para el rendimiento de la formación. Desafortunadamente, el proceso para determinar la tasa de aprendizaje y el impulso es principalmente ensayo y error.

La tasa de aprendizaje escala el gradiente y puede ralentizar o acelerar el aprendizaje. Una tasa de aprendizaje por debajo de cero ralentizará el aprendizaje. Por ejemplo, una tasa de aprendizaje del 0,5 disminuiría cada gradiente en un 50%. Una tasa de aprendizaje superior a 1,0 aceleraría el entrenamiento. En realidad, la tasa de aprendizaje casi siempre está por debajo de cero.

La elección de dos altas de una tasa de aprendizaje hará que su red neuronal no converja. Una red neuronal que no está convergiendo generalmente tendrá un alto error global que simplemente rebota alrededor, en lugar de converger a un valor bajo. Elegir una tasa de aprendizaje demasiado baja hará que la red neuronal tarde mucho tiempo en converger.

Al igual que la tasa de aprendizaje, el impulso también es un factor de escala. Momentum determina qué porcentaje del cambio de peso de la iteración anterior debe aplicarse a esta iteración. Momentum es opcional. Si no desea utilizar el impulso, simplemente especifique un valor de cero.

Momentum es una técnica que se añadió a la backpropagation para ayudar al entrenamiento a encontrar su salida del minima local. El mínimo local son puntos bajos en el gráfico de errores que no son el verdadero mínimo global. Backpropagation tiene una tendencia a encontrar su camino en un mínimo local y no encontrar su camino de vuelta de nuevo. Esto hará que el entrenamiento converja a un error indeseable más alto. Momentum le da a la red neuronal cierta fuerza en su dirección actual y puede permitirle forzar a través de un mínimo local.

Ahora estamos listos para conectar valores a la Ecuación 4.6 y calcular un delta de peso. Calcularemos un cambio de peso para la primera iteración utilizando la red neuronal que usamos anteriormente en este capítulo. Hasta ahora sólo hemos calculado degradados para un elemento de conjunto de entrenamiento. Todavía hay otros cuatro elementos de conjunto de entrenamiento para calcular. Sumaremos los cuatro gradientes

juntos para aplicar el entrenamiento por lotes. El degradado por lotes se calcula sumando el Gradientes cuál son Listado aquí.

0−. 0. 05554301180824532

0 . 016777795762852397

−0.05861906411780882

021940533165555356

Anteriormente en el capítulo calculamos el primer degradado, mencionado anteriormente. Si desea ver el cálculo de los demás, esta información se puede encontrar en la siguiente dirección URL.

<http://www.heatonresearch.com/wiki/Back_Propagation>

Sumar todo de estos Gradientes Produce el siguiente lote actualizar gradiente.

−0.07544358513197481

Para esta red neuronal usaremos una tasa de aprendizaje de 0,7 y un impulso de 0,3. Estos son sólo valores arbitrarios. Sin embargo, funcionan bien para entrenar una red neuronal XOR. La conexión de los valores e en la ecuación 4.6 da como resultado lo siguiente.

d E L gracias = ( 0 . 7 ∗ −0.0754) + ( 0 . 3 ∗ 0 . 0 ) = −0.052810509592382364

Esta es la primera iteración de entrenamiento, por lo que el valor delta anterior es 0.0. El impulso no tiene ningún efecto en la primera iteración. Este valor delta se añadirá al peso para alterar la red neuronal para la primera iteración de entrenamiento. Todos los demás pesos de esta red neuronal se actualizarán de la misma manera, según su gradiente calculado.

Esta primera iteración de entrenamiento reducirá ligeramente el error global de la red neuronal. Addi- tional las iteraciones de entrenamiento reducirán aún más el error. La siguiente salida del programa muestra el convergencia de éste neural red.

época #1 error : 0 . 3100155809627523

época #2 error : 0 . 2909988918032235

Epoch #3 Error : 0 . 2712902750837602

Epoch #4 Error : 0 . 2583119003843881

Epoch #5 Error : 0 . 2523050561276289

Epoch #6 Error : 0 . 2502986971902545

Epoch #7 Error : 0 . 2498182295192154

Epoch #8 Error : 0 . 24974245650541688

Epoch #9 Error : 0 . 24973458893806627

Epoch #10 Error : 0 . 24972923906975902

. . .

Epoch #578 Error : 0 . 0100027023745037777

época #579 error : 0 . 009947830890527089

rojo neuronal re S u l Ts :

1 . 0 , 0 . 0 , ONU C tu un l = 0 .9040102333814147 , es decir, un l = 1.0

0 . 0 , 0 . 0 , ONU C tu un l = 0 .09892634022671229 , es decir, un l = 0.0

0 . 0 , 1 . 0 , ONU C tu un l = 0 .904020682439766 , Yo D E Un L = 1.0

1 . 0 , 1 . 0 , un c tu a l = 0 .10659032105865764 , es decir, a l = 0.0

Cada iteración, o época, disminuye el error. Una vez que el error cae por debajo del uno por ciento, el entrenamiento se detiene. También puede ver la salida de la red neuronal para los datos XOR. Las respuestas no son exactamente correctas, however, está muy claro que los dos casos de entrenamiento que deben ser 1.0 están mucho más cerca de 1.0 que los otros.

# Resumen del capítulo

En este capítulo se le introdujo a backpropagation. La backpropagation es uno de los algoritmos de entrenamiento más antiguos, y más utilizados,disponibles para redesneuronales. La retropromputación funciona calculando un valor de degradado para cada peso de la red. Muchos otros métodos de entrenamiento también hacen uso de estos valores de degradado.

Hay ungrad ient para cada peso en la red neuronal. El cálculo de los degradados es un proceso paso a paso. El primer paso para calcular los degradados es calcular el error para cada una de las salidas de la red neuronal. Este error es para un ele ment conjuntode entrenamiento. Los degradados de todos los elementos del conjunto de entrenamiento se pueden agrupar por lotes más adelante en el proceso.

Una vez calculado el error de la capa de salida, los valores se pueden calcular para cada una de las neuronas de salida. Estos valores se denominan deltas de nodo para each de las neuronas de salida. Primero debemos calcular las deltas de nodo para la capa de salida. Calculamos las deltas de nodo para cada capa de la red neuronal que se trabaja hacia atrás a la capa de entrada. Esta es la razón por la que esta técnica se denomina backpropagation.

Una vez calculados los deltas del nodo es muy fácil calcular los degradados. Al final tendrás todos los degradados para un elemento de conjunto de entrenamiento. Si está utilizando el entrenamiento en línea, ahora utilizará estos degradados para aplicar un cambio en los pesos de la red neuronal. Si utiliza el entrenamiento por lotes, sumará los degradados de cada uno de los elementos set de entrenamiento en un único conjunto de degradados para todo el conjunto de entrenamiento.

La backpropagation debe proporcionarse una tasa de aprendizaje y un impulso. Ambos son elementos de configuración que tendrán un efecto importante en la velocidad de entrenamiento de su red neuronal. La tasa de aprendizaje especifica la rapidez con la que se deben actualizar los pesos. Demasiado alto de un

[4.3 Resumen del capítulo](#_TOC_250000)   [41](#_TOC_250000)

la tasa de aprendizaje hará que una red se vuelva inestable. Una tasa de aprendizaje demasiado baja hará que la red neuronal tarde demasiado en entrenarse. Momentum permite que la red neuronal escape del mínimo local. El mínimo local son puntos bajos en el gráfico de errores que no son el verdadero mínimo global.

Elegir valores para la tasa de aprendizaje y el impulso puede ser complicado. A menudo es sólo una cuestión de prueba y error. El método de entrenamiento de propgación resistente (RPROP) no requiere que no se establezcan paramaters. Además, RPROP a menudo entrena mucho más rápido que la contrapropagación. RPROP se cubrirá en el siguiente capítulo.